

# A hierarchical mechanism for the scalable deployment of services over large programmable and heterogeneous networks

Robert Haas<sup>a</sup>, Patrick Droz<sup>a</sup> and Burkhard Stiller<sup>b</sup>

<sup>a</sup>IBM Research, Zurich Research Laboratory  
CH-8803 Rüschlikon, Switzerland  
{rha,dro}@zurich.ibm.com

<sup>b</sup>Computer Engineering and Networks Laboratory  
ETHZ, CH-8092 Zürich, Switzerland  
stiller@tik.ee.ethz.ch

*Abstract*— This paper presents a novel mechanism that enables scalable service deployment over programmable heterogeneous networks. For ease of service deployment, it is expected that network management tools will require the network itself to participate in this task so as to be able to scale to very large number of network elements, with widely varying programmability levels. The type of services considered is very broad, ranging from routing services involving only selected nodes of a network to end-to-end services spanning the entire network. Successive steps of the mechanism to deploy new services into a network automatically are presented. The algorithmic structure of the mechanism described is then illustrated by several examples showing its applicability and complexity.

*Keywords*— Automated service deployment, programmable networks, network management.

## I. INTRODUCTION

Services are the key differentiators for Internet Service Providers. To enable interoperability and fast creation of new services, standardized Application Programming Interfaces (APIs) with their corresponding programmable network elements start to appear. In such an environment of networks with large number 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. Here, we present a new mechanism that captures the capabilities of a network to support a new service and organizes the deployment based on specific service policies, thereby addressing the programmability and heterogeneity aspects of the network.

The motivation of our approach is 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 the operator or the management platform alone could.

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 have not yet been considered in the context of deploying services. Let us briefly review the main activities of related work.

In [1], 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 [2], a framework to isolate between services deployed in different virtual active networks (VANs) is presented, whereas the creation of a VAN remains essentially a manual task. Active networks consider a packet-centric service-deployment paradigm, which is not necessarily suitable for a heterogeneous network if a predictable and coordinated deployment of services is required. Heterogeneity is considered

in [3], where an abstraction for links traversing non-active hops is used.

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 also summarizes bandwidth and delay characteristics to allow QoS routing. In [4], a complex node representation that captures the relevant characteristics of the nodes at the lower levels of the hierarchy is proposed. Likewise, distributed network management [5] uses hierarchical structures to better scale with large numbers of nodes and complex management tasks such as distributed monitoring with mid-level managers.

To accelerate the deployment of network protocols in the future, efforts have begun focusing on the standardization of interfaces in networking equipment, either in the form of control protocols for label switches (IETF GSMP) or media gateways (IETF MEGACO), or more generic APIs such as those described in [6, 7]. It is expected that in a heterogeneous network a variety of solutions are likely to coexist.

## II. SERVICE DEPLOYMENT

Large-scale service deployment requires an automated installation mechanism. The mechanism developed here includes an algorithm determining five sequential steps that allow the deployment of a wide range of services, and relies on a hierarchy.

### A. Supported Services

Control-plane services enable the proper transfer of payloads through a network, and include services that provide routing, QoS, and security. We classify control-plane services based on explicit or implicit addressing. Implicit-addressing-based signaling does not require a node to use the addresses of its peers that run the same service to correctly execute that service. Examples are in-band signaling such as IETF Differentiated Services (diff-serv) or out-of-band soft-state signaling such as IETF RSVP. Conversely, explicit-addressing examples include most routing protocols in which routing update messages are explicitly addressed to the peer routers.

The mechanism we propose addresses the deployment of control services in both categories. The subtle difference lies in the way implicit-addressing services need to be advertised so that data packets requiring such services are routed over a proper path. Clearly, it is expected that not all services are deployed over all possible paths. Depending on the services, they can be required at each hop on a path, at the edges of a path, at selected nodes in the network, etc. How the service needs to be installed to function properly is indicated in the *service-specific allocation policy* (SSAP).

## B. Service-Deployment Hierarchy

The service-deployment hierarchy is a key element of the mechanism. It is similar to a routing hierarchy [8] because it takes the topology into account, but does not address the routing issue, therefore it does not distribute reachabilities or routes.

Topology is the main factor in creating a hierarchy. Nodes in the same subnet are grouped together, assigned a peer group leader (henceforth denoted as group leader to avoid confusion with PNNI PGL), and at the next level of the hierarchy appear as a single node in the form of a logical node. This leads to a tree-like architecture in which the use of scopes limits the flooding of information to sub-trees.

The most suitable hierarchy may also depend on the type of service being installed. This is left for further research, and we assume in what follows that a single hierarchy is used for all services. Also, the exact procedure to put in place such a hierarchy is not detailed here.

## C. Capabilities Representation

This representation captures information needed in the deployment phase of a new service to decide whether a certain service is compatible with the resources it will require, in terms of how it interfaces with them and in terms of the quantity of resources it needs. This representation, therefore, includes

- a description of the base resources [6] and their utilization,
- a description of the higher-level resources, such as OS-resident services, and their utilization if applicable, and
- a reference to the type of APIs to access, configure and operate the above resources.

The IETF host-resource MIB [9] describes certain features such as processor, memory, and storage for a host computer. It can be extended to include all the necessary elements listed above.

Although the mechanism presented here is relatively independent of the specific representation of capabilities, the service to be deployed has to express its requirements in a way that is compatible with the representation of capabilities used in the nodes. The exact mechanism of the negotiation that takes place to find out whether a node is capable of running a certain service is beyond the scope of this paper. Hence, we deliberately concentrate here on the deployment mechanism.

## D. Service-Deployment Overview

Fig. 1 shows the five steps comprising the mechanism and the resulting deployment procedures when all or only some of the steps are executed. The mechanism is robust and allows a dynamic redeployment of a service based on changes in network capabilities or topology. This is similar to how virtual circuits or packets are rerouted when changes in the routing table occur.

The first step (*Solicitation*) consists of declaring the requirements of a new service to the network. The network nodes are asked to compare these requirements with their current capabilities, and return an answer in the form of a metric. Note that if the requirements of a particular service are well-known in advance, which generally is not the case in programmable networks, or have a generic form, then this solicitation step can be skipped.

The answer returned by the network nodes needs to be summarized (*Summarization*), otherwise the number of responses would make the system unscalable. Once the responses have been summarized up to the top level of the hierarchy, the deployment is executed by choosing the nodes appropriate to sup-

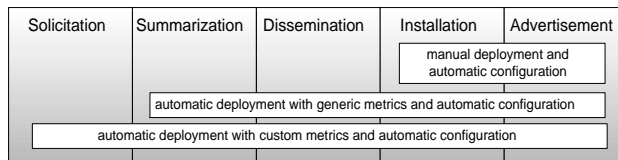


Fig. 1. The service-deployment mechanism divided into five steps.

port the service at each level, based on the SSAP (*Dissemination*).

Finally, the service is installed in the physical nodes selected (*Installation*). To allow configuration, nodes then advertise their installed services, and dynamically learn from other nodes advertising the same service (*Advertisement*). This allows, for instance, explicit-addressing control services such as routing protocols to autoconfigure the neighbor routers.

Note that the complete service life-cycle requires a removal step. However, here only the constructive service deployment is illustrated using a hypothetical deployment of a diff-serv++ service in a transit network.

### Step 1: Solicitation

This step is essentially a directed broadcast of the service requirements to the nodes of the network. In certain cases, some portions of the network can be ignored by the solicitation step: if the SSAP defines the edges or end points between which the service needs to be deployed, the solicitation step can direct the broadcast to all nodes between these edges or end points.

The goal of this step is to obtain a piece of information specific for the particular service that can easily be summarized. This information is a metric or a set of metrics.

Depending on the kind of metric they rely on, services are split into the following categories:

- Those requiring a set of custom metrics and/or custom summarization rules. The metrics are derived from the comparison of the services requirements with the actual capabilities within each network node. Therefore, they are not known in advance by the network, and
- those that only need a set of generic metrics and summarization rules, i.e., the metrics and their summarization are known in advance by the network.

It is clear that services sharing a common set of custom metrics but with having different summarization rules, will not be able to share any of the results obtained from the solicitation step.

Certain generic metrics can be advertised by default in the network, using their corresponding generic summarization rules. If such metrics are used for the deployment of new services with sufficient frequency, then advertising these metrics by default saves the solicitation step and the resulting summarization step. This can be interesting, especially for services that need to be set up in a short time period.

Values of the set of custom metrics result from the negotiation taking place when a node needs to compare its capabilities against the requirements of a certain service. The values can either simply indicate whether the service is supported (boolean), at what cost it can be supported (quantitative), or how well it is supported (qualitative).

The result of the solicitation step for our diff-serv++ example is shown in Fig. 2. Routers capable of supporting the service are shown in black.

### Step 2: Summarization

Such sets of metrics need to be summarized when being transmitted to the top level of the hierarchy. Basically, this is

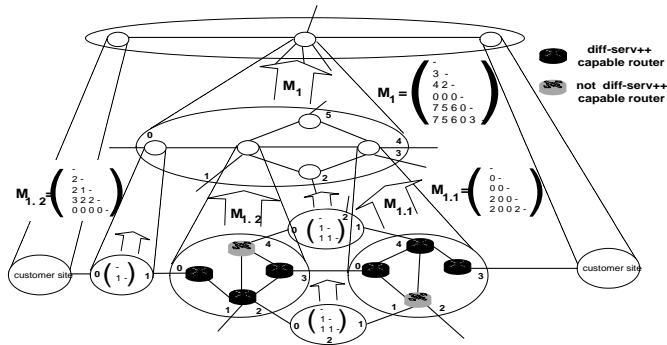


Fig. 2. Summarization step.

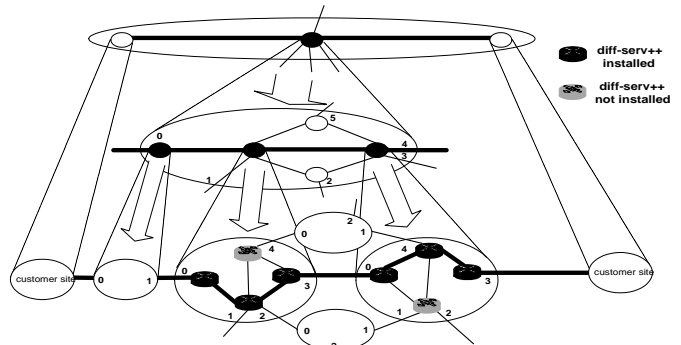


Fig. 3. Dissemination step.

how the mechanism is kept scalable. By making the summarization rules customizable, the summarized view can be made accurate enough for the deployment mechanism to work correctly. This step is repeated at regular time intervals (or when specific events take place), so as to cope with changes.

Fig. 2 shows the transition matrices [4] obtained successively at each level of the hierarchy, i.e.,  $M_{1,1}$ ,  $M_{1,2}$ , etc and finally  $M_1$ . Edges are numbered so that element  $m_{i,j}$  is defined as the number of hops on the shortest path between edges  $i$  and  $j$  that provides the required diff-serv++ service at each hop along the path. As the matrix is symmetric, only one half of it is shown.

### Step 3: Dissemination

The SSAP is used in this step to direct the deployment of the particular service. It contains topology-oriented actions, such as which nodes need to be enabled with the service. More specifically, a dissemination command to deploy the service is launched from the top-most node, and travels down the hierarchy, together with the associated SSAP. On each level, the group leader forwards the command only to those nodes where the service needs to be deployed, rather than executing a complete flooding. The group leader does not propagate the command to the nodes underneath itself if it is not selected to deploy the service.

As long as a service remains active, the nodes that deployed it have to keep state because the command from the top node is sent only once. This allows the service to be redeployed automatically when significant changes occur. The SSAP defines what is considered a significant change. To avoid redeployment of a service whenever a routing change occurs, certain heuristics can be used during the dissemination of the command so that the service is deployed not only on a certain path but on a set of paths instead. These heuristics are left for further study.

Inspecting the top matrix  $M_1$  of Fig. 2 in our diff-serv++ example shows that  $m_{4,0}$  is different from zero, meaning that the transit network is capable of supporting the diff-serv++ service between the two customer sites. Therefore, the dissemination takes place, as shown in Fig. 3, where the shortest path is selected at each level of the hierarchy based on the transition matrices provided by the lower level. The nodes on the path selected then forward the command downwards, and the same process repeats itself. The edges of the shortest path are shown as thicker lines at each level.

### Step 4: Installation

The command ultimately reaches the bottom level of the hierarchy, where the physical nodes are. Here, the service is installed independently on each node. The installation and a part of the configuration that take place at this moment are automated: for instance, a link to the code located in a repository is

provided in the SSAP together with configuration parameters. Conversely, if the code required for the service is reasonably small, it can be contained in the dissemination messages.

Work related to automating the installation of code, more specifically the use of script MIBs to automate installation and configuration of a newer version of a protocol, has for instance been described in [10].

In general, the specific code destined to each individual node to run a certain service in the network can vary depending on the capabilities proper to each node. The solicitation step has already verified that it is possible to run the service, whereas this step will now install the proper code.

In our example, the diff-serv++ code is loaded into each router selected.

### Step 5: Advertisement

Once installed, a service will either have to discover neighbors with which it exchanges control messages (explicit addressing) or be advertised so that data packets are routed towards their destination over nodes enabled with the service (implicit addressing). This configuration problem is not specific to our mechanism. So far it has mainly been addressed by manual operations. For instance, BGP routers need to be manually configured with the IP addresses of their peers.

In our example of an implicit-addressing service, routers advertise their installed diff-serv++ service, and this information is again summarized and distributed. Here, the transition matrices advertised in this step are identical to those created in the summarization step, as there is only one path between the two customer sites that can support the diff-serv++ service. Because routing needs to be aware of such services (we assume the nontrivial case where the service is not installed in all nodes), this summarized information has to be combined with routing to provide the appropriate routes. Note that having the service-deployment hierarchy aligned with the routing hierarchy will significantly ease the advertisement procedure of implicit-addressing services.

In [11], we compared the various automatic discovery techniques for explicit-addressing control services, and proposed to use PAR (PNNI Augmented Routing [12]) to advertise such services within a ATM-PNNI environment, as PAR is more robust and scalable than centralized directory services.

## III. EXAMPLES

### A. Hierarchical IP Routing

In this example, we show how the service-deployment hierarchy can benefit from an existing routing hierarchy such as PNNI. We illustrate this with the automated deployment of an IP routing service, composed of three hierarchy levels. This

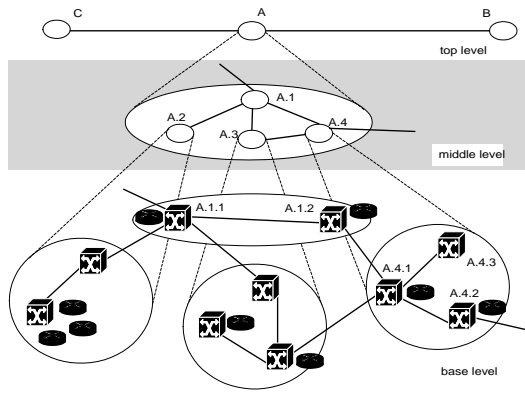


Fig. 4. The service hierarchy.

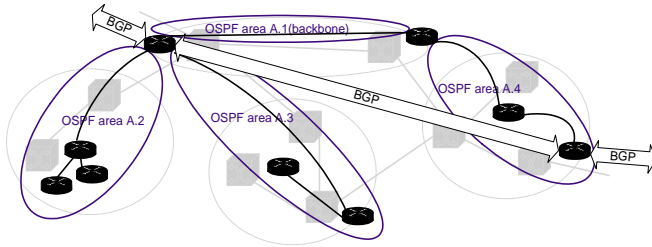


Fig. 5. A possible resulting IP routing hierarchy.

service is an explicit-addressing out-of-band control service. It is deployed over an existing ATM-PNNI network composed of ATM switches, to which IP routers are connected. The IP routing hierarchy will match the service hierarchy chosen here (which maps exactly to the PNNI hierarchy), therefore, automatically installing BGP and OSPF services at the appropriate layers. Note that from the viewpoint of IP the ATM network is considered as Non-Broadcast Multiple Access (NMBA), therefore each router could possibly peer with all other routers. Such a full-mesh is clearly not desirable because of scalability concerns.

Fig. 4 shows the service hierarchy, matching the PNNI routing hierarchy. It has a base, a middle and a top level. We skip the solicitation and summarization steps, and concentrate on the dissemination step, where a part of the auto-configuration of the service is performed. For instance, based on the topology connecting A.1, A.2, A.3, and A.4, the SSAP can choose to use A.1 as the OSPF backbone area. Then, in each OSPF area, routers interconnect with other routers in the same area, based on topology optimization methods such as described in [13]. The advertisements from all OSPF routers are scoped to their respective groups (their scope is limited to the base level), with the exception of the border-area routers of the backbone area, which have to be advertised into all other areas (their scope is limited to the middle level) [14, 15].

The routers chosen to run the BGP service will be advertised up to the top level, so that they can be seen from their peers in the B and C groups, as well as internally in A.

Fig. 5 shows a possible resulting IP routing hierarchy, with the underlying ATM network in gray. The Autonomous System is composed of four OSPF areas and two BGP speakers.

### B. Transparent Hierarchical Proxy Cache

An example of an explicit-addressing in-band service deployed at selected nodes only is a hierarchical transparent web proxy cache. Being transparent, HTTP clients need not be

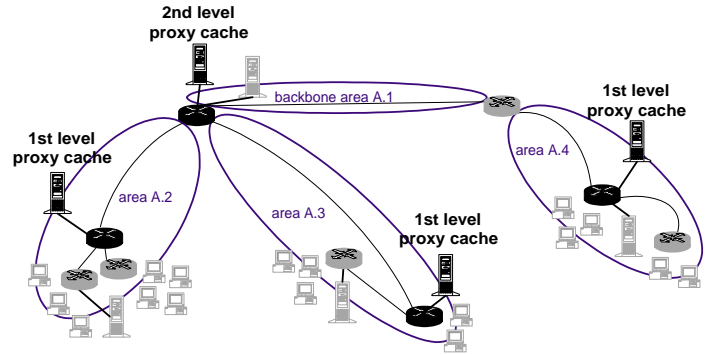


Fig. 6. Deployment of a transparent two-level proxy cache.

configured with the address of their proxy cache. Instead, the routers perform layer-4 switching and redirect HTTP requests to the proxy caches. To improve response time, two layers of caches can be used: the first-level cache contains the pages requested most often by the local user group, whereas the second-level cache contains the pages most often requested by all user groups.

We assume that an adequate routing and addressing functionality is already in place in the network before deployment of the service takes place. During the solicitation step, the specific requirements to support the service are sent to the nodes. Here, they include minimum processing capacity and storage on the directly attached servers, and capability of the routers to perform layer-4 switching. In Fig. 6, the node representing A.1, the backbone area, will receive a solicitation with the requirements corresponding to a second-level cache, whereas the nodes representing A.2, A.3, and A.4 will receive the requirements for a first-level cache. The resulting metrics, which can take the form of a boolean, are then sent up the hierarchy and summarized for each group. The summarization procedure is a logical-OR of the results from each individual router. This is done for each node A.1 through A.4. These metrics can then be summarized again, using a logical-AND of the previous results, so that node A will be informed whether the desired service can be deployed over the current infrastructure. Note that this example shows custom summarization rules that vary at each level of the hierarchy. In Fig. 6, the routers and servers selected by the deployment procedure are shown in black.

The system is transparent to the users, hence, they do not need to receive advertisements from this service. However, the first-level caches have to forward cache misses to the second-level cache. Therefore, the second-level cache originates advertisements carrying its address. These advertisements are scoped within the Autonomous System, and directed to the first-level caches only rather than being flooded throughout the network.

### C. Virtual Private Network (VPN)

Here, we illustrate the use of virtual outside links. The diff-serv++ example applied to transit networks, and therefore no such virtual outside links were required. The VPN interconnects subnetworks within the same network, and requires certain encryption capabilities at the end points as well as a certain QoS. The desired QoS is signaled by a protocol such as RSVP, and the nodes along the VPN therefore have to support it. We assume that the end points of the VPN are set statically, as shown in Fig. 7 by the letters A, B, and C.

During the solicitation step, the SSAP includes addresses of the desired end points for the VPN so that the appropriate

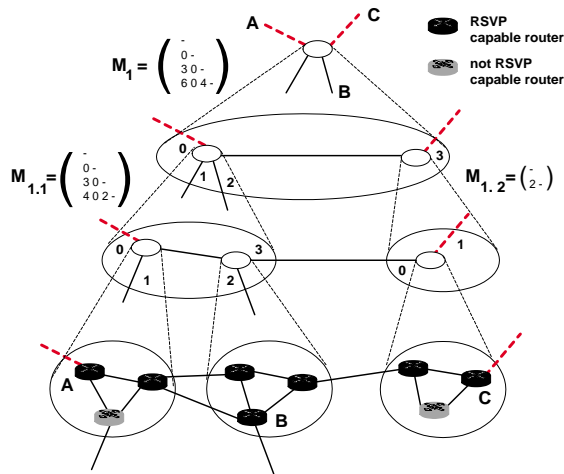


Fig. 7. VPN deployment using virtual outside links.

virtual outside links (in thick dashed lines) are created in the case of  $B$  and  $C$  during the summarization step. Otherwise, the top-level view of the hierarchy would be totally unaware whether it is possible to interconnect  $A$ ,  $B$ , and  $C$  with RSVP-capable routers. The transition matrices shown are similar to those in the diff-serv++ example. In matrix  $M_1$ , which is built out of  $M_{1,1}$  and  $M_{1,2}$ ,  $m_{2,0} = 3$ , which is the cost of the VPN between  $A$  and  $B$ . Similarly, the cost between  $B$  and  $C$  is  $m_{3,2} = 4$ , and between  $A$  and  $C$   $m_{3,0} = 6$ . The dissemination phase can then begin by choosing to deploy the VPN between  $A$  and  $B$ , and  $B$  and  $C$  so as to minimize the overall cost.

#### IV. APPLICABILITY

Various examples have revealed that a service hierarchy is required to keep the deployment scalable when targeting large numbers of nodes. Although the number of hierarchy levels is not limited by the mechanism, the resources in the network to maintain this hierarchy are bounded. ATM and IP networks commonly use three levels of hierarchy to aggregate routes. A fourth level of hierarchy already exists, although it is hidden from the routing protocols: distributed routers (clusters) appear as a single node with a single IP address from the outside. Extending the service hierarchy into such nodes can help automate the placement of functions within clusters.

We did not take link metrics explicitly into account when performing the service-deployment mechanism. The complexity of shortest-path routing with multiple metrics can be high or even become NP-complete, especially if multiple metrics, additive as well as restrictive, are used. Although the mechanism presented here allows a dynamic definition of the metrics to be used, this has to be balanced against the complexity that can arise if these metrics are not chosen appropriately.

By using generic metrics and generic summarization rules, we have shown that the solicitation step can be avoided, provided no virtual outside links are needed. If the network collects and summarizes these metrics by default, then deployment of a new service can start immediately with the dissemination step. If custom metrics and/or summarization rules are required, the solicitation step can be performed using a directed broadcast instead of a complete broadcast, provided the region(s) in the network where the service will have to be deployed can be identified.

Organizing the automated deployment of services is a key step towards the intelligent network infrastructure. Openness and programmability are beginning to appear in network equipment and, therefore, it is necessary to provide generic mechanisms that can enable the deployment of any type of service. In this paper, we have proposed a novel mechanism that addresses the deployment of a wide range of services within programmable and heterogeneous networks. The mechanism collects all capabilities available in the network and compares them with the requirements for a specific service, resulting in the creation of appropriate metrics. 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 and configuration are achieved, avoiding time-consuming and error-prone manual operations. Moreover, the mechanism is robust and capable of adapting to dynamic changes in the network, ensuring that a service remains available once deployed. Most importantly, when networks grow in size and heterogeneity, it 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.

In addition, it allows the experimentation involving a wide variety of new services outside the typical simplified testbed networks. 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 or DNS (Domain Name Service). The needs for a per-service specific service-deployment hierarchy are being evaluated against the cost of putting such hierarchies in place on-demand.

#### REFERENCES

- [1] Campbell, A.T., Kounavis, M.E., Vilella, D.A., Vicente, J., Miki, K., De Meer, H.G., and Kalaichelvan, K.S., "Spawning Networks", IEEE Network Magazine July/August 1999.
- [2] Brunner, M., and Stadler, R., "Virtual Active Networks - Safe and Flexible Environments for Customer-Managed Services", Proc. 10th IFIP/IEEE Int'l Workshop on Distributed Systems (DSOM'99), Zurich, Switzerland, October 1999.
- [3] Sivakumar, R., Han, S., and Bhargavan, V., "A Scalable Architecture for Active Networks", OPENARCH 2000, Tel-Aviv, Israel, March 2000.
- [4] Iliadis, I., and Scotton, P., "Transition Matrix Generation for Complex Node Representation", Proc. IEEE ATM Workshop'99, pp. 489-500, Kochi, Japan, May 1999.
- [5] Xin, T., and Xiao, D., "An Approach for Hierarchical Network Management Architecture Based on SNMPv2", Smartnet'99, The Fifth IFIP Conf. on Intelligence in Networks, Thailand, November 1999. Internet RFC 2805, April 2000.
- [6] Biswas, J., et al. "Application Programming Interfaces for Networks", IEEE PIN1520 Working Group Draft White Paper, www.ieee-pin.org.
- [7] CPIX Forum, "Common Programming Interfaces", www.cpixforum.org.
- [8] Awerbuch, B., Du, Y., and Shavitt, Y., "The Effect of Network Hierarchy Structure on Performance of ATM PNNI Hierarchical Routing", Computer Communications 23(10), May 2000.
- [9] Grillo, P., and Waldbusser, S., "Host Resources MIB", Internet RFC 1514, September 1993.
- [10] Quittek, J., and Kappler, C., "Practical Experiences with Script MIB applications", The Simple Times 7(2), www.simple-times.org, November 1999.
- [11] Haas, R., Droz, P., and Bauer, D., "PNNI Augmented Routing (PAR) and Proxy-PAR", Computer Networks 34(3), September 2000.
- [12] ATM Forum, "PNNI Augmented Routing (PAR) Version 1.0", af-ra-0104, January 1999.
- [13] Frelechoux, L., Osborne, M., and Haas, R., "Topology Optimization of IP over ATM", Proc. 1st IEEE European Conf. on Universal Multiservices Networks (ECUMN 2000), October 2000, Colmar, France.
- [14] Przygienda, T., Droz, P., and Haas, R., "OSPF over ATM and Proxy-PAR", Internet RFC 2844, May 2000.
- [15] Droz, P., and Przygienda, T., "Proxy-PAR", Internet RFC 2843, May 2000.