Autonomic Service Architecture using Virtual Services

Ramy Farha[1], Myung Sup Kim[2], Alberto Leon-Garcia[3], and James Won-Ki Hong[4]

[1,2,3]Dept. of ECE, Univ. of Toronto, 40 St. George Street, Toronto, ON, Canada
[4]Dept. of CSE, POSTECH, Pohang, Korea

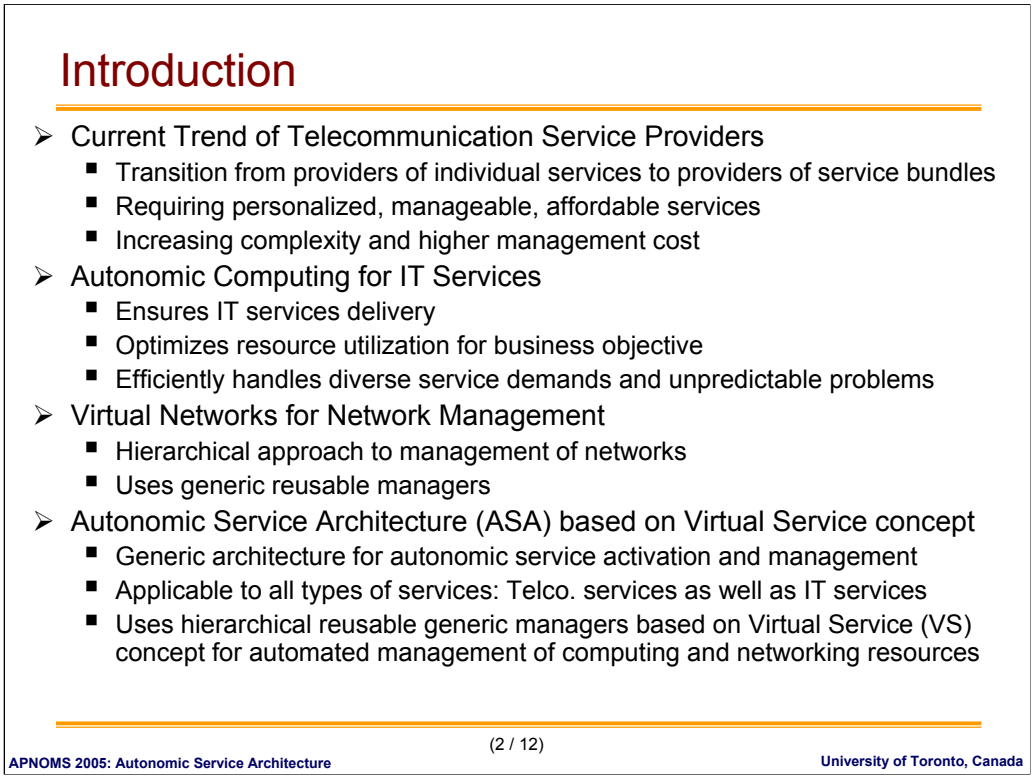
[1]rfarha@comm.utoronto.ca, [2]myungsup.kim@utoronto.ca,
[3]alberto.leongarcia@utoronto.ca, and [4]jwkhong@postech.ac.kr

Abstract

Traditional telecommunications Service Providers (SPs) are undergoing a transition to a shared infrastructure in which multiple services will be offered to customers. Currently, such service control and management techniques are mostly manual, and need human intervention, which leads to slow response times, high costs, and customer dissatisfaction. Efficient service control and management is a fundamental aspect of the promise of next-generation telecommunication services, which is still challenging. In this paper we propose an Autonomic Service Architecture (ASA) to address this challenge. A key feature of ASA is the automated management of computing and networking resources, providing self-management functionalities for multiple services in a service provider's domain. We developed ASA based on our Virtual Service (VS) concept which views services as a composition of a hierarchical structure of other VSs and underlying resources. ASA is composed of hierarchical autonomic service brokers (ARBs) managing those VSs. In addition, we focus on ASA's autonomic operations through the four major possible disturbances to an ARB's operation: Customer, SP, Services, and Resources. We also illustrate an example of ASA applied for the management of a Voice over IP (VoIP) service.

Keywords:

Autonomic Service Management, Self-management, Resource Management, Virtual Services, VoIP



Telephone and cable companies are reacting to competitive pressures by transitioning from being providers of individual services (voice, data, video…) to providers of service bundles. The aim of these bundles is to provide personalized, manageable, and affordable services. This strategy is motivated by the need to protect existing mature markets, e.g. wire-line telephone service or television distributions, and to participate in growing markets, e.g. cellular telephone or broadband access. The trend is to offer a real-time self-serve capability that allows customers and service providers (SPs) to mix-and-match from the existing and future services. This changing landscape presages an increase in the complexity of the service management and a jump in the maintenance costs of such service management approaches.

Currently, such service management techniques are mostly manual, needing regular human intervention. This leads to slow response times, high costs, and customer dissatisfaction. SPs face the challenge of reducing the costs and complexity of providing services. Efficient service management is a fundamental aspect of the promise of next-generation telecommunication services, which require a new service delivery framework that can exploit the capabilities of IP networks and provide the required service richness, agility and flexibility. From the information technology (IT) world, autonomic computing [1] is touted as the means to providing a rich set of IT services over a common computing infrastructure. A key feature of autonomic computing is the automated management of computing resources, providing self-management functionalities. The application of autonomic management principles to ensure the delivery of telecommunications services remains largely unexplored. From the networking world, Virtual Networks (VNs) [2] have emerged as possible solutions to the management of networks using a hierarchical generic manager approach. The application of the hierarchical generic manager principles to activate and manage different telecommunications services remains largely unexplored. In this paper, we propose an Autonomic Service Architecture (ASA), based on our Virtual Service (VS) concept, to address the need for autonomic service management.

The main contribution of this paper is to propose ASA, a generic architecture for autonomic service activation and management. ASA is applicable to all types of services, by proposing an autonomic generic approach. Such a generic approach is possible because of the introduction of the VS concept, which is another contribution of this paper. VS allows ASA to apply to all types of services.

The rest of this paper is structured as follows. First, we summarize the related work. Second, we describe the VS concept, which is composed of a hierarchical structure of services and resources. Third, we illustrate ASA based on the VS concept introduced previously. Fourth, we detail the generic managers forming ASA, called Autonomic Resource Brokers (ARBs), and elaborate on the autonomic properties of ASA. Fifth, we present an example of ASA applied to a real service, in this case Voice over IP (VoIP). Finally, we conclude this paper, and present our ongoing and future work.

Related Work

Related Work	Proposed ASA
IBM Autonomic Computing (AC) > Focuses on computing resources for IT services delivery > Define four aspect of self-management: Self-configuring, Self-optimizing, Self-healing, Self-protecting	expand to telecommunications services
Autonomic Communication > Focuses on individual network elements' functionalities. > Bottom-up Approach	consider computing and networking resources focus on services
Virtual Networks > Focuses on hierarchical generic managers for large scale networks	use a hierarchical architecture of generic Autonomic Resource Brokers (ARBs)
Autonomia > Provides dynamically programmable control and management	
Automate > Enables development of autonomic Grid applications	virtualize physical resources into quantifiable resource metrics
Oceano > Provides a prototype for a scaleable use of virtualized resources	propose generic architecture for all services
HP Adaptive Control > Provides an architecture for a service-oriented control system	

(3 / 12)

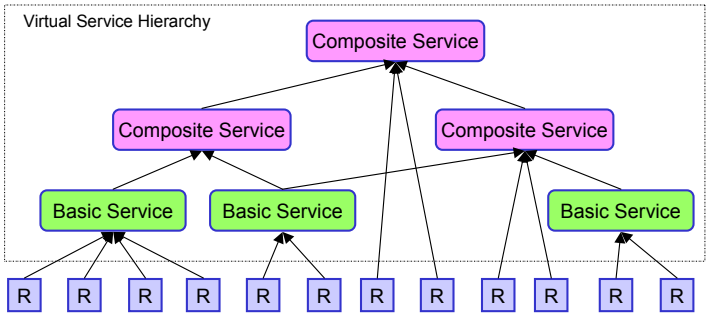
APNOMS 2005: Autonomic Service Architecture — University of Toronto, Canada

As mentioned in the previous slide, the first work in the new wave of autonomics has been the Autonomic Computing proposal by IBM [1]. However, IBM's approach exclusively focuses on computing resources for IT services delivery. Our work aims to expand this basic view to include telecommunications services, which consist of both computing and networking resources. Another proposal called Autonomic Communication [3] has similar aims to IBM's Autonomic Computing, except that it focuses on individual network elements, and studies how the desired element's behavior is learned, influenced or changed, and how it affects other elements. Our work is focused on services, and therefore is a top-down approach as compared to this bottom-up approach. Furthermore, we consider the interplay of both networking and computing resources to offer next-generation services. As mentioned previously, another related area is that of Virtual Networks [2, 4], defined as a hierarchical approach for the management of large scale networks, using a generic reusable manager. Our work expands the scope of Virtual Networks to that of Virtual Services, where all types of services are managed by generic managers called Autonomic Resource Brokers (ARBs), by virtualizing physical resources into well-defined metrics.

In addition, research projects such as Autonomia [5], Automate [6], and Oceano [7] are using the autonomic concept in various ways. Autonomia provides dynamically programmable control and management to support development and deployment of smart applications. Automate enables development of autonomic Grid applications that are context-aware, and capable of self-configuring, self-composing, and self-optimizing. Oceano is developing a prototype for a scaleable infrastructure to enable multi-enterprise hosting on a virtualized collection of hardware resources. Dynamic adaptability and computing resource allocations are used to ensure that customers' SLAs are met. The difference between our work and these approaches is that they consider particular services to which their design is appropriate. We propose a generic architecture which applies to all services in current and future networks. The closest work to this effort, is done by HP [8]. It proposes an architecture and some algorithms for a service-oriented adaptive control system that constantly re-evaluates system conditions and re-adjusts service placements and capacities. The control system is organized as an overlay topology with monitoring and actuation interfaces to underlying services. The major difference from our work is that, like the previous approaches, it is limited to a particular scenario, in this case an overlay topology of servers providing computing capabilities.

Virtual Service

- Service: engagement of resources for a period of time according to a contract (SLA) between customers and SPs
- Resource: physical and logical components forming service
- Players involved in the delivery of a service: Customers, SPs
- Virtual Service: Basic Service and Composite Service



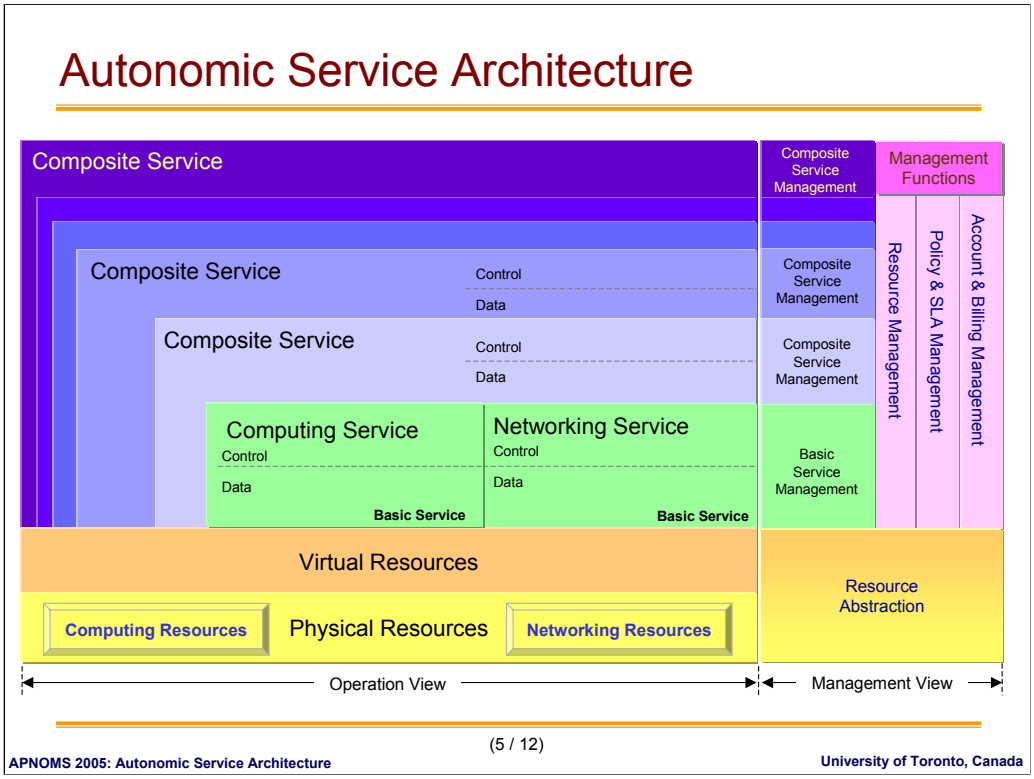
(4 / 12)

APNOMS 2005: Autonomic Service Architecture — University of Toronto, Canada

We define a service as the engagement of resources for a period of time according to a contractual relationship between the customers and the SPs, which the two players are involved in the service delivery process. We define a resource as components used to construct these services. These resources range from physical resources such as routers, switches, links, or servers, to logical resources such as overlays and peer-to-peer networks, or grids. When customers purchase a service from a SP, they can themselves offer this service to other customers, becoming SPs to those customers. We also define applications as services offered to the end-users, which use one or more services as the building blocks. Service Level Agreements (SLAs) are the contracts that govern the service delivery process. Service management mainly ensures that SLAs are met, and that the necessary resources are provided to each service.

These definitions are driven by our approach that "everything is a service", from complex multimedia services, to simple query/response. We identify two types of services however: Basic services, which cannot be broken down anymore into other services and mainly consist of the underlying physical resources, and Composite services, which are composed from the underlying resources, as well as the Basic and Composite services. In Composite services, the Basic and Composite services can be seen as logical resources. All services, whether Basic or Composite, are referred to as Virtual Services (VSs). Our service view is therefore a hierarchy of VSs. The creation of a VS is actually a composition process where Basic services and/or Composite services are used.

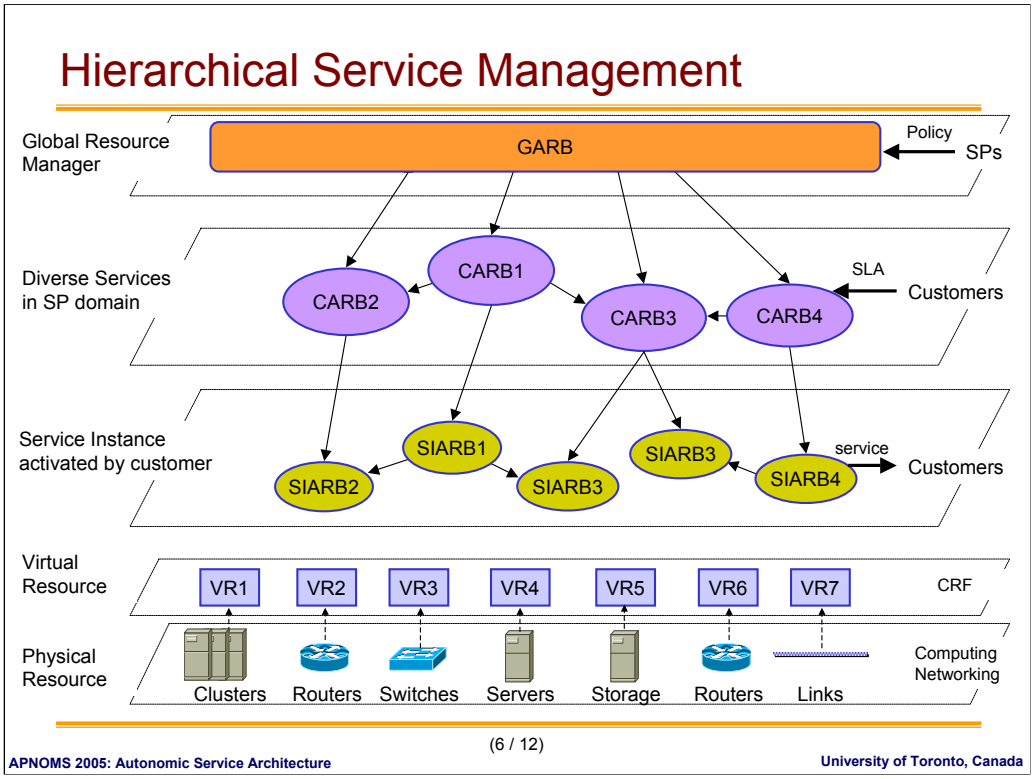
Fully automating the delivery of the service is un-realistic since some manual actions will be needed to complete delivery. In that regard, we divide the operations in the SP's domain to ensure service delivery into two categories: manual and autonomic. In the manual operation part, the basic composition of the service is involved, as well as manual adjustments. In the autonomic operation part, the runtime adjustment of the service is performed in a fully autonomic way. After customers and SPs agree on the SLA, ASA will manage the service in order to satisfy the SLAs without SP's intervention, unless the problems incurred are too complex to be handled by runtime autonomic adjustments, in which case manual adjustments are needed.



Traditionally, services are divided into three planes: Data, Control, and Management [9]. Data plane denotes functions involved in the delivery of information between endpoints. Control plane denotes functions involved in laying the ground for the delivery of information, such as signaling and session setup. Management plane denotes functions involved in managing the operation of a service, and in managing the networking and computing resources needed for the delivery of information delivery between endpoints. This involves, among others, resource allocation, policy changes, SLA monitoring, accounting and billing adjustments.

ASA unifies the view of different services in a layered view shown in the above slide, which corresponds to the VS hierarchy presented in the previous slide. The lowest layer consists of physical resources engaged in the delivery of the service. The middle layer consists of an abstraction of the physical resources into virtual resources according to metrics specifying the characteristics of the physical resources. The upper layers consist of VSs using these underlying resources. In those layers, the services offered to the end-users are referred to as applications. Vertically, the ASA architecture is broken down into two views: Operation and Management. In the Operation view, a VS consists of the control and data planes at the different layers of ASA, while in Management view, a VS consists of the service management functions needed.

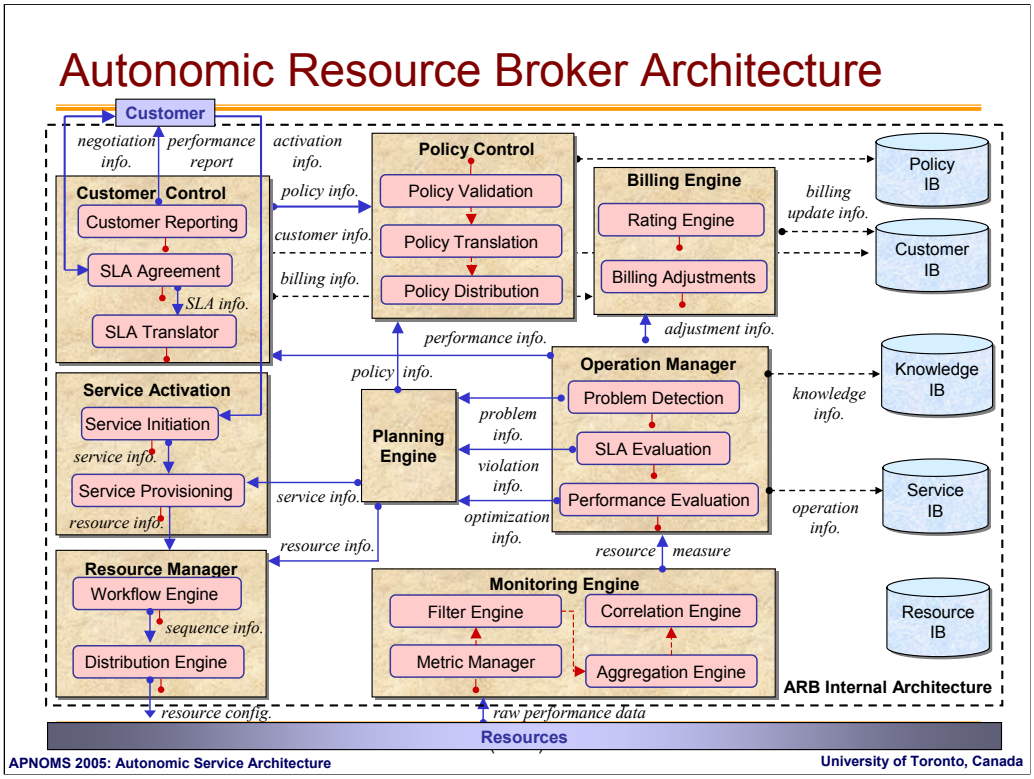
The Physical Resources layer consists of the physical resources that the SP has at its disposal. These resources are either computing and/or networking. The Virtual Resources layer consists of an abstraction of the physical resources into virtual resources. These virtual resources allow the SP to deal with resources at its disposal to create services independent of the actual physical resources. Every service views its needed resources in a unified format, called the Common Resource Format (CRF), which we define as an XML-based array of metrics needed by upper layer services. The motivation for CRF is that it translates heterogeneous physical resources into a common format, which allows ASA to be generic and service-independent. This virtualization depends on the types of resources involved, whether they consist of single computing or networking resources (routers, switches, links, servers), clustered resources (cluster of servers), and/or distributed resources (overlay network, grid). In some situations, VSs such as Virtual Networks [10], which are bought from other SPs according to SLAs, become logical resources at the disposal of the purchasing SP. The process of VS composition is hierarchical and recursive, and continues until the VS is offered to end-users. Each VS has its own independent management entity, which we call an Autonomic Resource Broker (ARB). ARBs are the key concept of ASA, and will be detailed next.



The main task of ASA consists of managing the resources available to the SP in order to meet fluctuations in service demands. Resource management is the major function of the ASA framework. In addition, the ASA framework should allow Policy & SLA management, as well as Accounting & Billing management. All these functions are performed by the Autonomic Resource Broker (ARB), a self-managing entity whose role is to ensure automated delivery of the services. The resources are therefore controlled and managed by ARBs, following existing policies and SLAs between customers and SPs. In the ASA framework, the Autonomic Resource Brokers (ARBs) are the analogy of autonomic managers, which implement a particular autonomic control loop (monitor, analyze, plan, and execute), control the managed objects, and finally ensure automated management of VSs. "Child" ARBs (for "Child" VSs) are created automatically by "Parent" ARBs (for "Parent" VSs) by allocating the appropriate amount of resources in accordance with SLAs. The resources are controlled and managed by ARBs according to SP policies and SLAs.

This slide illustrates a possible ARB hierarchy within a SP's domain. As mentioned previously, the physical resources are quantized using CRF. When customers activate new service instances, these service instances are managed by SIARBs (Service Instance ARBs). The multiple service instances of a particular service offered by a SP are managed by CARBs (Composite ARBs). In addition, some CARBs, called Basic CARBs, manage the virtual resources directly, or the basic services using those virtual resources. Others, called Composite CARBs, manage composite services. The different services offered by a SP are managed by a GARB (Global ARB), which handles all the resources available at the SP's disposal.

Initially, the GARB manages all resources at the SP's disposal. When the SP desires to introduce a new service, the GARB spawns a new CARB to manage this service, and spawns all CARBs needed to manage the VSs needed by this service. The "spawning" concept borrows from our previous work on Virtual Networks [2]. For this newly created service, the CARB spawns a new SIARB every time a customer activates an instance. Each instance also needs resource provisioning according to SLAs. More details on the operation of ARBs will be provided in the following slides. Note that our implementation will be based on a Service-Oriented Architecture (SOA) [10] for interactions between ARBs and underlying resources, in order to leverage existing and future management protocols (SNMP, DMTF, CIM, etc.) [11].



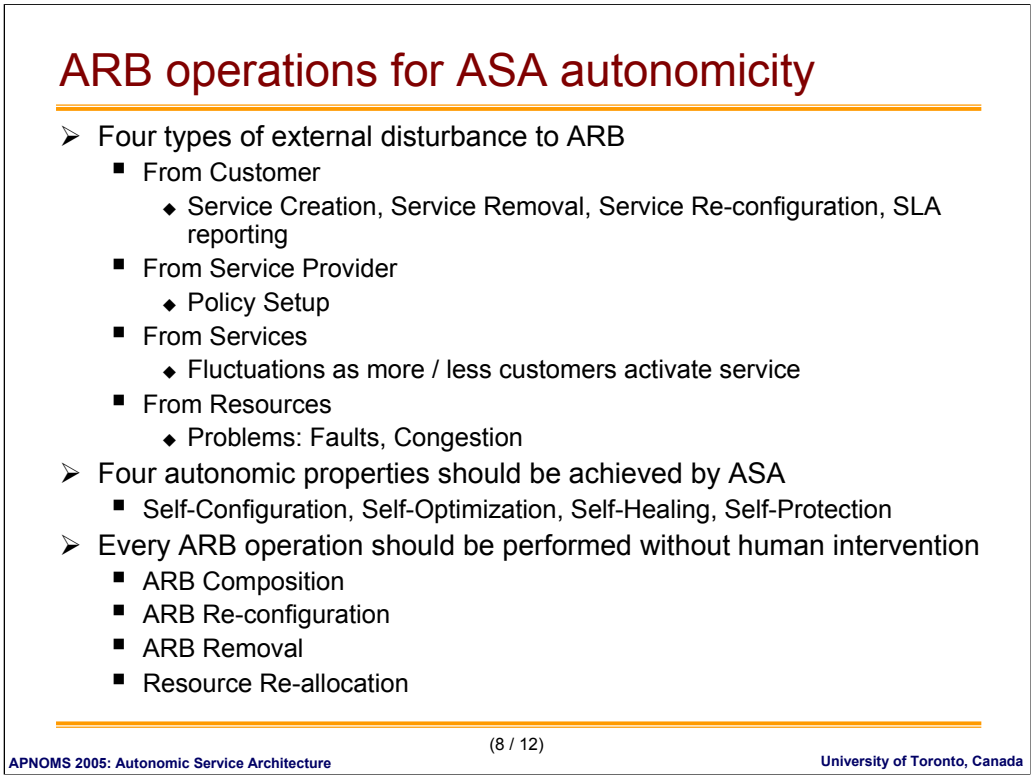
This slide illustrates the detailed architecture of ARB which is common in all types of ARBs and all types of services. Note that the ARB is a logical architecture. ARBs could be distributed, their components also be implemented in different locations, working together towards a common goal. Information Bases (IBs) are needed to store the information needed for the delivery of the services. Information Bases can be classified into five logical groupings: Customer IB (CIB) for information related to customers, Service IB (SIB) for information about the service instances activated by the customers, Resource IB (RIB) for information about the resources, Policy IB (PIB) for policies created at runtime or entered manually, and Knowledge IB (KIB) for information for use in case problems arise and remedy actions.

Several policies are created initially by SPs, or at runtime as a result of customers activating services through Policy Control block. Existing policies are updated as a result of service demand and load variations. ASA should allow different types of policies (If/Then, Goal, and Utility) to coexist [12]. Customer Control is in charge of two main functions: SLA Agreement to allow customers to activate services, and Customer Reporting to keep customers informed about performance. A SLA negotiation between customer and SP is performed through the Customer Control block. And we are currently defining a unified interface for the Customer Control block which can be applied all kinds of telecommunication services.

The SLA Initiation and the Service Provisioning components in the Service Activation block translate high-level SLA Agreements into objective function that optimizes the service provisioning process by choosing the VSs appropriately. A composed service is represented by a workflow of services using an appropriate language such as BPEL [13]. The Service information is sent to the Resource Manager so that the appropriate resource allocation can occur.

The Monitoring Engine collects raw performance data from underlying resources. The Operation Manager performs problem determination, SLA evaluation, and performance optimization. The Billing Engine bills customers and makes adjustments when need be, such as when the SLA is violated. The adjustment's amount depends on policies. The Planning Engine is the brain of the ARB, which determines the optimal resource allocation on the fly according to the kaleidoscope of resource state, and makes appropriate changes / updates to policies in order to keep ARB's operation as planned.

The ASA architecture are compliant to the autonomic features of IBM autonomic computing and extends its functionality of ASA to the business level. The Monitoring Engine, the Operational Manager, the Planning Engine, and the Resource Manager take the four functions of autonomic control loop (monitor, analyze, plan, and execute). The Customer Control, the Service Activation, and the Resource Manager take charge of the SLA negotiation and service provisioning.



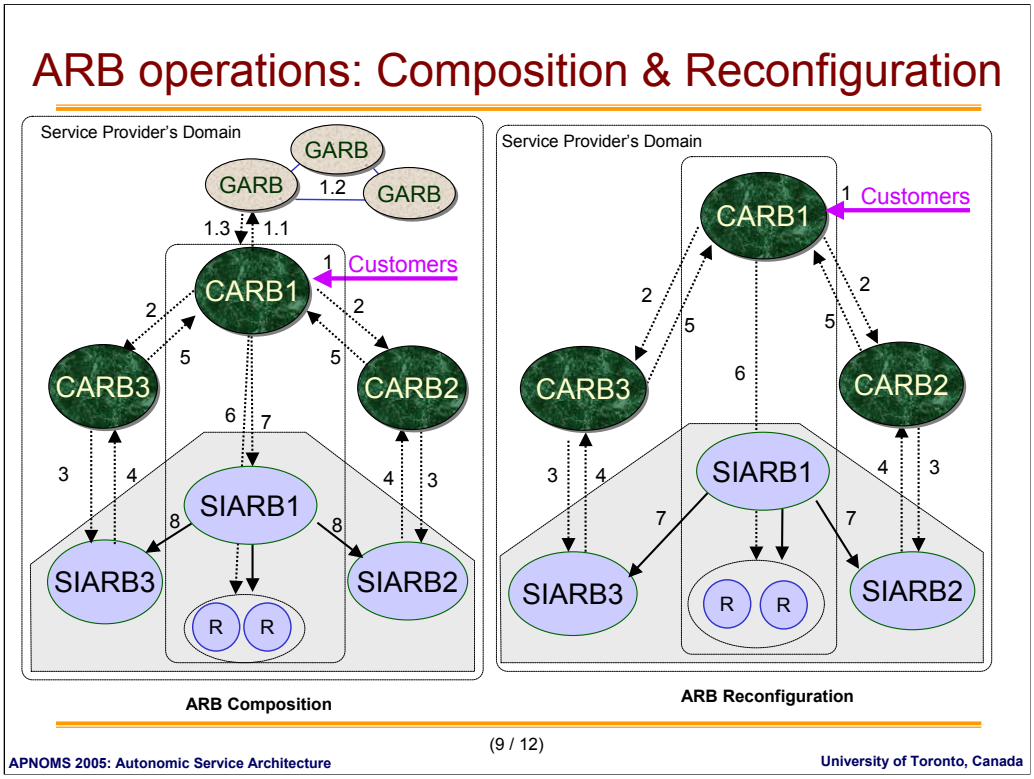
ASA therefore consists of a hierarchy of ARBs managing VSs, which provide ASA with its autonomicity property. Each ARB, except GARB, is created by its "Parent" ARB according to a contract (SLA). When the "Child" ARB is created, the "Parent" ARB allocates the appropriate amount of resources to it from the resources at its disposal. The customer in turn can offer the VS created as a new service to customers, becoming a SP to them. Once an ARB is created, it is subject to several disturbances, from its customers, SP, services offered, and underlying resources. The actions taken by the ARB in response to these disturbances ensure that its operation is autonomic. Possible actions include: Creation of a "Child" ARB, Reconfiguration of an existing "Child" ARB, Removal of a "Child" ARB, or Changes to resource allocation (in case of changing resource dynamics due to load fluctuation or problems). In this and the following two slides, we describe the ARB operations in detail, to validate the autonomicity of ASA.

Customer Disturbance: The information entered by the customer in the SLA Agreement is forwarded to the SLA Translator, which creates runtime policies in the appropriate format and stores them in the PIB. These policies regulate the "Child" ARB composition, its re-configuration in case of unsatisfactory performance detected by the Operation Manager, its removal in case customers terminate the service instance, and its resource allocation in case of problems such as failures or congestion in the underlying resources which are detected by the Operation Manager. This shows the self-configuring, self-optimizing, and self-healing properties of ASA.

SP Disturbance: The policies manually entered by the SP through the Policy Control are changed to the appropriate format, and are stored in the PIB. These policies are later used to automate the operation of ASA according to these specified goals (Customer Control, Service Activation, Planning Engine, Monitoring Engine, Operation Manager, Billing Engine, Resource Manager). This shows the self-configuring property of ASA.

Services Disturbance: When the service demand fluctuates, dynamic resource allocation and re-allocation based on SLAs occur by the Planning Engine. Performance is evaluated periodically by the Operation Manager. If this performance is not deemed optimal, reconfigurations can take place to ensure optimal resource use by the Resource Manager. In some cases, predictions of future demands happen and ASA can anticipate outages, congestion, or any kind of problems, using the Knowledge accumulated in the KIB as an indicator. These are the self-configuring, self-optimizing, and self-protecting properties of ASA.

Resources Disturbance: The faults, overloads, and congestion detected from the measurement infrastructure by the Monitoring Engine, are fed in parallel to the Problem Detection, SLA Evaluation, and Performance Evaluation components of the Operation Manager. The results from these components are fed into the Planning Engine along with relevant policies from the PIB, so that it decides on the appropriate actions to remedy to the situation at hand. Sometimes, prediction of future resource problems is possible by using the KIB, in addition to learning and inferencing techniques. These are the self-healing, self-optimizing, and self-protecting properties of ASA.

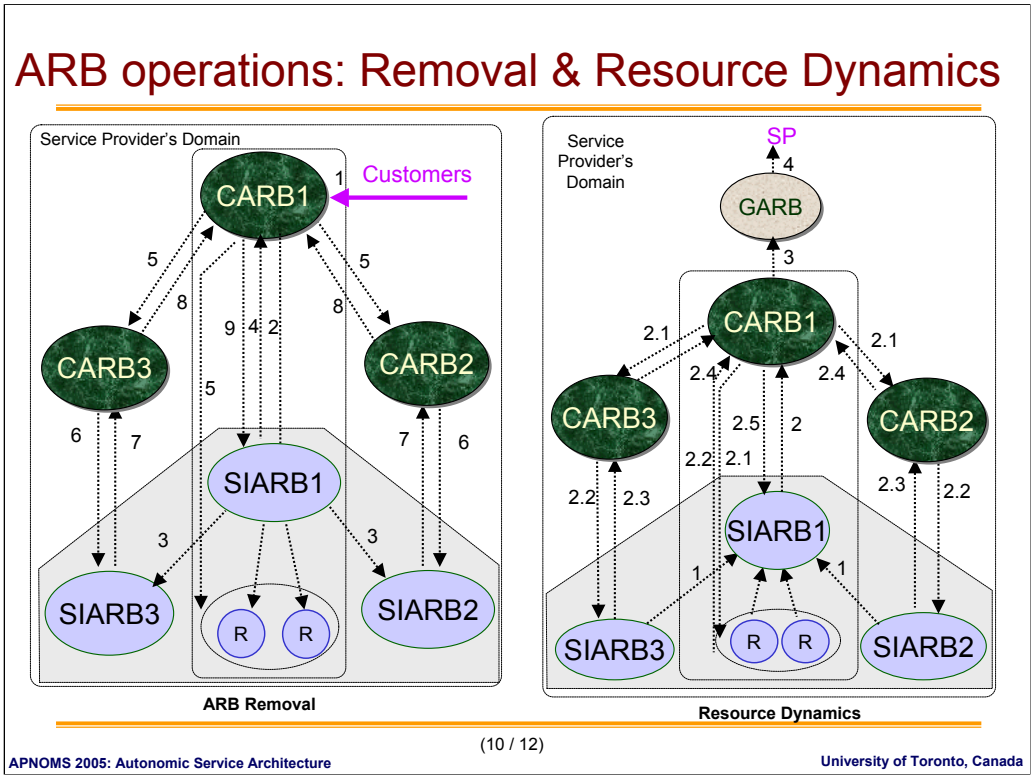


When customers activate a new service instance, they interface with the Composite CARB in charge of the service to be activated. In the above slide it is represented by CARB1. Hence, CARB1 needs to compose and provision a new service instance for the customer. Customers send the service activation request to the CARB handling the service needed, according to SLAs. CARB1 verifies that the SLA is valid, and determines the type and amount of underlying resources needed by the service instance. In some cases, it contacts its "Parent" ARB, GARB, to locate needed resources that are not available in the SP's domain but can be found in other domains. It determines that CARB1 needs resources from CARB2 and CARB3, in addition to underlying resources.

CARB1 sends a resource allocation request to the underlying resources needed by the ARB instance created. First, CARB2 and CARB3, both components of the composite CARB1, are contacted with the amount of resources needed. CARB2 and CARB3 create new SIARB instances, SIARB2 and SIARB3, with the appropriate amount of resources provisioned. SIARB2 and SIARB3 notify their "Parent" ARBs, in this case CARB2 and CARB3 respectively, of the new status and the completion of the new SIARBs' creation. CARB2 and CARB3 notify CARB1 of the new SIARBs created that are needed by the new service instance SIARB1. CARB1 sends a resource allocation request to the underlying resources needed by SIARB1. CARB1 notifies SIARB1 of the underlying resources allocated to it, as well as the other SIARBs of which it consists. SIARB1 notifies SIARB2 and SIARB3 of the composition completion.

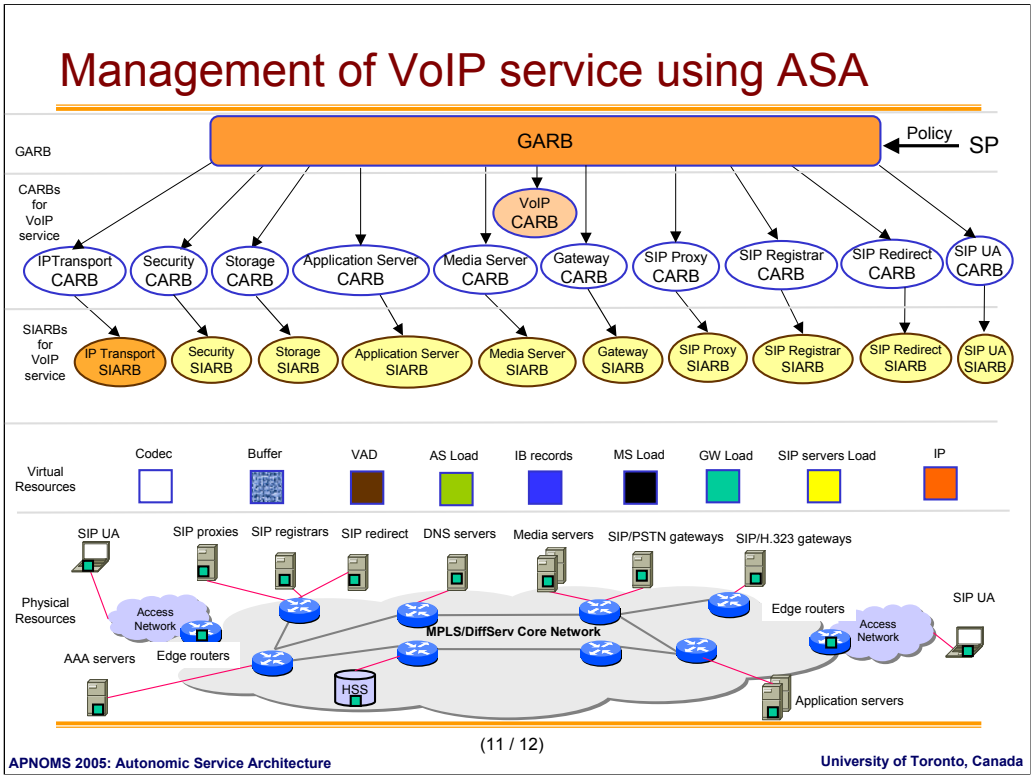
When customers need to reconfigure the service instance, they send a reconfiguration request to the Composite CARB1.. Hence, CARB1 needs to re-provision the service instance of the customer, SIARB1. CARB1 determines the type and amount of underlying resources needed by SIARB1. It determines that CARB1 uses resources from CARB2 and CARB3, in addition to underlying resources, so reconfigurations of SIARB2 and SIARB3 are needed.

CARB1 sends a resource re-allocation request to the underlying resources needed by SIARB1. First, CARB2 and CARB3, both components of the composite CARB1, are contacted with the new amount of resources needed. CARB2 and CARB3 contact their child ARBs, SIARB2 and SIARB3, with the appropriate amount of resources needed. SIARB2 and SIARB3 notify their parent ARBs, in this case CARB2 and CARB3 respectively, of the new status and the completion of the resource re-allocation. CARB2 and CARB3 notify CARB1 of the SIARB changes needed by SIARB1. CARB1 sends a resource allocation request to the underlying resources needed by SIARB1. Finally, SIARB1 notifies SIARB2 and SIARB3 of the reconfiguration completion.



When customers finish using the service instance activated, they send a release request to the composite CARB1, which plays the role of the SP from the point of view of the customer. Hence, CARB1 needs to remove the service instance of the customer, SIARB1. CARB1 sends a resource release request to the underlying resources needed by SIARB1, and to SIARB1 itself. SIARB1 forwards the resource release request to SIARB2 and SIARB3. SIARB1 notifies its parent ARB, in this case CARB1, of release request completion. CARB1 sends a resource removal request to its CARB components, in this case CARB2 and CARB3. CARB2 and CARB3 forward the resource removal request to their child ARBs, SIARB2 and SIARB3 respectively. SIARB2 and SIARB3 notify their parent ARBs, CARB2 and CARB3 respectively, of the removal completion. CARB2 and CARB3 notify CARB1 of the removal completion.

When the performance is unsatisfactory, changes to resource allocation are needed. In the case of Composite CARBs, problems are detected at the lowest layer, the virtual resources layer, or by the SIARBs constituting the customer's SIARB1. They are first sent to the SIARB managing those resources, SIARB1. If SIARB1 cannot solve the problem at hand, it needs to forward it to its parent ARB, in this case CARB1. CARB1 attempts to make the changes needed to fix the problem, by re-allocating additional underlying resources under its control to SIARB1. For this purpose, it contacts its components CARBs, CARB2 and CARB3, which try to obtain additional resources and allocate them to their child ARBs, SIARB2 and SIARB2 respectively. If CARB1 cannot solve the problem at hand, it needs to forward it to its parent ARB, the GARB. The GARB attempts to make the changes needed to fix the problem, by re-allocating additional underlying resources under its control to CARB1. If the problem is not solved, the service provider is contacted for manual intervention.

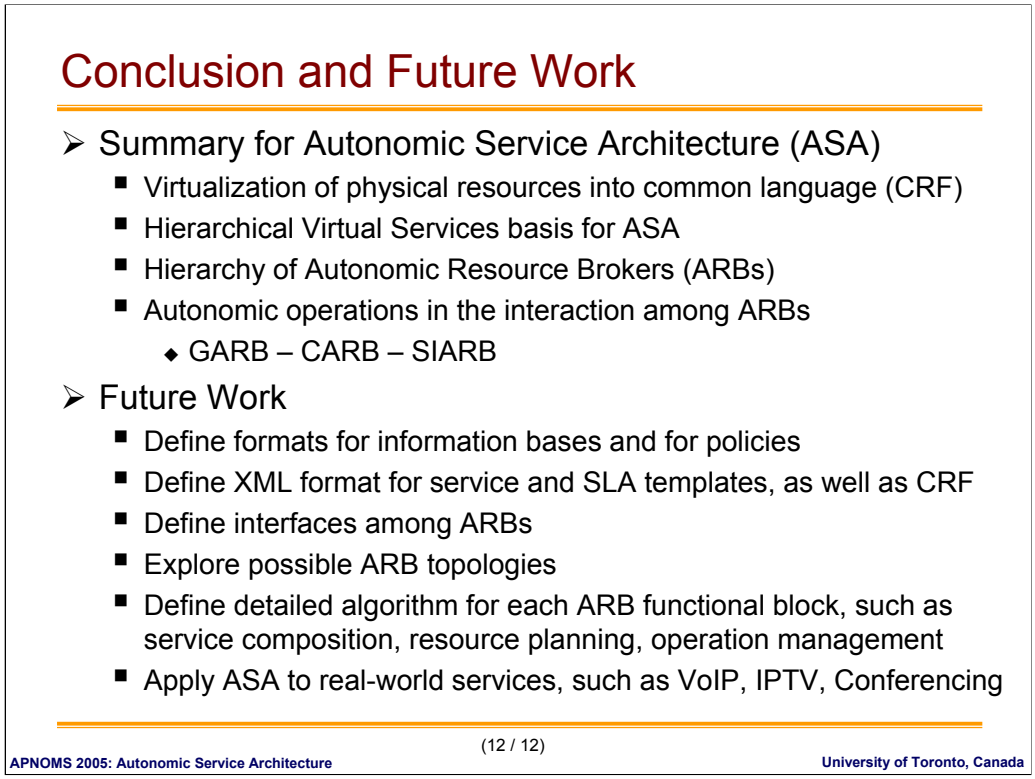


To illustrate ASA's feasibility, we consider its use for a SIP-based Voice over IP (VoIP) service offered by a telecommunication SP. Assume that this SP buys IP transport from a network service provider (NSP). Applying our Virtual Service (VS) view, VoIP is a composite service consisting of the following VSs and virtual resources: SIP User Agents (UAs) at customer premises, SIP Signaling Servers (Proxy, Redirect, and Registrar), IP packet transport (e.g. MPLS/DiffServ, bought from NSP, as mentioned previously), Media & Application Servers, and Gateways. All these resources are managed by the GARB. These resources at the disposal of the SP can be shared among various services, such as Gaming, IPTV, and VoIP. These services are managed by their corresponding CARBs, for example, the IPTV CARB for the IPTV service. The GARB regulates the resource allocation between the different CARBs managing those services, in order to satisfy the requirements of each service offered to customers. In ASA, all these resources are managed by basic CARBs and the VoIP service is managed by a Composite CARB, consisting of several Basic CARBs (for VSs such as SIP UAs, IP packet transport, SIP Signaling, Media and Application Servers). The VoIP CARB is created by the SP, and the corresponding resources are provided from the GARB resource pool in accordance with the VoIP service description sent from the SP. Using the ASA architecture, we can efficiently allocate the resources to multiple hierarchically constructed services in a telecommunication service domain.

SPs manually enter policies about the use of these resources, such as maximum number of calls allowed per SIP Proxy Server, rules for codec choice in UAs, or billing adjustments to be performed when SLAs are violated.

A customer subscribes to the VoIP service offered by this SP. He can choose the Gold class, which corresponds to one-way delay less than 150ms, and a packet error rate less than 0.001 (Class 0 as defined by ITU-T [14]). Based on this SLA data, target values for delay, jitter, and packet loss needed to match the desired voice quality is automatically determined using the E-Model [15]. The CARB selects the appropriate codec at the customer's premise, chooses the size of the receiver's play-out buffer, decides on the SIP servers to use in order to setup VoIP calls, and informs the edge routers (ERs) that they need to mark the voice packets accordingly. This ensures that the self-configuring and self-optimizing properties of autonomic systems are provided.

Monitoring network elements, SIP and Media Servers, SIP UAs is required to quickly detect problems (e.g. congestion due to increase in demand, failure of equipment such as the SIP or media servers, SLA violation due to an increase in delay, etc.). Remedy actions are taken by the Planning Engine, such as changes to the Call Admission Control policies at ERs or at SIP Proxies, re-directing SIP traffic from a congested SIP Proxy to a lighter one by modifying the corresponding entries in the DNS server at the sender, changes to the packet marking scheme at ERs, adjustments to the size of the receiver's play-out buffers, modifications to the sender's speech codec (more bandwidth efficient or lower packetization delay). This ensures that the self-healing and self-protecting properties of autonomic systems are provided.



The journey to a fully autonomic service architecture is still in its early stages. This paper illustrates our view of how SPs can approach this issue. We proposed a generic autonomic service architecture (ASA). ASA will allow SPs to reduce the costs of service delivery, by managing their resources through ARB operations and automatic allocation / de-allocation of resources to those ARBs. ASA is based on two main concepts: virtualization of physical resources using a common language (CRF), and autonomic service delivery using a hierarchy of Autonomic Resource Brokers (ARBs) which manage the Virtual Services (VSs) consisting each service delivered to customers. CRF allows the service delivery to be extended for all types of services by using the appropriate metrics which should cover the majority of services' needs. The ARB hierarchy allows the algorithms (filtering, service provisioning, resource allocation, problem detection, SLA violation, performance optimization, action planning) to be easily modifiable according to each service's needs. The VS concept allows ASA to expand to next-generation services by allowing flexible, scaleable, and recursive composition out of existing VSs. We are currently elaborating the aforementioned algorithms, and implementing ASA in our Lab. The ASA is still a conceptual architecture, but its realization for real-world services is underway. First, we are defining formats for the information bases, and for most importantly for policies. Second, we are defining an XML format for service and SLA templates, as well as CRF. Third, we are defining interfaces among ARBs, and exploring several ARB topologies and assessing the best in a particular situation. Finally, we are developing algorithms for each ARB functional block.

REFERENCES

[1] Jeffrey O. Kephart et. al., "The Vision of Autonomic Computing," IEEE Computer Magazine, Vol. 36, No. 1, Jan. 2003, pp. 41-50.

[2] A. Leon-Garcia et. al., "Virtual Network Resource Management for Next-Generation Networks," IEEE Communications Magazine, Vol. 41, No. 7, July 2003, pp. 102-109.

[3] "Autonomic communication," http://www.autonomic-communication.org/.

[4] Xiangdong Dong et. al., "Autonomia: an autonomic computing environment," Proc. of the IEEE International Performance, Computing,and Communications Conference (IPCCC) 2003, Phoenix, Arizona, Apr. 2003, pp. 61-68.

[5] M. Agarwal et. al., "AutoMate: enabling autonomic applications on the grid," Proc. of the Autonomic Computing Workshop, Seattle, Washington, Jun. 2003, pp.48-57, .

[6] K. Appleby et. al., "Oceano: SLA based management of a computing utility," Proc. of the IEEE/IFIP International Symposium on Integrated Network Management (IM) 2001, Seattle, Washington, May 2001, pp. 855-868.

[7] Sven Graupner et. al., "Adaptive Control Overlay for Service Management," Proc. of the IEEE Design of Self-Managing Systems, San Francisco, California, June 2003.

[8] A. Leon-Garcia et. al., "Communication Networks," McGraw-Hill, ISBN: 007246352X, 2004.

[9] H. Kreger, "Web Services Conceptual Architecture," White Paper, May 2001.

[10] J. Schonwalder et. al., "On the Future of Internet Management Techniques," IEEE Communications Magazine, Vol. 41, No. 10, Oct. 2003, pp. 90-97.

[11] K. Appleby, et. al., "Policy-based automated provisioning," IBM Systems Journal, VOL 43, NO 1, Jan. 2004, pp. 121-135.

[12] IBM Corporation, "Business Process Execution Language for Web Services," http://www.ibm.com/developerworks/library/ws-bpel, 2004.

[13] ITU-T, "International Telecommunications Union", http://www.itu.org

[14] ITU-T, "The E-Model, a computational model for use in transmission planning." Recommendation G.107, 2002.