Multicast Injection for Application Network Deployment

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 task. Application networks, applications being serviced through multiple interconnected service nodes disseminated across the Internet for better performance, fault tolerance and availability, as well are costly to set up. In order to provide a network-enabled application service, a number of surrogate servers have to be provisioned.

In this paper we propose a mechanisms to dynamically deploy an application network: multicast injection. Currently employed dynamic deployment mechanisms, SNMP per-node configuration, is a centralized model that can not scale or be as fault tolerant as more distributed mechanisms such as multicast injection.

We perform simulations to compare its efficiency in terms of deployment request success ratio, unused allocation percentage and traffic vs. deployment resource allocation requests. We show that multicast injection has a higher success ratio with lower bandwidth consumption at deployers' locations.

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. [4], "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. [18]; 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.

An application network is a set of interconnected application entities providing an application layer service. Application networks have been shown to improve service performance, availability and fault tolerance. Content Distribution Networks as Akamai [1] are examples, but peer-to-peer systems as Freenet [8], fall into this definition too. Each service agent provides an equal service and communicates with others to maintain a consistent service or to propagate client data to other service agents. We will use through this article the term surrogate for these service agents as it is widely used, though it implies the existence of an origin master service entity, which needs not always be the case.

Application network services are provisioned on the Internet either by users replicating servers without any coordination or by centralized management stations configuring a number of proprietary servers. In the first case service providers are not able to specify which service level they desired, whereas the management stations model does not scalable and requires resources provider to give a lot of control on their resources to managers. Our solution, multicast injection, aims at ease service deployment in an scalable and cost-effective way; thereby allowing networked application creators to provide its services to more users, and benefiting users with a wider service offer.

1.1. Application Network Deployment

Application network deployment is defined as the process of setting up or creating an application network service. All steps since a new application network service is specified till it is provided to users. Application networks require a number of surrogates to be provisioned from a pool of surrogates, as web sites are provisioned on web hosting centres or virtual private networks are provisioned on virtual network provider's infrastructure.

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.

Our first task is to define which functionality is demanded from this framework. Deploying a service requires: obtaining service specifications, mapping specifications to resources, discovering resources,

0-7695-1321-2/01 $10.00 © 2001 IEEE

gathering resources, configuring resources, activating service, and providing a management interface. Providing this functionality requires 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 properties 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 properties from this framework: efficiency and cost-effectiveness, for highest resource revenue, scurity, to avoid resource misuse.

1.2. Problem Scope

A framework for provisioning an application network poses several technical challenges, several of which can be solved by already proven technologies; we will first differentiate which issues are particular to this problem.

Resource agents, in a framework for application network deployment have to permit for dynamic service activation and service level enforcement. Service activation means launching remotely a service on a surrogate. Its main issue is security, proposals such as active caches [5] or the extensible proxy service framework [19] are defining requirements nodes environment must fulfill for secure remote service activation. Service level enforcement is a harder issue, how to provide and maintain quality of service to applications is subject to intensive research, research such as [15] is setting the path for QoS web services. We are not discussing farther these issues.

Besides end-to-end security, safety and manageability have to be considered. There exist several researches already discussing the security and management functionality required to make a reality programmable network nodes [4], so we are not discussing it further.

Provisioning is formulated as a resource allocation problem, a number of scarce resources have to be allocated to a number of services requesting resources. But resource allocation is only one of the tasks of the provisioning process; provisioning consist in gathering a number of resources required to build a service, configuring them and activating the service. There exist a number of studies proposing algorithms for resource allocation on a set of distributed resources which optimize distributed web services, such as those undertaken by Aron et al [3], Korupolu et. al. [11], or Kelly et al. [10]. Research undertaken in this paper does not propose novel resource allocation algorithms. It makes use of simple algorithms to provide a good enough resource allocation with available data at deployer entities. Heuristics for optimization are subject of future research.

We have identified a framework component that provides functionality specific for this problem requiring extensive research to meet every party requirement: the mechanisms that provide for an efficient and cost-effective deployment. This research is about the mechanisms to accomplish application network provisioning, about how to dynamically deploy an application networks. Deployment mechanisms make service deployers discover and gather resources, plus activate services on appropriate nodes. It is not a trivial issue; scale and dynamics of the Internet, wide range of application specifications, and quantity and characteristics of surrogates and deployers are a complex set of dimensions to be met by those mechanisms in a cost-effective and efficient way. Our goal is to propose mechanisms for dynamic deployment on the Internet that are able to meet the scalability, adaptability, multiplicity and heterogeneity requirements of the Internet environment.

In section 2 we explain which is the design space we set for evaluation and in section 3 we describe which solutions exists and which we propose. Finally in section 4 we describe the performance evaluation we have carried out and in section 5 we present and discuss the results of such simulation.

2. Design Space

Our goal shall be to design mechanisms for dynamic application network deployment that scale efficiently and cost-effectively in those value ranges that can be encounter on the real world. There are three kinds of parameters that affect performance of those mechanisms: in first place specifications of application networks to be provisioned, in second place characteristics and multiplicity of deployer entities and surrogates; thirdly, characteristics of the Internet will condition those dynamic deployment mechanisms. The ranges of values for these design space parameters are:

- **Application networks specifications parameters**: per node storage, which varies from Kbytes

to Gbytes; total service traffic, from Kbits to Gbits, network regions where service is to be provided: from some regions to every region; maximum distance between service nodes and client regions: from nearer surrogates to surrogates at any distance; maximum application network diameter: from surrogates at most 1 hop away from each other (a mesh) to surrogates n hops away from further surrogate (n level hierarchy); number of nodes providing service: from one to tens of thousands; service duration: from seconds to years; finally service demand variations will condition how often allocations are-evaluated.

- **Surrogate capabilities parameters**: storage and network bandwidth, which are the scare resources; service regions, surrogates can service different regions with different service levels depending on network path characteristics from surrogate to region, depending on their network location, they can service from a few regions to hundreds of regions (however there is no good characterization of what is a region, it has to be subject of research); finally the number of surrogates dictates the total amount of resource and traffic that can be provided.
- **Deployer parameters**: number of deployer entities is a limiting factor since deployers condition each other efficiency by contending for shared surrogates. Although the number of surrogates and deployer entities is determined by service offer and demand, we can expect at least as many surrogates and as many deployer entities as number of service regions. Any surrogate located nearer than any other to a service region will be chosen for service that region. Deployers will be located nearer application providers, which again will be evenly distributed over every service region.
- **Internet parameters:** are scale and dynamics; scale affects specifications of services, and the number of deployer entities and surrogates as previously commented. It also affects latency and traffic characteristics of the deployment mechanisms; dynamics of the Internet affect the pace at which services must be deployed, causing services not to be deployed on optimal locations by the time deployment terminated due to changes on the availability of resources in the meanwhile. It also affects service levels provided to clients due to variations in best surrogate election.

3. Dynamic Deployment Mechanisms

Deployment mechanisms make service deployers discover and gather resources, plus activate services on appropriate nodes. A first solution represents the method currently used for provisioning services that require resource allocation at multiple nodes, such as virtual private networks or content distribution networks. It is the SNMP management station based method, in which a centralized entity monitors, chooses and configures resource agents [7]. As every centralized system, it is not the best solution in terms of scalability or fault tolerance.

The method we propose is multicast injection. Our approach aims at taking advantage of the announce/listen communication mechanisms of multicast to provide robustness and adaptability to the system [17], and the traffic aggregation capability that provides better scalability. Beside this mechanisms lends itself to a distributed resource allocation by autonomous resource agents, thereby eliminating the bottleneck and single point of failure that centralized resource allocation involves.

3.1 Centralized Configuration Mechanism

This is the method currently used [3] [7]. Per surrogate configuration deployment involves the following steps:

1. Deployer continuously monitors surrogates resources, / agents publish their resources,
2. Service provider request service deployment,
3. Deployer calculates resource allocation,
4. Per surrogate configuration,
5. (At timeout), every surrogate response is ok OR deployer 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.

3.2 Multicast Injection Mechanism

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.

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 information that other agents publish. Deployer entities inject service specifications (including a reference to application binaries and data) into the

system at a global multicast channel. Deployers can only expect that enough and appropriate resource agents accept the service activation request, else they can inject a cancel service request, appropriate agents are those that have enough resources and can provide client with good enough service. Surrogate entities implement an agent that listens to deployer requests, calculates if they can provide the service and publishes on an application specific multicast channel where the service has been allocated. After a while if they do not listen that enough resource agents have allocated resources for that service they cancel the application activation.

3.3 Comparison

	Advantages	Disadvantages
Centralized Configuration (SNMP)	-More control	-Deployer computation -Deployer traffic -Stale information -Unused allocations
Multicast Injection	-Faster activation -More adaptable -Simple deployers -Robust system -Loose relations	-More unused allocations -Network traffic

Figure 1 - Mechanisms Comparison

Centralized configuration presumes more control by the deployment entity over surrogates, since surrogates are passive entities controlled by deployers. In contrast, multicast injection presumes a looser relation between deployers and surrogates. Surrogates 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 surrogates can be available for multicast injection dynamic deployment.

Centralized configuration requires high bandwidth in a centralized location; as well it has to implement a centralized resource allocation algorithm, which can require high computational resources on Internet scales. Multicast injection makes use of a distributed allocation algorithm, which makes it more scalable and fault tolerant.

When many deployers content for surrogates, with the SNMP method 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; contention probability will increase as the number of deployers and request load increases. With multicast injection it shall not happen, since resources are allocated as soon as they are discovered, however allocated resources might have to be released is not enough resources are found to set up the service. This unused allocations block others requests which do not make use of more resources allocated to them reaching to deadlock situations. This phenomena, called thrashing, has been observed in other scenarios [14]. We shall evaluate how serious this phenomenon is.

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. Multicast injection puts less resource requirements on deployer entities, while permitting surrogates to have more autonomy, thereby allowing application creators to easy deploy application network and resource providers to easy provide their resources for third party service deployment.

4. Simulated Evaluation

4.1. Metrics

Our experiments aim at comparing efficiency of both deployment mechanisms. Our first metric for comparison is success ratio, defined as the number of "average application" deployed to number of request for application deployment. An "average application" represents an application whose surrogate resource requirements are the media. Other metrics for comparison are unused allocated resources and traffic created.

- **Success ratio** depends on service specifications, stricter specifications means an application has higher probability of failing to be deployed; the more deployers more contention for surrogates and higher probability of failure, and surrogates capacity and network characteristics.
- **Unused allocations** occur whenever a resource is allocated but does not start providing a service since its deployment is cancelled due to not finding enough resources. It can be a problem if they represent a large percentage of resource occupancy time.
- **Traffic** created is proportional to number of requests, number of surrogate nodes, requests size, interactions per request, monitoring request rate and reallocation interval. We want to measure traffic at every deplorer's location, and total traffic created on the network.

4.2. Simulation Topology, Surrogates and Deployer Parameters

We have simulated both deployment mechanism using the ns-2 network simulator [13], on it we have configured a wide area network with 5 top level regions, each region has an exchange point connected to 2 international exchange points by 1 Mbit/50ms delay links, each of the 5 international exchange points is connected to 2 other international exchange points by 1Mbit/100ms delay links. Surrogates and deployer are evenly distributed in all regions connected to local exchange points by 4 Mbits/20ms delay links. There are 25 service regions and 25 surrogates, one at each service regions. Surrogates parameters are: 1000 Mbytes storage, 10 Mbits service traffic capacity, 1 region at distance 1, 2 regions at distance 2, 4 regions at distance 4, and so on. There are 5 deployers one at each wide area top level regions, each deployer has deployment requests with a Poisson arrival rate with interval media ranging from 1 request every 50 seconds to 1 request every 8 seconds. Figure 2 shows this scenario.

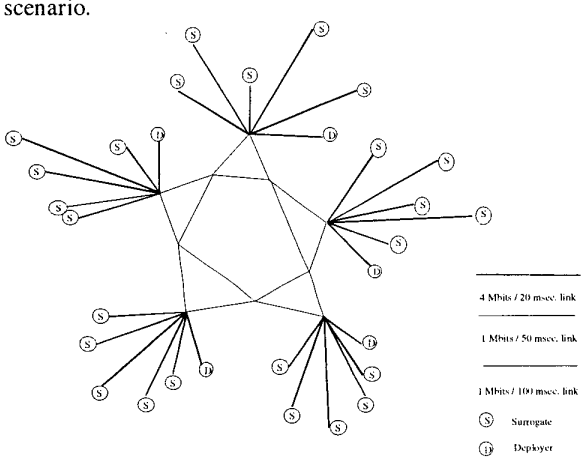

Figure 2 - Topology configuration used for simulation

We are not evaluating the effect of different allocation algorithms, though deployment efficiency depends on the option chosen. Allocation algorithms that find smaller surrogates sets for an application specification permit more applications to be deployed. Allocation algorithms that coordinate allocation among different applications can optimise overall service level increasing deployment efficiency. Also depending on how often allocation calculation is done, the reallocation interval, application networks will adapt making use of enough resources. In figure 3 we show which allocation algorithm is being implemented on these simulations for both type of mechanisms.

Centralized Configuration Allocation Algorithm	**Multicast Injection Allocation Algorithm**
Deployer { *:: select_nodes {* *Foreach service region {among those with enough resources, allocate on nearer surrogate}* *}* *::deploy_timeout {* *If some region not serviced {send cancel-request}* *}* *}*	*Resource Agent {* *::receive_mcast_injection {* *If enough resources && near some service regions {multicast_service_match}* *}* *::receive_service_match {* *If peer_agent nearer than this to some service region {stop service in that region}* *}* *::deploy_timeout {* *If not every region serviced {stop-local-service}* *}* *}*

Figure 3 - Allocation Algorithms in Simulation

4.3 Deployed Application Networks Parameters

Deployer entities accept deployment requests with these application network specification parameters:

- Per node storage, the require storage at each surrogate for application data and code. It can be the whole data set size or a fraction of it, downloading on demand least used data.
- Total traffic, represents the expected total network traffic caused by client to server interactions. It can be expressed as traffic-region pairs. Deployers in this simulation will assume traffic is equally distributed from all regions.
- Network regions, where service is to be provided, currently we are considering it is a list of AS numbers.
- Maximum distance, between service nodes and client regions, represents required service level. Selecting larger distances means worse services are tolerated. Currently, distance is the number of Internet hops between client regions and surrogates.
- Number of nodes providing service, it determines fault tolerance and availability of service, however it represents a higher cost.
- Service duration, it will express a determination to make use of surrogates for a specified period, which can be changed later.

On the simulations every application network to be deployed is specified to request 100 Mbytes storage per surrogate, 5 Mbits traffic capacity, service is required at 5 random regions (all with equal probability), service has to be provided from 5 surrogates, maximum distance between surrogate and client is 1 hop, therefore each

service can only be serviced by those five surrogates at each requested region, and service life time has a negative exponential distribution with an average of 100 seconds. It is a very low value for normal Internet service life times, this value intent to represent the extreme case of very popular flash-crowd services.

5. Results

We run simulations of both SNMP per node configuration and multicast injection with deployment request rates of each deployer from 1 request every 50 seconds to 1 request every 8 seconds. We set deploy cancel timeouts at 3 seconds for both types of mechanisms. We run simulations of centralized configuration deployment with monitoring intervals of 5 and 30 seconds.

Simulations were run for 2000 seconds time, leaving 200 seconds for warm-up before start collecting data.

5.1. Impact of Deployment Resource Demand

We have obtained the number of successfully deployed applications for different deployment loads. Deployment load represents the number of average application deployment request per second multiplied by the service time of an average application. With five deployer each with deployment requests with a Poisson arrival rate with interval media ranging from 1 request every 50 seconds to 1 request every 8 seconds, total deployment load range from 10 average applications to 60 average applications. (1 request every 50 seconds * 100 seconds duration each average application service * 5 deployers = 10 average applications load). With 25 surrogates each with 10 Mbits of bandwidth resources, a maximum of 50 average applications can be deployed.

In figure 4 as expected success ratio keeps quite high for moderate deployment loads and 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 deployed on those surrogates.

SNMP centralized configuration has worst behaviour at high load than multicast injection because it makes use of stale information. Deployers consider many surrogates as fully allocated and reject deployment requests. If we increase the monitoring rate from 30 seconds to 5 seconds the number of successfully deployed applications increases, however at the cost of higher monitoring traffic. With multicast injection surrogates allocate resources autonomously, increasing success rate. As it has been discussed this mechanism can lead to a high number of allocations that are cancelled without being used because not enough surrogates are found.

As well stale resource information in SNMP 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.

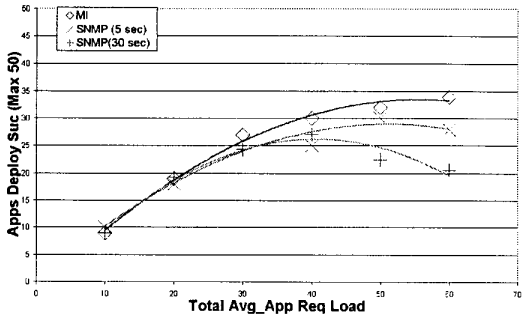

Figure 4 –Deploy Success vs. Resource Demand

In figure 5 unused allocations, as percentage of total possible allocations, has been plot against deployment load. 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. It has very low values for SNMP centralized configuration mechanism, since deployment cancellations only take place when two deployers content for same surrogates at the same time, which has low probability on the simulated topology that contains only five surrogates. Multicast injection shows higher unused allocations values, however they represent only a very small percentage of total resources. It seems to increase exponentially as deployment request load increases, however at highest simulated request loads, 60 average applications deployment load which is above the total capacity, it only represents a 3.3 % of total resources. This is so because service lifetimes (media 100 secs.) are much higher that cancellation timeouts (3 secs.), therefore for a given request load most of the time surrogates are servicing applications, and cancellations consume very low resources levels.

In figures 6 and 7 we present traffic created by each type of mechanisms on every deployers location and the total traffic created on the network by all deployers as the sum over all links. Per node configuration generates quite a lot of traffic at deployer locations, whereas multicast injection requires much smaller connectivity at deployers.

Multicast injection produces a large amount of network traffic on the whole network that increases with demand.

Centralized configuration generates traffic to monitor status of each surrogate. It does not increase as request load goes up. However it increases at the monitoring interval is increased, which is required to obtain fresh information from surrogates. Multicast injection generates most of its traffic by multicasting service specifications requests. It grows with increased request loads, but it keeps very low values at deployer locations. Total traffic created by multicast injection grows linearly as number of deployment request load, but it does not represent a large percentage of total network capacity.

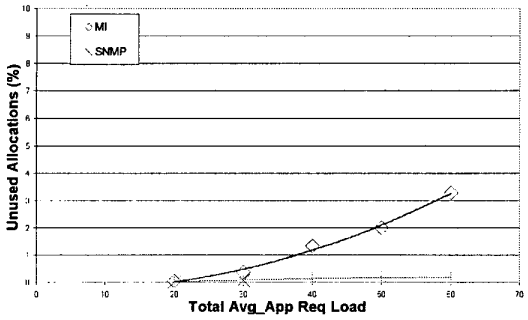

Figure 5 - Unused Allocations vs. Resource Demand

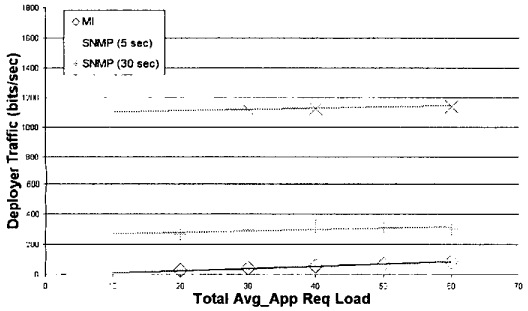

Figure 6 - Deployer Traffic

5.2. Impact of Number of Deployers

We have run the same simulation varying the number of deployer entities from 5, 10 and 20. To keep total deployment load constant we have decreased each deployer request rate proportionally by 1, 2 and 4.

In figure 8 we see how many application networks are successfully deployed as the number of deployment entities changes from 5 to 20 at different deployment loads. With a moderate request load of 30 applications (all deployers request 60% of resources) every deployment mechanisms provides a constant success ratio as the number of deployment entities increases. However when we consider a total request load equivalent to 100% of resource available, SNMP centralized configuration deployment mechanisms decreases its success ratio as the number of deployment entities increases, it is worst for low monitoring rates. It occurs due to increase resource contention with increased number of deployment entities. In figure 9 SNMP centralized configuration deployment increases total network traffic proportionally to the number of deployment entities. This increase is due to the monitoring traffic.

Multicast injection scales better with number of deployment entities, since the total network traffic created is not proportional to the number of deployment entities.

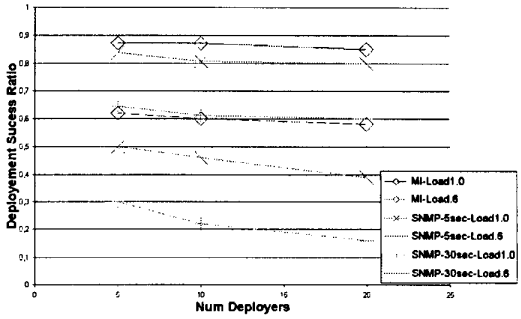

Figure 8 - Success Ratio vs. # Deployers

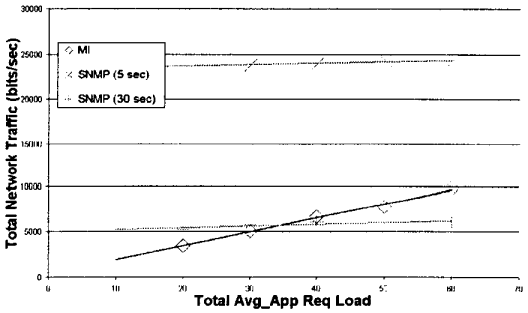

Figure 7 - Total Network Traffic

6. Related work

There is little research on how provisioning is accomplished. Mechanisms for dynamic deployment are being studied at the Xbone project [20] at ISI, at the Darwin project [6] at CMU, and at the Globus project [9], a joint effort of various universities and research centres. The Xbone project is developing a system for overlay dynamic deployment. It allows virtual networks to be set up on the Internet to behave as virtual private networks or to isolate experimental services from the Internet. The Xbone is developing a protocol for dynamic deployment using multicast expanded searches and unicast secure configuration. Darwin project has developed a resource allocation protocol, Beagle, to

dynamically deploy virtual meshes, collection of networking resources. One of its main uses is multiparty videoconferencing with quality of service set up. Globus 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.

Commercial CDNs must have implemented such deployment mechanisms; however there is no published document on their operation, though we presume they will have implemented the SNMP centralized approach.

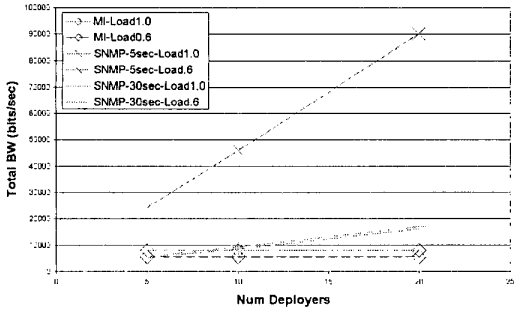

Figure 9 - Total Traffic vs. # Deployers

The Content Networks Internetworking IETF working group is currently specifying requirements for a distribution signalling protocol [2] which will allow a web content to be distributed and serviced from a number of surrogates on different CDN. However it will implement, as all internetworking protocols, only a minimum set of functionality for basic interoperability.

In either kind of research areas, there has not been any published performance evaluation of deployment mechanisms as the one undertaken in this paper.

7. Summary & Future Work

Our simulations show that multicast injection is more scalable that SNMP per node centralized configuration, specifically it overcomes these centralized configuration deployment drawbacks: 1) bandwidth needed at SNMP based deployers' locations to obtain the state of surrogates up-to-date is orders of magnitude larger than that needed for deployment with multicast injection method; 2) computation resources needed at deployer entities are much larger for SNMP based deployment that for multicast injection due to centralized resource allocation algorithm to be implemented; 3) as well SNMP per node configuration is not scalable with the number of deployer entities, the more deployment entities, the higher contention and traffic required to obtain fresh resource status information.

Multicast injection achieves at least equal success ratio for any deployment request load while generating much less traffic at deployers' location. It even achieves higher success ratio at deployment request load near and above the maximum capacity of resource agents. Additionally we have observed that multicast injection incurs in a negligible penalty of unused allocated resources caused by its distributed allocation. Percentage of unused allocations is low at even very high deployment load, therefore thrashing will not happen or it will be prevented simply by input load limiting mechanisms.

Further work we have planned is evaluating the effect of different allocations constraints, i.e. allocation permitted on surrogates that are farther from the service region, which will increase the number of surrogates that can be allocated, but at the cost of service level decreases. We also expect to study heuristics to improve and/or accelerate allocations of different specifications, i.e. when number of surrogates required differs from number of service regions. Finally a good characterization of service regions is an important research to realize application network provisioning with proposed dynamic deployment mechanisms, work such as that of Krishnamurthy [12] on clustering of web clients is a good starting point in that direction.

8. References

[1] Akamai Inc. "FreeFlow", Dic 1999, http://www.akamai.com

[2] Amini L., Thomas S., Spatscheck O., "Distribution Peering Requirements for Content Distribution Internetworking", IETF Inet Draft, draft-amini-cdi-distribution-reqs-00.txt, Feb. 2001.

[3] Aron M., Druschel P., Zwaenepoel W. "Cluster reserves: A mechanism for resource management in cluster-based network servers", Proc. ACM SIGMETRICS 2000.

[4] Campbell A.T., De Meer H.G., Kounavis M.E., Miki K., Vicente J.B., Villela D., " A Survey of Programmable Networks", ACM SIGCOMM Comp. Comm. Rew, April 1999.

[5] Cao P.,. Zhang J., Beach. K., "Active Cache: Caching Dynamic Contents on the web", Proceedings of IFIP International Conference on Distributed Systems Platforms (Middleware'98), September 1998.

[6] Chandra P. et. Al., "Darwin: Customizable Resource Management for Value-Added Network Services", 6 th IEEE Intl. Conference on Network Protocols (ICNP'98), 1998.

[7] Cisco, "Cisco Provisioning Center White Paper" July 2000. http://www.cisco.com/warp/public/cc/pd/nemnsw/pvcr/tech/provs_wp.htm

[8] Clark I. "A distributed decentralised information storage and retrieval system". 1999. http://freenet.sourceforge.net/Freenet.ps

[9] Foster I,. Kesselman C., Lee C., Lindell B., Nahrstedt K., Roy A., "A Distributed Resource Management Architecture that

Supports Advance Reservation and Co-Allocations", In International Workshop on Quality of Service, 1999.

[10] Kelly T., Reeves D., "Optimal Web cache sizing: scalable methods for exact solutions", Proceeding 5 th Web Caching and Content Distribution Workshop, Lisbon, May 2000.

[11] Korupolu M.R., Dahlin M., "Coordinated Placement and Replacement for Large-Scale Distributed Caches", Proceeding of IEEE Workshop on Internet Applications, July 1999.

[12] Krishnamurthy B., Wang J., "On Network -Aware Clustering of Web Clients", ACM SIGCOMM, August 2000.

[13] McCanne S., Floyd S., "CCB/LBNL/VINT Network Simulator nsv2", http://www.mash.cs.berkeley.edu/ns/ Aug 1998.

[14] Mitzel D., Estrin D., Shenker D., Zhang L., "A Study of Reservation Dynamics in Integrated Service Packet Networks", Proc. IEEE Infocom, April 1996.

[15] Pandey R., Barnes J.F., Olsson R., "Supporting Quality Of Service in HTTP Servers", Proceeding of the 17 th SIGACT-SIGOPS Symp. on Prpc of Distributed Computing, June 1998.

[16] Rabinovich M., Aggarwal A., "RaDaR: A Scalable Architecture for a Global Web Hosting Service", WWW8 Conference, Toronto May 1999.

[17] Raman S. and. McCanne,S., "A model, analysis, and protocol framework for soft state-based communication," in ACM Sigcomm, 1999, September 1999.

[18] Tennenhouse D.L., Wetherall D.J., "Towards an active networks architecture", Proceedings Multimedia Computing and Networking, San Jose CA, 1996.

[19] Tomlinson G., et al., "Extensible Proxy Service Framework" Inet-Draft draft-tomlinson epsfw- 00.txt, July 2000.

[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.