

# On Service Deployment on Peer-to-Peer Networks

O. Ardaiz, F. Freitag, L. Navarro

Computer Architecture Department, Polytechnic University of Catalonia, Spain  
{oardaiz, felix, leandro}@ac.upc.es

## Abstract

*Introduction of new services on the Internet is a laborious, time-consuming process. Peer-to-peer networks also present that same challenge; even augmented due to the larger set of potential service nodes and new services that can be created. A framework for service deployment will facilitate service introduction in both cases.*

*In this paper we present our investigations on two of the components of an Internet service deployment framework: simple service specification interface and mechanisms for service deployment. As well we present issues we have encounter when trying to implement this framework for service deployment in peer-to-peer networks. We have designed services templates to accommodate new possible services in peer-to-peer networks. We show how to extend peer-to-peer networks to support our proposed multicast injection service deployment mechanisms by adapting query flooding mechanisms and implementing multicast group communication mechanisms. Finally we present our prototypes and performance simulations on service deployment.*

## 1. Introduction

"The introduction of new services into existing networks is usually 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. [2], "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. [19]; these cites expose the relevance of facilitating service introduction in a network. That is, enabling an easy, efficient and secure introduction of new services will promote service development, flooding the Internet with new services for the benefit of end users. It is similar to the way UNIX open system has promote the development of host-based services. Service deployment is one of the technical challenges of the Internet that is being addresses

from various areas: from network engineering to system integration.

Peer-to-peer networks main characteristic is that all entities have similar capabilities; all entities are peer of each other. Peer-to-peer computing considers every entity as client and provider of services. However entities providing or requiring an equal service or resources group together with other peers, aggregating resource capacities, aggregating information resources, aggregating decision agents, etc. The service that is formed by such aggregation constitutes a many-to-many service where every peer can make use of every other peer capabilities. Peer-to-peer strengths are two: 1) entities obtain services from entities on a peer to peer relation, achieving total desintermediation; and 2) peers provide resources to other peers, which can join forces to achieve services not possible by any single peer.

Peer-to-peer services have to be introduced into the system similarly as today's Internet services: either on clients demand or by centralized management stations. In the first case service providers cannot plan service quality; the management stations model is not scalable and requires resources provider to give a lot of control on their resources to managers. Our solution, multicast injection, permits an easy-to-use, scalable and cost-effective service deployment at Internet networks and peer-to-peer networks. Thereby allowing peer-to-peer services creators to provide its services to more users, and benefiting users with a wider service offer.

### 1.1 Related Work

The problem of introduction of services into a network has been addressed with different approaches from active networking to open signaling to programmable networks [2]. Most of them have concentrated on providing environments where services can be remotely activated in a secure way. Research on mechanisms for network deployment is being studied at Xbone [20] in ISI and Tempest [15] in Cambridge University. They allow virtual networks to be set up on the Internet to behave as virtual private networks or to isolate experimental services from the Internet. Research on how to dynamically deploy a service is also being conducted at the Darwin project [3] at CMU which attempts to dynamically instantiate virtual

meshes, collection of networking resources, to be used for multiparty videoconferencing with quality of service set up. Globus project [10], a joint effort of various universities and research centers, has developed a protocol for advance reservation and co-allocation of resources in computational grids. It allows grid applications to obtain the set of required resources for its execution with QoS guarantees.

Our work concentrates on the development of efficient and cost-effective deployment mechanisms, and easy-to-use framework interfaces.

As well we are broadening the scope for service deployment to peer-to-peer networks. As we will discuss currently service deployment on peer-to-peer networks is accomplished in an on-demand basis when user download application and manually configure their service peer, as in most file-sharing applications i.e. Freenet [6]; or by a centralized manager that monitor peers state and remotely activate services, as in several computational intensive applications i.e. Entropia Internet Grid [8]. We propose a deployment mechanism that is more adaptable, and a simple service deployment interface for peer-to-peer network.

## 2. Internet Service Deployment Framework

### 2.1 Framework Requirements

To allow for ease creation of services requires the development of a framework that provides basic building blocks with which to construct a service deployment system. This framework should support the automation of every task required to deploy a service. We attempted to reuse the Xbone system for overlay deployment to deploy services, however it was not an ideal solution [1]. It required many fundamental changes in its deployment mechanisms, and it was designed for different allocation strategies, so we set ourselves to develop a framework for service deployment from scratch. Our first task was to define which functionality is demanded from this framework.

Deploying a service requires: obtaining service specifications, mapping specifications to resources, discovering resources, gathering resources, configuring resources, activating service, and providing a management interface.

Providing this functionality requires an architecture composed of resource agents at resource providers' nodes and deployment managers at service providers' nodes. Resource agents are responsible for publishing resources, mediating between resources and service providers' deployment managers, configuring resources, activating services, and returning management interfaces. Deployment managers are responsible for obtaining

service specifications, mapping specifications to resources, discovering resources, gathering resources, trading with resource agents on behalf of service providers, and managing overall deployment operation.

Service providers demand these characteristics from this framework:

- Usability, to allow for an easy service introduction,

Efficiency and cost-effectiveness, for rapid and cheap service provision,

- Manageability, to govern services once deployed,
- Safeness and fault tolerance, to assure service integrity and availability.

Resource providers demand these characteristics from this framework:

- Efficiency and cost-effectiveness, for highest resource revenue,
- Security, to avoid resource misuse.

There exist much research from active networks to programmable networks and even commercial system that provide security and management functionality to programmable nodes [2], which can be integrated in a framework for service deployment. However we have identified two framework components that provide functionality specific for this problem requiring extensive research to meet every party requirement. On the one hand it is required a simple service specification interface that allows for easy service deployment requests, on the other hand deployment mechanisms are required to provide for an efficient and cost-effective deployment.

### 2.2 Service Templates

Defining specifications of a service on the Internet is a laborious task. A service provider must calculate which resources capabilities and number of resources to use, sub-networks where service has to be provided, organization among resource, etc.

A service description language can be defined to facilitate service specification as it has been done to define overlay networks in [3] or [15] that would allow to specify service requirements in terms of number and type of basic resources, connections among resources, and service components. However the main goal of the service deployment framework is to facilitate service provider's deployment of their services. Reviewing the evolution of traditional programming environments we find that a programming languages is a complicated technique for many users since it offers more functionality that what an average user requires. For this reason, application frameworks such as Microsoft Foundation Classes or communication frameworks such as [9] have been introduced. These programming frameworks allow a service creator to implement their service starting from a

skeleton template, which already provides much of the required functionality. They only have to adapt it for their special service requirements. An important benefit of templates is that system components implementing those templates functionality will be reused many times, therefore programming errors can be reduced and systems components stream-lined.

We propose a service deployment framework that provides a set of predefined service templates. Service providers will select a template and fill it in to create his service specification.

For Internet services we have identified two service templates that can be instantiated to most Internet services: the dissemination service template and the access service template.

The dissemination service template allows the creation of content distribution networks, multicasting networks, video on demand services, and notification services. Service providers that choose this template know they will obtain a service specifically allocated, configured and organized for information dissemination. Service providers have to select which resource capabilities are required at nodes where to activate its services: storage and bandwidth. They have to select which Internet regions they want their service available, which is the expected traffic caused by service clients, which service engine to launch at every node: web surrogate cache, streaming server, video server, notification server, etc.

The access service template allows the creation of index services, proxy services, adaptation services or alarm services. This service gathers information from a number of resources that have some of these capabilities: sensors, crawlers, web servers, or even databases. Service providers have to indicate from which areas those resources have to be gathered. The service engine will filter, index, cache or adapt that information to be provided locally to an organization or remotely to any user. Service deployers have to indicate which is the expected maximum data traffic to be aggregated.

Deployment managers and resource agents deploying a service use specific algorithms for resource allocation, service organization and resource discovery for the corresponding dissemination or access templates. Dissemination services require resource allocation algorithms optimized for network traffic balancing, service organization algorithms optimized for information dissemination and resource discovery algorithm optimized for storage and traffic resource discovery. Access services require resource allocation algorithms optimized for data clustering, service organization algorithms optimized for faster response time, and resource discovery algorithms optimized for information searching.

## 2.3 Deployment Mechanisms

Deployment mechanisms make service deployers discover and interact with resource providers, and activate the service on several nodes. The first method is the SNMP management station; the second method we propose is multicast injection. The former represents the method currently used for provisioning services that require resource allocation at multiple nodes, such as virtual private networks or content distribution networks. In it in which a centralized entity monitors, chooses and configures resource agents [4]. As every centralized system, it is not the best solution in terms of scalability or fault tolerance. Multicast injection considers resource agents as active entities that can decide autonomously whether to accept or discard a service activation request based on local policies and other nodes decisions. Therefore deployer entities can only inject service specifications (including a reference to application binaries and data) into the system and expect that enough and appropriate resource agents accept it, else they can inject a cancel service request. Obviously injection has to be implemented with some multicast communication scheme.

The SNMP based method, per surrogate configuration deployment, involves the following steps:

1. Deployer continuously discovers and monitors surrogates resources / advertise their resources,
2. Provider requests service deployment,
3. Deployer calculates resource allocation,
4. Per surrogate configuration,
5. (At timeout), every surrogate response is ok OR deployer sends per surrogate rollback.

It is a centralized system where a deployer has to gather information from every surrogate in order to calculate resource allocations for every deployment requests. It requires computational and bandwidth resources in a centralized location, which is subject to failure. In addition, since more deployers will be contending for surrogates, a deployer can be denied allocation of resources in a surrogate that was thought to be available but another deployer got its resources a little earlier.

Our proposal, multicast injection deployment has to take the following steps:

1. Provider request service deployment,
2. Deployer injects application service specifications in a global mcast channel,
3. Surrogates map spec -> allocate resource and mcast service match on application channel OR do nothing,
4. Surrogates compare published matches with itself -> service activation OR cancel service and release resources,
5. (At timeout) every region serviced OR surrogate cancels service and release resources.

Clearly this method requires less resource on deployer entities, while permitting surrogates to have more autonomy. A problem of this solution is the level of under-utilization of surrogate resources due to conditional allocations while deployment takes place. In figure 1 we present advantages and disadvantages of both deployment mechanisms, which are discussed next.

	Advantages	Disadvantages
<b>Centralized Configuration</b>	-More control	-Deployer computation -Deployer traffic -Stale information -Unused allocations
<b>Multicast Injection</b>	-Faster activation -More adaptable -Simple deployers -Robust system -Loose relations	-More unused allocations -Network traffic

Figure 1 - Mechanisms comparison

Per node configuration presumes more control by the deployment entity over resource agents, since resource agents are passive entities controlled by deployers. In contrast, multicast injection presumes a looser relation between deployers and resource agents. Resource agents subscribe to deployers at will, retaining its autonomy on local configuration actions. It is easy to establish relations with more nodes when least requirements are put on both parties, therefore larger sets of resource agents can be available for multicast injection dynamic deployment. Per node configuration has to implement a centralized resource allocation algorithm, which can require high computational resources when computing on Internet scales. However multicast injection makes use of a distributed allocation algorithm, which makes it more scalable and fault tolerant. Per-node configuration is not as scalable since as the number of nodes grows it requires increased traffic capacity at the deployer site to continually monitor every resource. Multicast injection is more scalable than per node configuration since its traffic and computational requirements do not create a bottleneck at deployers. Stale resource information in per node configuration deployment causes deployer entities to select nodes that have been allocated, and not to consider for allocation nodes that have already available resources. Because multicast injection does not use stale information it is more responsive and adaptable than per node configuration deployment. Multicast injection requires simpler deployers since it just sends a service deployment request: they do not have to be continuously monitoring the state or individually configure every surrogate. It is

also more robust since deployers do not have to detect and recover from every surrogate failure. Unused allocations occurs in both cases when surrogates are allocated and released shortly afterwards without providing any service in that period, due to being unable to find enough surrogates for deployment. This effect has to be less important in per node configuration since there allocations are only requested to a subset of surrogates.

### 3. Service Deployment in Peer-to-Peer Network

#### 3.1 Peer-to-Peer Services Templates

Peer-to-peer networks main characteristic is that all entities have similar capabilities; all entities are peer of each other. Therefore on a peer-to-peer network communications and transactions among entities can take place by direct exchange between peer A and peer B.

In a peer-to-peer network exchanges are one-to-one, and thus services are basically one to one, there is no need for intermediation servers that service several entities. However entities providing or requiring an equal service or resources group together with other peers, aggregating resource capacities, aggregating information resources, aggregating decision agents, etc. The service that is formed by such aggregation constitutes a many-to-many service where every peer can make use of every other peer capabilities. But in fact all services are many-to-many since a many-to-many service provided by only one peer becomes a one-to-one service.

All peers providing MP3 file access provide a MP3 file sharing service to every other equal peer, since peers are clients and servers at the same time. All peers that provide their CPU cycles for parallel application execution provide a computational service to each other equal peer, since any peer providing CPU cycles can initiate a parallel application over the rest of peers. All peer that provide their networking capacity for information distribution provide a distribution service to any other equal peer, any of those peers can request other peers to distribute their information. All peers of my company that collaborate at a workgroup provide a collaboration service to every other equal peer.

We propose to classify P2P services as follows, trying to group together service with similar resource capabilities requirements, resource allocation requirements and similar communication schemes. This classification aims at defining different service templates for service deployment. Each category needs a service template, since their resource requirements, allocation algorithms and communication schemes are different.

There will be five different service templates for service deployment in peer-to-peer networks: collaboration

template, workflow template, computational template, distribution template, and access template. Template will simplify service specification by service creators and they will allow streamlining of peer resource agents.

- *Collaboration Services*: provide for unstructured unplanned communication among peer nodes. Peers need to have good connectivity to each other, and sometimes aggregation and routing capabilities. Peers have multimedia capabilities for collaboration. Multicasting is a must to reduce network traffic, which can be implemented at application layer. Examples of such systems are Groove Networks [13].
- *Workflow Services*: provide for goal-driven processes requiring decision support at several peers. Peers require asynchronous communication capabilities and a substrate supporting workflow actions processing.
- *Computational Services*: provide for computationally intensive applications. Peers provide computing resources and scheduling capabilities to start, execute, resume and stop processes. Communication among peers is sporadic event notifications. Examples are grid such as Entropia Internet Grid [8].
- *Distribution Services*: provides for dissemination of information. Peers provide distribution capabilities. Peers have to have storage and good network connectivity resources. Garnet Consulting provides a white paper on it [11].
- *Access Services*: provides for information aggregation. Peers provide information from local file systems, sensor, and filtering capabilities. The service provides shared access to all that information with filtering, caching, or searching functionality. Examples are Gnutella [12] and Freenet [6].

### 3.2 Peer-to-Peer Services Deployment Mechanisms

Currently service are introduced in peer-to-peer networks in an on-demand basis when user download application and manually configure local application peers, as in most file-sharing applications; or by a centralized manager that monitor peers state and activate applications making use of peer resources, this mechanism is similar to the centralized configuration scheme, examples are computational intensive Internet Grids.

Our multicast injection service deployment on a peer-to-peer network will demand resources on peer nodes, which can accept or deny service activation requests based on local allocation policies and other peers decisions.

Implementing our multicast injection deployment mechanisms on peer-to-peer networks requires providing these mechanisms on peer-to-peer network:

- Peer-to-peer multicast injection,
- Peer-to-peer group communication.

**3.2.1 Peer-to-Peer Multicast Injection.** Multicast injection operates as a resource discovery mechanism whereby deployer peers localize other peers willing to provide resources to build that service or to participate in it. Deployment managers need to discover resources that matches service specification in which to activate the service.

There exist several discovery mechanisms on peer-to-peer networks that could be adapted for multicast injection. Purist peer-to-peer networks like Freenet [5] or Gnutella [12] implement discovery by flooding queries between every peer; it implements timeouts to avoid loops, but its scalability for information discovery seems at disarray unless better algorithms are implemented [18]. Hybrid systems such as Napster or computational grid make use of a centralized directory that indexes every peer resources; discovery of peer willing to participate in a service will imply querying that directory, which assembles more to the centralized configuration system.

For a start, query flooding seems a good mechanism to implement multicast injection in peer-to-peer networks. Each peer interested in providing its resources to services of a specific template will connect to other peers that already provide that type of services. If a peer needs a new service it will inject its service specification by propagating its service specification with a flooding message modified for service injection. On receiving a service injection message, peers will check whether they meet resource constraints and will join that service intra-service group communication channel. They will publish their intentions to activate that service, finally on examining other nodes intentions they will decide whether to activate that service. If they decide not to do so, they will keep listening for some time in case other nodes fail, and substitutes are required.

**3.2.2 Peer-to-Peer Group Communication.** Peers upon checking that they can provide a service join an intra-service group communication channel. They publish in that channel their intention to activate that service and the resource they will provide. Any other peer that intends to activate that service will examine every other peer intentions and will decide whether to activate that service. Peers that do not activate the service will keep listen in that channel waiting for the opportunity to activate the service if some other peer fails or if demand grows.

Intra-service group communication among all service peers is best implemented with some kind of multicast mechanisms. We do not know of any peer-to-peer platform that implements a generic peer-to-peer multicast mechanism. However it will be very valuable, apart from being used for service deployment, it can reduce traffic among peers and provide developers a simple paradigm to implement much functionality of group applications.

Multicasting for intra-service communication requires two simple operations support by peers: group join and group send. Group join operation is implemented by flooding a join message it to every other peer; every peer that receives it saves a reference from that peer in a table indexing groups with peer members. Group send operation proceeds to lookup that table and send the message body to those peers that joined the group. This group communication mechanisms can be implemented by doing the contrary to what Heimbigner did in [14] adapting Gnutella-like query flooding functionality to publish /subscribe middleware.

Though it seems very straightforward, its efficiency can be compromised when there is a large number of groups, groups are very large, or the network is very large. But luckily this problem has been thoroughly researched for the design of IP-multicast. There exist many algorithms that provide efficient implementations of IP-multicasting communication scheme, which can be reused for peer-to-peer multicasting [20].

## 4. Experiments

### 4.1 Implementation

We have developed the Xweb framework prototype that allows for deployment of web distribution services. A web distribution service operates as a set of interconnected reverse cache servers called surrogates that provide web service to web client scattered all over the demanded service coverage area.

It provides service creators a dissemination service template. Web application creators have to fill in web service characteristics such as minimum storage and bandwidth at resource nodes, number of service points, expected total clients traffic, Internet regions where service has to be provided, maximum distance between clients and surrogates, etc.

Dynamic deployment mechanisms make deployment managers and resource agent interact for efficient and cost-effective service deployment. There are two possible deployment mechanisms: centralized configuration or multicast injection. Centralized configuration performs SNMP-style surrogate configuration, multicast injection uses IP multicast for service injection and intra-service communication. We show a simulated performance evaluation of both mechanisms below. Resource agents configure web surrogates at resource peers to start servicing a web application on service provider behalf. As well they control resource usage for appropriate service responsiveness by limiting number of services provided at each surrogate and monitoring service response time. Once a web service is deployed a management interface is returned to its creator. Currently services location is expressed as a list of Internet autonomous systems numbers.

In fact this Xweb prototype can be considered a peer-to-peer system that could provide for peer-to-peer service deployment, it would be the Xp2p. If we consider that at every node there is a resource agent that provides resources to every other node, and there is a deployment manager that allows for service deployment on other nodes, the Xweb can be defined as a peer-to-peer network that provides user deployed services. The only mayor difference from a service deployment peer-to-peer network as we have specified on section 3, is the use of IP layer multicast. However as we have discussed peer-to-peer networks can easily accommodate multicasting mechanisms for multicast injection and group communication.

We are evaluating peer-to-peer development platforms such as Endeavors' Magi [7] or the peer-to-peer Intel Working Group Initiative [17], for the implementation of such service deployment framework. We have designed the Xp2p service deployment interface, which is shown in figure 3 with the distribution template selected and a service defined and ready to be deployed. Other service templates require other resource capabilities and other service engines to be defined, which would appear on selecting those templates.

### 4.2 Performance Evaluation

We have evaluated both deployment mechanisms in a simulation on the ns-2 network simulator [16]. We simulated deployment of web distribution services, demanding resources on 5 randomly selected peers from 25 randomly scattered peers on a typical WAN network topology. Request loads range from 20 average services per second to 60 average services per second (an average service is a service with media resource demands and media service duration). On figure 2 we show which allocation algorithms were used to allocate resources in each simulation. These are very basic algorithms that try to allocate resources on the nearest peer possible or they fail.

Centralized Configuration Allocation Algorithm	Multicast Injection Allocation Algorithm
<pre> Deployer_peer { ::select_peers {     Foreach service region     {among those peers with     enough resources, allocate     on nearest} } ::deploy_timeout {     If some region not     serviced {send cancel-     request} } } </pre>	<pre> Surrogate_peer { ::receive_mcast_injection {     If enough resources &amp;&amp; near     some service regions     {multicast_service_match} } ::receive_service_match {     If sending_peer nearer than     this to some service region {stop     service in that region} } ::deploy_timeout {     If not every region serviced     {stop-local-service} } } </pre>

Figure 2 - Allocation algorithms implemented in simulation

## XP2P Service Deployment

This page allows you to deploy a new peer-to-peer service. Please fill out:

General Properties		
Service Name	<input type="text" value="may-discourse"/>	With assigned local domain (f.e.mydomain.p2p) will constitute the service URI.
Service Template	<input type="checkbox"/> Collaboration <input type="checkbox"/> Workflow <input type="checkbox"/> Computation <input type="checkbox"/> Distribution <input type="checkbox"/> Access	Collaboration template creates workgroup services, Workflow template creates goal-pursuing services, Computational templates creates CPU intensive services, Distribution templates create information disseminate services, Access template creates information aggregation services.
Service Properties		
Peers Capabilities	<input type="text" value=" &gt; 1 Gbyte Storage"/> <input type="text" value=" &gt; 1 Mbit BW"/>	Capabilities of peer nodes that will build the service.
Peers Coverage	<input type="text" value="Home"/> <input type="text" value="Office"/> <input type="text" value="Company"/> <input type="text" value="State"/> <input type="text" value="Country"/>	Administrative domain of peers to be used on application.
Number of Peers	<input type="text" value="5"/>	Number of peers forming service.
Max. Traffic	<input type="text" value="2"/>	Mbits, Maximum data traffic expected at any peer.
Deployment Properties		
Service Engine	<input type="text" value="Web surrogate cache"/> <input type="text" value="Video Streaming"/> <input type="text" value="Video on demand"/> <input type="text" value="Alert"/>	Service to be activated at each peer.
Service Manager	<input type="text" value="10.12.13.14"/>	Management station that will manage this service.
Start Time	<input type="text" value="18:00"/>	GMT, service must be deployed before this time.
Duration	<input type="text" value="30"/>	days, service must be active for this period.
Evaluation Interval	<input type="text" value="60"/>	min., interval at which peers selection has to be re-evaluated.
<input type="button" value="Submit"/>		

Figure 3 – Xp2p Service Deployment Interface

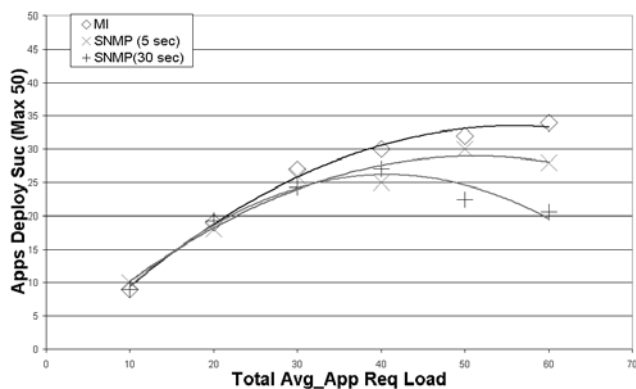


Figure 4 -Deployed services vs. req. load

On figure 4 and 5 we show which is the success rate and percentage of unused allocations of both deployment mechanisms. As expected success ratio keeps quite high for moderate deployment loads and it decreases for high deployment loads. Since we are allocating surrogates only when they are one hop away from the region requested, the rate of successfully deployed applications keeps quite below the maximum number of applications (50) that can be deployed on those surrogates. Centralized configuration has worst behavior at high load than multicast injection because it makes use of stale information. Deployers consider many surrogates as fully allocated and reject deployment requests, however multicast injection tries to allocate resources on every case, increasing its success rate. As it has been discussed this mechanism can lead to a high number of allocations that are cancelled without being used by applications. Multicast injection avoids that contention by allocating resources as soon as they are requested, however some resources can be allocated and not used leading to thrashing and deadlock situations; on figure 5 we see that this percentage is very low even at high loads. This is a consequence of service times being much longer than allocations intervals, therefore undesirable thrashing and deadlock shall not occur during normal operation of deployment systems.

These figures tell us that multicast injection mechanisms adapts to resource availability variations, without causing much resource under-utilization.

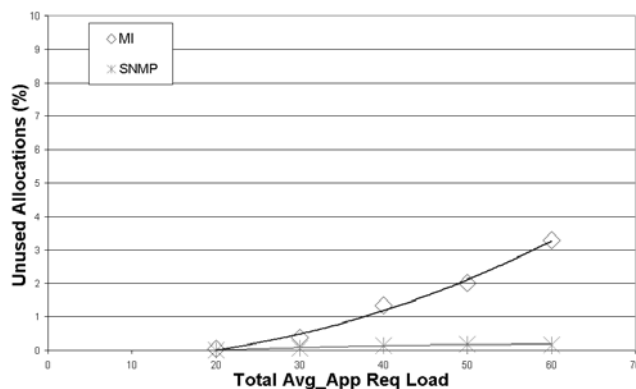


Figure 5 -Unused allocation vs. req. load

## 5. Summary

In this paper we have exposed our framework for Internet service deployment and have characterized how to extend peer-to-peer networks to support dynamic service deployment. We present which new deployment functionality is required to deploy new services of peer-to-peer networks, and how to adapt peer-to-peer communications mechanisms for multicast injection service deployment.

Our Xweb service deployment framework enables the introduction of dissemination or access type Internet services by using simple to use service templates and multicast injection mechanisms. As simulation shows, our proposal multicast injection behaves very well at high load scenarios, being successful in deploying more services than centralized configuration mechanisms. Our multicast injection for service deployment is a good solution that provides good adaptability to resource variations on Internet environment.

Also we have provided a set of templates for service deployment on peer-to-peer networks; distribution, access, collaboration, computational and workflow templates allow for a simple specification of peer-to-peer services for deployment. We propose that is very easy to extend peer-to-peer network to enable them for service by simply adding a modified query flooding mechanism for service injection, and to introduce some form of group communication mechanism similar to IP multicast.

This service deployment framework will provide service creators the possibility to make their service available to larger users populations, and users will have wider and more elaborate sets of services to choose.

## 6. References

- [1] Ardaiz O., Touch J., "Web Service Deployment Using the X-bone", Proceedings of Spanish Symposium on Distributed Systems SEID 2000, Sept 2000.
- [2] Campbell A.T., De Meer H.G., Kounavis M.E., Miki K., Vicente J.B., Vilella D., " A Survey of Programmable Networks", ACM SIGCOMM Computer Communication Review, April 1999.
- [3] Chandra P. et. Al., "Darwin: Customizable Resource Management for Value-Added Network Services", 6 th IEEE International Conference on Network Protocols (ICNP'98), 1998.
- [4] Cisco Inc., "Cisco Provisioning Center White Paper" July 2000.  
[http://www.cisco.com/warp/public/cc/pd/nemnsw/pvcr/tech/provs\\_wp.htm](http://www.cisco.com/warp/public/cc/pd/nemnsw/pvcr/tech/provs_wp.htm)
- [5] Clarke I., Sandberg O., Wiley B., Hong T., "Freenet: A distributed Anonymous Information Storage and Retrieval System", Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley 2000.
- [6] Clarke I., "The Freenet Network Project", <http://freenet.sourceforge.net/>, Jul. 2000.
- [7] Endeavors Technology Inc., "Peer-to-Peer Architectures and the Magi™ Open-Source Infrastructure", [http://www.peer-to-peerwg.org/downloads/collateral/proposals/ETI\\_P2P.pdf](http://www.peer-to-peerwg.org/downloads/collateral/proposals/ETI_P2P.pdf), December 2000.
- [8] Entropia Inc. "Entropia Internet Grid", April 2001.
- [9] Fayad M., Schmidh D. " Object-Oriented Application Frameworks" Guest Editorial, Communication of ACM Vol 40 N 10 Oct 1997.
- [10] Foster I., Kesselman C., Lee, C. Lindel B.I, Nahrstedt K., Roy A. "A Distributed Resource Management Architecture that Supports Advance Reservation and Co-Allocations", In International Workshop on Quality of Service, 1999.
- [11] Garnet Consulting, "The Emergence of Distributed Content Management and Peer-to-Peer Content Networks", <http://marketplacena.gartner.com/010022501oth-NextPage.PDF>, January 2001.
- [12] Gnutella,"Gnutella", <http://gnutella.wego.com/>, June 2000.
- [13] Groove Inc., "Groove Networks", <http://www.groovenetworks.com/>, April 2001.
- [14] Heimbigner D., "Adapting Publish/Subscribe Middleware to Achieve Gnutella-like Functionality", SAC'2001, Las Vegas NV 2001.
- [15] Isaacs R., Leslie I., "Support for Resource-Assured and Dynamic Virtual Private Networks" IEEE Jrl. on Sel. Areas in Communication Vol. n. 2001.
- [16] McCanne S., Floyd S. " CCB/LBNL/VINT Network Simulator -ns (v2)", <http://www.mash.cs.berkeley.edu/ns/>.
- [17] Peer-to-Peer Working Group, <http://www.peer-to-peerwg.org/>, Oct 2000.
- [18] Sripanidkulchai K., "The popularity of Gnutella queries and its implications on scalability", <http://www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html>, February 2001.
- [19] Tennenhouse D.L., Wetherall D.J., "Towards an active networks architecture", Proceedings Multimedia Computing and Networking, San Jose CA, 1996.
- [20] Touch J., Hotz S.,"X-bone: a System for Automatic Network Overlay Deployment" Third Global Internet Mini Conference in conjunction with Globecom'98, Nov. 1998. <http://www.isi.edu/x-bone>.
- [21] Wittmann R., Zitterbart M., "Multicast Communication: Protocols, Programming, and Applications" Morgan Kaufmann Publishers; ISBN: 1558606459. May 15, 2000.