Integrated Service Deployment for Active Networks

Matthias Bossardt[1], Takashi Egawa[2], Hideki Otsuki[3], and Bernhard Plattner[1]*

[1] Computer Engineering and Networks Laboratory, ETH Zürich, Switzerland
{bossardt|plattner}@tik.ee.ethz.ch
[2] NEC Networking Laboratories, 4-1-1 Miyazaki, Miyamae-ku, Kanagawa, 216-8555 Japan
t-egawa@ct.jp.nec.com
[3] Communications Research Laboratory, 4-2-1 Nukui-Kitamachi, Koganei-shi, Tokyo, 184-8795 Japan
otsuki@crl.go.jp

Abstract A key feature of active networks is the capability to dynamically deploy services. In this paper, we present a scheme to classify service deployment mechanisms of existing or future active network architectures. Distributed algorithms (services), as being implemented in active networks, can be described based on active packets or as distributed programs running on active nodes. Although both programming models are basically equivalent, some services are more naturally implemented in either way. This paper proposes an active node architecture that supports the implementation and deployment of services according to both programming models. We point out that a combination of in-band and out-of-band service deployment is needed to dynamically deploy services implemented in either model. Furthermore, we argue that composing services from service logic implemented in either programming model is beneficial for the design of efficient and flexible services. We reason that a service abstraction in the form of a service description language is necessary to cope with real world scenarios.

1 Introduction

The difficulties to deploy new services in IP networks led to the development of active networks. Historically, active networking technology was based on an active packet (or "capsule") model, where packets — in addition to the ordinary payload — carry program code, which is executed in appropriate execution environments (EEs) of active nodes [15].

Active packets have some advantages that are hardly achieved by other technologies. One merit is that the processing of each packet can be programmed separately. Another merit is that a certain class of services is easily implemented and *deployed*. Examples of such services include different flavors of multicast, congestion avoidance algorithms and others. The main advantage of active packets is that they do not require any service deployment infrastructure.

Research, however, showed also that active packets are not practical for some type of services [1]. Examples include services that interact with many packet flows, as is the

* This work is partly funded by ETH Zürich, and Swiss BBW under grant number 99.0533.

J. Sterbenz et al. (Eds.): IWAN 2002, LNCS 2546, pp. 74–86, 2002.

case for firewalls filters, or long lived, usually rather complex, services that provide control functionality (e.g. routing daemons). Moreover, the expressiveness of active packet programming languages is often restricted due to security or performance concerns. Thus, there is a clear need to extend an active packet based network with dynamically deployable service components.

In this paper, we analyze service deployment mechanisms for active networks according to a novel classification scheme. Based on this analysis, two complementary service deployment schemes are selected, which we propose to integrate in order to overcome the limited expressiveness of active packets and to support more generic services. Furthermore, we motivate a service abstraction in the form of a service description language. As a result, a flexible active network architecture is obtained.

The following sections are organized as follows. In section 2, the design space of service deployment in active networks is analyzed and existing active network systems are classified. Section 3 discusses an integrated approach to service deployment and motivates a service abstraction in the form of a service description language. Finally, section 4 concludes the paper.

2 Exploring the Service Deployment Design Space

The potential of active networks to provide a wide variety of services must be supported by an appropriate service deployment architecture. In fact, active networks are unmatched in their flexibility to accommodate new services, because they allow programmability not only in the management and control planes, as is the case for programmable networks, but also in the data plane. In this section, we explore the design space of service deployment and discuss different approaches found in the literature.

2.1 Two Levels of Service Deployment

Service deployment in a network can be subdivided into two levels:

1. The network level where nodes that run service components are identified.
2. The node level where software must be deployed within the node environment.

A comprehensive framework must support service deployment at both levels. Due to the loose coupling between the two levels, the design of service deployment mechanisms at each level can be tackled to a great extent independently. The design space is similar for both levels (cf. Figure 1). It is essentially defined by two logically orthogonal axes being described in the following sections.

2.2 In-Band versus Out-of-Band Service Deployment

The x-axis of the service deployment design space defines the way service logic is associated with certain nodes and distributed in the network. The designer has the choice between in-band and out-of-band service deployment. In-band deployment refers to a system, where the service logic is distributed in the same way as payload data. Out-of-band deployment, on the other hand, refers to an architecture where service deployment and payload data use logically and/or physically distinct communication channels. That is, service deployment information is exchanged via the control or management plane.

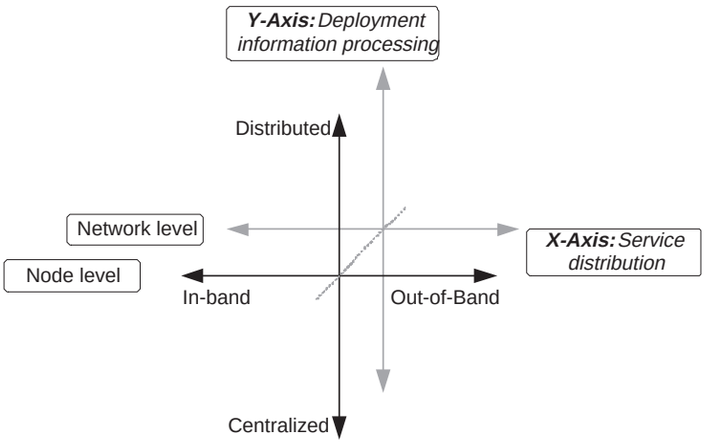

Fig. 1. Design space of service deployment frameworks

2.2.1 Network Level. Service deployment at the network level consists of identifying appropriate nodes that match the service requirements and are eligible to execute service logic.

In an in-band approach, service deployment is inextricably linked to the way packets are routed through the network. Only nodes a packet is routed to are eligible to execute service logic. In such a scheme, however, it may be possible to influence the location of service logic by executing/selecting different (e.g. service-specific) forwarding rules on the nodes. The main motivation for an in-band approach is that no specific service deployment infrastructure is needed. Active packets are a typical representative of network-level in-band service deployment.

In an out-of-band approach, on the other hand, node capabilities and topology information is gathered via management or control communication channels. Once the suitable nodes are identified, the same channels are used to allocate network level resources and to trigger the node level service deployment. Existing approaches to network-level out-of-band deployment include [14], as discussed in section 2.4.

2.2.2 Node Level. Service deployment at the node level consists of installing and configuring the appropriate service logic on a node.

An in-band approach combines service logic and payload data in the same *active* packet. The service logic is executed in adequate execution environments (subject to availability) on each node the packet traverses.

An out-of-band approach retrieves the service logic from some code server or cache. Packets carrying payload data do not include the service logic. It is, however, possible that such packets carry references to service logic, which is subsequently installed on the node.

Some existing systems use active packets for in-band node-level deployment [8, 7, 13]. Other systems, such as [3, 6], apply active packets to out-of-band node-level deployment.

2.3 Distributed versus Centralized Service Deployment

The y-axis of the service deployment design space distinguishes between centralized and distributed service deployment mechanisms. In this context, the design choice refers to the method of deployment information processing.

2.3.1 Network Level. Service deployment at the network level includes identifying suitable nodes that match service requirements.

A strictly centralized approach collects status and configuration information of nodes and topology information of the network at a central location, e.g. a network management station, and selects nodes that are to execute service logic.

In large networks, or when the network topology is highly dynamic — as is the case in ad-hoc networks — a distributed approach is required to cope with the complexity of the service deployment process. A distributed scheme may work similar to hierarchical routing algorithms. It extends the latter by more generic information exchange and aggregation, which includes not only link state information, but also node capabilities relevant to service deployment. Another approach to distributed network level service deployment may be based on mobile agents that cooperate in order to find a set of suitable nodes in the network. At the network level, active packets can also be classified as a distributed service deployment method because forwarding rules in active networks usually depend on node local decisions. An exception would be if some source-routing type of forwarding were used in the active packets.

2.3.2 Node Level. Deploying services at the node level consists of selecting service logic that match node capabilities. Capabilities of nodes include the offered set of EEs and node-resident services, such as inter-EE communication facilities. At the node level, code modules must be installed and configured such as to correctly provide the specified functionality.

In a centralized approach this task is done by an entity similar to the network management station, which is also used for network level deployment. Another centralized approach to node level deployment is the case where a sender generates an active packet including code or direct references to code that can not be modified as the packet traverses the network.

In a distributed approach each node involved in a particular service performs the node level deployment separately. Such an approach may use a node independent service description, specifying the functionality to be implemented on a particular node. A distributed approach is also possible with active packets. That is, an active packet may modify its code or references to code based on some node state as it traverses the network. As a consequence the service logic that is finally deployed at a specific node depends on (distributed) state information.

At first sight, this dimension of the service deployment design space could also be considered as an early versus late binding issue. Using a node independent service description could also be classified as late binding, whereas including service code in active packets may be considered as early binding. In the case of active packets that are able to modify their own code, however, this classification would no longer be applicable. Therefore, we prefer the classify node level approaches into centralized and distributed.

2.4 Discussion of Existing Active Network Systems

In this section, we discuss existing active network systems from a service deployment perspective and classify them within the presented design space. The list of an analyzed systems is not exhaustive. The systems were chosen in a way to get a sampling of different design decisions. Table 1 summarizes our analysis, the details of which can be found in the following paragraphs. We observe that active networks typically use a distributed, in-band approach at the network level. Greater variety can be found in the approaches for the node level. The chosen approach depends on targeted services, as well as performance and security considerations.

	network level	node level
ANN	in-band, distributed	out-of-band, either$^{\alpha}$
ANTS/PAN	in-band, distributed	out-of-band, either$^{\alpha}$
Chameleon	N/A	out-of-band, distributed
HIGCS	out-of-band, distributed	N/A
PLANet/SwitchWare	in-band, distributed	either$^{\beta}$, either$^{\alpha}$
Smart Packets	in-band, distributed	in-band, centralized
Stream Code	in-band, distributed	in-band, either$^{\alpha}$

α : Choice between centralized and distributed is service specific
β : Choice between in-band and out-of-band is service specific

Table 1. Classification of service deployment in existing systems

2.4.1 ANN/Router Plugins. The *Active Network Node (ANN)* [3] architecture makes use of active packets to deploy services. These packets feature a reference to a *router plugin*, which contains the service logic. If not cached locally on a node, router plugins are fetched from a code server and installed on the node.

Using active packets, the service deployment mechanism of this system can be classified as a *distributed, in-band* approach at the network level. It is distributed because the service logic is installed on active network nodes traversed by active packets. Whether an active network node is traversed or not depends on the forwarding tables in the nodes, which are set up by distributed routing algorithms. It is in-band because necessary information is contained in active packets, which also carry payload data.

At the node level, an out-of-band mechanism is used. That is, router plugins are fetched from a code server, using a different logical communication channel. As plugins

may modify all fields of active packets, the choice between a centralized or distributed approach is left to the service designer, as explained in section 2.3.2.

2.4.2 ANTS/PAN. As far as service deployment is concerned, *ANTS* [6] and *PAN* [10] use a similar architecture. From a service deployment viewpoint, ANTS is similar to ANN. It also uses active packets, which contain a references to *code groups*, i.e. the service logic.

At the network level, the same comments as for ANN apply. At the node level, however, it is interesting to note that while using an out-of-band approach, ANTS efficiently mimics an in-band deployment mechanism. That is, if an active packet arrives at a node, service logic is — if not cached locally — retrieved from the cache of the previously visited node. Therefore, active packets and service logic generally follow the same path. The out-of-band approach, however, allows for an efficient use of network bandwidth because the service logic follows the first active packet of a stream. Subsequent active packets of the same stream will make use of the cached service logic. Therefore it is not necessary to transmit the service logic with each active packet. As in the case of ANN, the choice between a centralized or a distributed approach is left to the service designer.

2.4.3 Chameleon. *Chameleon* [11, 12] is a service deployment framework for the node level. The main goal of Chameleon is to support heterogeneous networks, that is a network may feature different flavors of active nodes. Therefore, it is not useful that active nodes get the service logic or a direct reference to it, because they are possibly lacking an adequate execution environment. Chameleon features a component-based service model. The *service creation engine* composes the service logic based on node independent service descriptions on each node separately. The components of the resulting service logic are dynamically loaded from code servers or caches. The number and type of components to be installed depends on node capabilities and types of available execution environments.

Chameleon does not describe the network level of service deployment. It may be combined with different existing approaches, e.g. a modified version of HIGCS (see 2.4.4) or using active packets containing a reference to a node independent service description.

At the node level, Chameleon represents a *distributed, out-of-band* approach to service deployment. It is obviously out-of-band as the service logic is dynamically loaded from code servers. More important is the fact that it is a distributed approach. Since each active node performs the node level deployment autonomously, service components that match node capabilities and take advantage of specific node features can be selected. The selection of components is constrained by node independent service descriptions.

2.4.4 HIGCS. In [14], the authors describe a network level service deployment framework for programmable networks. Although not targeted to active networks, a modified version of the main contribution of this work, the *Hierachical Iterative Gather-Compute-Scatter (HIGCS)* algorithm, may be applied to active networks as well. HIGCS allows to match node capabilities against service requirements, resulting in a set of appropriate nodes that are subsequently configured to implement the service.

HIGCS uses a *distributed, out-of-band* approach at the network level. Similar to hierarchical routing schemes, nodes build clusters and elect a cluster leader to aggregate information (e.g. node capabilities) and to represent it to the upper hierarchy level. The exchange is based on a specific control protocol (e.g. an extension to a hierarchical routing protocol) and therefore an out-of-band approach. It is distributed because cluster leaders process (aggregate, distribute) information relevant to service deployment within their cluster.

As HIGCS is intended for the network level deployment, a discussion of the node level service deployment is not applicable.

2.4.5 PLANet/SwitchWare. *SwitchWare* [8] is an architecture that combines active packets with *active extensions* to define networked services. Active packets contain code that may call functions provided by active extensions. Active extensions may be dynamically loaded onto the active node. *PLANet* implements this architecture using a safe, resource-bounded scripting language (called PLAN [4]) in the active packets and a more expressive language to implement active extensions.

At the network level, service deployment is implemented in a *distributed, in-band* way, using active packets similar to ANTS and ANN.

An interesting service deployment characteristic of the SwitchWare architecture is found at the node level. In fact, both in-band and out-of-band service deployment is used to combine the advantages of both worlds. Active packets contain code (in-band) that can be used like a glue to combine services offered by dynamically deployed active extensions (out-of-band). Similar to ANN and ANTS, the content of active packets may be modified by active extensions. Therefore, the choice between a centralized and distributed way of deployment is left to the service designer.

2.4.6 Smart Packets. *Smart Packets* [7] are active packets that contain code in a compact, machine-independent representation. *Sprocket*, the language Smart Packets are programmed in, is targeted to the management plane.

At the network level, Smart Packets use a *distributed, in-band* approach to service deployment. It is distributed because it makes exclusive use of standard IP forwarding at the network level. There is no central entity that determines on which nodes the service will be deployed (i.e. Smart Packets will be evaluated). It is in-band because the same mechanism (IP forwarding) is used for both packet forwarding and selection of nodes to run the service logic.

As opposed to ANTS and ANN, Smart Packets include the service logic in the active packet, not only a reference to it. As a consequence, they are a typical in-band service deployment representative at the node level. From an efficiency view point this makes sense because the management plane is targeted where no streams of packets are expected. That is, Smart Packet have the characteristics of agents that roam through the network and mainly interact with the Management Information Base (MIB) of the visited active nodes. The node level service deployment is organized in a centralized way: the originating node inserts the code to be executed on the active nodes in the active packet. Modification of this code is, to the best of our knowledge, not possible.

2.4.7 StreamCode. *StreamCode* [13] uses a hardware-decodable instruction set for describing the service logic in active packets. It has a StreamCode Execution Environment (SC-EE) where StreamCode programs are executed. This environment aims at high performance, and is basically designed for the data path. Since functions available in this environment are limited by program size restrictions and computational time bounds, complex services cannot be executed. Such services are provided in a EE, where long lived programs, such as routing daemons, are executed. EE programs may write their results into special area of the SC-EE. SC-EE programs may control themselves by reading out the result in the area.

At the network level, this system is a representant of a *distributed, in-band* approach to service deployment, with similar service deployment characteristics as other typical active packet based systems.

At the node level, we classify StreamCode as an *in-band* approach. It is in-band because the service logic is embedded in the packet. Active packets can not be directly modified, but they can be dropped and new packets may be generated by a service component in the EE. Hence, the choice between centralized and distributed deployment is left to service designer.

3 Integrated Service Deployment

In this section, we reason about the conceptual benefits of an integrated approach to service deployment. That is, why it is useful to use both in- and out-of-band service deployment in active networks. Furthermore, we present our approach to a hybrid system, which features an abstraction of service implementations, i.e. a service description language. We argue that the description language is necessary to cope with the requirements of real world scenarios.

3.1 Why Combining In- and Out-of-Band Service Deployment?

To better understand the benefits of a combined approach to service deployment, it is important to note that there are two equivalent models to program a distributed system, as observed by Wall in [2]. First, a *component based* model (sometimes also referred to as *discrete*) where service logic is installed on network nodes and exchange messages to be processed by service logic on other nodes. Second, an *active messages* based model where service logic moves from node to node and executes on the visited nodes. Wall observed that for a specific service (or distributed algorithm) one of those models is usually preferable over the other, because it leads to a more natural implementation of the algorithm. As a consequence, he suggested the development of a hybrid system that would naturally combine both programming models[1].

Being a generic platform for the implementation of distributed algorithms, active network nodes should — for the same reasons — support both programming models.

[1] In Wall's system, moving code exists as a concept only. In an implementation, he suggests to simulate moving code by pre-installing the code on all nodes and let the messages — similarly to ANN and ANTS — carry a reference to the code.

In fact, active networks can be considered as an architecture, which extends Wall's system — a combination of *active messages*[2] and component based logic — with dynamic service deployment. To support both programming models in active networks, a combination of in- and out-of-band service deployment is necessary.

On hybrid nodes, integrated service deployment may be performed in several ways. Firstly, embedded in the program, active packets may contain references to a service descriptor, which describes service logic to be installed out-of-band. That is, from a network level perspective, the service is deployed in-band. This deployment method is preferable if a maximum of flexibility for service definition is to be left to the end-user. Secondly, nodes are contacted by a management system requesting the deployment of service logic in a set of active nodes. In this scenario, network level deployment is performed out-of-band, which is advantageous if a network provider wants to keep full control over the network.

Architectures such as ANN, ANTS and SwitchWare, use a combination of in- and out-of-band service deployment, where, at the network level, in-band deployment is implemented with active packets and, at the node level, out-of-band deployment with some type of code components: *(*router plugins) in ANN, *code groups* in ANTS, and *active extensions* in SwitchWare. Having a closer look at those systems unveils some interesting aspects of combined approaches.

In ANN and ANTS, a combined approach is motivated by a gain in **efficiency**. More precisely, it enables code caching and, as a result, avoids latency for the installation of code and waste of bandwidth. That is, active packets contain references to code, which is retrieved from another cache or code server and cached locally. For subsequent packets containing the same references the locally available code can be used.

In SwitchWare, the main motivation is to allow for **secure extensibility** of the active packet programming language. That is, users are able to easily introduce new services in the network by programming active packets (in-band), whereas network-operators are enabled to extend the functionality of the packet programming language in a controlled, out-of-band way. Active packet programs are untrusted, but written in a restricted and safe language, while the components installed out-of-band are trusted, after having undergone some security checks.

In the next section, we propose an hybrid approach that combines in- and out-of-band deployment in a more generic way.

3.2 Integrated Service Deployment for Hybrid Systems

In this section, we argue that service deployment for hybrid systems has not been taken care of in a way that allows using them in real world scenarios. We illustrate an integrated approach to service deployment with our hybrid system based on Chameleon [12] and StreamCode [13].

[2] In the area of active networks, the term "active packets" is common to refer to a similar concept as Wall's "active messages". Henceforth, we will use active packets.

3.2.1 The Chameleon/StreamCode Hybrid. In our hybrid system *Chameleon* provides out-of-band service deployment, including a service description language dedicated to active networks. *StreamCode*, on the other hand, provides the means necessary for in-band deployment, as discussed in section 2.4.

The Chameleon/StreamCode hybrid node (cf. Figure 2) is characterized by at least two EEs: an active packet EE and a component-based EE.

EEs are usually targeted to a certain type of functionality. As shown in section 2.4, some EEs are optimized for the management plane, whereas others are adapted for the control or even data plane. There is generally a tradeoff between the abstraction level (or richness) of the API and the performance provided by an EE. Since the requirements for an EE vary widely from plane to plane, there is no single EE design that is superior to others for all interesting services. As a consequence, our active node must provide more than one EE, if it is to support the whole spectrum of services.

In our active node implementation, the StreamCode-EE (SC-EE) plays the role of an active packet EE. The component based EE uses Java as the underlying technology. Those two EEs are required for the separation of data and control/management plane. Communication between EEs is possible through a reserved area of shared memory. In particular, StreamCode programs may access results of complex computations of service logic in the component-based EE. Furthermore, the component based EE may generate StreamCode packets to communicate with the SC-EE. In this way the computations in the data plane are decoupled from the processing in the control/management plane.

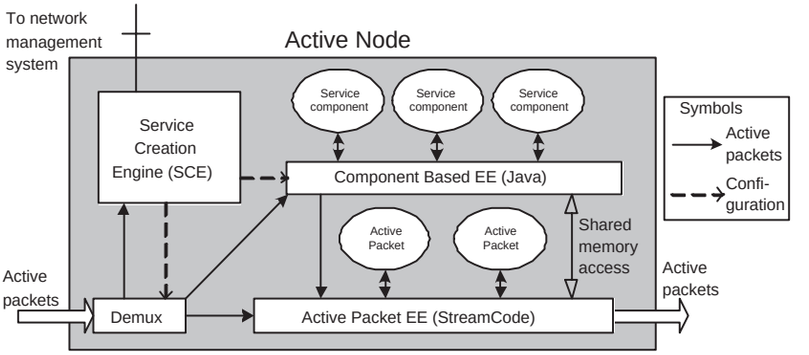

Fig. 2. Functional architecture of a Chameleon/StreamCode hybrid node.

Up to this point, our node is, from an architectural viewpoint, similar to ANN or SwitchWare nodes. There are, however, two distinctive architectural differences.

Firstly, Chameleon provides means to abstract service functionality from a service implementation. This abstraction comes in the form of a service description language.

Secondly, Chameleon/StreamCode nodes feature a *service creation engine* (SCE). The SCE is able to interpret *node independent* service descriptors and, based on this, to compose a service implementation that is *dependent* on the node environment.

An important consequence of the abstraction provided by the service descriptors and the node local mapping process (provided by the SCE) is the ability to deal with heterogeneous active nodes. That is, with active nodes running sets of EEs that may vary from node to node. Another benefit of the proposed architecture is the ability to compose services from components, which, in turn, enables code reuse.

We describe these features and their benefits in the following paragraphs.

3.2.2 Active Network Service Description Language (ANSDL). Particular service logic may be implemented in different ways, depending basically on the set of EEs available on a specific node. The role of the ANSDL is to abstract service functionality from its implementation. The main goal is to provide a description of service logic in a node independent way.

In [12], we proposed an XML-based language to address this issue. The details of the language are beyond the scope of this paper. We provide, however, a short overview.

The ANSDL centres around two main abstractions: *containers* and *connectors*. Containers abstract service logic as a black box with interfaces. Connectors abstract the binding among containers. Containers can be composed of other (sub-)containers bound to each other via connectors. We assume that container names capture the functionality of the service logic.

There are two types of service description documents, both include a container name and number and types of interfaces. Documents of the first type contain information about sub-containers — described in other service descriptors — and their interconnection. Documents of the second type contain a reference to service logic and the type of EE the logic is to be executed in.

The connectors act as an abstraction of the binding between containers. In this way, it is possible to abstract protocols describing the interaction between service components. An active packet language, such as StreamCode or Plan, complements this description with the possibility for dynamical binding. In the Chameleon/StreamCode hybrid node, for example, the connectors between SC-EE and components in the Java-EE are mapped to shared memory. StreamCode programs allow to access the results of computations in the Java-EE in a flexible way.

Clearly, a particular description language must be supported by all active nodes. Therefore, it is subject to standardization. Because of the practical difficulties to agree on a standard node environment, we believe it is much easier and more reasonable to standardize a description language.

3.2.3 Service Creation Engine (SCE). Each active node is responsible for selecting suitable service logic by matching the container's requirements (attributes in the descriptors) against node capabilities. During the matching process, descriptors of subcontainers are interpreted until all descriptors contain direct references to service logic (e.g. router plugins, code groups, active packet code etc.). This mapping task is per-

formed by the SCE, which is available on each node. The result depends on the node environment (available EEs, node-resident services, etc.)

4 Conclusion

In this paper, we explored the design space of service deployment in active networks and proposed a new scheme to classify different approaches according to their 1) service distribution mechanism and their 2) method of deployment information processing. The proposed scheme was then applied to a number of service deployment approaches described in the literature. As to our knowledge, there is no previous work that explores the design space of service deployment in active networks in a systematic way.

Moreover, we presented an approach showing how active packet based networks can be integrated with an out-of-band service deployment framework. We argued that both active packet based networks — which are a typical realization of in-band service deployment — and component based technology, using out-of-band service deployment have advantages and shortcomings. Fortunately, they are rather complementary. Hence, integration is beneficial as it results in a highly flexible, though efficient network architecture. Furthermore, we described our own hybrid approach to service deployment that can deal with heterogeneous active networks, because the service deployment framework is able to interpret abstract, i.e. node independent, service descriptions.

5 Acknowledgments

We would like to thank Michael Hicks and the anonymous reviewers for the detailed and insightful comments on an earlier version of this paper.

References

[1] D. Wetherall. Active network vision and reality: lessons from a capsule-based system. In Symposium on Operating System Principles (SOSP'99), December 1999.

[2] D.W. Wall. Messages as Active Agents. In ACM Symposium on Principles of Programming Languages (POPL), Albuquerque, New Mexico, January 1982.

[3] Dan Decasper, Guru Parulkar, Choi, S., DeHart, J., Wolf, T., Plattner, B., A Scalable, High Performance Active Network Node, IEEE Network, Vol. 13(1), 1999.

[4] M. Hicks, P. Kakkar, J.T. Moore, C.A. Gunter and S. Nettles. PLAN: A Packet Language for Active Networks. In ACM SIGPLAN International Conference on Functional Programming Languages, 1998.

[5] M. Hicks, J.T. Moore, D.S. Alexander, C.A. Gunter, S.M. Nettles. PLANet: An Active Internetwork. In IEEE Infocom Conference, 1999, New York, USA.

[6] D. J. Wetherall, J. V. Guttag, D. L. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In IEEE Openarch'98, San Francisco, USA, April 1998.

[7] B. Schwartz, A.W. Jackson, W.T. Strayer, W. Zhou, R.D. Rockwell, C. Partridge. Smart Packets for Active Networks. ACM Transactions on Computer Systems, Vol. 18(1), February 2000.

[8] D.S. Alexander, W.A. Arbaugh, M.W. Hicks, P. Kakkar, A.D. Keromytis, J.T. Moore, C.A. Gunter, S.M. Nettles, J. M. Smith. The SwitchWare Active Network Architecture. IEEE Network Special Issue on Active and Controllable Networks, vol. 12 no. 3, pp. 29 - 36.
[9] D. Decasper, Z. Dittia, G. Parulkar, B. Plattner. Router Plugins - A Software Architecture for Next Generation Routers. IEEE/ACM Transactions on Networking, February 2000.
[10] E.L. Nygren, S.J. Garland, M.F. Kaashoek. PAN: A High-Performance Active Network Node Supporting Multiple Mobile Code Systems. In IEEE Openarch'99, March 1999.
[11] M. Bossardt, L. Ruf, R. Stadler, B. Plattner. Service Deployment on High Performance Active Network Nodes. In IEEE NOMS 2002, Florence, Italy, April 2002.
[12] M. Bossardt, L. Ruf, R. Stadler, B. Plattner: A Service Deployment Architecture for Heterogeneous Active Network Nodes. In 7th IFIP SmartNet 2002, Saariselkä, Finland, April 2002.
[13] T. Egawa, K. Hino and Y. Hasegawa. Fast and Secure Packet Processing Environment for Per-Packet QoS Customization. In IWAN 2001, September 2001.
[14] R. Haas, P. Droz, B. Stiller. Distributed Service Deployment over Programmable Networks. In DSOM 2001, Nancy, France, 2001.
[15] D.L. Tennenhouse, and D.J. Wetherall. Towards an Active Network Architecture. Computer Communication Review, Vol. 26, No. 2, April 1996.