

Cost- and Quality-of-Service-Aware Network-Service Deployment

Robert Haas¹, Patrick Droz¹ and Burkhard Stiller²

¹IBM Research, Zurich Research Laboratory
CH-8803 Rüschlikon, Switzerland
{rha,dro}@zurich.ibm.com

²Computer Engineering and Networks Laboratory
ETHZ, CH-8092 Zurich, Switzerland
stiller@tik.ee.ethz.ch

Abstract: *This paper presents the information aggregation methods enabling cost- and QoS-aware service deployment. It is expected that network management tools will require the network itself to participate in the service deployment task so as to adapt to heterogeneous network-nodes cost, QoS, and capabilities. The aggregation methods are part of a set of hierarchically-distributed computations, for which a formal description is presented. Four types of information that can be handled by the mechanism are introduced and illustrated by examples from three service-deployment categories.*

Keywords: *Cost and quality of service awareness, information aggregation, automated service deployment*

1. Introduction

The importance of fast, optimized, and reliable deployment of new services into a network is a key issue for Internet Service Providers. Simultaneously, network elements in general offer an increasing spectrum of capabilities, some with dedicated specialized functions, others with programmable behaviors, in soft- or hardware. These two driving forces render the deployment of services in a network more complex to manage. As a first step to enable interoperability and faster creation of new services, standardized Application Programming Interfaces (APIs) for network elements are being defined. But in such an environment of networks with large numbers of nodes that need to be enabled with new services and that have widely varying capabilities and resources, it is necessary to define and provide a way to organize the deployment of new services at a network level.

Here, we present the information aggregation methods of a framework used to capture the dynamic capabilities of a network and to organize its deployment based on specific service-allocation policies, thereby addressing the programmability and heterogeneity aspects of the network.

To our knowledge, this framework is the first to rec-

ognize and propose solutions to this particular problem. The motivation of our approach can be viewed as similar to that of quality-of-service (QoS)-aware routing protocols. Such protocols replace lengthy and error-prone manual steps of provisioning resources within a network to guarantee a certain level of QoS. By handling the information related to the available QoS in the network internally, routing protocols can perform this task more efficiently than any operator or management platform alone could. In addition, the cost of deploying a service can be computed in a distributed fashion as a function of performance, thereby allowing the most profitable option to be chosen.

This paper is structured as follows: Section 2 reviews related activities. Section 3 introduces a formalism for hierarchically-distributed computations, illustrated with an example. Section 4 presents the various basic information types, and introduces aggregation methods for services of the three service-deployment categories. Section 5 contains a brief summary of this contribution as well as an outlook.

2. Related Work

There are only few known activities that focus on the automated deployment of services over large heterogeneous programmable networks. Hierarchical architectures have been used in routing protocols and network management, but not yet considered in the context of deploying services. Let us briefly review the main activities of related work.

In [1], the need for a distributed and programmable management platform for the future Internet is presented, and simple navigation patterns for mobile agents are described. The underlying hierarchical structure in our contribution can be viewed as another, more sophisticated navigation pattern used by mobile agents to perform the specific task of scalable service deployment.

In [2], the need for an automated design, creation, and deployment of network architectures is presented, and a

high-level methodology to spawn virtual network architectures based on the Genesis profiling system that relies on distributed object technology and centralized profile databases is proposed. In [3], a framework to isolate between services deployed in different virtual active networks (VANs) is presented, whereas the creation of a VAN essentially remains a manual task.

Hierarchical structures are used in IP and ATM networks to aggregate and propagate routing information. While IP networks only aggregate routing information, with two to three levels of hierarchy, ATM-PNNI [4] also summarizes bandwidth and delay characteristics to allow QoS routing. In [5], a complex node representation that captures the relevant characteristics of the nodes at the lower hierarchy levels is proposed. Likewise, distributed network management [6-9] uses hierarchical structures to better scale with large numbers of nodes and complex management tasks such as distributed monitoring with mid-level managers. In [10, 11], the discovery process of distributed services over wide-area networks exploits a hierarchy of service brokers to propagate end-hosts queries and to cache advertisements of installed media gateway services.

Dynamic composition and deployment of services in the context of end-to-end application sessions are addressed in [12-14]. This applies for instance to the setup of a network path for a multimedia session based on the availability and cost of image transcoders and compression service components active in intermediate network nodes.

3. Hierarchical Distributed Service-Deployment

This section concentrates on the formalization of the service-deployment mechanism using an approach similar to [15], albeit with specific enhancements. We then illustrate how it is implemented for service deployment of the *path-based* category.

From a high-level perspective, the service-deployment mechanism can be broken down into five steps. Figure 1 shows these steps and the resulting deployment procedures when all or only some of the steps are executed. Using only the last two steps in Fig. 1 leads to a *manual deployment and automatic configuration* of a service. This is how services are generally deployed in networks today. With the intermediate solution, the result is an *automatic deployment with generic metrics and automatic configuration*. Only when all five steps have been executed will an *automatic deployment with custom metrics and automatic configuration* result.

Computations are distributed in a logical hierarchy de-

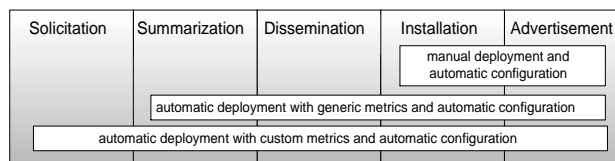


Fig. 1. The five steps of the service-deployment mechanism.

scribed in [16]. Layers in the hierarchy are built by recursively grouping logical or physical nodes and representing them by a single logical node at the next layer of the hierarchy.

3.1. Hierarchical Iterative Gather-Compute-Scatter (HIGCS) Algorithm

The mechanism is divided into a sequence of iterations each consisting of gather, compute, and scatter phases. The sets of destinations and origins relevant for the messages exchanged in the scatter and gather phases, respectively, are obtained from the compute phase. Note that these iterations can occur indistinctively in logical nodes of the hierarchy as well as in nodes of the underlying physical topology.

Service-deployment HIGCS messages exchanged during these iterations follow the tree-like topology of the service hierarchy. The logical node of a group is merely responsible for communicating with its underlying peer group members of the scatter set. The logical node neither has to monitor all members of its group nor perform all computations centrally. But without loss of generality, we assume in the following descriptions that the scatter set is always determined by a central computation performed by the logical node rather than distributing this computation among group members. In addition, here we consider it more appropriate to focus on the distributed computations taking place along the hierarchy dimension itself, than on those along both cross-hierarchy and group-internal dimensions.

Note that this approach of organizing distributed computations can be elegantly implemented with mobile agents following the navigation model extracted from the hierarchical structure.

Each node executes iterative computations based on a tuple $\{(G_i, C_i, S_i) | 0 \leq i \leq (k-1)\}$, where k is the total number of iterations. The gather set G_i is defined here as the set of (logical or physical) nodes from which messages are expected. The compute phase C_i executes once the $iMsg$ messages have been received from all nodes in G_i . The scatter set S_i denotes the (logical or physical) nodes to which $oMsg$ messages are sent once the compute phase C_i completes. $oMsg_n$ messages can differ

depending on their destination node n in the S_i set. Similarly to [15], node attributes are assumed to be available during the compute phase. This includes, for instance, the hierarchy level at which the node is located.

The generic signaling message format used by the service-deployment mechanism is defined in Table I.

TABLE I
THE HIGCS SERVICE-DEPLOYMENT MESSAGE FORMAT.

Parameter	Generic value
<i>serviceId</i>	Instance of service deployed
<i>hierarchyId</i>	Identifier of service hierarchy
<i>srcId</i>	Source of message
<i>destId</i>	Destination
<i>iterationId</i>	Current iteration
G_i	Gather set for iteration i
C_i	Compute function for iteration i
S_i	Scatter set for iteration i
<i>servSpec</i>	Service specification
<i>sMetric</i>	Solicited metric
<i>iMetric</i>	Installed metric
...	Other service-specific info

3.2. HIGCS Service-Deployment Computations

To perform service deployment, iterations are associated with the steps as described in Fig. 1. For that purpose, we define the following general behaviors for C_i :

- C_0 selects the set of underlying nodes S_0 that have to be solicited (null set if executed on a physical node),
- C_1 summarizes information gathered from the set of underlying nodes G_1 (on a physical node, this information is created), and places it in *sMetric*,
- C_2 selects the set of nodes S_2 where the service is to be deployed (on a physical node, the service is installed),
- C_3 summarizes the results from the deployment on the set of nodes G_3 (on a physical node, this information is obtained locally), and places it in *iMetric*.

Clearly, these functions will be implemented differently for each service to be deployed, as we shall describe next. The service-specific allocation policy is contained in the implementation of the C_1 and C_2 functions.

Table II shows the computations executed for the deployment of a hypothetical path-based service on all nodes of a path between two customer sites (represented by the A and C top-level nodes in Fig. 2). For the sake of simplicity, the straightforward update of other fields in the *oMsg* (such as *srcId*) and fields that do not change compared to the *iMsg* message are not shown.

Figure 2 illustrates the result after all four HIGCS iterations have completed. The selected path is shown

TABLE II
 $C_0, C_1, C_2,$ AND C_3 FUNCTIONS.

C_0	DS-solicit
S_0	\leftarrow SelectAllNodesBetween(<i>iMsg.ends</i>)
<i>oMsg_n.ends</i>	\leftarrow SelectNeighborNodes($n \mid n \in S_0$)
G_1	\leftarrow S_0
C_1	DS-summarize
S_1	\leftarrow GetLogicalNode()
<i>oMsg.sMetric</i>	\leftarrow if IsLogicalNode() then SummarizeMetrics(<i>iMsg_j.sMetric</i> , $\forall j \in G_1$) else CreateMetric(<i>iMsg.servSpec</i>)
G_2	\leftarrow S_1
C_2	DS-disseminate
S_2	\leftarrow if IsLogicalNode() then SelectNodesOnShortestPath(<i>iMsg.ends</i>) else null
<i>oMsg_n.ends</i>	\leftarrow SelectNeighborNodes($n \mid n \in S_2$)
G_3	\leftarrow S_2
C_3	DS-install
<i>oMsg.iMetric</i>	\leftarrow if IsLogicalNode() then SummarizeInstalledMetrics(<i>iMsg_j.iMetric</i> , $\forall j \in G_3$) else InstallService(<i>iMsg.servSpec</i>)
S_3	\leftarrow GetLogicalNode()

as a thicker line. The first iteration was initiated by node B , which received the initial HIGCS message with *iMsg.ends* set to $\{A, C\}$. Logical node $B.5$ and router $B.2.4$ did not participate in the service deployment, as they are not on a path between A and C . All other nodes had been solicited, and nodes $B.1.1$ and $B.3.4$ were excluded as their capabilities did not match the requirements of that service. Of the remaining routers, $B.2.2$, $B.2.3$, $B.3.1$, $B.3.2$, and $B.3.3$ are finally enabled with the service, as they are on the shortest path between A and C .

4. Information Aggregation for QoS- and Cost-Aware Deployment

The service deployment framework allows QoS-aware deployment of services. In addition, the cost dimension is considered, enabling charging mechanisms for the deployment of services. First we classify the various basic information types that are relevant to the service-deployment framework, list the three types of services considered, and then show how these basic information types can be used in the context of service deployment of all three categories.

4.1. Basic Information Types

In general, services make use of one or a combination of the following basic types in order to achieve proper deployment:

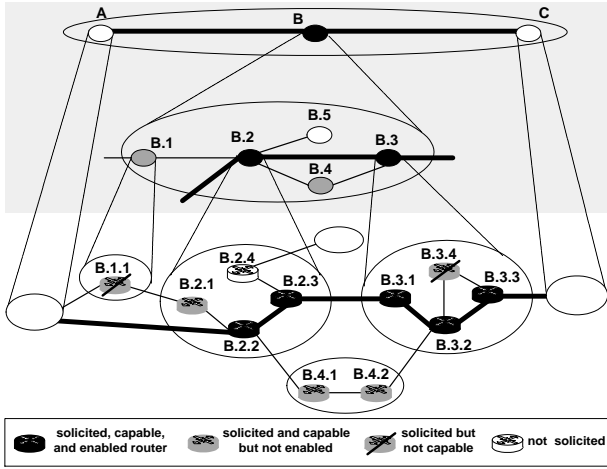


Fig. 2. Result of the HIGCS computations

- topology: description of the connectivity,
- capability: description of node capabilities,
- performance: measure of topology (such as bandwidth or delay) or node capability (such as CPU speed) usage, and
- cost: administrative measure for using the above resources.

Note that when this information is transported towards the top of the hierarchy, it can be abstracted and aggregated differently depending on service needs. One advantage of the service-deployment framework proposed here is to let services define the information they require to execute their deployment, and how it should be aggregated. This can be viewed as an application of active code for the network control plane.

Other aggregation methods, such as used in [11] (Bloom filters) for service discovery, differ in their purpose from the approach presented here. Whereas here only relevant capabilities are extracted from the network thanks to the solicitation step, Bloom filters aggregate all installed services in the network, and achieve scalability only at the price of false positive advertisements.

Services are classified into three categories depending on the type of locations where they are to be deployed, namely:

- path-based,
- node-based, and
- path- and node-based.

Each category determines what information of the four basic types is aggregated and how it is done.

4.2. Path-Based Deployment

For instance, the example in Section 3.2 uses link abstractions at the upper layers of the hierarchy. For simplicity, we assumed in that example that those logical links had already been added to the hierarchy before the first iteration of the service deployment started.

For path-based deployment, routing decisions have to be taken successively at each layer of the hierarchy (executed by `SelectNodesOnShortestPath` in Table II). Among the various suitable data representation, transition matrices offer an accurate representation of the cost of traversing a logical node.

Transition matrices used in path-based service deployment contain topology-, capability- and performance and/or cost-related information. For instance, the transition matrix generated for node B in Fig. 2 is (the matrix is symmetric):

$$\mathcal{T}_B = \begin{pmatrix} 0 & \dots & \\ 0 & 1 & \dots \\ 0 & 5 & 1 \end{pmatrix}$$

Elements $t_{i,j}$ indicate that there is connectivity between border nodes $B.(i+1)$ and $B.(j+1)$, with a path composed of nodes capable of running that particular service. In addition, the value of $t_{i,j}$ corresponds to the performance of that path in terms of number of hops. More sophisticated transition matrices having elements composed of vectors describing the efficient frontier (set of minimal paths in terms of delay and bandwidth between two border nodes) are presented in [17].

In order to make the deployment of a path-based service cost-aware, a measure indicating the cost to deploy the service can be used, similarly to the hop count performance measure. Each node computes this measure, which is added to the transition matrix.

4.3. Node-Based Deployment

We consider here the deployment of a standalone service, such as a web-cache, over the network shown in Fig. 2. This type of service does not require routing over the network, therefore topology-oriented information is not necessary. Node B initiates the deployment procedure to select the router with the best processing capability to support that particular service. The summarization proceeds by computing the max of the processing capabilities among nodes in each group, and is repeated successively at each higher layer of the hierarchy.

A cost-aware alternative to the above deployment specifies the exact required performance, rather than requesting the best performance. Next, solicited routers compute the corresponding cost to achieve that level of per-

formance. The summarization then proceeds by choosing the minimum cost, successively at each layer of the hierarchy.

In both these cases, requirements (in terms of cost or performance, or as a combination of acceptable minimal performance and maximal cost) do not allow the network manager the flexibility to evaluate cost as a function of performance, before eventually deciding on the desired performance. For this reason, we introduce the summarization of cost functions: based on policy rules, the cost of enabling a device in the network with a certain service can be plotted against the requested capabilities and associated performance. In addition, this cost can depend, for instance, on the particular customer requesting the service.

Figure 3 shows three cost functions: (a) is the hypothetical cost function for a device using hardware assists to perform the desired service; (b) is the cost function of a device performing the same service in software; and (c) is the summarized cost function for these two devices, assuming they are the only members of their group, and the service needs to be enabled in only one of the two nodes. The summarized cost function is a \min of the two cost functions, and can in turn be summarized recursively at each hierarchy level. Note that summarization of cost functions does not always necessarily map to a \min of functions; this depends on the specific service requirements.

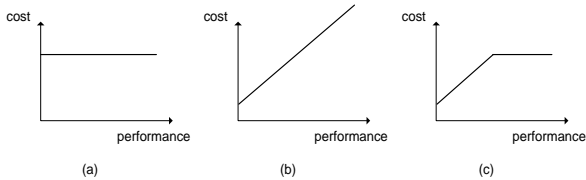


Fig. 3. (a) hardware cost-function, (b) software cost-function, and (c) summarized cost-function

4.4. Path- and Node-Based Deployment

As an example of a service deployed both along paths and at selected nodes with differing requirements, we consider the deployment of a Virtual Private Network. Encryption capabilities are required at the VPN endpoints, and QoS treatment of packets is ensured by RSVP-enabled nodes between those endpoints.

In order to accommodate both path- and node-based characteristics, we choose to augment the transition matrix presented in Section 4.2 with the necessary node information. The extended transition matrix is defined as follows:

$$\mathcal{T} = (\mathcal{M}; \mathcal{P}).$$

Elements $m_{i,j}$ indicate the shortest number of RSVP-capable hops between border nodes i and j . Elements $p_{i,j}$ indicate the shortest path from a node in domain D_j that fulfills the requirements of the VPN-endpoint service specification to border node i . The VPN interconnects n domains D_n . Logical nodes represent these domains.

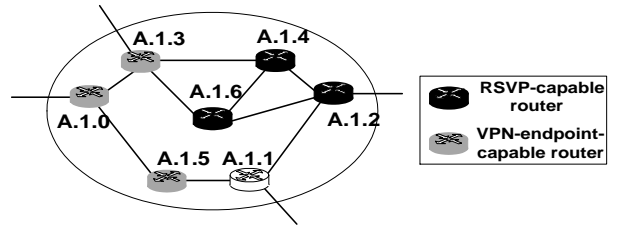


Fig. 4. A group of nodes solicited for the VPN service.

Figure 4 shows a group of nodes with their capabilities. For simplicity, it is assumed that all VPN-endpoint-capable nodes are also RSVP-capable, but not vice-versa. The extended transition matrix for this group is [assume $m_{i,j}$ corresponds to $(A.1.i - A.1.j)$]

$$\mathcal{T}_{A.1} = \begin{pmatrix} 1 & \dots & & & 1 \\ 0 & 0 & \dots & & 0 \\ 4 & 0 & 1 & \dots & 3 \\ 2 & 0 & 3 & 1 & 1 \end{pmatrix}.$$

If we assume that logical node $A.1$ represents one domain and that logical node $A.2$ (not shown) represents another domain, then the \mathcal{P} matrix of the extended transition matrix for logical node A , composed of $A.1$ and $A.2$, will have two columns $\mathbf{p}_{:1}$ and $\mathbf{p}_{:2}$, one for each domain. When summarizing such extended matrices, a new column is appended for each domain considered.

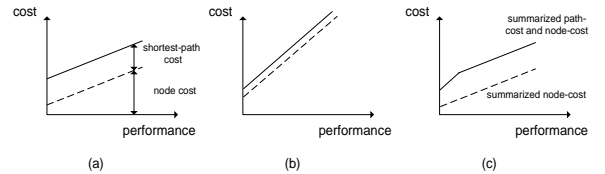


Fig. 5. (a) first cost-function, (b) second cost-function, and (c) summarized cost-function

The \mathcal{P} matrix does not account for the cost of enabling routers with VPN-endpoint capability. Similarly to Section 4.3, we introduce a cost function for each potential

VPN endpoint. To simplify the summarization of information, we choose to add the node cost to the shortest-path cost. Figure 5 shows an example of two node-cost functions (in dashed lines) (a) and (b). The cost of the shortest path is added to each node-cost function. The first segment of the summarized path- and node-cost function (c) shows that (b) is the lowest-cost VPN-endpoint, although (a) has a lower node cost. In the second segment, (a) is the lowest-cost VPN-endpoint.

5. Conclusion

Organizing the automated deployment of services is a key step towards an intelligent network infrastructure. Openness and programmability are starting to appear in network equipment and it is therefore necessary to provide generic mechanisms that can enable the deployment of any type of service.

In this paper, we have introduced three general categories of service deployment, together with their four basic information types, and proposed aggregation methods that address the cost- and QoS-aware deployment of a wide range of services in programmable and heterogeneous networks. Network-node capabilities are evaluated in terms of the requirements for a specific service, resulting in the creation of appropriate metrics. Administrative cost and cost functions can be summarized, allowing customers to compare cost versus service performance before opting for a particular service performance. Therefore, the deployment of the service can be based on a summarized view of these metrics without loss of relevant information for that specific service. The deployment task is then distributed throughout the hierarchy, following a service-specific allocation policy. Thus an automated deployment is achieved, avoiding time-consuming and error-prone manual operations. When networks grow in size and heterogeneity, the deployment mechanism can still capture the essential data for deploying any particular service, while retaining its scalability thanks to the use of summarization and dissemination across the service-deployment hierarchy.

On-going investigations include the relationship between the requirements for a generic service-deployment hierarchy and the various kinds of existing hierarchies, such as routing, management, or domain name service (DNS), and the means to evaluate the performance of such a service-deployment hierarchy over large-scale networks using simulation.

References

- [1] R. Kawamura and R. Stadler, "Active Distributed Management for IP Networks," *IEEE Communications Magazine*, 38(4):114–120, April 2000.
- [2] A. Campbell, M. Kounavis, D. Villela, J. Vicente, K. Miki, H. D. Meer, and K. Kalaichelvan, "Spawning Networks," *IEEE Network Magazine*, July/August 1999.
- [3] M. Brunner and R. Stadler, "Virtual Active Networks – Safe and Flexible Environments for Customer-Managed Services," In *Proc. 10th IFIP/IEEE Int'l Workshop on Distributed Systems (DSOM'99)*, Zurich, Switzerland, October 1999.
- [4] ATM-Forum. P-NNI V1.0. af-pnni-0055.000, March 1996.
- [5] I. Iliadis and P. Scotton, "Transition Matrix Generation for Complex Node Representation," In *Proc. IEEE ATM Workshop'99*, Kochi, Japan, May 1999, pp. 489–500.
- [6] Y. Yemini, G. Goldszmidt, and S. Yemini. "Network Management by Delegation," In *Proc. Int'l Symp. on Integrated Network Management*, April 1991, pp. 95–107.
- [7] K. Lim and R. Stadler, "A Navigation Pattern for Scalable Internet Management," In *IFIP/IEEE International Symposium on Integrated Network Management (IM'01)*, Seattle, WA, May 2001.
- [8] J. Schonwalder, J. Quittek, and C. Kappler, "Building Distributed Management Applications with the IETF Script MIB," *IEEE J. Sel. Areas Commun., Special Issue on Network Management and Operations*, 18(5), May 2000.
- [9] J. Quittek and C. Kappler, "Practical Experiences with Script MIB Applications," *The Simple Times*, 7(2), November 1999. www.simple-times.org.
- [10] D. Xu, K. Nahrstedt, and D. Wichadakul, "QoS-Aware Discovery of Wide-Area Distributed Services," In *First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, Brisbane, Australia, May 2001.
- [11] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An Architecture for a Secure Service Discovery Service," In *Mobile Computing and Networking*, pp. 24–35, 1999.
- [12] K. Nahrstedt, D. Wichadakul, and D. Xu, "Distributed QoS Compilation and Runtime Instantiation," In *Proceedings of the IEEE/IFIP International Workshop on QoS (IWQoS'2000)*, Pittsburgh, PA, June 2000.
- [13] A. Nakao, L. Peterson, and A. Bavier, "Constructing End-to-End Paths for Playing Media Objects," In *OpenArch 2001*, Anchorage, Alaska, March 2001.
- [14] S. Choi, J. Turner, and T. Wolf, "Configuring Sessions in Programmable Networks," In *Infocom 2001*, Anchorage, Alaska, April 2001.
- [15] Y. Chae, S. Merugu, E. Zegura, and S. Bhattacharjee, "Exposing the Network: Support for Topology-Sensitive Applications," In *OPENARCH 2000*, Tel Aviv, Israel, March 2000.
- [16] R. Haas, P. Droz, and B. Stiller, "A Hierarchical Mechanism for the Scalable Deployment of Services over Large Programmable and Heterogeneous Networks," In *ICC 2001*, Helsinki, June 2001.
- [17] D. Bauer, J. N. Daigle, I. Iliadis, and P. Scotton, "Efficient Frontier Formulation for Additive and Restrictive Metrics in Hierarchical Routing," In *ICC 2000*, New Orleans, June 2000.