Distributed Service Provisioning Using Stateful Anycast Communications

Tim Stevens, Joachim Vermeir, Marc De Leenheer, Chris Develder, Filip De Turck, Bart Dhoedt, Piet Demeester
Department of Information Technology, Ghent University - IMEC - IBBT
Gaston Crommenlaan 8 bus 201, 9050 Gent, Belgium
Tel.: +32 9 331 49 39, Fax: +32 9 331 48 99
Email: tim.stevens@intec.ugent.be

***Abstract*—Notwithstanding IP anycast's introduction in Internet standards dates back to 1993 and its more recent adoption in IPv6 standards, its use in production environments is limited to date. This is mainly because native IP anycast lacks routing scalability and does not support session-based communications, thereby limiting its applicability to single request-response services such as DNS. For this reason, we propose a transparent anycast overlay architecture that retains the strengths of native anycast and neutralizes above-mentioned limitations. The resulting proxy infrastructure unleashes the power of anycast by opening up new opportunities for transparent distributed service provisioning.**

Taking into account user demands, available resources, network overhead and anycast infrastructure costs, we provide near-optimal heuristics for the placement of proxy nodes and dimensioning the infrastructure in large networks. We show that even modest overlay infrastructures, consisting of a small number of proxy routers, provide an effective stateful anycast solution where the detour via the proxy routers is negligible in terms of extra network load. Furthermore, simulation results illustrate that server state aggregation in the proxy nodes lessens control plane overhead, which contributes significantly to service robustness.

I. Introduction

IP anycast enables communication between a source host and one member of a group of target hosts, usually the one nearest to the source [1]. As such, anycast is considered as a powerful tool for realizing transparent, scalable and reliable communications with connectionless distributed network services. The use of replicated DNS root servers listening to a common—anycast—IP address is an example application where anycast has been proven useful [2].

At present, there are limitations that prevent widespread adoption of IP anycast in general, and its adoption for network service provisioning more specifically. First, session-oriented services (including all applications implemented on top of TCP) cannot take advantage of this addressing mode, because subsequent packets from the same source host (and session) may be routed towards a different target host. In a sense, application layer anycast [3] alleviates this issue, albeit at the expense of losing IP anycast transparency. Another anycast limitation is its poor global routing scalability due to the fact that routes to anycast groups cannot be aggregated: widespread adoption of end-to-end native IP anycast would undoubtedly lead to huge and unmanageable routing tables. Possible solutions for this issue have been proposed by D. Katabi et al. [4] and H. Ballani et al. [5].

In this paper, we present a proxy-based architecture that enables IP anycast for session-oriented network services. Using this approach, advanced distributed network services can be scaled to a large number of consumers, and this in a transparent way from an end-user perspective. In addition to network state and metrics, the proxy infrastructure uses server state information to forward service requests to the most suitable location, which is not possible using only IP anycast. Furthermore, anycast group state changes (e.g., a new anycast group, a failing server) are completely hidden from the routing substrate, thereby maintaining IP routing scalability.

In a number of ways, the proposed anycast proxy architecture resembles PIAS (Proxy IP Anycast Service) [5] and most PIAS features and benefits remain valid for our architecture. Whereas Ballani et al. focus on global routing scalability and motivate part of the PIAS design by relying on BGP route stability, we tailor our anycast proxy architecture to the needs of a service provisioning platform, with explicit support for session-based communications.

After describing the anycast proxy architecture, we propose a near-optimal heuristic approach to tackle anycast infrastructure dimensioning and proxy placement in large networks. Based on several parameters, including anycast proxy infrastructure costs, network operational costs and infrastructure component capacities, we transform the problem into a—well known—Fixed Charge Network Flow Problem (FCNFP) [6], which is solved by means of a Dynamic Slope Scaling Procedure (DSSP) proposed by Kim and Pardalos [7]. Using this solution technique, we can show that a modest anycast overlay provides an effective solution, even in large networks. Furthermore, we investigate server state aggregation in the anycast infrastructure using discrete event simulation. Results illustrate that aggregation has a significant impact on service robustness and drastically reduces control plane overhead.

The remainder of this paper is structured as follows. Section II details the anycast proxy architecture. We expose the optimization problem and heuristic approaches for infrastructure dimensioning and proxy unit placing in Section III. In Section IV, service robustness resulting from server state aggregation is discussed and control plane overhead observations are presented. Section V summarizes the main results of this paper.

II. Anycast architecture

A. IP anycast limitations

Because IP anycast forwards packets to the nearest member of an anycast group, it could prove useful as a transparent *service discovery* primitive. For single request-response services such as DNS, it can even support the entire service and increase service scalability by means of implicit coarse-grained load balancing between the anycast group members.

Despite these promising features, the use of IP anycast is not widely adopted and production use is essentially limited to DNS root server replication [2]. According to Ballani et al. [5], the main reason for this is the *lack of IP routing scalability* inherent to native anycast. First, IP anycast routes cannot be aggregated and widespread adoption would lead to an explosive growth of IP routing tables. Secondly, anycast group dynamics (i.e., joining and leaving members) necessitate frequent changes to a relatively slow converging IP routing configuration, eventually leading to network instability.

With the intention to use IP anycast as a cornerstone of a *transparent distributed service provisioning* platform, two additional limitations arise:

(i) IP anycast does not support session-based communications;
(ii) IP routing metrics are static and it does not support multiple constraint routing.

Taking into account both the strengths and weaknesses of IP anycast, we propose an anycast overlay architecture based on PIAS [5] to realize a transparent and scalable service provisioning platform.

B. Architecture overview

The design objectives for a regional anycast proxy infrastructure supporting session-based network services differ from a global anycast overlay infrastructure such as PIAS [5] or OASIS [8]. In addition to the PIAS objectives, we wish explicit session support in the proxy routers and a dynamic, session-based bonding between client proxies and server proxies (and hence, between clients and servers). Contrary to OASIS, we wish that anycast network services are completely transparent from a client perspective, on each layer. Contrary to both PIAS and OASIS, we target a regional anycast solution (e.g., deployed in access and aggregation networks) and not a global one. On a smaller scale, anycast infrastructure dynamics are easier to deal with and maintaining session state in the overlay is feasible.

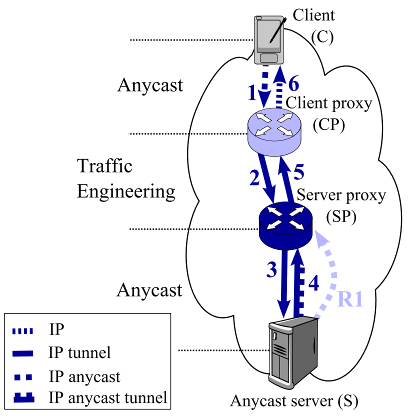
The overlay infrastructure consists of a combination of two types of nodes: *client proxies* (CP) and *server proxies* (SP). Both client proxies and server proxies are special routers advertising their proximity to the anycast IP range into the routing substrate. By doing this, the proxy routers force IP packets with an anycast destination address to pass through the overlay. When a client initiates a new session to an anycast destination, the closest client proxy registers the new session and selects an appropriate server proxy. The server proxy receiving the new session then selects the most suitable server to handle the request. Using control plane messages, servers update their state information to the server proxies, whereafter server proxies distribute aggregated state information from all neighboring servers to the other proxy routers.

Fig. 1 depicts the steps involved in setting up a session between a client and a target anycast server through the proxy system. Step R1 registers a server with unicast address **S** for the service offered by anycast address **A** and port **b**. Note that this registration uses native anycast to reach the *closest* server proxy (SP). Next, a client can initiate a session by sending a packet addressed to the anycast service of choice (step 1). When the packet arrives at the *closest* client proxy (CP), it is tunneled to a suitable SP (step 2), where it is tunneled again towards a target server (step 3). The return path (steps 4, 5 and 6) is realized in the same way. *Stateful tunneling* occurs twice in each direction and is necessary to guarantee session continuity. The IP tunnels cannot be avoided on the return path because both the CP and SP have to monitor the session state, for which packets have to traverse the system in both directions. This contrasts with PIAS, where the return path does not pass the proxy infrastructure. Another PIAS dissimilarity can be found in the communication phase between SP and target anycast server, where IP tunneling is preferred over network address translation (NAT), as this preserves end-to-end connectivity. This is important for IPsec support and application layer services that experience difficulties traversing NAT gateways. On the downside, the packet overhead increases on the path between the SP and target server (in both directions) due to the extra IP header. Because the second tunnel on the return path (step 5) is unavoidable, this is not an extra limitation on the return path.

C. Overlay architecture benefits

Since the described architecture is based on PIAS, the design goals outlined by Ballani et al. [5] are also achieved. More specifically, the overlay architecture drastically improves IP anycast routing scalability because a single IP range can be allocated for all anycast services, thereby aggregating all anycast services into a single routing table entry. This is particularly important when the number of anycast groups increases. Additionally, anycast group dynamics do not directly interact with the routing substrate, adding to overall routing stability. Besides path length, server proxies can use server state information (e.g., CPU load, memory) to select the most appropriate target for a service request. Based on aggregated server state information distributed by the server proxies, client proxies can also forward a request to the most suitable server proxy. In Section IV, simulation results demonstrate that the dissemination of aggregated state information between proxies increases service robustness. At the same time, control plane overhead decreases, leading to improved system manageability.

Stateful communications are not explicitly supported by the proposed overlay mechanism since steps 1 and 4 in Fig. 1 cannot guarantee that subsequent packets from the same session arrive in the same proxy. However, in practice



Step	Packet headers (From → To)
R1	S:b → A:c
1	C:a → A:b
2	CP:C:a → SP:A:b
3	A:C:a → S:A:b
4	S:A:b → A:C:a
5	SP:A:b → CP:C:a
6	A:b → C:a

Fig. 1. Anycast communication through the proxy system. In the table capitals refer to IP addresses and lowercase characters point to the TCP/UDP port used.

there are two reasons why the overlay infrastructure suffices to support stateful communications. First, the number of proxies is relatively small compared to the number of network nodes, meaning that a single link or router failure is unlikely to cause a client (or server) to swap to another proxy node. Secondly, the distance between a client (or server) and its closest proxy node is usually significantly smaller than the end-to-end distance between a client and a server, thereby reducing the chances for a failure on the path segment between client (or server) and proxy node.

III. Heuristic infrastructure dimensioning

A. Problem statement

Equipped with the anycast architecture outlined in Section II, we wish to determine how many proxies are needed and where they should be attached to the network for a given client and server configuration. More formally, given a network $G(V, E)$, a set of source sites $S \subset V$ and their demands d_i, a set of server sites $T \subset V$ and their capacities c_j, edge weights $w_e : e \in E$, determine how many CP (resp. SP) are needed, and where they should be attached to the network. Additionally, determine which target sites t_j need to be opened. The optimization process should balance network operational costs (related to flow unit processing costs for regular edges (w_e) and flow unit processing costs for proxies and servers $u^{\cdots}$), proxy infrastructure costs (determined by the fixed charge f^{CP} (resp. f^{SP}) associated with each CP (resp. SP)), and server site opening costs f^{t_j}. The parameters for this optimization problem are summarized in Table I.

B. Solution techniques

In [9], we address the optimal placement and dimensioning of such an anycast architecture using an integer linear program (ILP) solved by a branch-and-bound algorithm [10]. Unfortunately, due to the complexity of the formulation, an exact solution can only be computed for relatively small networks (up to 300 nodes). For this reason we propose two heuristic methods to solve this problem: CP and SP *separated* and *combined* optimization. Contrary to the global optimization performed by the exact ILP, both heuristics decouple the proxy placement problem from the traffic engineering between the proxies (see Fig. 1), which results in a two-step optimization plan:

(i) Find suitable CP and SP locations and determine which target sites to use;
(ii) Optimize the flow between CPs and SPs.

In fact, step (ii) does not contribute to the proxy placement and dimensioning optimization, but allows us to examine the efficiency of the proxy locations determined in step (i). Using the proxy locations provided by step (i), a regular ILP minimizes the total network flow transportation cost between the proxies in step (ii):
Minimize

$$f(x) = \sum_{e \in E} w_e x_e$$

subject to

$$\sum_{e \in out(n_i)} x_e - \sum_{e \in in(n_i)} x_e = b_i, \forall n_i \in V \tag{1}$$

$$x_e \geq 0$$

In Eq. 1, b_i stands for the residual flow in node n_i, which might be positive (CP), negative (SP) or zero (regular node). In the ILP, x_e denotes the flow over edge $e \in E$. For this formulation, we assume all edges are directed. If this would

TABLE I
Model parameters

Variable	Description
$G(V, E)$	network topology, V and E denote the sets of vertices and edges
S	set of source sites $s_i (S \subset V)$
T	set of target sites t_j $(T \subset V)$
w_e	edge weight $(\forall e \in E)$
d_i	aggregated demand from source site s_i (units of flow)
c_j	capacity of target t_j (units of flow)
$f^{\cdots}$	fixed charge for opening a CP (f^{CP}), SP (f^{SP}) or target (f^{t_j}) edge
$u^{\cdots}$	unit processing cost for a CP (u^{CP}), SP (u^{SP}) or target (u^{t_j})

not be the case, undirected edges can easily be replaced by two directed edges.

For step (i), the actual proxy placement, we propose two network transformations that allow us to reformulate the problem as a *Fixed Charge Network Flow Problem* (FCNFP). FCNFP is a well-known subclass of the minimum concave-cost network flow problems and is known to be $\mathcal{NP}$-hard [7]. This problem can be formulated as follows:
Minimize

$$f(x) = \sum_{e \in E} f_e(x_e)$$

subject to

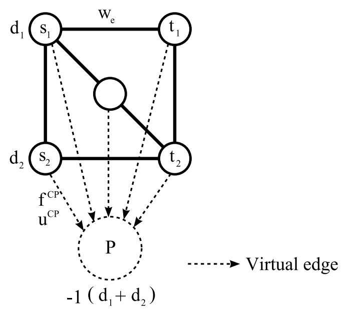
$$\sum_{e \in out(n_i)} x_e - \sum_{e \in in(n_i)} x_e = b_i, \forall n_i \in V$$

$$0 \leq x_e \leq c_e$$

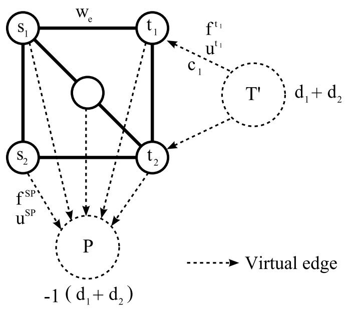
where c_e is the maximum capacity of edge $e \in E$ and $f_e(x_e)$ is defined as follows:

$$f_e(x_e) = \begin{cases} 0, & x_e = 0 \\ f_e + u_e x_e, & x_e > 0 \end{cases} \tag{2}$$

In Eq. 2, f_e and u_e denote the fixed charge and unit flow transportation cost for edge $e \in E$. To overcome the computational complexity inherent to the FCNFP, approximation techniques have been developed. For our heuristic approach, we employ the Dynamic Slope Scaling Procedure (DSSP) introduced by Kim and Pardalos [7].



(a) Client proxy selection



(b) Server proxy selection

Fig. 2. Transformed network topology for separated client and server proxy optimization.

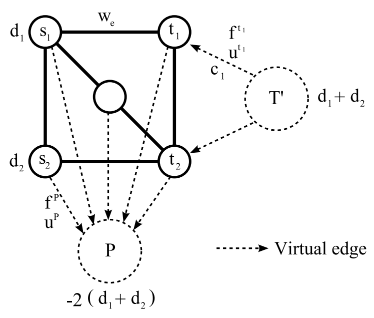


Fig. 3. Transformed network topology for combined optimization.

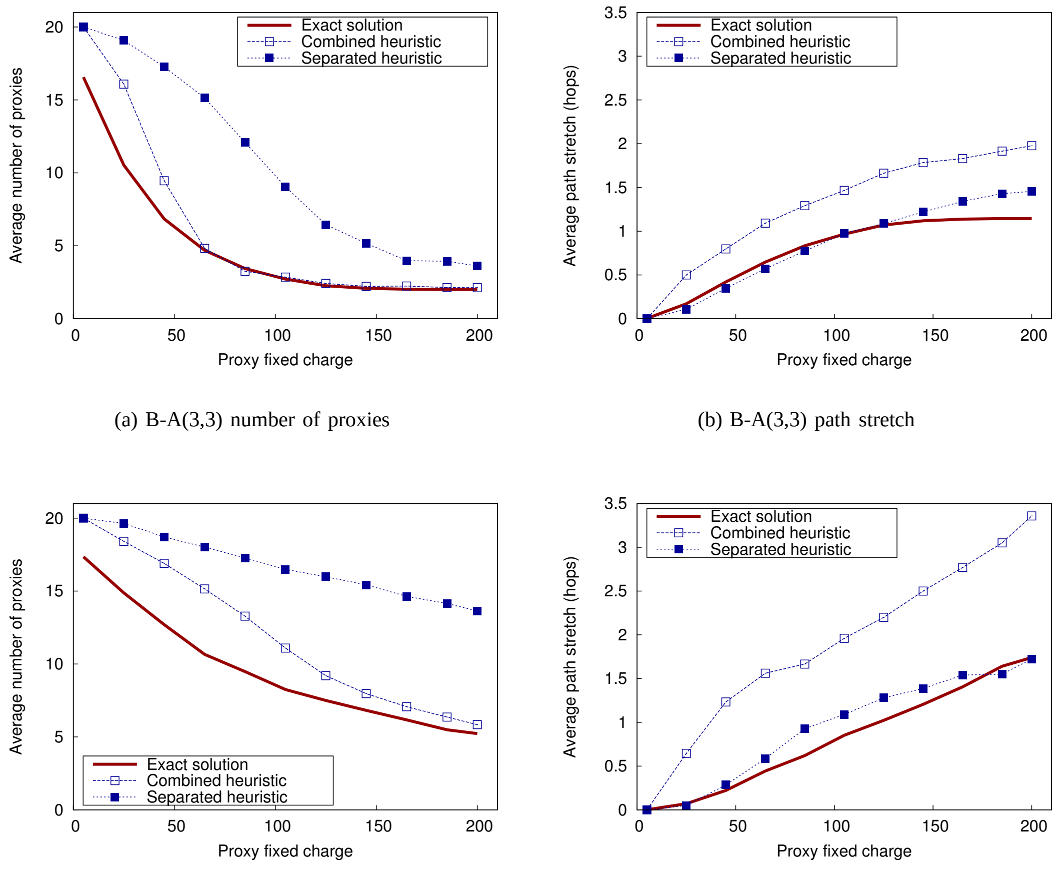
Fig. 2(a) depicts the first transformation: the original network is drawn in bold and the cumulative arcs and nodes for the transformation are plotted using dotted lines. Two source nodes s_i with demand d_i are located on the left and two possible targets t_j with capacity c_j on the right. By assigning a fixed charge f^{CP} to all virtual edges connecting the virtual sink P to the real network nodes, solving the FCNFP yields the optimal CP locations; each virtual edge with positive flow reveals a CP location. SP locations are discovered using a similar transformation shown in Fig. 2(b), with one additional virtual node T' representing the server locations. Virtual edges connecting T' to the server sites are capacitated to reflect server capacity limitations.

The separated optimization heuristic finds CP and SP locations using two FCNFP instances reflecting the transformed networks depicted in Fig. 2. Since clients and servers connect to the nearest proxy node without making a distinction between CP and SP, an additional merging routine is necessary to guarantee correctness: a client proxy is supposed to connect to a CP, whereas a target server has to connect to a SP. This is achieved by augmenting CP or SP proxies to proxies supporting both where necessary. After applying this routine, unused proxies can be dropped.

Combined optimization applies both network transformations together to create a single FCNFP instance. This approach is depicted in Fig. 3. Initially, each arc to the virtual proxy node P carrying positive flow is both a CP and SP. Unused proxy functionality is removed after solving the FCNFP. In this case, the fixed charge $f^P = f^{CP} + f^{SP}$ associated with opening a virtual proxy edge can be determined based on the cost to combine a CP and SP on the same node.

C. Results

The purpose of this evaluation section is twofold: one aspect we want to investigate is the optimality of the heuristic dimensioning approaches discussed in the previous section, another item of interest is the dimensioning and placement behavior in small and large networks.



(a) B-A(3,3) number of proxies

(b) B-A(3,3) path stretch

(c) Lattices number of proxies

(d) Lattices path stretch

Fig. 4. Comparison of exact optimization and both heuristic approaches for 100 node Barabási-Albert graphs and square lattices. For each class of graphs, results are averaged out over 100 instances. The number of installed proxies and the path stretch are related to the fixed charge for installing a proxy.

TABLE II
PARAMETER VALUES

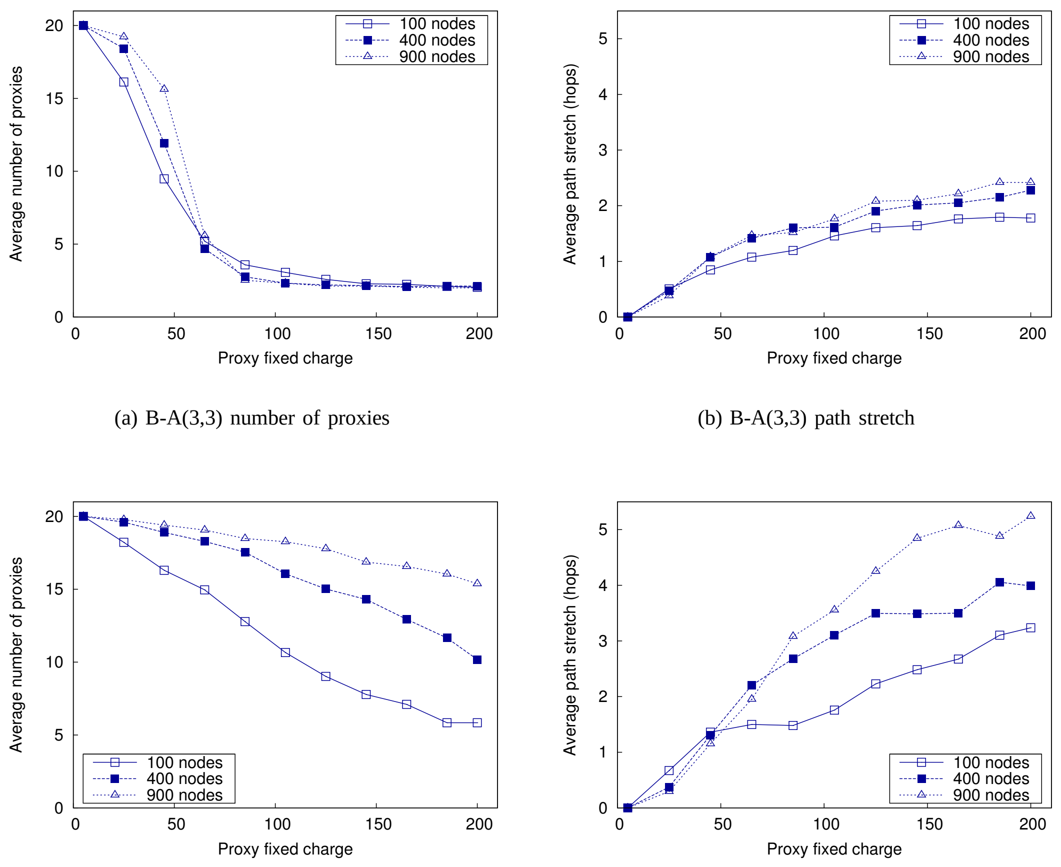
Parameter	$\|V\|$	$\|S\|$	$\|T\|$	d_i	c_j	f^{t_j}	$u^{\cdots}$
Value	100	10	10	100	100	0	0

Throughout this section two classes of random graphs are used: Barabási-Albert random graphs[1] and square lattices, both with discrete uniformly distributed edge weights w_e drawn from the set $\{0.1, 0.2, ..., 0.9\}$. These two types of graphs cover a wide range of random graphs: lattices are artificial networks with a regular structure, whereas B-A graphs are small world networks that are often used to model large networks (e.g., the Internet) in a realistic way.

A comparison between both heuristic approaches and the exact optimization ILP [9] is depicted in Fig. 4. Results are averaged out over hundred iterations and input parameters for the optimization model are shown in Table II. Routers providing network access to client sites and target sites are selected randomly and total client demand equals total server capacity. Obviously, an increasing fixed charge for installing a proxy (either a CP or SP) leads to less proxies being installed and a growing path stretch. From Figs. 4 and 6, the following conclusions are drawn:

1) Both heuristics follow the same trend as the exact optimization and provide good approximations for the exact approach. It is possible that heuristics provide better results than the exact ILP for one of the sub-problems (e.g., on Fig. 4(b)); the combined solution cost is at least as high as the cost computed by the exact ILP, however.
2) Separated optimization generally yields results with a smaller path stretch, at the expense of installing more proxies. Combined optimization suggests a smaller number of proxies and a larger path stretch. This is mainly due to the merging procedure for the separated heuristic changing the proxy configuration after the FCNFP optimization process. On the contrary, the combined heuristic initially overestimates the infrastructure costs by coupling CP and SP functionality. Afterwards, the infrastructure costs can

[1]In this paper **B-A(3,3)** stands for the class of Barabási-Albert random graphs [11] with three initial nodes, and during the growing process new nodes are connected by three edges



(a) B-A(3,3) number of proxies (b) B-A(3,3) path stretch

(c) Lattices number of proxies (d) Lattices path stretch

Fig. 5. Average number of proxies and average path stretch related to the fixed charge for installing a proxy. Results are shown for 100, 400 and 900 node networks using the combined optimization heuristic.

often be reduced when excess (unused) functionality is removed.

3) Contrary to the exact ILP, the heuristic approaches enable optimized anycast infrastructure dimensioning and placement for large networks consisting of more than 2000 nodes (see Fig. 6). Execution times are measured with modern PC hardware, using ILOG CPLEX [12] branch-and-bound software.

Fig. 5 shows optimization results of the combined optimization for 100, 400 and 900 node networks. Apart from the number of nodes, the simulation input parameters from Table II are used. This leads to the following observations:

1) In small world networks, a relatively small number of proxies suffices to achieve a solution with low network overhead. This is shown on Fig. 5(a), where the number of installed proxies decreases rapidly to a minimum proxy configuration as the fixed charge increases. Due to the small world properties which B-A(3,3) graphs obey, the corresponding path stretch (see Fig. 5(b)) does not increase significantly for larger networks.
2) For square lattices, an increasing number of nodes results in an increase for the average distance between nodes. As such, this artificial type of network can be seen as a worst-case scenario for deploying the anycast infrastructure. Figs. 5(c) and 5(d) illustrate that the total proxy infrastructure cost depends significantly on the network size. Fortunately, real large networks have small world properties [13], meaning they are not susceptible to this issue.

IV. State aggregation: Key to service robustness

In order to evaluate service robustness and control plane scalability, a discrete event simulator supporting the distinct anycast components was built. This way, target server selection behavior and control plane overhead for exchanging server state can be investigated for varying proxy locations.

For any distributed service provisioning platform, selecting a target server based on a combination of network and resource state can significantly improve efficiency. Unfortunately, injecting resource state information in the target selection process is not as easy as one might think. Resource state information is usually volatile, necessitating frequent update messages.

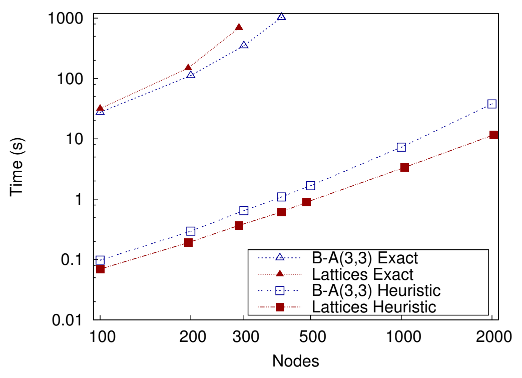


Fig. 6. Comparison of execution times for the exact dimensioning ILP and the combined optimization heuristic. A log scale is used for both axes.

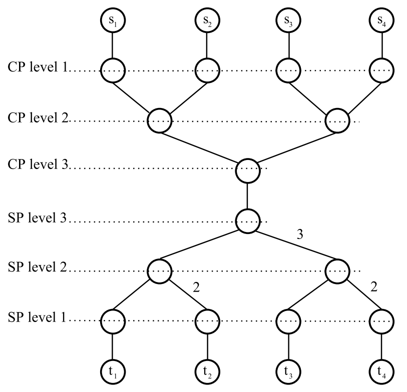


Fig. 7. Simulation network topology

For larger networks, state information messages might even be stale upon arrival, necessitating routing under inaccurate state information. Moreover, increasing the number of resources or the status update frequency can stress the control plane [14]. When resource state information is injected in the routing substrate to achieve total transparency towards client nodes, even routing instability may occur due to constant routing table updates. The proposed anycast overlay effectively shields these dynamics towards the routing substrate and preserves service transparency.

Fig. 7 depicts the network topology used for investigating the control and data plane behavior of the anycast infrastructure. On top, four client sites initiate sessions to the anycast address of the four target servers below. The link propagation time equals one time unit, unless indicated otherwise. In the resource aggregation tree, propagation time is chosen in such a way that the distance from a source s_i to target t_j is smaller than the distance to t_{j+1}. As such, native IP anycast would forward all requests to the same target (t_1). Both CP and SP can be placed on three levels, as indicated on the figure. CP use a simple scheduling policy: *"Select the closest SP with a free resource, and if all resources are busy select the SP with the largest number of connected resources."* SP employ a similar scheduling policy to select the actual target server. Table III summarizes the simulation setup. The combined session request inter-arrival time of all clients behind a single aggregating client network node (s_i) is supposed to be negative exponentially distributed, as such session arrivals in the system are characterized by a Poisson process. Job (session) duration is also assumed to be negative exponentially distributed. These assumptions are motivated in [15]. Servers update their SP every 50 time units. Taking into account the average session duration (10^4 time units), this is a high update frequency.

TABLE III
SIMULATION SETUP. SERVER CAPACITY IS 120% OF THE AVERAGE LOAD.

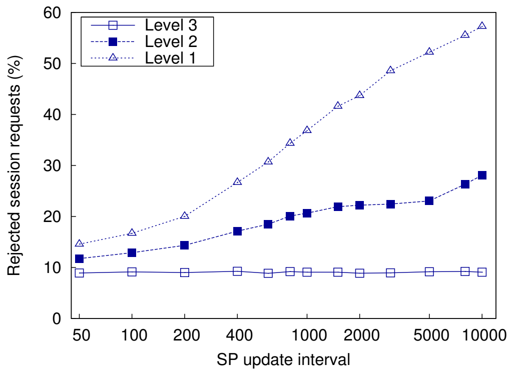
	Parameter	Value
client	Job inter-arrival time	Neg. Exp. with $\frac{1}{\lambda} = 10^3$
	Session duration	Neg. Exp. with $\frac{1}{\lambda} = 10^4$
resource	Parallel sessions	12
	Job queue length	0
	$S \rightarrow SP$ update interval	50
global	Simulation duration	10^7
	w_e (link propagation time)	1 (Unless indicated otherwise on Fig. 7)

From the simulation results shown in Fig. 8, we draw the following conclusions:

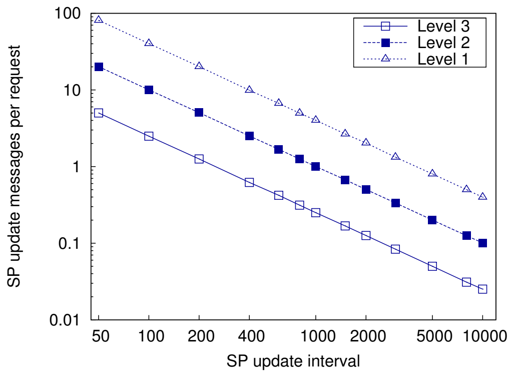
1) Fewer proxies (proxies placed on level 3 in Fig. 7) lead to a very stable system in terms of session acceptance rate, even for a low SP $\rightarrow$ CP update frequency. In fact, the situation for CP and SP placed on level 1 more or less corresponds to a no-proxy system where resources update all clients directly. In this case, the tradeoff between a high session rejection rate for a large SP $\rightarrow$ CP update interval (Fig. 8(a)) and a high number of update messages for a small update interval (Fig. 8(b)) clearly illustrates the problem described in the first and second paragraph of this section. In Section III, we show that an increasing proxy cost quickly leads to fewer proxies being installed, a result that harmonizes well with this system stability observation.
2) A network event is defined as a single action performed in the simulator (e.g., forward a request to the next hop, send an update message to the next hop). As such, the total number of events generated per session request gives a good indication of the global network load. Fig. 8(c) shows that the global network load of the level 3 configuration might exceed that of the other configurations (with more proxies), despite the smaller number of SP $\rightarrow$ CP update messages for the same update interval (Fig. 8(b)). This is due to the larger hop count between servers and their SP for the level 3 configuration and the high S $\rightarrow$ SP update frequency. If network resources are limited, this may be a good reason to increase the number of proxies to place them closer to the resources.

V. CONCLUSION

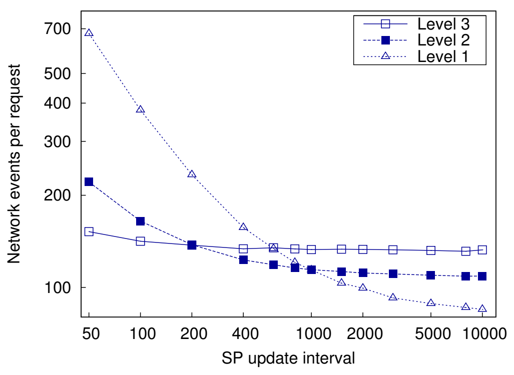
Routing scalability concerns and the lack of stateful communications support in native IP anycast prevent its widespread



(a) Rejected sessions



(b) Update messages sent



(c) System events

Fig. 8. Simulation results for a decreasing SP $\longrightarrow$ CP update frequency.

adoption for scalable and transparent service provisioning. In this paper, we presented a lightweight proxy infrastructure to overcome these issues and unleash the power of this addressing mechanism. Contrary to native anycast, these proxies hide anycast group dynamics towards the routing substrate. Moreover, server selection can take into account both network and resource state, which is not possible using IP anycast.

Dimensioning studies have shown that a relatively small number of proxies suffices to effectively accommodate an anycast-based service provisioning platform, especially in networks with small world properties such as the Internet and large provider networks. Furthermore, server state aggregation in the proxy infrastructure adds significantly to overall service robustness, while decreasing the control plane overhead.

Acknowledgment

The work presented in this paper is supported by the EU through the IST Project Phosphorus (www.ist-phosphorus.eu). The Phosphorus project is funded by the European Commission under the FP6 contract no. 034115.
Chris Develder thanks the FWO for his post-doctoral grant.

References

[1] C. Partridge, T. Mendez, and W. Milliken, “RFC 1546: Host Anycasting Service,” November 1993.
[2] S. Sarat, V. Pappas, and A. Terzis, “On the Use of Anycast in DNS,” *SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 394–395, June 2005.
[3] E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee, “Application-Layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, pp. 455–466, August 2000.
[4] D. Katabi and J. Wroclawski, “A Framework for Scalable Global IP-Anycast (GIA),” *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 3–15, October 2000.
[5] H. Ballani and P. Francis, “Towards a Global IP Anycast Service,” *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 301–312, October 2005.
[6] P. Gray, “Exact Solution of the Fixed-Charge Transportation Problem,” *Operations Research*, vol. 19, no. 6, pp. 1529–1538, October 1971.
[7] D. Kim and P. Pardalos, “A Solution Approach to the Fixed Charge Network Flow Problem Using a Dynamic Slope Scaling Procedure,” *Operations Research Letters*, vol. 24, no. 4, pp. 195–203, May 1999.
[8] M. Freedman, K. Lakshminarayanan, and D. Mazières, “OASIS: Anycast for Any Service,” in *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI 06)*, San Jose, California, United States, May 2006.
[9] T. Stevens, F. De Turck, B. Dhoedt, and P. Demeester, “Achieving Network Efficient Stateful Anycast Communications,” in *Proceedings of the 21st International Conference on Information Networking (ICOIN 2007)*, Estoril, Portugal, January 2007.
[10] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*. New York, United States: Wiley-Interscience, 1988.
[11] A. Barabási and R. Albert, “Emergence of Scaling in Random Networks,” *Science*, vol. 286, pp. 509–512, October 1999.
[12] ILOG CPLEX, “http://www.ilog.com/products/cplex/,” 2006.
[13] R. Albert and A. Barabási, “Statistical Mechanics of Complex Networks,” *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97, January 2002.
[14] T. Korkmaz, M. Krunz, and J. Guntaka, “OSPF-based Hybrid Approach for Scalable Dissemination of QoS Parameters,” *Elsevier Computer Networks*, vol. 46, no. 2, pp. 273–293, October 2004.
[15] K. Christodoulopoulos, M. Varvarigos, C. Develder, M. De Leenheer, and B. Dhoedt, “Job Demand Models for Optical Grid Research,” in *Proceedings of the 11th Conference on Optical Network Design and Modelling (Accepted for publication)*, Athens, Greece, May 2007.