2. Background & State of Art

In this chapter we provide background on concepts and technologies required to understand this thesis work. We describe state of art of related technologies and classify them according to properties we seek our system to hold. Finally we compare the most related work and comment on how it influences this dissertation.
2.1. Overlays and Application Networks

Overlays are networks constructed over another network, called the base network. From such definition it is derive a property with important consequences later in this work: several overlay can coexist over the same base network. Overlays are used for different purposes. Overlays are created to provide application or network services to a subset or to all clients of the base network with properties such as server proximity, redundancy, load balancing, consistency, security, multicast communication, scalable addressing, etc. Overlays can be classified by the functionality they provide: network services vs. application services, and how they provide this functionality: network layer vs. application layer overlays.

Fig. 2.1. Multiple overlays over a base network

Network service overlays differ from application service overlays in these characteristics: per node service, address scheme, overlay goal, coordination rules and connectivity. Network service overlays provide a data packet transmission service; network service nodes are packet-forwarding engines. Whereas application service overlays provide end-user applications with complex functionality such as information caching, remote redirections, session management, etc., application service overlays nodes are full application server with access to persistent storage, graphical user interfaces, etc. Network service overlay nodes are identified by a network specific address scheme, usually either Ipv4 or
Ipv6. Application service overlay nodes are identified by application specific address schemes: Universal Resource Identifiers, Channels Ids, etc. Network service overlays try to minimize end-to-end packet transmission latency and minimize overall bandwidth cost. Application service overlays try to optimise user experience decreasing response time and providing service redundancy. Coordination rules are very different. Network service overlays contain routing rules to be applied to every packet, whereas application service overlays are coordinated by complex rules applied to application messages, “if message can not be processed here forward to x”, “if load increases create a replica nearer to clients”, etc. To achieve service goal using such coordination rules, node services are connected in a specific way. Network service overlays topology is a bi-directional non-cyclic spanning tree, whereas application service overlays are unidirectional hierarchies or meshes.

<table>
<thead>
<tr>
<th>Application Service Overlay</th>
<th>Network Service Overlay</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node services:</strong></td>
<td></td>
</tr>
<tr>
<td>Application server requiring permanent storage,</td>
<td>Packet transmission implemented by forwarding engine</td>
</tr>
<tr>
<td>and complex services.</td>
<td></td>
</tr>
<tr>
<td><strong>Addressing:</strong></td>
<td></td>
</tr>
<tr>
<td>Application specific: URL, channels ID,...</td>
<td>Network specific: Ipv4, Ipv6,...</td>
</tr>
<tr>
<td><strong>Overlay Goal:</strong></td>
<td></td>
</tr>
<tr>
<td>Good user experience: low response time, service</td>
<td>Minimize e2e latency, minimize bandwidth usage,...</td>
</tr>
<tr>
<td>redundancy, ..</td>
<td></td>
</tr>
<tr>
<td><strong>Connectivity Topology:</strong></td>
<td></td>
</tr>
<tr>
<td>Unidirectional hierarchies and meshes.</td>
<td>Bidirectional non-cyclic spanning trees.</td>
</tr>
<tr>
<td><strong>Coordination rules:</strong></td>
<td></td>
</tr>
<tr>
<td>Session redirection, content switching, replicate,</td>
<td>Packets routing.</td>
</tr>
<tr>
<td>caching rules, agents,…</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1.Application vs. network service overlays

Overlays can be built using network layer technology or application layer technology. This choice affects location of overlay nodes on the base network, and connectivity mechanisms among overlay nodes. Network layer overlay nodes are kernel processes executing in base network nodes accessing several base links. Application layer overlay nodes are user space processes executing in host machines which are located at edge nodes of the
base network, more or less distance to the core depending on its connection latency. A table of communication channels between pair of nodes defines connectivity among overlay nodes. Communication channels between two nodes using network layer technology are created using tunnels [Sim95]. Tunnels operation is as follow: source overlay nodes encapsulate service packets, containing destination overlay address plus data, in a network layer packet whose destination address is a base node where it is the corresponding destination overlay node. Source nodes implement a mechanism that marks tunnelling packets for destination deencapsulation. Destination nodes implement a mechanisms that deencapsulate packets marked as tunneled, read destination overlay address, and forward such service packet to the corresponding destination overlay node. A tunnel is thus defined by a 4-tupla \{source overlay address, source base address, destination base address, destination overlay address\}, which has to be configured at both tunnel end nodes. There exist several types of network tunnels, which are specified on different Internet RFC documents [Han94][Sim95][Ham99].

Whereas application layer overlay communication channels are application layer tunnels: UDP port pairs or TCP virtual circuits [Pos80][Pos81]. Source overlay nodes encapsulate service data, containing destination overlay address plus data, in a transport layer data payload whose destination address is a network layer address plus a port number. In that port there is an application process which read destination overlay address, and forward such service packet to corresponding destination overlay node. Port communication abstractions were invented so that several applications could connect from and to the same machine.

<table>
<thead>
<tr>
<th>Node locations:</th>
<th>Application layer overlay</th>
<th>Network layer overlay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hosts connected at edge.</td>
<td>Intermediate nodes.</td>
<td></td>
</tr>
<tr>
<td>Links:</td>
<td>TCP virtual circuits and UDP port pairs.</td>
<td>IP tunnels.</td>
</tr>
</tbody>
</table>

*Table 2.2. L7 vs. L3 Overlays*
From this classification we can create a matrix where four different types of overlay are possible:

- **Layer three overlay providing network service, network overlay**: have been created for incremental introduction of network service in non-contiguous networks.
- **Layer seven overlay providing network services, P2P network overlays**: incremental introduction of network service in non-contiguous networks implemented at application layer. Network services that have been introduced are multicast and active networks.
- **Layer seven overlay providing application services, which we call application network overlay**: are distributed applications which have been connected and coordinated for server proximity, redundancy, scalability, consistency, etc.
- **The fourth cell of table 2.3. might be filled by an overlay created in an active network where a distributed application service is provided by active network technology.**

<table>
<thead>
<tr>
<th>Network Service</th>
<th>Layer 3 overlay</th>
<th>Layer 7 overlay</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network overlays</strong></td>
<td>Mbone, 6bone, VPN,...</td>
<td><strong>P2P network overlays</strong> : Narada, Abone, Alan,....</td>
</tr>
<tr>
<td><strong>Application Service</strong></td>
<td>--</td>
<td><strong>Application networks</strong> : Proxy-caches hierarchy, chat networks, P2P applications: file sharing,...</td>
</tr>
</tbody>
</table>

*Table 2.3. Overlays Classifications*
2.1.1. Network Overlays

Network overlay have been constructed to obtain services the Internet as its base communication network did not provided. An overlay permits a new network service to be activated on a number of non-adjacent nodes. Those overlays provide a service non-existing in today’s Internet: multicast routing, Ipv6 addressing, secure data transmission, etc. Network overlays are constructed to introduce new network services in the Internet for a number of network clients, attempting to achieve such service goals: minimizing bandwidth usage, minimizing end-to-end latency by establishing connectivity among service nodes and coordinating them. However connectivity and coordination is done in a manual basis, resulting in non-optimal overlays in many cases. Connectivity is achieved using network layer tunnels; overlay nodes require support for tunnelling mechanisms, and virtual interfaces since equal addresses are use in many cases for base and overlay networks. Network overlays were proposed long ago for experimentation of new network services [McG82], [Hag89].

![Fig. 2.2. Network overlay](image)

*Fig. 2.2. Network overlay*
**Mbone, 6bone**

MBone [Eri94] is one of the very first operational large scale network overlays; it was created to access session of IETF meeting using multicast routing services. Even today any Internet node can create a tunnel and connect to it. 6bone is a similar overlay built to create a testbed for IPv6 experimentation [Gua00].

**VPN**

Virtual Private Networks VPN [Fer98] is the solution for corporate users that cannot establish secure group communications in today’s Internet. They make use of secure tunnels built with IP-SEC [Ken98], which are being offered by most Internet provider.

**Research Projects: Xbone, TEMPEST**

However all those overlays are manually created, projects as Xbone [Tou01] and TEMPEST [Van97] are providing mechanisms to automate deployment of such overlays, so that new overlays are introduce easily and adjustment of overlays to base network conditions and demand variations are dynamic. Xbone is researching infrastructure support for several concurrent overlays. TEMPEST researches algorithms to optimise overlay mapping in the base networks [Isa01].
2.1.2. Peer-to-peer Network Overlays

Since creating network overlays has demonstrated to be a burdensome task requiring complex tunnelling support by nodes. It were proposed network overlays built at the application layer with network services running as user programs in host machines, and TCP tunnels for connectivity among service instances. Such peer-to-peer network overlays have been built to provide multicast services, secure communications and active networks. All of these overlays suffer the same problem: they cannot map on the underlying base network, and create a lot of redundant traffic.

Application layer multicast overlays proposed to distributed live content in real time are Yoid [Fra00], Narada [Chu00] and Overcast [Jan00]. They differentiate on algorithms and metrics used to build topology and route packets. Other overlays aim at creating a testbed for active networks: Abone [Rie98b] and ALAN application layer active networks [Fry99].

![Fig. 2.3. Peer-to-peer network overlay](image-url)
2.1.3. Application Networks

Application networks are applications being serviced through multiple interconnected service nodes disseminated across the Internet to provide a good user experience in terms of better performance, fault tolerance and availability. Application servers provide a high level service requiring persistent data access, session management, cooperative spaces, etc. Application servers follow coordination rules for request forwarding, replica consistency, etc. Application servers communication through UDP / TCP connections, creating create an application layer topology. Application networks are overlays, because they are over other network: the Internet. Several such application networks coexist over the Internet.

Application networks have appeared as a means to provide application services with more quality of service. There exist application networks whose nodes are at the edge of the Internet: service nodes are near to users providing low latency and high throughput service to users; others provide multiple access points for redundant services, other provide data caching permitting faster data access to user requesting equal data.

Large companies that set up application servers around the world are creating some application networks. While other are created by Internet individual users which install an application server at their personal computer, and make it available to the rest of application users, so called peer-to-peer networks. This operation is very simple, because of this many application networks are being provisioned this way; however it does not make a cost-efficient use of resources since servers installed might not be near to users demand, and server connection might not be the most appropriate.

Application networks differentiate from network overlays and peer-to-peer network overlays on what are service instances at each node and which coordination rules are used. They are similar in that they all provide a self-contained service built on the coordination and connectivity among several service instances. As well multiple overlays of every kind can be
present over the same base network. From these similarities many of the contributions to application networks can be applied to the other type of overlays.

![Application network diagram](image)

**Fig. 2.4. Application network**

**DNS, NTP, News, Chat Networks**

A name service was needed to provide a highly available, efficient and scalable service on top of the Internet [Moc81]. A hierarchy of servers was created to provide such a service. DNS clients could connect to a number of servers; DNS servers connected to each other to resolve names. Network time servers are also connected worldwide for synchronization [Mil85]. News servers were also connected worldwide to distribute news messages [Kan86]. Chat networks connect chat servers for large-scale distributed chat rooms [Oik93].

**Content Distribution Networks**

Due to Web network traffic explosion in early 90’s, layer seven overlays were constructed in the form of hierarchies of proxy caching servers [Cha96] that routed web request to server containing cached web pages to reduce network traffic and improve user response time. As well replicated server networks were propose and created [Obr96][Lig97], which are the
predecessors of commercial content distribution networks, CDNs, such as Akamai [AKA02]. CDNs are large scale worldwide hosting services to distribute content efficiently to global audiences, decreasing latency and improving availability. Many CDN use accelerator servers, caching servers front-ending web servers to provided most popular web pages from fast memory which contact origin servers to obtain not so popular web pages or dynamic pages [Bes95]. Others use satellites for content distribution [Rod00]. More recently CDNs are providing streaming video services. Research was initiated by project such as Radar [Rab99], and DSI [Bec98], having being continued by many.

**Peer-to-peer application networks**

Peer-to-peer applications are cooperative networks build linking resources provided by end users. They connect their personal computers to take advantage of a large number of distributed resources for file sharing, computing power sharing, etc. In its origins coordination was provide by a centralized server that gathered all peers’ availability information, as Napster [Ora01]. Second generation peer-to-peer networks such Gnutella [Ora01] and Freenet [Cla00] communicate many to many to discover files in a decentralized fashion. Third generation peer-to-peer networks use DHT distributed hash table algorithms [Dab01]. Several applications are being created with this architecture: group communication system [GRO01], large scale archives as OceanStore [Kub00]. Even basic services frameworks that need to be standardized are being created such as JXTA [Gon01].
2.2. Programmable Internet Infrastructures

"The introduction of new services into existing networks is a manual, time consuming and costly process", "there is an increasing demand to add new services to networks to match new application needs" Campbell et al. [Cam99]. "vendors are hesitant to support service before they gain user acceptance, yet the utility of network services is dependant on their widespread availability" Tennenhouse et al. [Ten97]; "much human coordination is necessary to achieve functionality enhancements in large-scale deployments" Estrin et al. [Gov98]. These citations expose the relevance of facilitating service introduction by means of a programmable Internet infrastructure.

A programmable infrastructure is composed of a large number of nodes where new programs can be activated through a common interface. Since these nodes are distributed throughout the Internet, a remote service activation mechanism is incorporated. Programmable infrastructures are represented by a number of open boxes that provide an interface that permits to configure new service instances, or to program a new service figure 2.5. Such interface must be remotely accessible so that service providers can remotely create new services.

Fig. 2.5. Programmable infrastructure
It is a step ahead of local operating systems that allows sharing of local resources by user-activated applications or services. UNIX open computer programming platform [Ker84] has promoted the development of thousand of end applications on end user machines by providing a common interface for program development and execution. A programmable Internet infrastructure permits sharing of a large number of distributed resources by different services activated by different users. Internet programmable infrastructures will promote third party service development for the benefits for service developers with shorter service introduction time and responsiveness to demand variations, and benefited end-users with more and better services. Such change in the Internet would promote third party service development, and would provide users with more and better services; to promote development of new third party distributed services, which will in turn flood the Internet with new services for the benefit of end users. A system that facilitates service introduction will make it easy for service creator to provide their service to clients, thereby promoting development of new and innovative services. Besides it will benefit end users by providing them with a wide range of services to choose.

<table>
<thead>
<tr>
<th>Benefits of Programmable Internet Infrastructures:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ease incorporation of new services.</strong></td>
</tr>
<tr>
<td><strong>Responsive modification of services.</strong></td>
</tr>
<tr>
<td>Expected Consequences:</td>
</tr>
<tr>
<td><strong>Promote services development.</strong></td>
</tr>
<tr>
<td><strong>More and better services.</strong></td>
</tr>
</tbody>
</table>

*Table 2.4. Benefits and expectations of programmable infrastructure*
Nodes of a programmable infrastructure provide some programming abstractions with which to construct services. These programming abstractions are functions to be used by most services. Node will support either a configuration interface to create new services combining those abstractions, or programming languages for development of service programs with those abstractions. The ultimate function of the programmable infrastructure is to give services access to resources. Different types of resources can be provided by a programmable infrastructure, and different types of resources are demanded by services: network, storage, processing resources. Security and safety are two objectives looked for: a safe system provides protection against errors of trusted users, while a secure system protects against errors introduced by untrusted users. The difference of a programmable infrastructure with a conventional network is that a portion of the control plane is intentionally exposed, and this can lead to far more complex threats than exist with an inaccessible control plane. As well we have to distinguish between node-safe and network-safe programs, a program that can be safe for a node cannot be safe network wide. Security capabilities in many programmable networks projects are inherited from the Java secure programming environment on which are built.

There are two main types of programmable infrastructures: those supporting application layer services and those supporting network layer services. They differentiate in those characteristics that distinguish an application service from a network service: service requirements, programming abstractions, programming language, resource requirements and security support. Type and requirements of services to be introduced on a programmable infrastructure will affect every aspect of the programmable infrastructure. Application services are application directly access by end users which are designed to satisfy user requirements. They require good response times, data persistence, etc. Usually are fairly complex services that are implement in high-level language. They make use of high level
programming abstractions that simplify access to resources. Besides they require a variety of resources. Lastly but not less important application services require security end to end.

Network services are mainly packet forwarding services, which are required to maximize network throughput, with the smallest service interruption. These services are programmed in low-level languages to maximize throughput. There are a few programming abstractions that every service can make use such as routing tables, link interfaces or forwarding engines. Its principal resource is network capacity, but CPU and RAM memory are important resources. Security can be integrated at the link level or per packet.

<table>
<thead>
<tr>
<th>Layer 7 Programmable Infrastructure</th>
<th>Layer 3 Programmable Infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Programmable Service:</em></td>
<td>End user application</td>
</tr>
<tr>
<td><em>Service requirements:</em></td>
<td>User requirements: response time, persistence,...</td>
</tr>
<tr>
<td><em>Activation:</em></td>
<td>Per server or per session.</td>
</tr>
<tr>
<td><em>Programmable abstractions:</em></td>
<td>Process, server, virtual memory, socket, user interface,...</td>
</tr>
<tr>
<td><em>Programming language:</em></td>
<td>High level: C++, java, perl,...</td>
</tr>
<tr>
<td><em>Resources:</em></td>
<td>CPU, permanent storage, bandwidth, communication</td>
</tr>
<tr>
<td><em>Security:</em></td>
<td>Per session, per application.</td>
</tr>
<tr>
<td></td>
<td>Packet forwarding</td>
</tr>
<tr>
<td></td>
<td>Throughput, fault tolerance,...</td>
</tr>
<tr>
<td></td>
<td>Per process, session or packet.</td>
</tr>
<tr>
<td></td>
<td>Routing tables, link interfaces, forwarding engines,...</td>
</tr>
</tbody>
</table>

*Table 2.5. Network service vs. application service programmable infrastructure*
2.2.1. Application Layer Programmable Infrastructures

Research projects coming from various fields have created systems with similar properties to an application layer programmable infrastructure: distributed system researcher did propose the WebOS [Vah98], which aimed to allowed for generic program activation on global programming infrastructure. High performance computing created the Grid [Fos99] to harvest computing cycles from computers distributed throughout the Internet. Internet services researchers have adopted active mobile technology to build programmable infrastructures: AS.1 [Ami98] that allowed for web service transducer deployment on proxy server, Active Caches [Cao98] framework that allows for server application caching and execution. Storage infrastructures have been proposed to accommodate static data services [Bec98].

Internet-scale OS

Distributed operating system researchers objective was a global operating system that considered all resources on a global network, as the Internet, to be operating system resources. One of this projects WEBOS [Vah98] at Univ. of Washington aimed at creating a set of basic operating system services to support for geographically distributed, highly available, incrementally scalable, and dynamically reconfigurable applications. They have created a Rent-A-Server application with those primitives to dynamically automate web servers based on users demand. Some Internet-scale OS were reconverted into computational grids: infrastructures for automatic execution of parallel applications, such as Legion [Gri97].

Grids
Grids are used to harvest processing resources for computationally intensive applications. These resources are obtained from several nodes, thereby making it an ideal system for execution of parallel applications. Grid toolkit software is being created by Globus [Fos97], and grid testbeds are being created GUSTO[GUS00], IPC [Joh00], DataGrid [DAT01].

**Active Services**

Several active service frameworks have been proposed based in mobile secure agents technology. They aim at defining the equivalent of active networks technology at the application layer [Gov98]. The Active Service Framework at Berkeley [Ami98] enables clients to download and run application layer service agents at strategic locations inside the network; a multimedia transducer active service was prototyped, client requests launched a transducer services on any of several active proxy server, multimedia streams were transported to this nodes which downgraded stream quality for clients. Active caches [Cao98] cache programs download from origin servers for local execution.

**Distributed Storage Infrastructure**

Static data services with very large demands suggested the need for an storage only large scale infrastructure. Distributed storage infrastructure [Bec98] was in Internet2 project to create such large scale and distributed storage; it has been transformed into the logistic backbone Lbone [Bas02]. Also an economy based Intelligent Storage Market [Chu99] has been proposed.

**Proprietary programmable infrastructure**
Some proprietary application service providers systems allow for activation in several nodes of the proprietary infrastructure of clients applications. Systems such as Ejasent Application Service Network [EJA01] allow for on demand activation of UNIX application server on the network of Ejasent hosts. Proprietary Content Distribution Networks are modified to provide new services adding programming interfaces to their network servers. The goal is to increase number of services provided by CDN beyond content caching. The OPES (Open Pluggable Edge Services) [Bar02] aims at defining a framework for automatic activation of application layer intermediate services, such as transcoder, translators, virus scanners, at edge servers, such as content distribution networks servers. OPES services are executed on a per request basis, either on site or by remote callout mechanisms. Also many network management tools have been extended for remote software distribution and activation [Car98], though they are very close solutions.
2.2.2. Layer 3 Programmable Networks

One can consider a spectrum of possible choices from highly dynamic to more conservative levels of programmability. At one end of this spectrum, capsules [Wet98] carry code and data enabling the uncoordinated deployment of protocols. Capsules represent the most dynamic means of code and service deployment into the network. At the other end of the spectrum there are more conservative approaches to network programmability based on quasi-static network programming interfaces using RPCs between distributed controllers [Laz96] to deploy new services. Between the two extremes lie a number of other methodologies combining dynamic plug-ins, active messaging and RPC. Different approaches have a direct bearing on the speed, flexibility, safety, security and performance at which new services can be introduced into the infrastructure. A complete, though not very up to date, overview of programmable networks is found in [Cam99].

Active Networks

Active networks have been proposed by Internet community to allow for run-time per-packet service activation; it was established by DARPA in a series of research projects [DAN96]. The network activation approach taken by DARPA for programmable networks originates in a technological push from recent advancements in active mobile technologies such as Java, which permit to inject securely programs into the network thereby allowing for faster service creation; an demand pull from new applications which benefit from processing and storage capabilities in the path between application origin and end users in the form of caching, resolution transducers, or language translators.

Capsules [Wet98] was one of the very first and more extreme active networks, capsules are network packets carrying code and data enabling the uncoordinated deployment of
Switchware [Ale98] supports active extension which are dynamically loaded into active routers in a secure way, active packets only carry lightweight programs, therefore there is much less requirements for testing and verification. The CANEs project led by researchers at University of Kentucky and Georgia Tech. aim to define and apply service composition rules as a general model for network programmability [Bah97]. A composition method is used to construct composite network services from components. A composition method is specified as a programming language with enhanced language capabilities that operates on components to construct programmable network services. Other requirements of active networks have been researched subsequently: programming languages as PLAN [Hic98], node operating systems NodeOS [Pet00]. Finally the A-bone [Ric98b] overlay is being built to enable large-scale active networks experiments, it allows deployment of active networks execution environments and applications through an a-bone node API [Ric98a].

---

Fig. 2.6. Active Network

(Cortesy of DARPA http://www.darpa.mil/ato/images/programs/)
Open Signalling Networks

Open signalling has being proposed by telecom industry to construct fully programmable networks from lower to higher layers of the ISO reference data communication model by providing an open interface to telecommunications infrastructure, it evolved around the Open Signalling working group [OSI97]. Open signalling approach taken by telcos, and it predecessor open architectures, are follows up of traditional telecom networks signalling systems: SS7, Intelligent Networks and TINA management framework. Telecommunications networks provide QoS services, especially for voice telephony, however introduction of new services demostrated a hard task. On the other side on the Internet it is possible to allow for new services to be provided from an end point. The open signalling approach allows for providing some interface to network infrastructure for service creator to provide a new service. Open signalling aims at modifying network functionality by exposing a set of node APIs. This APIs allow network operator and users to select and customize nodes functionality. This interface should allow for the separation of network devices software and hardware businesses ensuring that the end users get the full benefit of competition in the marketplace, similar to OS APIs.

The most emblematic and precursors implementation of an open signalling customisable infrastructure is Xbind [Laz96]. Xbind network nodes present an API for remote customisation of node functionality through interface figure 2.3. Xbind research prototypes have shown ATM service creation [Laz96], and mobile service creation [Kou00]. There is an standardization effort by IEEE, standard P.1520 for network application interfaces [Bis98] to open up networking devices for middleware service programmability. Several forums have been established to develop such standard and its successors: Parslay, MSSF multiservice switching forum, etc.
Fig. 2.8. Xbind open signalling infrastructure interfaces
(Courtesy of Univ. of Columbia http://comet.ctr.columbia.edu/xbind/images/)
2.3. Comparison of Related Work

Related work has been classified according to the Internet layer where new functionality is being proposed. In following section 2.5.1., we discuss related work providing functionality at the application layer. In section 2.5.2., we discuss related providing functionality at network layer. Besides either section is sub-classified along coordinates in fig.2.10. The y-axis represents which control is exercised on services. On one extreme service instances are isolated from each other, without any control on its resource utilization or service quality. On the other extreme several entities connected and co-ordinated forming an overlay for efficient resource utilization, guaranteed service level, and other properties, provide a service. Service control is achieved because service instances are located, connected and coordinated for a common goal. Which nodes provide a service, and how they are connected and coordinated determine service quality, resource utilization, etc. in a per overlay basis. The x axis represents how programmable is the system, how easy it is to introduce a service; it can go from manual intervention being required; to programmable nodes where new service code can be activated on-the-fly, in between on-the-fly configuration of nodes.

![Fig.2.9. Coordinates for Classification](image)

Connected & Coordinated Service Instances

Service Control

(quality/service, resource utilization)

None

(Uncoordinated nodes)

Programmability

(on the fly Programming / Configuration)

Manual Modifications

(to temporal/spatial/functional variation)

B → C

A

Fig.2.9. Coordinates for Classification
Overlays are situated at point B in figure 2.10. Overlays are made up of a group of connected and coordinated entities that provide a service. By means of coordination service control is achieved. Coordination permits a level of service quality with determined resource utilization. As well overlays function as contained units with distinct properties from other overlays and the base network: they can be isolated from the rest of the system with its own coordination rules to provide its own service level. Control of services is achieved by coordinating an overlay.

**Fig.2.10. Overlay coordination facilitates service control**

Programmable infrastructures would be situated at point A in 2.10. Programmable infrastructures are represented by a number of open boxes, which provide an interface that permits to configure new service instances, or to program a new service. Such interface must be remotely accessible so that service providers can remotely create new services. In fig. 2.11., a service is introduce by configuring or programming a service instance on nodes of a programmable infrastructure.
Finally point C in figure 2.9. would be a system that evolves from an overlay, point B, by adding programmability. Overlays are created manually connecting and coordinating a group of nodes, therefore changes in demand require extensive work to adjust overlays. However in point C overlays are dynamically composed while retaining control to adjust its configuration and service level to demand variations. C evolves from static overlays by allowing new overlay services to be activated in a number of nodes, and connecting and coordinating those instances. On the other hand a system in C evolves from a programmable infrastructure, point A, adding service control by introducing services as overlays. In an Internet scale programmable infrastructure, huge number of existing resources and service to be provided makes it impossible for controlled service creation. A mechanism for controlled service introduction, such as overlays, will facilitate service introduction.

Fig.2.12. *Overlay service deployment facilitates controlled service introduction*
2.3.1. Comparison at Application Layer

Application layer services have an evolution a) towards application layer networks, and b) towards programmable application infrastructures, figure 2.13. Application layer networks are created connecting and coordinating a number of application servers. Mechanisms for connecting several server instances, TCP/UDP port abstraction, and to set up several servers in one node, multitasking, are standard in most operating system, because of these application networks have proliferated. However manual federation of independent individual servers creates most existing application networks: i.e. Peer-to-Peer networks, Web caches hierarchies or mirrored web services. Resource providers independently decide to activate an application service and connect it to other service instances already part of the network. This operation is very simple; because of this many application networks are being created. On the other hand manual coordination is not very effective: it does not make a cost-effective use of resources (servers installed might not have much demand), it does not provide efficient service (servers installed might not be near to users demand, server connection might not be the most appropriate), nor there are many new different services (servers are manually activated).

Existing application layer programmable infrastructures provide functionality for remote service activation, however they lack mechanisms for service coordination. Projects such as WebOS [Vah98] have tried to create an Internet wide Operating System by extending functionality of operating systems to enable them for the large scale. However they lack mechanisms for service control in the large scale: coordination mechanism, allocation mechanisms, etc. Active network technologies have also been proposed for remote activation of application layer services. AS1 [Ami98] clients create services by on demand activation on a nearby server; Active Caches [Cao98] and OPES [Bar02b] framework permit remote activation of cacheable servlet and edge servers programs. None of them permits coordination
among service instances; therefore resources utilization and response times are not controlled. Service can only be controlled if service instances are coordinated. Those programming infrastructures facilitate deployment of a service in any single node. However in an Internet scale programmable infrastructure, huge number of existing resources, and service to be provided would make it impossible for controlled service creation. Scale, complexity and security risks of the resulting infrastructure make it unviable. This thesis aims at solving that problem, how to deploy a coordinated application network in a programmable Internet service infrastructure.

Computational grids [Fos99a] are used to permit allocation of processing resources at several nodes for execution of computationally intensive parallel application. Little coordination is required among parallel applications instances, so they provide no support for coordinated application network deployment.

There exist some proprietary application network where application servers are dynamically reallocated, with more or less programmability. Hosting infrastructures as Radar [Rab99], and as Ejasent [EJA01] permit application to reserve resources capacity for a service in a number of nodes, it can provide the best possible service to users, since the nearest service point from any user provisions every application. As well some network management systems provide support for remote software distribution and activation [Car98]. However these proprietary solutions can only scale as much as the proprietary infrastructure allows, and are only as flexible as its owner permits. In fact in some cases for simplicity, services are allocated resources and activated on every service node, making a wasteful use of resources for very large number of service points or very large number of services.
Fig 2.13. Application Layer Related Work
2.3.2. Comparison at Network Layer

To make a good overview of related work we also classify research projects at layer 3. Networking technology is evolving in two dimensions a) towards network virtualization. b) towards infrastructure programmability, fig. 2.14.

Network overlays are being researched as a means to provide network services to a group of clients over a number of non-contiguous nodes in a controlled manner. Overlays such as Mbone or 6bone are slowly being created by manual configuration and connection of end clients to a core backbone. Some argue that this technologies have not taken up because it is too difficult to set up, and propose to create network overlays using application layer technologies: Yoid [Fra00] and Narada [Chu00] provide multicasting services, and Abone [Ric98a] provides active service with peer-to-peer overlays. Projests such as Xbone [Tou01] are dealing with the real problem: how to set up easily network overlays by dynamic deployment. The Xbone is intended as a system for deployment of different overlay networks: Mbone, 6bone or Abones. Initial releases permit deployment of virtual private networks implemented as IpSec tunneled infrastructures.

Internet programmability is being pursued with different levels of programmability from open signalling infrastructures such as Xbind [Laz96], to programmable infrastructures for per packet or per service activation as active networks [Ten97]. Both types of programmable infrastructures present an equal problem: how to introduce a service in a controlled way in such complex and security risky infrastructure when there are thousands of nodes where to activate a service and/or thousands or service creators interested in deploying a service. Beside there is a performance penalty, especially for active nodes, which is decreasing initial attention and interest. There has even been some commercial implementation by Nortel Networks [Nor01], but this are merely adds-on to existing products with more marketing that real usefulness.
Open signalling researches have realized that services should be introduced in a controlled manner, being deployed as an overlay. Such overlay should not interfere with other services and behave as a unit. Xbind project has continued research on support for network virtualization developing spawning networks [Kou01]. Other infrastructures that provide network virtualization are Tempest [Isa01]. As well Darwin [Cha98a], a resource allocation infrastructure, is being extended for virtual network service creation [Lim99].

Also some active networks researchers realized virtualization helps to deploy new services: the solution that has been proposed is to define a high level programming abstraction that allows programming at once a whole networked service. Such high level abstraction, which provides as well isolation, can be controlled easily. VAN [Sil98] Virtual active networks have been proposed as the virtual network abstraction in active network environment. Two different research groups have proposed Creation of VANs: Netscript project at Columbia supports virtual active networks [Sil98], abstractions that can be systematically composed, provisioned and managed. A generic VAN provisioning interface, and deployment of virtual active service networks is proposed in [Bru99].

Adding some programmability support to overlay deployment systems such as Xbone permits deployment of services in a controlled manner: extensions to Xbone permit deployment of network overlays with web proxy caching services [Ard00] and active network overlays [Wa02] where active services can be isolated.
All nodes provide equal service

Programmability

Fig 2.14. L3 Related Work State of Art
2.4. Conclusions of Related Work

We have discussed that currently application networks are created by manual service activation and connection by resource provider or at proprietary infrastructures. Besides application layer programmable networks allow uncoordinated service activation. At network layer overlay deployment systems are being developed. There is a gap, not cover by any existing system, which this thesis studies: a system that permits coordinated application network deployment on a programmable infrastructure. Dynamic deployment of application networks will facilitate service providers to introduce new application layer services in a coordinated manner for efficient resource usage and responsiveness to demand variations. This dissertation will provide a framework for application network deployment in the Internet.

Application networks today are manually created as a consequence they lack flexibility to adapt to demand variations. The work presented in this thesis will make it possible faster, more efficient and more diverse application network introduction because it automates deployment of application networks. Because application network creation, modification and removal is automated with no manual intervention required, they will be created, modified and removed in shorter time. Services created are more efficient since they are automatically allocated resources and coordinated for efficient resource usage and according to specific specifications, allowing for automatic adjustment. As well more and diverse services will be provided to users; service developers will be incentive to create new services if services can be provided easier. Existing proprietary application networks which have mechanism to adjust to demand variations cannot cope with every demand as an open system, which can gather more resources.
### Application Networks

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manual creation</td>
<td>• Automated deployment</td>
</tr>
<tr>
<td>• Slow modifications</td>
<td>• Fast and efficient adjustment</td>
</tr>
<tr>
<td>• Few applications</td>
<td>• Easy and fast to introduce new applications</td>
</tr>
</tbody>
</table>

*Table 2.6. Benefits of Thesis Work*

Programmable infrastructures were design with a similar goal, however they only provide a programming abstraction to activate services in a per node basis. A system for application network deployment shall make use of a programmable infrastructure. Some of them will be evaluated to be reused as the programmable infrastructure for application networks. The work presented in this thesis improves functionality provided by existing application layer programmable infrastructures, facilitating creation of services as coordinated application networks. Services are created in a controlled manner, provisioned with enough resources and coordinated conforming to some predetermined specifications: resource usage, service level, etc. Application network deployment will require functionality not provided by any programmable infrastructure due to the need for controlled connectivity and coordination among application instances.

Finally the work presented here concentrates on deployment of application layer services. Several projects are researching how to provide similar functionality at the network layer. Especially Xbone system for network overlay deployment [Tou98a][Tou01] influences this thesis work. Differences are a direct consequence of differences between network overlays and application overlays as shown in section 2.1: programmable infrastructure support, resource requirements, service complexity, coordination mechanisms, connectivity mechanisms, security threats, etc.
Grids are used for on demand execution of parallel application in several computers distributed throughout the Internet. Though they present similarities with this thesis work, again differences are a consequence of differences between parallel applications and application networks: coordination mechanisms, connectivity requirements, and service specifications.

<table>
<thead>
<tr>
<th><strong>Related Work</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• AS1 &amp; WebOS programmble application infrastructures</td>
</tr>
<tr>
<td>• Xbone system for nework overlay deployment</td>
</tr>
<tr>
<td>• Globus toolkit for on demand paralel application execution in grids</td>
</tr>
</tbody>
</table>

*Table. 2.7. Thesis main influences*