Available online at www.sciencedirect.com
ScienceDirect
Computer Communications xxx (2006) xxx–xxx
computer communications
www.elsevier.com/locate/comcom

A generic architecture for autonomic service and network management ☆

Yu Cheng [a,*], Ramy Farha [a], Myung Sup Kim [a], Alberto Leon-Garcia [a], James Won-Ki Hong [b]

[a] Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ont., Canada M5S 3G4
[b] Department of Computer Science and Engineering, POSTECH, Pohang, Republic of Korea

Received 19 December 2005; received in revised form 20 June 2006; accepted 22 June 2006

Abstract

As the Internet evolves into an all-IP communication infrastructure, a key issue to consider is that of creating and managing IP-based services with efficient resource utilization in a scalable, flexible, and automatic way. In this paper, we present the Autonomic Service Architecture (ASA), a uniform framework for automated management of both Internet services and their underlying network resources. ASA ensures the delivery of services according to specific service level agreements (SLAs) between customers and service providers. As an illustrative example, ASA is applied to the management of DiffServ/MPLS networks, where we propose an autonomic bandwidth sharing scheme. With the proposed scheme, the bandwidth allocated for each SLA can be automatically adjusted according to the measured traffic load and under policy control for efficient resource utilization, while SLA compliance over the network is always guaranteed.

Keywords: Autonomic service management; Service level agreement; DiffServ/MPLS networks; Bandwidth sharing; Quality of service

1. Introduction

Internet protocol (IP) networks have been growing dramatically in size and functionality in the past decade, and are evolving into a global service communication infrastructure. In addition to the traditional best-effort data services, quality of service (QoS) guaranteed telecommunication services have started to be deployed over IP networks, for example, Voice over IP (VoIP). To reduce the time-to-market of new Internet services and lessen the operation/development/capital costs of service providers (SPs), it is necessary to develop a new service delivery framework by which the SPs can create and deploy QoS guaranteed services over IP in a scalable, flexible, and automatic way. From the information technology (IT) world, *autonomic computing* [1,2] is touted as the means to providing a rich set of IT services over a common computing infrastructure. The key feature of autonomic computing is the automated management of computing resources, encompassing the characteristics of self-configuration, self-optimization, self-healing, and self-protection. The application of autonomic management principles to ensure the delivery of telecommunications services over IP networks is largely unexplored. In this paper, we introduce an Autonomic Service Architecture (ASA) to address this need.

Many of the studies on autonomic computing or autonomic management [3–6] focus on the application of the autonomic concept to a certain service or application environment. In this paper, we propose a generic architecture, ASA, for autonomic service delivery over IP networks.

☆ This work is supported by a Postdoctoral Fellowship and a Discovery Grant from Natural Sciences and Engineering Research Council of Canada (NSERC).
* Corresponding author. Tel.: +1 416 978 4764.
E-mail addresses: y.cheng@utoronto.ca (Y. Cheng), ramy.farha@utoronto.ca (R. Farha), myungsup.kim@utoronto.ca (M.S. Kim), alberto.leongarcia@utoronto.ca (A. Leon-Garcia), jwkhong@postech.ac.kr (J. Won-Ki Hong).

0140-3664/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.06.017

ASA is driven by our view that "every thing is a service", from complex multimedia applications to simple IP packet delivery, and all the services are organized into a service hierarchy. Using this service perspective, ASA provides a uniform framework for service and transport network management. The underlying IP packet delivery and queueing are considered as *basic services*, upon which upper layer applications are built as *composite services*. We will show that ASA enables the network to orchestrate by itself the service, resource, billing, and fault management under the high-level policy guidance, where interaction with the human network managers is limited to specifying the services according to customer needs and establishing the management policies according to the QoS and revenue objectives.

To illustrate ASA's operation, we apply it to manage a multiprotocol label switching (MPLS) [7] based differentiated services (DiffServ) [8] network. DiffServ/MPLS is a promising IP network infrastructure due to its scalable QoS management and its traffic engineering capability [9]. We will demonstrate the VoIP service delivery through a virtual network (VN) [10] over the DiffServ/MPLS transport network, which is managed following the hierarchical service composition used by ASA. A representative framework for resource management and traffic engineering in DiffServ/MPLS networks is the TEQUILA architecture proposed in [11]. In the TEQUILA architecture, a DiffServ/MPLS network is operated in a "*first plan, then take care*" fashion, first through off-line planning and dimensioning and subsequently through dynamic operations and management functions for self-optimization. It will be shown that the TEQUILA architecture maps to the generic ASA framework, behaving as an instance of ASA for the management of DiffServ/MPLS IP transport networks.

Although ASA supports autonomic management by proposing a generic architecture, this architecture needs to be solidified and materialized through specific technologies, such as *service level agreement* (SLA) negotiation, policy control, efficient resource allocation, as well as automatic account and billing management. Note that SLAs are contracts between SPs and customers that define, among others, the services provided, the metrics associated with these services, acceptable and unacceptable service levels, liabilities on the part of the SP and the customer, and actions to be taken in specific circumstances. SLAs are critical in guaranteeing service delivery. Service management must ensure that necessary resources are provided to meet the SLA.

In this paper, we also investigate implementation details for autonomic service and resource management in the DiffServ/MPLS network. We propose an MPLS label stacking technique and path-oriented bandwidth management to support VN-based service provisioning. Particularly, we focus on the efficient resource management, which is a critical problem for all SPs desiring higher revenue. ASA is a SLA-centric management model, where it can be considered that the resources are shared by all SLAs over the network. We develop an autonomic *bandwidth borrowing* scheme for efficient inter-SLA resource sharing in a DiffServ/MPLS network. With bandwidth borrowing, the network can automatically adjust the resource allocation to each SLA when the traffic load conditions deviate from the engineered operation point or the high level policies change, so that the spare capacity in underloaded SLAs[1] can be exploited and QoS specification of all the SLAs are always guaranteed.

The remainder of this paper is organized as follows. In Section 2, we give a review of related work. Section 3 describes ASA. Section 4 illustrates the operation of the autonomic resource broker (ARB), the key component of ASA. Section 5 shows how ASA is applied to the management and control of a DiffServ/MPLS network. Section 6 presents the bandwidth borrowing scheme. Section 7 presents the computer simulation results. Section 8 gives the concluding remarks.

2. Related work

2.1. Autonomic computing in IT services

The IBM Autonomic Computing Architecture [1,2] is the pioneer work in the new wave of autonomics, which defines an abstract information framework for self-managing IT systems. In the information framework, an autonomic system is a collection of *autonomic elements*. Each autonomic element consists of an *autonomic manager* (AM) and the *managed resource* (MR). The communication between the AM and the MR is done through the MR's management interfaces, which exposes two types of hooks, *sensors* and *effectors*. The sensors are used by the AM to obtain the internal state of the MR, and the effectors are used by the AM to change the behavior of the MR. The AM enables self-management of the resource using a "*monitoring, analysis, planning, and execution*" control loop, with supporting knowledge of the computing environment, management policies, and some other related considerations. The autonomic computing information model only provides the conceptual guidance on designing self-managed systems; in practice, the information model needs to be mapped to an implementable management/control architecture. Specifically, measurement techniques, rule engines, planning methodologies, dynamic resource allocation techniques, and access/security management schemes need to be developed for autonomic elements, and a scalable management/control plane is required to coordinate the autonomic elements into a self-managing system.

[1] When the actual traffic load associated with a certain SLA is more or less than the engineered traffic load, the terms "overloaded" and "underloaded" are used, respectively, to indicate the loading status of an SLA.

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

Various approaches have been investigated to implement an autonomic computing system. The Autonomia architecture [3] provides dynamically programmable management/control to support development/deployment of smart applications. In Autonomia, a mobile agent system (MAS) [12,13] is installed on a computer server (which intends to provide service components to applications) for a resource-independent execution environment; an application management editor (AME) is provided to users for defining the component management and composition rules; a centralized application delegated manager (ADM) is responsible for automated registration/discovery of components, automated configuration of applications/resources, and application adaption to deal with component failure. However, Autonomia is not a fully autonomic architecture, as the components in Autonomia are not self-managed autonomic elements, the properties and states of which are pre-configured through the AME. Moreover, the centralized management/control plane (i.e. the ADM) suffers from scalability in large scale networked computing systems.

The AutoMate architecture [4,14] is an materialization of the autonomic-element based information model to enable autonomic grid applications [15]. In AutoMate, each autonomic element (consisting of the computational element and the element manager) encapsulates rules, constraints, and mechanisms for self-management, where three classes of ports are defined for interactions with other autonomic elements: the *functional port* defining the computational behavior of the element, the *control port* exporting the sensors and effectors to the element manager, and the *operational port* defining the interfaces to inject and manage policy rules. A multiagent infrastructure [14] consisting of peer element managers and a composition manager is used to formulate autonomic applications as dynamic composition of autonomic elements, in a distributed manner under the policy control. While the AutoMate project makes significant contributions in the element interface design and rule engine design for dynamic application composition, it does not provide a general implementation structure for the element manager (i.e. the autonomic manager in the vision of autonomic computing).

The Oceano project [5,16], joint work between IBM and the University of Berkeley, aims to design and develop a scalable, manageable computing utility infrastructure, which consists of a farm of massively parallel, densely packaged servers interconnected by high-speed, switched LANs. Particularly, dynamic resource allocation techniques, according to the "monitoring, analysis, planning, and execution" control loop, are developed to accommodate planned and unplanned fluctuation of workloads under the constraints of SLA.

The HP vision for the Adaptive Enterprise [6,17] and the Microsoft Dynamic Systems initiative [18] are related industry efforts that recognize that self-managing components and systems are vital to the future of IT. Moreover, both of the works emphasize the importance of *resource virtualization*; rather than directly mapped to physical resources, the applications/services are associated with virtualized resources which may consist of computing/storage/networking resources from multiple devices. The ability to decouple workloads from the physical resources greatly facilitates dynamic resource sharing.

It is noteworthy that all the above mentioned projects mainly focus on the autonomic management of computing resources for IT services delivery. In this paper, we expand the autonomic view to include telecommunications services, which consist of both computing and networking resources. In addition, the proposed Autonomic Service Architecture exploits related research contributions to achieve a generic autonomic architecture supporting heterogenous services in current and future IP networks. Particulary, a virtual resources layer is used to separate services from physical resources, upon which services are composed hierarchically according to our view "everything is a service". Each service in ASA is encapsulated as an autonomic element, and all the autonomic elements interact with each other in the service hierarchy automatically and adaptively under the policy control. Moreover, we design an autonomic resource broker to serve as the autonomic manager, which is the key enabler of the ASA.

2.2. Policy-based management

One critical aspect of an autonomic system is the principle of *policy-based management* [19,20]. In general, policies represent the high-level service objectives and operation control logics that can determine the behavior of managed systems. The promise of policy-based management is that the operation of computing/networking resources can be guided to follow certain rules, and dynamically configured so that they can achieve certain goals and react more nimbly to their environment [20].

The Distributed Management Task Force (DMTF), jointly with the Internet Engineering Task Force (IETF), develops the policy information model based on the Common Information Model [21] that provides a consistent definition and structure of data using object oriented techniques. The CIM Policy Model defined by DMTF [22] and the Policy Core Information Model (PCIM) defined by the IETF [23,24] both facilitate unified and consistent representation of policies across a wide spectrum of technical domains, including policies related to configuration and usage of devices and applications. A specific mapping of the CIM Policy Model in the Autonomic Computing Architecture, including components for policy creation, storage, evaluation, and enforcement, is presented in [20]. The design of ASA in this paper also follows the principle of policy-based management.

2.3. Automation in IP services and network management

While the autonomic concept has attracted much attention in the IT domain, efforts are being made in the tele-

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

communications domain to bring as much automation as possible into IP network and service management. The TEQUILA architecture [11] is operated in a ''first plan, then take care'' fashion for network management, which is in fact consistent with the ''monitoring, analysis, planning, and execution'' autonomic control loop proposed by IBM. By considering TEQUILA as an instance of ASA for the IP transport service management, the generality of ASA and the convenience of DiffServ/MPLS infrastructure for automatic management can be demonstrated.

The notion of virtual network [10] (or similarly the service overlay network [25]) has been widely studied for scalable IP service deployment and efficient resource management. A VN can purchase certain amount of networking resources from the IP network provider (NP) via the bilateral VN–NP SLAs to build a logical service delivery network. The VN then in turn behaves as a SP to sell QoS guaranteed services to customers. With such an approach, good scalability can be achieved as the NP is freed from the high-level application/service management, which will be addressed by specific VNs. Moreover, a VN can be further subdivided to form a hierarchical architecture where different services can be flexibly deployed. The VN concept can be seamlessly integrated into ASA's virtual service concept as a way of building composite services from basic services or other composite services.

AT&T's MPLS OAM architecture [26] proposes the Concept of Zero that aims to bring full automation for every human-to-computer interaction currently required for setting up and maintaining network services. However, AT&T's design only focuses on the network management and leaves the service management part open. AT&T's work also gives a discussion of the technologies required to achieve automatic management of an MPLS network. We have the same concern that the implementation of ASA requires autonomic technologies in various areas, and also puts emphasis on the application of ASA for service management over DiffServ/MPLS networks. Particularly, we propose a novel autonomic inter-SLA resource sharing technique for efficient resource utilization and QoS guarantee.

In [27], Bouillet, Mitra, and Ramakrishnan propose a SLA management architecture based on virtual partitioning [28] for efficient resource utilization. At each link, virtual partitioning is implemented for resource sharing among overloaded and underloaded SLAs. The cost of virtual partitioning is that the QoS of the underloaded SLAs can not be guaranteed. SLA violation for underloaders is a serious problem, which could encourage malicious overloading. Therefore, the authors propose to use a *penalty payment* from the service provider to the customer to compensate the possible QoS or SLA violations incurred in the virtual partitioning. However, the penalty scheme is not a completely satisfying solution from the customers' perspective. Customers would always prefer to have guaranteed quality of service as well as a fair billing system. To the best of our knowledge, there is currently no such resource allocation technique available that can achieve a resource utilization close to virtual partitioning while guaranteeing the QoS of all SLAs involved in the resource sharing. Therefore, the bandwidth borrowing scheme is proposed for the above objective. It is noteworthy that VP based resource sharing is a static design, where a pre-configured VP scheme is applied for resource sharing in all the possible load conditions; such static design is the basic reason that leads to SLA violations. However, the proposed bandwidth borrowing adopts the methodology, i.e. ''boundary resource commitment determines link resource sharing'', which is consistent with SLA-centric manage principle and convenient for adaptive adjustment of resource allocation.

3. Autonomic service architecture

In this section, we will present ASA according to a layered view, where services are built on top of virtual and physical resources. The players involved in the delivery of a service are the customers and the SPs. After customers and SPs negotiate the services needed and their corresponding SLAs, ASA will manage these services in order to ensure satisfactory service delivery without SP's intervention. It is noteworthy that some manual actions will still be needed to complete the service delivery process, but they are limited to few high level operations such as establishing management policies and specifying new services. If the problems incurred are too complex to be handled by the autonomic system, manual adjustments are also required.

In ASA, 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. Resources can be physical or logical components used to construct services. When customers purchase any service from a SP, they can also offer the purchased service to other customers, becoming SPs to those customers. A SP can also behave as a customer to a peer SP to negotiate inter-SP resource commitment needed to achieve end-to-end (E2E) QoS support.

As mentioned earlier, the ASA design is driven by our view that ''everything is a service'', from complex multimedia services to IP packets delivery. The layered autonomic service architecture is shown in Fig. 1. The lowest layer consists of the actual physical resources, which are involved in the delivery of the service. The middle layer consists of an abstraction of the physical resources into virtual resources, which specify characteristics of the physical resources. The upper layers consist of the services (basic and composite), which are hierarchically and recursively composed by using these underlying physical and virtual resources.

Vertically, services delivered by ASA could be broken into two views: *operation* and *management*. The operation view consists of the control and data planes present from traditional service views [29] at different layers, while the management view consists of the management functions needed to manage the services delivered. Management

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

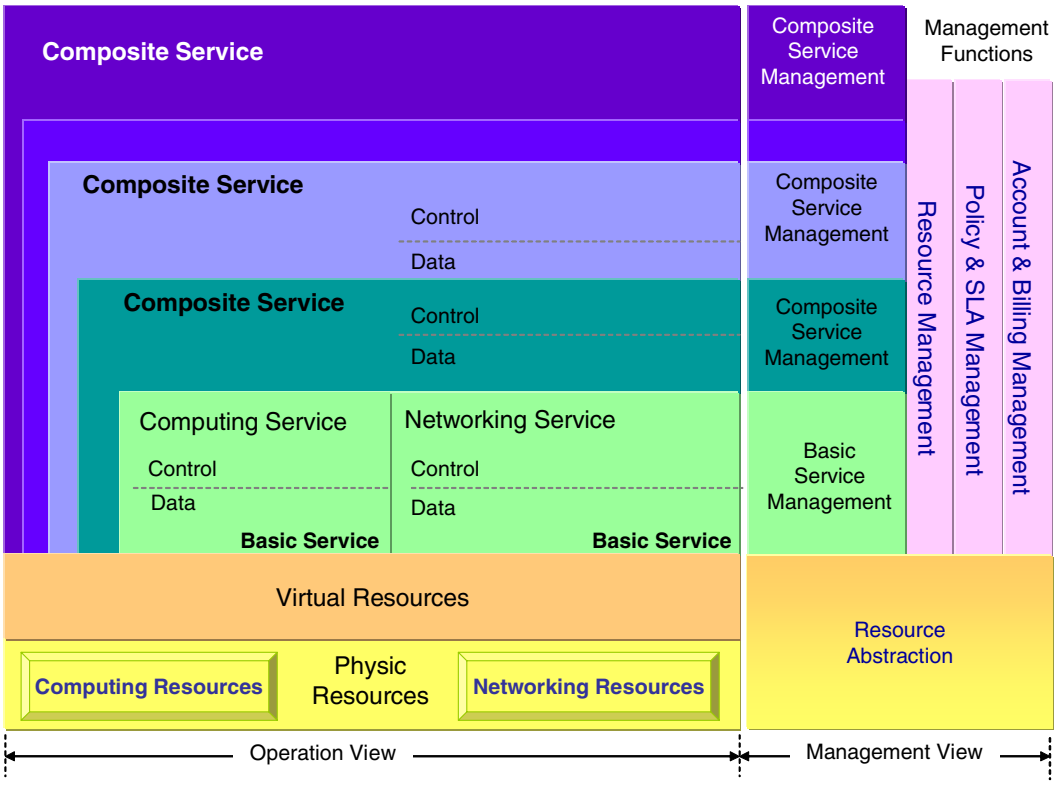

Fig. 1. Autonomic service architecture layered view.

functions mainly involve resource management, policy and SLA management, as well as accounting and billing management. Note that an underlying measurement infrastructure is needed at some control points in the SP's domain, to gather raw performance data.

3.1. Operation view

The operation view mainly consists of the iterative layering of services (basic and composite), on top of the underlying logical and physical resources.

(1) *Physical Resources Layer*: This layer consists of the physical resources that the SP has at its disposal, which include computing and networking resources (e.g., routers, switches, links, servers, and storage devices).
(2) *Virtual Resources Layer*: The ASA encompasses heterogeneous computing, storage, and networking resources to support various IT and telecommunication services. The virtual resource layer provides a uniform and consistent interface to all the physical resources, which would simplify the resource management, service composition, and dynamic resource sharing. Standards activities within the grid computing and the web computing communities have recently converged in the Web Services Resource Framework (WSRF) [30,31], aimed towards the development of common web-service based representations of all resources. However, the web-service representations of networking resources (e.g. routers, queues, links) are still open issues; an web-service interface for optical links is presented in [32]. Moreover, the eXtensible Markup Language (XML) used by the web-service representation is verbose, and the impact of the processing overhead (due to parsing the XML contents) on certain realtime resource control needs further studies. Regarding these open problems, we assume that the virtual resource interface may be at last agreed on a more efficient Common Resource Format (CRF). The details of CRF will be studied in our future work.
The virtual resource layer also includes an adapter to implement the physical to virtual translation, i.e. the translation from a lower-level proprietary format to the CRF format. The specific translation implementation depends on the types of physical resources involved, which include three categories: (a) *single resources* consisting of a single physical resource such as a router or a servers, (b) *clustered resources* consisting of multiple physical resources clustered together at the same geographical location, and (c) *distributed resources* consisting of multiple physical resources, geographically dispersed, but which can be virtualized to look as an aggregate resource.
(3) *Basic Services Layer*: In some cases, the virtual resources are offered to customers directly by SPs as basic networking services, for example as QoS-guar-

anteed IP transport services. In other situations, basic services, which are bought from other SPs according to SLAs, become virtual resources at the disposal of the purchasing SP.

(4) *Composite Services Layer:*: Composite services consist of several basic services and/or composite services. The uppermost layer composite services are offered directly to customers. Service composition is hierarchical and recursive, and continues until the desired composite service is ready to be offered to customers. VN [10] *is one example of a composite service.*

3.2. Management view

The main task of ASA consists of autonomically managing the resources at the SP's disposal in order to meet service demands fluctuations. All management functions (resource, policy, SLA, accounting and billing management) are performed by autonomic entities called Autonomic Resource Broker (ARB), which are self-managing, and whose role is to ensure automated delivery of services. A key concept in the IBM autonomic computing architecture is the autonomic element, a component that is responsible for managing its own behavior in accordance with high-level policies, and for interacting with other autonomic elements. In the ASA framework, ARBs are the analogy of IBM's autonomic elements. The ARB is the component responsible for managing services in accordance with the policies and SLAs, by interacting with underlying resources, and with other ARBs to provide or consume services.

The ARB hierarchy is shown in Fig. 2. When customers activate service instances, these 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). The different services offered by a SP are managed by a GARB (Global ARB), which handles all the resources available at this SP's disposal. Note that our approach is based on service-oriented architectures [33] for interactions between ARBs and underlying resources to leverage existing and future management protocols [34].

4. Autonomic resource broker architecture

Autonomic resource brokers are the autonomic components, which constitute the autonomic service architecture. Fig. 3 shows the ARB's internal architecture.

4.1. Information base

In order to perform autonomic service management, ASA needs to maintain necessary information about the service environment. Information Bases are classified into five logical groups:

- Customer Information Base (CIB): contains information related to customers, such as personal information, list of the subscribed services, and updated bill.
- Service Information Base (SIB): contains information about the service instances activated by customers, such as parties involved (customers and

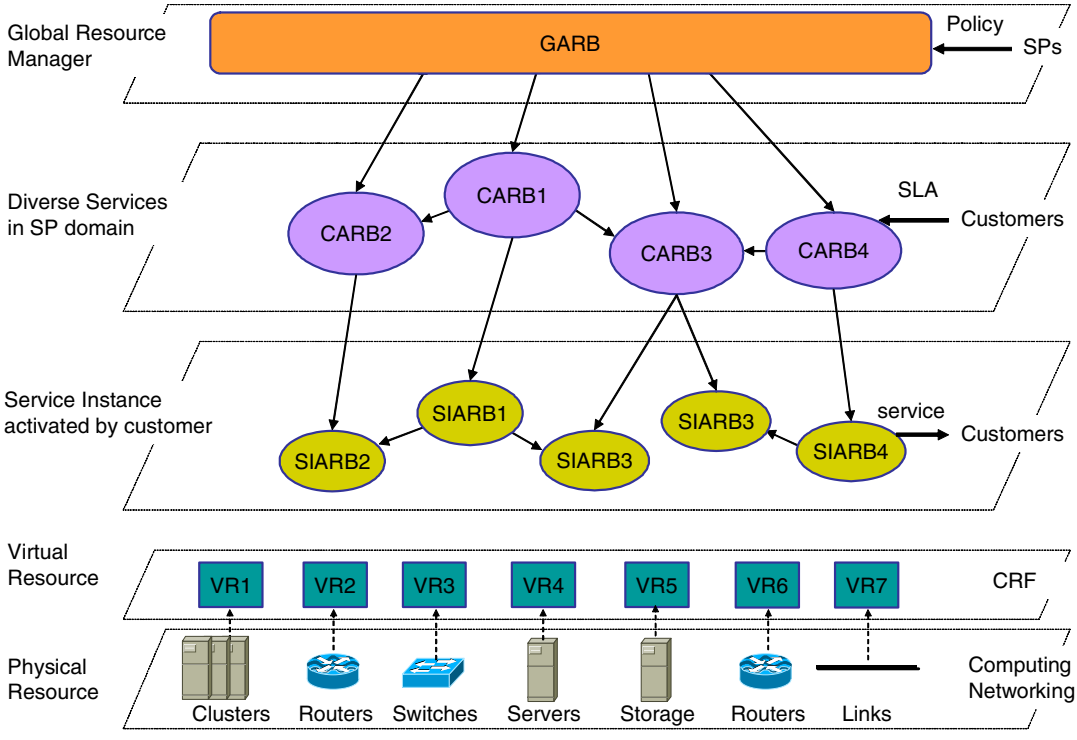

Fig. 2. Hierarchical management view.

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

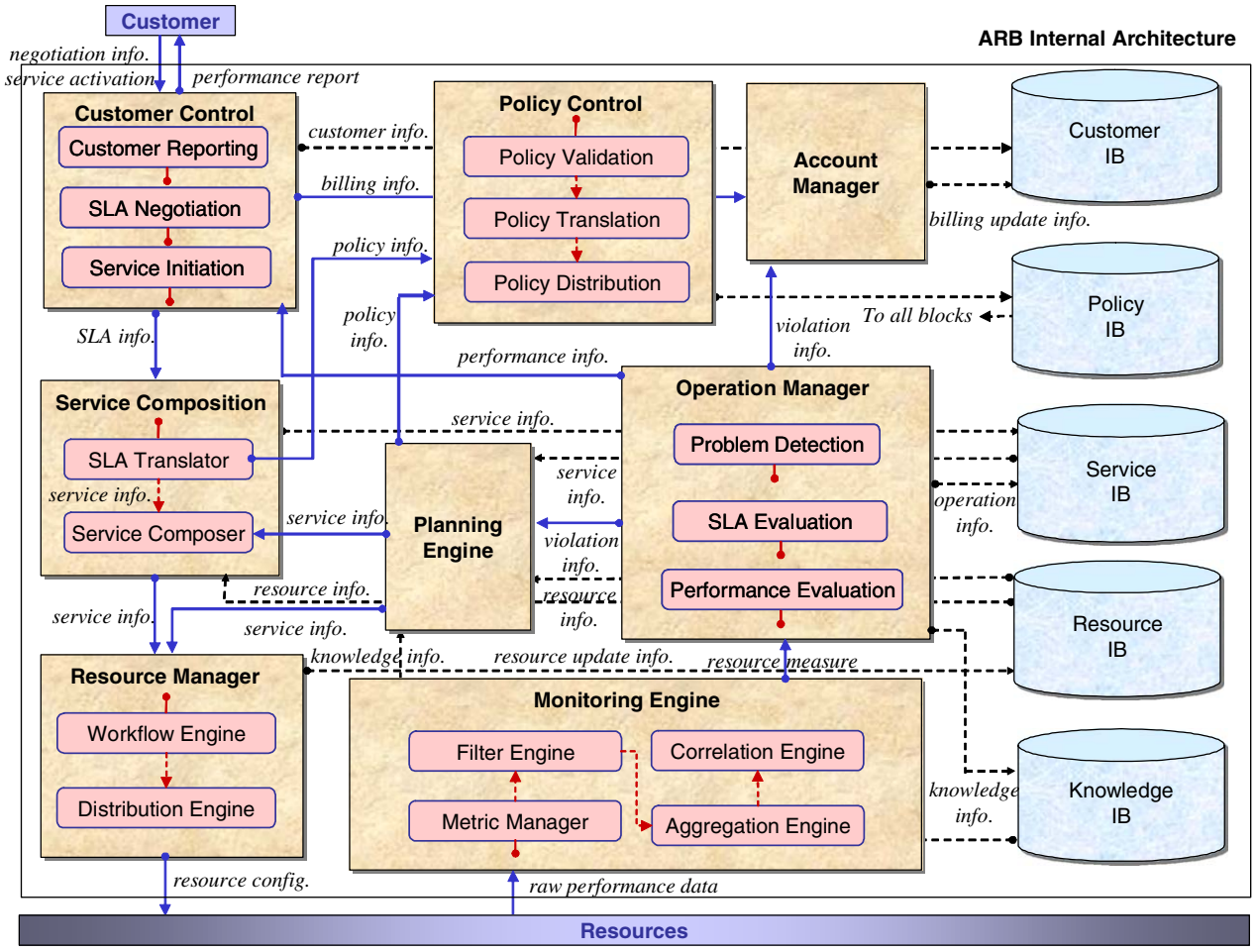

Fig. 3. Autonomic resource broker architecture.

SPs), SLAs regulating the service delivery, types of resources needed, amount of each resource type needed, billing plan for the service, and operation history.

- Resource Information Base (RIB): contains information on resources available at the SP's disposal at a given time, such as types of resources and updated quantity available of each.
- Policy Information Base (PIB): contains the policies created at runtime, or predefined by human operators. These policies are service-based, and used to regulate the operation of each ARB component, such as monitoring engine, performance evaluation, SLA evaluation, problem detection, planning engine, customer reporting, service composition, resource manager, account manager, as well as to provide SLA templates for the services offered.
- Knowledge Information Base (KIB): contains information for use in case problems arise. Remedy actions can be taken based on previous occurrences of the problem, such as problem description, cause of problem, time of occurrence, parties involved, solution elaborated, and effect of solution taken.

4.2. Policy control

ASA uses policies to make decisions and choose a certain course of actions. These policies could be created initially by SPs, or at runtime as a result of service activations. These policies can be updated when service demands change and loads vary. Policy Control includes the following actions:

(1) *Policy Validation*: Sometimes, the creation or update of policies could lead to conflicts, redundancy, inconsistency, and infeasibility problems. The role of this component is to ensure no such problems occur and to remedy them if possible.
(2) *Policy Translation*: This component interprets policies and translates them to an understandable format prior to use.
(3) *Policy Distribution*: This component distributes policies to the ARB components that need them.

4.3. Customer control

Customers and SPs need to interact in order to buy the services with a certain SLA, and then to use these services.

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

Customer Control constitutes this interaction. The SLA is initially negotiated between the two parties, and later on customers activate services, expecting the performance level for which they had subscribed. This performance is also reported to the customers if needed.

(1) *SLA Negotiation*: SLA negotiation consists of two main steps: (a) generating the contract by the SP – SPs keep categorized contract templates so that the contracts can be customized according to the customer types using policies; (b) approving the contract by the customers – customers fill the contract fields appropriately, and the SP checks validity of the entries. If correct, the contract is translated into Customer Info. and Billing Info., which are kept in the CIB. The information entered by the customer upon service subscription, is called the Negotiation Info., and, once the negotiation is successful, is sent as SLA Info. to the Service Composition component for SLA translation.
(2) *Service Initiation*: Once the SLA has been agreed upon and stored for the particular service and the customer specified, service instances can be activated by customers. The activation is done via the Service Initiation component, which retrieves the Service Info. using the SLA Translator in the Service Composition component, and sends it to the Service Composer.
(3) *Customer Reporting*: The SLA allows customers the access to monitoring service performance, so that they have the freedom to switch SPs if performance is not satisfactory.

4.4. Service composition

This component handles composition of services upon service activation by customers, and identifies the resources required for each activated service instance.

(1) *SLA Translator*: Based on the SLA Info. received from the Customer Control component, some policies could be created on the fly. In addition, this SLA Info. is translated into a list containing the type and amount of resources needed to support the activated service instance. This is part of the self-configuring aspect of ASA.
(2) *Service Composer*: When the service activated by the customer is composite, the Service Composer needs to optimize the composition by choosing the basic services or lower-level composite services needed and their quantities appropriately. The optimization problem for service composition is formulated by the SP initially, according to the service type and resources that the service needs, and then it is solved to maximize or minimize a certain objective function depending on policies and goals set forth by SPs, e.g. maximize revenue or minimize operation/capital cost. The resulting Service Info. is then sent to the Resource Manager for the appropriate resource allocation. This is part of the self-configuring and self-optimizing aspects of ASA.

4.5. Resource manager

Necessary resources need to be allocated to the service instance activated. The appropriate ARBs and/or the underlying resources are informed of the decision to provision the needed resources. At this point, the SP domain's self-configuration to support services is completed.

(1) *Workflow Engine*: The resource allocation process is converted to a workflow of actions, for example, setting up new customer account, allocating media/signaling servers, determining the QoS class over IP, and configuring the scheduler and buffer manager at each hop to guarantee QoS.
(2) *Distribution Engine*: The actions decided by the workflow Engine are distributed to the appropriate ARBs and/or the underlying resources, and their execution is controlled. The distribution engine is the interface between ARB and the underlying logical and physical resources.

Here, we would like to emphasize that in the aforementioned description, we considered service activation to be done by a general customer. The customer may be a single or a family, or other types of customers such as a corporation, a content provider, or a VN. The Service Composition and Resource Management normally incur network dimensioning and network configuration when large customers are considered, or incur connection admission control and per-flow resource management when smaller customers are considered.

4.6. Monitoring engine

The self-management operation of the ARB is based on the raw performance data collected by the Monitoring Engine. This component monitors the underlying resources, both physical and virtual, and generates aggregate metrics such as the actual customer data rate. The measurement results are forwarded to the Operation Manager for analysis.

(1) *Metric Manager*: There is a need to quantify raw measurement data in a common format understandable by the ARB components in order to make appropriate decisions. This component ensures that the raw measurement data collected conform to the CRF format.
(2) *Filter Engine*: To avoid overloading the ARB components with raw measurement data, and to filter out

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

unwanted data, filtering is needed. The tradeoff is between precision and overhead of measurements. The more precise the results need to be, the more measurements we need to perform.

(3) *Aggregation Engine*: The measurement results after filtering could be aggregated if a new metric which is a combination operation (e.g. summation, average, maximum, or minimum) of the collected measurements is needed.

(4) *Correlation Engine*: The Correlation Engine correlates filtered measurements and detects complex situations, using techniques such as spatial/temporal correlation, and prediction.

4.7. Operation manager

In order to take appropriate corrective actions which will ensure optimal operation in the SP's domain, an ARB component is needed to analyze the measurements sent by the Monitoring Engine, in order to detect any abnormal behavior that results from faults, PSLA violation, and sub-optimal performance. Operation Manager and the Planning Engine are the main components that give the ARB its self-optimizing, self-healing, and self-protecting characteristics.

(1) *Problem Detection*: Faults can occur when computing or networking components fail. Overloads can occur when the demand on a particular component exceeds the capacity of the component. Congestion can occur when the performance of some components degrade because of excessive load. This is part of the self-healing aspect of ASA.

(2) *SLA Evaluation*: SLAs are evaluated, and the violations detected are sent to the Planning Engine for appropriate planning, and, if needed, to the Account Manager for proper adjustments to the customer bill stored in the CIB. This is part of the self-optimization aspect of ASA.

(3) *Performance Evaluation*: When the service operation is satisfactory, ARB ensures that resources are optimally allocated by using optimization functions. This is part of the self-optimization aspect of ASA.

4.8. Planning engine

This component is the brain of the ARB to achieve the self-optimizing, self-protecting, and self-healing aspects of autonomic systems. The inputs to the Planning Engine are:

- Service Info., i.e. performance requirements for services (SLAs), obtained from the Service Information Base.
- Policy Info. that constrains solutions, i.e. policies that restrict allocation of resources, obtained from the Policy Information Base.
- Resource Info. extracted from the existing resource pool that keeps track of available resources at the SP's disposal, obtained from the Resource Information Base.
- Knowledge Info. that contains previous comparable situations, where the advocated solutions could be used instead of elaborating new ones, obtained from the Knowledge Information Base.
- Violation Info. consisting of the results passed on by the Operation Manager, such as problem detection, SLA violations, and sub-optimal performance.

The outputs that can be generated are:

- Changes to the Service Composer, for instance, resources needed to meet service requirements, re-allocation plans to improve service performance, in the form of Service Info.
- Changes to the Resource Manager when the problems do not require re-composition of the service instance.
- Changes to the policies that regulate the operation of the ARB components, in the form of Policy Info.

4.9. Account manager

The account manager makes adjustments to the bill of a particular customer, when the SLA is violated. The adjustments depend on billing policies set by SPs. This is part of the self-healing aspect of ASA.

5. Service example: VoIP over DiffServ/MPLS

In this section, we illustrate ASA's operation by applying it for autonomic management of a SIP-based VoIP service that is delivered over a DiffServ/MPLS IP transport network. We consider that the IP transport network supports multiple types of upper-layer services; one of the services is VoIP. According to ASA, the VoIP service is a composite service consisting of the following virtual resources: SIP User Agents (UAs) at the customer premises, SIP Signaling Servers (proxy, redirect, registrar), Gateways, Media Servers, and IP packet transport (which is a VN purchased from the DiffServ/MPLS network). The VoIP service network is managed by a composite CARB, consisting of several basic CARBs (for SIP UAs, IP VN, Gateways, SIP and Media Servers). SPs manually enter policies related to criteria such as maximum number of calls allowed per SIP Proxy Server, rules for codec choice in UAs, the upper bound of new call blocking probability, or billing adjustments to be performed when SLAs are violated. To create a QoS guaranteed VoIP service network over the IP transport network, the VoIP SP is considered as a customer to the transport network, and all the related resources should be properly planned and allocated from the corresponding component CARBs. Resources allocated from each CARB are managed by a SIARB which is put at the disposal of the VoIP CARB. Fig. 4 shows the VoIP service managed by ASA.

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

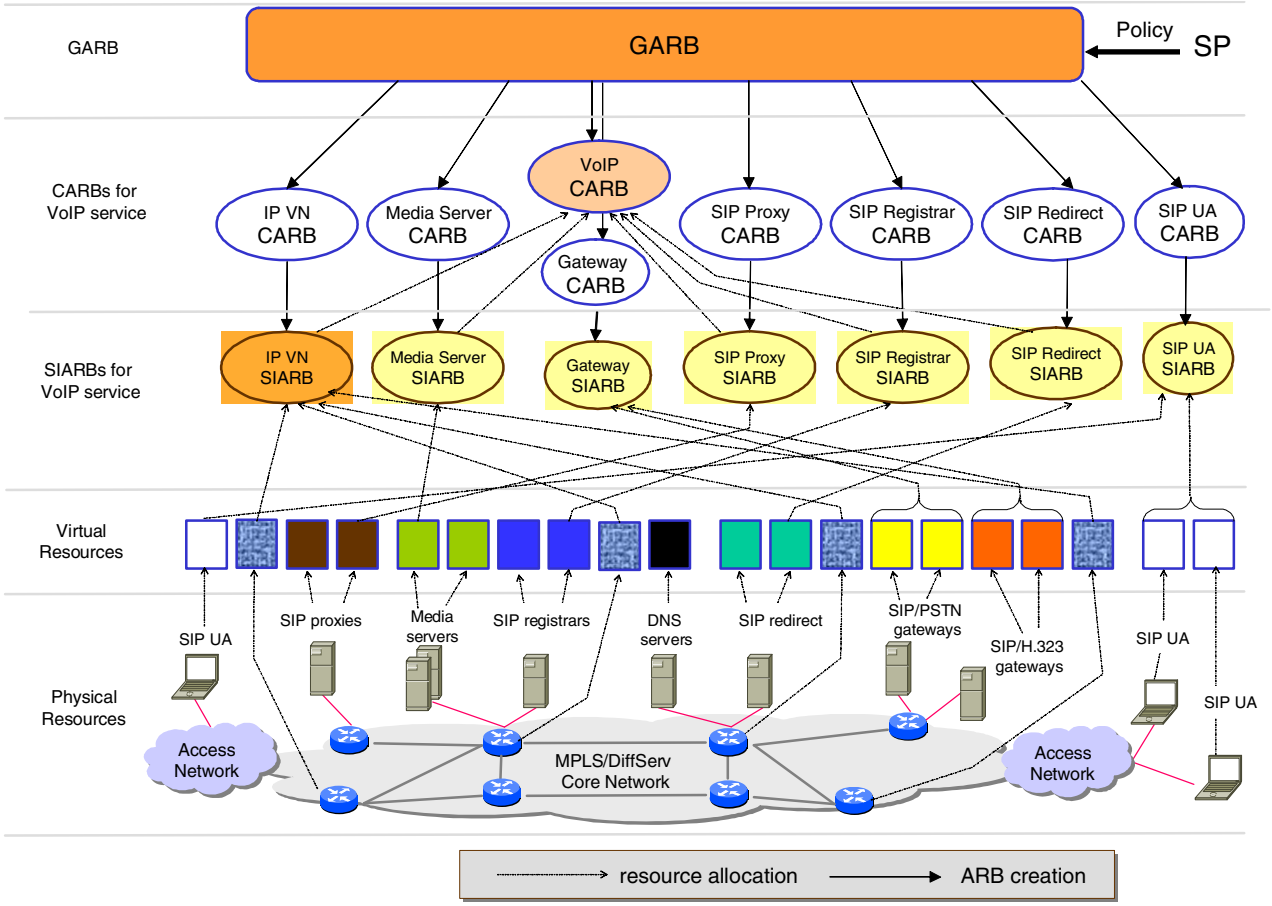

Fig. 4. Autonomic management of VoIP over DiffServ/MPLS.

5.1. VoIP service instance activation

When the VoIP service network is established, a customer can subscribe to the VoIP service with a SLA, and the service is composed and managed by the VoIP CARB. The SLA negotiation may result in the customer asking for the VoIP premium service at a given Mean Opinion Score. This SLA Info. is passed on to the SLA translator, which translates it into Service Info. consisting of target values for E2E delay, jitter, and packet loss, needed to match the desired voice quality.

The Service Info. is sent to the Service Composer, which checks whether the VoIP service network has enough resources to hold the new call through a call admission control procedure. If admission decision is positive, the service composer will then select the appropriate codes at the customer's premise, choose the size of the receiver's play-out buffer, use the appropriate silence suppression/voice activity detection (VAD) at the receiver, decide on the SIP servers that the signaling messages traverse to setup the connection, determine the DiffServ Class for IP transport and inform the edge routers (ERs) that they need to mark the voice packets accordingly. The Resource Manager ensures that the appropriate actions are taken by notifying the underlying resources involved. Within the VoIP service network, each aggregation of voice calls is managed by a SIARB, and related customer, service, and resource information is memorized in the corresponding information bases. The aggregation size depends on issues such as management overhead and scalability.

5.2. ARB of the DiffServ/MPLS transport network

In the previous section, a generic ARB architecture was shown. The generality of ARB is shown by applying it to manage a DiffServ/MPLS network. The scenario considered is that VNs are created over the IP transport network to support upper layer services, i.e. VoIP. A VN is a subset of physical or virtual resources allocated to a group of customers, and VNs can be spawned from other VNs by recursively allocating resources [10]. According to ASA, IP packet delivery is considered as a basic service which consists of the underlying bandwidth, buffering and routing resources, and VN as a composite service which is built on the basic IP transport service. In this context, a VN is a customer of the transport network provider, and purchase QoS guaranteed IP transport services via VN–NP SLAs.

In ASA, the DiffServ/MPLS network is also managed by a CARB. As we mentioned in Section 2, TEQUILA is an architecture for DiffServ/MPLS network management, consistent with the self-management principle. In Fig. 5,

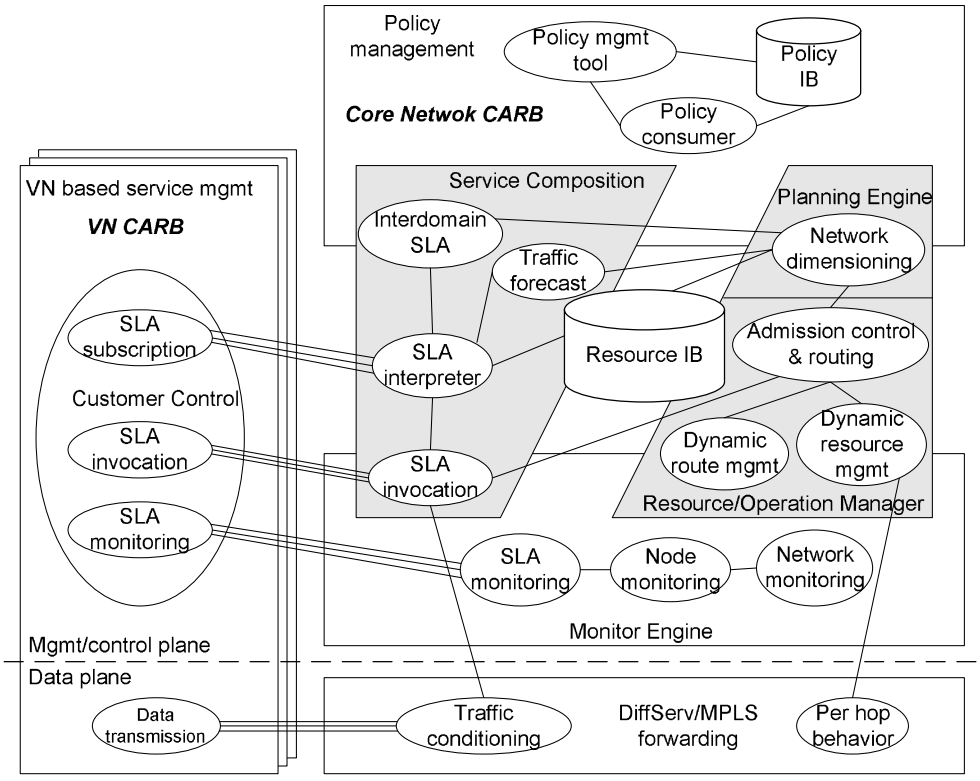

Fig. 5. The CARB for the DiffServ/MPLS core transport network.

we show that TEQUILA can be mapped to the ARB architecture to function as the core network CARB. In order not to clutter the figure, we do not show the Account Manager, Customer IB, Service IB, and Knowledge IB there.

In Fig. 5, the DiffServ/MPLS network provides IP transport service to multiple VNs. Each VN negotiates a SLA with the network provider to purchase a certain amount of bandwidth (for simplicity, here we only consider bandwidth for resource allocation) with certain QoS guarantee between some ingress/egress pairs of the core network. The SLA interpreter (or SLA translator) will map the boundary resource commitment to internal network resource requirements according to the MPLS trunk deployment. The SLAs and the corresponding internal resource requirements will be saved as Service Info. in the Service IB. SLA invocation refers to the phase that the service composer determines whether the requested resources are available or not in the network given the current network configuration; if yes, the service composer will notify the resource manager to configure related network devices for resource allocation.

The traffic forecast module is the "glue" between the service provisioning part and the resource management part of the CARB. The module estimates the long-term traffic load in each DiffServ service class based on the current and some historical SLA subscription information (saved in the Service IB), and forwards such traffic load information to the network dimensioning module. By knowing the network topology, the network dimensioning will determine the label switched path (LSP) deployment over the network and calculate the bandwidth provisioning directives for each class at each link. The network dimensioning directives are forwarded to the admission control and routing module as "soft" resource partitions, leaving space for traffic fluctuations to be handled by dynamic route and resource management techniques.

We have mentioned that ARB is an analogy of the autonomic element. In Fig. 5, resource management is indeed operated according to the "monitoring, analysis, planning, and execution" control loop, which is the principle for autonomic element design [1]. The monitoring and planning (network dimensioning) parts in Fig. 5 are self-explanatory. The SLA interpreter and traffic forecast modules analyze the resource requirements from VNs. On-line resource allocation is then executed by the admission control/routing module and dynamic route and resource management modules, following the network planning guidelines. The autonomic nature of the system is also reflected in that the dynamic management modules can in turn affect the network dimensioning and SLA negotiation. For example, when certain device failure happens within the network, the dynamic route management module will try to re-route affected traffic flows to other LSPs and the dynamic resource management module will adjust the bandwidth allocation (within the policy allowed range) on related links to serve those rerouted flows. If the dynamic management modules are not capable enough to solve the problem, they will send a request for network redimensioning. If the problem still remains unsolved, an SLA renegotiation procedure has to be started. The above inter-

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

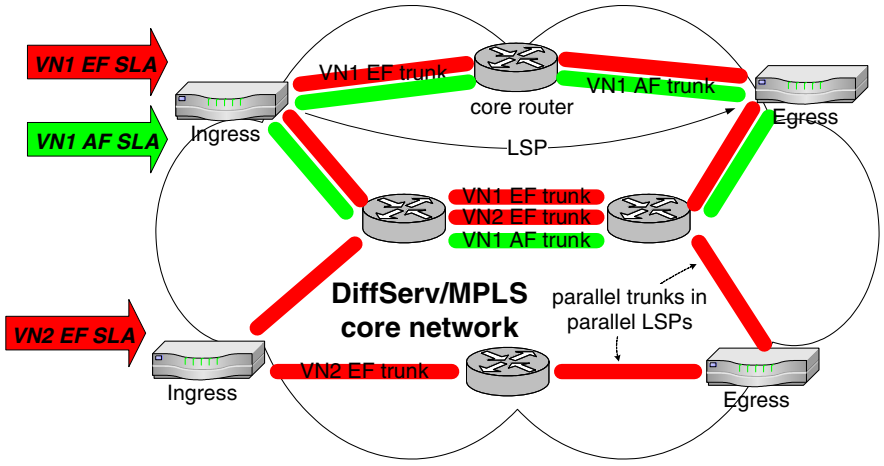

Fig. 6. The path-oriented infrastructure and MPLS trunk deployment.

action process can also be triggered if the traffic load changes and consistently deviates from the original traffic load estimation. In the autonomic control loop, the monitoring part is essential for problem determination.

5.3. Path-oriented bandwidth management

In the remainder of this section, we present a path-oriented bandwidth management scheme that can be used by the core network CARB to implement the VN-based service provisioning over a DiffServ/MPLS network.

(1) *Two-layer label stacking*. In order for the core network CARB to allocate network resources to a VN, a mechanism is required for resource partitioning. To support VN based resource management in a DiffServ/MPLS network, we propose a two-layer label stacking scheme to achieve both DiffServ-aware traffic engineering (TE) [9] and VN identification. With label stacking, an LSP between a pair of ingress/egress points can comprise a set of microflows from different VNs. Consider a certain LSP is used to deliver traffic from both VN A and VN B. During forwarding, the outer label determines where to forward the packet and the DiffServ per-hop behavior (PHB) [8], and the label switched router (LSR) only applies label swapping to this outer label. The inner label is for VN identification and checked to measure VN based bandwidth usage and QoS performance. Such information is used for VN–NP SLA monitoring. The label stacking scheme can also be used to support the routing and forwarding when VNs span multiple DiffServ/MPLS domains as proposed in [35], where the ingress routers of a domain can use multiple routing/forwarding tables to separate VN based BGP routing information and use a two-layer label stacking within a domain.

(2) *Path-oriented bandwidth allocation*. In Fig. 5, at the VN–NP interface the SLA interpreter is responsible for translating the VN–NP SLA resource commitment to a uniform *per-VN, per-class, per-ingress/egress pair* SLA format[2] for network dimensioning. Here, we present a path-oriented dimensioning approach, which is also the basis for the autonomic inter-SLA resource sharing to be presented later. We assume that there exists a routing algorithm which can set up several parallel paths for each ingress/egress pair, and the paths are fixed as LSPs. All traffic traversing an ingress/egress pair is distributed among these LSPs for load balancing purposes. A *traffic trunk* is defined as a logical pipeline within an LSP, which is allocated a certain amount of capacity to serve the traffic associated with a certain SLA. Therefore, an LSP between an ingress/egress pair may carry multiple traffic trunks associated with different SLAs, and traffic belonging to different trunks can be discriminated by the label stacking scheme mentioned earlier. The path-oriented infrastructure and MPLS trunk deployment is illustrated in Fig. 6.

In this path-oriented approach, all the per-VN, per-class, per-ingress/egress resource commitments are mapped to bandwidth allocation at each traffic trunk by network dimensioning. The dimensioning problem is normally formulated as an optimization problem subject to the constraints: (1) the total bandwidth allocated to parallel traffic trunks associated with an SLA should meet the SLA resource commitment; and (2) the total bandwidth allocation at a link does not exceed the physical link capacity. At each router, the total bandwidth allocation for a DiffServ PHB is derived by summing the bandwidth allocation of all the trunks of the same class that traverse that router along a given output port. With feasible bandwidth allocation for each PHB, the specific scheduling algorithm,

[2] Virtual networks can be dedicated according to QoS classes, application types or different customer organizations. Therefore, a VN–NP SLA may require IP transport services from different DiffServ service classes.

for example the priority queueing or weighted fair queueing (WFQ), can be designed and configured correspondingly to guarantee the resource allocation and packet level QoS requirements.

In the core network CARB, the network topology, trunk deployment, network dimensioning results and real-time bandwidth usage are tracked in the Resource IB. Such information is used to support connection (flow, or call) admission control. According to the DiffServ terminology, a centralized entity, the *bandwidth broker* [11], performs resource management and network configuration; therefore, bandwidth broker is the core network CARB in the DiffServ context. Hereafter, ''core network CARB'' and ''bandwidth broker'' will be used interchangeably in the DiffServ context for convenience. In the path-oriented environment, admission control and routing are correlated and jointly controlled by the same module as shown in Fig. 5. Each time a new connection request arrives at a certain ingress router, it is forwarded to the bandwidth broker. By checking the stored status information, the controller will select a traffic trunk according to the routing algorithm and make an admission decision according to the resource availability of the selected trunk. The decision will then be delivered back to the corresponding ingress router. If accepted, flow related information is stored in the Service IB associated with each edge router, and bandwidth usage information is updated in the Resource IB. A detailed design of the data structures used in the bandwidth broker and the edge routers is presented in [36].

Admission control can be measurement-based or analysis-based. Measurement based approach can achieve high resource utilization. However, measurement-based admission control in a VN environment incurs new issues. When multiple VNs share the same DiffServ network, the bandwidth is only virtually partitioned in the control/management plane. In the data plane, traffic from all VNs is aggregated into DiffServ classes. So even if some spare capacity is detected by measurement, it is really hard to tell which VN the spare capacity belongs to, and fairly distributing the spare capacity among VNs is also difficult. Therefore, we suggest an analysis and measurement combined admission control approach for efficient QoS and resource management. Specifically, each VN independently calculates an initial effective bandwidth [27,37] for its traffic flows to encapsulate the packet level QoS and design its admission control algorithm to guarantee the connection level QoS, i.e. the connection blocking probability. In the transport network, the bandwidth broker determines on which traffic trunk the accepted flow should be placed. The bandwidth usage and achieved QoS for the aggregated traffic are to be measured, and such information is used to tune the effective bandwidth to take the statistical multiplexing gain into account. With effective bandwidth based resource allocation, bandwidth usage and leftover capacity of each trunk (and therefore of each VN) can be readily obtained by tracking the arrival and completion of connections. It is noteworthy that the effective bandwidth based packet-level and connection-level QoS management is particularly suitable for the VoIP service.

6. Autonomic inter-SLA resource sharing

The autonomic service architecture is a SLA-centric management system. At the SLA level, the transport network resources are shared by a set of SLAs. For each SLA, the resource requirement is determined in accordance with the management policy to guarantee QoS under an engineered traffic load (which is the estimated long-term average traffic demand). The planning component of the core network CARB will find an optimal solution to accommodate all the SLA resource requirements by the network dimensioning as we discussed in last section. However, due to the random nature of traffic, the network dimensioning is effective only over the long-term horizon. In operation, the short-term traffic load may be higher or lower than the engineered load in an SLA, i.e. SLA is *overloaded* or *underloaded*, respectively. With hard static resource partitioning, the overloaded SLAs will suffer degraded QoS while the spare resources in the underloaded SLAs are wasted. Therefore, in the core network CARB, dynamic resource management is used to handle the traffic load fluctuation for higher resource utilization and better QoS.

As we mentioned in Section 2, the SLA management scheme based on virtual partitioning [27] is efficient in resource utilization, but may lead to SLA violation due to the static configuration independent of actual network conditions. In this section, we present an adaptively self-configuring and self-optimizing resource sharing technique called *bandwidth borrowing* for DiffServ/MPLS networks, by which the SLA compliance and high resource utilization can be achieved simultaneously. For convenience, we consider a connection or a flow's packet level QoS is encapsulated by the notion of effective bandwidth, and term a bandwidth guaranteed connection/flow as a *call*. A SLA handles QoS and resources at the call level. The scheme in [27] also considers SLAs handling call level QoS.

6.1. SLA with call-level differentiation

To facilitate resource sharing, we propose that the definition of an SLA be extended with a statement of the QoS and resource commitment in an underloaded period and a call-level differentiation agreement as follows:

(1) A nominal capacity is allocated in the SLA according to the target call arrival rate to satisfy the target call blocking probability (CBP).
(2) During operation, according to the actual call arrival rate measured by a traffic monitor at the ingress router, two resource utilization states are associated with an SLA. The SLA is said to be in a *lendable* state, if the actual call arrival rate is less than the engineered load and the target CBP can be satisfied

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

with a smaller serving capacity. Such a smaller QoS guaranteeing bandwidth is defined as the QoS ensuring (QoSE) bandwidth. Otherwise, the SLA is in the *unlendable* state.

(3) In the lendable state, the QoSE bandwidth of the SLA is guaranteed to its traffic flows to meet the QoS. The unused bandwidth within the nominal capacity can be exploited by all related SLAs.
(4) In the unlendable state, the nominal capacity of the SLA is guaranteed to its traffic flows. Furthermore, the SLA may accept overloaded traffic by borrowing bandwidth from the lendable SLAs. The traffic flows accepted with borrowed bandwidth are tagged as *out profile* calls, and the flows accepted with the nominal capacity are considered as *in profile* calls.
(5) When an SLA returns to the unlendable state from the lendable state, the QoSE bandwidth is increased to the nominal capacity to claim back resources of the SLA. Some out profile flows from the borrower SLAs may be preempted during the bandwidth claiming.

In the above SLA definition, the possible preemption of the out profile calls is considered as the QoS differentiation between the in profile traffic and the out profile traffic (The in profile calls cannot be preempted). The counterpart differentiation scheme at the packet level is the assured forwarding PHB [38]. Such call-level differentiation efficiently utilizes the spare capacity as well as avoids the malicious overloading. The call-level differentiation can bring a more customer-friendly service model. When a flow is to be served as an out profile call, a message can be sent to the customer before the actual service regarding the SLA load status and flow admission status. The customer can then determine to continue or try at a later time, or send the most important information first.

Note that the call-level differentiation concept and bandwidth borrowing concept presented in the above SLA definition can be applied to any service scenario where a bandwidth requirement can be determined based on the QoS specification, traffic load, and control policies. Here, we consider SLAs handling call level QoS for the sake of concreteness.

6.2. Call admission control

The proposed bandwidth borrowing scheme can be conveniently implemented by the traffic trunk based resource allocation that we discussed in Section 5.3. In the Resource IB, each traffic trunk has an information record including the nominal capacity from network dimensioning, the QoSE bandwidth (the SLA QoSE bandwidth is distributed among associated trunks), and the current bandwidth usage. If an SLA is in unlendable status, the QoSE bandwidth is set as the nominal capacity. A traffic trunk is considered as *notfull* if current usage is less than the QoSE bandwidth, otherwise as *full*. The traffic trunk information records will be used by the bandwidth broker to determine whether a call can be accepted as in profile or out profile. Three basic principles guiding the admission control are:

(1) In an SLA, if the current resource usage (the SLA bandwidth usage can be obtained by summing up all related trunk usages) is less than the QoSE bandwidth, the new call will be treated as in profile and its acceptance is guaranteed.
(2) Out profile calls can be accepted by fully exploiting unused bandwidth. The unused bandwidth along a selected LSP will be checked link by link.
(3) If the bandwidth broker finds that there is no bandwidth available for an in profile call, some out profile calls should be preempted.

6.3. Dynamic spare bandwidth distribution

In the bandwidth borrowing scheme, the spare bandwidth (nominal capacity minus the QoSE bandwidth) is calculated at edge routers for related SLAs and then distributed to associated traffic trunks. The spare capacity at a certain link can be indirectly obtained by summing the spare capacity over all the trunks traversing the link. It is obvious that the distribution of spare capacity directly determines the resource utilization that can be achieved. A straightforward approach is that the spare capacity from an SLA is evenly distributed to related traffic trunks. The even distribution may not be the best solution, because (1) the bandwidth borrowing may not happen on all the routes, and (2) the traffic loads and resource sharing levels on different routes, and therefore on different links, are different. Ideally, the QoSE bandwidth (correspondingly the spare bandwidth) should be distributed in such a way that leads to the maximum resource utilization. It is very difficult, if not impossible, to derive a centralized, optimal on-line distribution technique. Therefore, we propose a distributed algorithm implemented at edge routers to dynamically adjust the spare bandwidth distribution in the network. The basic idea for the dynamic spare bandwidth distribution is as follows.

Consider a certain SLA supported by multiple traffic trunks. Initially, the QoSE bandwidth (and correspondingly the spare bandwidth) is evenly distributed to all the traffic trunks. When a traffic flow is admitted onto a traffic trunk, at each link along the LSP holding the new flow, traffic trunks having spare capacity are searched, which are termed as *lender trunks*. Note that the lender trunks may belong to different lendable SLAs. Each lender trunk searched will push a certain amount of QoSE bandwidth to randomly selected one of its fellow parallel trunks (belonging to the same SLA), i.e. to pull the spare capacity within the SLA to the links currently having new arrivals. Such pushing/pulling procedure is executed in all the SLAs. Seen from the network level, each LSP tries to grab the

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

spare capacity within the network with a *pulling force*; such pulling force is proportional to the traffic load on the LSP. In the pulling competition, those paths with stronger pulling force that reflects heavier traffic load obtain a larger part of the spare capacity. At the same time, the over-pulling will be confined so that each LSP has a chance to be the winner of the pulling competition when traffic load changes dynamically. Due to the length limit of the paper, we omit the details of the spare bandwidth distribution, which are presented in [36].

At this point we want to emphasize that the bandwidth borrowing scheme is designed with the objective that the DiffServ/MPLS core network CARB can be implemented as an example of an ASA-compliant system. In bandwidth borrowing, the traffic load will be monitored and the bandwidth reservation to guarantee QoS will be adjusted adaptively. Moreover, the bandwidth sharing over the network is also dynamically adjusted according to network status and traffic load variations. Although here we mainly consider the call-level QoS, the call-level differentiation and bandwidth borrowing concept can be applied in accordance very general management policies.

7. Performance evaluation

In this section, we use a case study to demonstrate the performance of the bandwidth borrowing scheme. The units used for related measures are second for time, capacity unit (c-unit) for link/trunk/SLA capacity and efficient bandwidth usage, call/second for call arrival rate, and c-unit/call for effective bandwidth. The network topology, SLA and trunk deployment for the case study are shown in Fig. 7, where 8 edge routers and 1 core router are connected through 12 links. Five SLAs are supported by this network, and each SLA is served with parallel traffic trunks. Assume Poisson arrivals for each SLA, and exponentially distributed call holding times with mean of 1. The effective bandwidth associated with each SLA is equal to 1. Erlang-B formula is used to calculate the CBP, i.e. $\mathrm{CBP} = E$ (call arrival rate, SLA capacity). The target CBP for each SLA is 0.01. The engineered call arrival rates for SLA-1 to SLA-5 are $(\lambda_p^1, \lambda_p^2, \lambda_p^3, \lambda_p^4, \lambda_p^5) = (46.9, 29, 29, 29, 29)$ and the corresponding SLA capacity planning is $(C_1, C_2, C_3, C_4, C_5) = (60, 40, 40, 40, 40)$ to achieve the target CBP for the engineered traffic load. The capacity of each link is 60 for links 1 and 2, and 40 for the other links. We run Monte Carlo simulations to estimate the CBP for each SLA.

7.1. Efficient inter-SLA resource sharing

In the first example, the simulation starts at $t = 0$, and ends at 48,000. Traffic for each SLA starts with the engineered call arrival rate, and the call arrival rates for some SLAs are changed at certain moments to create the overloaded and underloaded periods. The actual call arrival rate for each SLA, λ_d^i $(i = 1, \ldots, 5)$, and the corresponding QoSE bandwidth for each trunk, R_j $(j = 1, \ldots, 11)$, are given in Table 1. The measured call blocking probability for each SLA with bandwidth borrowing is presented in Fig. 8. From Table 1 and Fig. 8, we have the following observations.

(1) During the time period of (0, 6000), all the SLAs maintain the specified rates and no inter-SLA resource sharing happens, i.e. the network resources

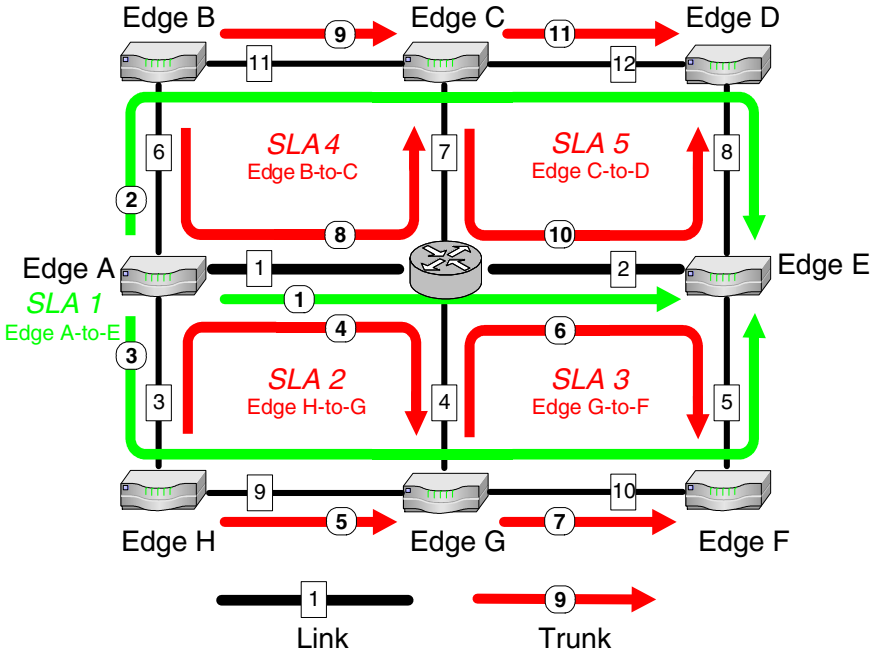

Fig. 7. Network topology, SLA and trunk deployment for simulation.

Table 1
The call arrival rate and QoSE bandwidth for each SLA

t	λ_d^1	(R_1, R_2, R_3)	λ_d^2	(R_4, R_5)	λ_d^3	(R_6, R_7)	λ_d^4	(R_8, R_9)	λ_d^5	(R_{10}, R_{11})
0	46.9	(20,20,20)	29	(20,20)	29	(20,20)	29	(20,20)	29	(20,20)
6000	62.6	(20,20,20)	29	(20,20)	29	(20,20)	29	(20,20)	29	(20,20)
12,000	62.6	(20,20,20)	14.4	(11,12)	29	(20,20)	29	(20,20)	14.4	(12,11)
36,000	62.6	(20,20,20)	14.4	(11,12)	29	(20,20)	29	(20,20)	29	(20,20)

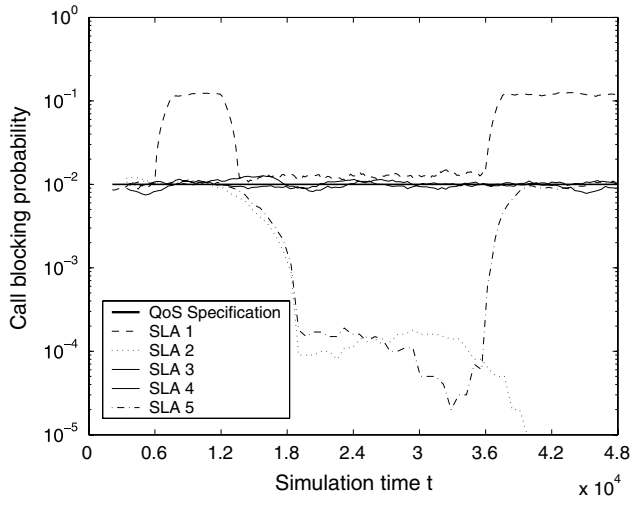

Fig. 8. The performance with bandwidth borrowing.

are utilized by all the SLAs in a *complete partition* (CP) manner. Each SLA achieves the target CBP of 0.01.

(2) After $t = 6000$, SLA-1 becomes overloaded. The bandwidth borrowing does not happen until $t = 12000$ when SLA-2 and SLA-5 become underloaded. During the time period of (6000, 12,000), SLA-1 has the CBP of $E(62.6, 60) \approx 0.1208$. Other SLAs continue with the engineered call arrival rate and achieve the target QoS.

(3) During the time period of (12,000, 36,000), SLA-2 and SLA-5 become underloaded. As $E(14.4, 23) \approx 0.01$, the QoSE bandwidth is set as 23 (with spare capacity of $40 - 23 = 17$) and at first evenly distributed between the two parallel trunks as 11 and 12 (fractional division of a capacity unit is assumed impossible). With bandwidth borrowing, SLA-1 can utilize the spare capacity from SLA-2 and SLA-5 along trunk-1, but not along trunk-2 and trunk-3, because there is no spare bandwidth on link-6, link-11, link-10, and link-5. Therefore, the bandwidth pushing/pulling algorithm will adjust distribution of the spare capacity in SLA-2 and SLA-5 to 16 c-units on trunk-4 and trunk-10, and 1 c-unit on trunk-5 and trunk-11, respectively. The 1 unit of spare capacity reserved on trunk-5 and trunk-11 is to maintain the "spare" property of the routes, which is the strategy to avoid over-pushing. The spare capacity of 16 on link-1 is shared between SLA-1 and SLA-2 according to the *complete sharing* principle, and on link-2 between SLA-1 and SLA-5. As the new call arrival rate at trunk-1 is much larger than that on trunk-4 and trunk-10, it grabs almost all the spare capacity to achieve the CBP of $E(62.6, 60 + 16) \approx 0.0125$. The simulation results match very well with the numerical estimations. The spare bandwidth of 1 U on trunk-5 and trunk-11 can only be accessed by SLA-2 and SLA-5, respectively. In addition to some spare capacity from trunk-4 and trunk-10, both SLA-2 and SLA-5 can achieve the CBP much smaller than $E(14.4, 24) \approx 5.7 \times 10^{-3}$, which are also demonstrated by the simulation results.

(4) After $t = 36{,}000$, SLA-5 changes back to the unlendable state, and both QoSE bandwidths of trunk-10 and trunk-11 are then reset to 20 to claim back the lent out bandwidth. Since the spare capacity on link-2 is not available anymore, the bandwidth borrowing along trunk-1 also stops. When bandwidth borrowing stops, SLA-2 exclusively utilizes the spare capacity, achieves the CBP of $E(14.4, 40) \approx 1.48 \times 10^{-8}$.

(5) For the CAC without bandwidth borrowing, each SLA always exclusively uses its nominal capacity. No inter-SLA resource sharing happens. The difference between the borrowing and non-borrowing scenarios exists in the time period of (12,000, 36,000), where the underloaded SLA-2 and SLA-5 in the latter achieve the CBP of near 0, but the throughput only increases from $14.3(\approx 14.4 \times [1 - E(14.4, 24)])$ to $14.4(\approx 14.4 \times [1 - E(14.4, 40)])$ as compared with the former. On the other hand, the bandwidth borrowing can trade the slightly (almost unnoticeably) degraded QoS of underloaded SLAs for an obvious throughput increase in the overloaded SLA-1, from $55.0(\approx 62.6 \times [1 - E(62.6, 60)])$ to $61.8(62.6 \times [1 - E(62.6, 60 + 16)])$.

(6) With bandwidth borrowing, high resource utilization is achieved with SLA compliance, that is, the CBP in underloaded and normally loaded cases never exceeds the target specification of 0.01.

7.2. Bandwidth borrowing compared to virtual partitioning

As we mentioned in Section 2, SLA management based on virtual partitioning can also achieve efficient resource

Table 2
Call blocking probability and efficient bandwidth usage in different bandwidth sharing schemes

Schemes	SLA-1	SLA-2	SLA-3	SLA-4	SLA-5	EBU
CP	0.3769	1.48×10^{-8}	0.0100	0.0100	1.48×10^{-8}	144.6668
VP	0.1718	0.0356	0.0688	0.0690	0.0367	159.4450
BR	0.1459	0.0028	0.0086	0.0087	0.0027	166.3356

utilization compared to the CP scheme, but may lead to QoS or SLA violation in some cases. In this example, we demonstrate that bandwidth borrowing (BR) can achieve higher resource utilization than VP, while SLA compliance is always guaranteed in a heavily loaded network.

The network dimensioning is still as given in Fig. 7. We consider a scenario where SLA-1 is overloaded and SLA-2 and SLA-5 are underloaded, and the on-line measured call arrival rates for SLA-1 to SLA-5 are $(93.8, 14.4, 29, 29, 14.4)$. The QoSE bandwidths of SLA-2 and SLA-5 are calculated equal to 23 and each has an initial even distribution of $(11, 12)$ over its associated 2 trunks. We compare the performance of the CP, VP, and BR schemes. Under each resource sharing scheme, the CBP for each SLA is measured, by which the total efficient bandwidth usage (EBU) over the network is calculated according to the approach given in [27]. All the results are presented in Table 2. Note that the results for the BR scheme are conservatively measured, where all the preempted calls are treated as blocked calls.

With the CP scheme, traffic service in each SLA works independently. The CBP is directly obtained from the Erlang-B formula and serves as the performance benchmark. Obviously, CP leads to the worst QoS and resource utilization due to the static resource allocation. Both VP and BR schemes can significantly improve the resource utilization. However, in the VP scheme, the overload SLA-1 leads to QoS violation in all the other SLAs, including both the underloaded and normally loaded ones. With BR scheme, the resource utilization is even better than the VP and SLA compliance is guaranteed. The good performance of the BR scheme stems from the more aggressive bandwidth usage by out profile calls,[3] dynamic adjustment of spare bandwidth distribution, and protection of the in profile flows via preemption scheme.

8. Concluding remarks

This paper presents a general approach to autonomic service management using ASA, which will allow service providers to reduce the costs of delivering services to customers, and to manage services and network resources under a uniform framework. ASA is based on two main concepts: virtualization of physical resources using a Common Resource Format, and autonomic service delivery using a hierarchy of Autonomic Resource Brokers. The CRF allows the service delivery framework applicable to all types of services by using the appropriate resource metrics corresponding to each service's needs. The hierarchical service view makes ASA expandable to next-generation services by allowing flexible, scalable, and recursive service composition out of existing virtual resources and services.

The ASA is a conceptual architecture, and its realization for real-world services needs the development of automation techniques in various areas. Specifically, in this paper, we apply ASA to the management of a VN-based service over a DiffServ/MPLS network. We present a path-oriented implementation for VN-based bandwidth management, based on which an autonomic inter-SLA resource sharing scheme, the bandwidth borrowing scheme, is proposed. By monitoring the actual traffic load conditions, the bandwidth borrowing scheme can automatically adjust the resource allocation to each SLA when necessary, so that the spare capacity in underloaded SLAs can be exploited and QoS specification of all the SLAs are always guaranteed.

The journey to a fully autonomic service architecture is still in its early stages. We are currently elaborating ASA's design. First, we are defining formats for the information bases, and most importantly for the management policies. Second, we are defining an XML format for service and SLA templates, as well as the CRF interface for virtual resources. Third, we are defining interfaces among ARBs, and exploring several ARB topologies and assessing the best in a particular situation. Fourth, we are developing algorithms for each ARB functional block. Finally, we are investigating the application of ASA for peer-to-peer service management.

[3] In bandwidth borrowing, all the unused bandwidth in the network can be exploited to accept traffic flows. The aggressive bandwidth usage will be preempted when necessary. QoSE bandwidth is just a "soft" state, which determines when the preemption should be executed.

References

[1] J.O. Kephart, D.M. Chess, The vision of autonomic computing, IEEE Computer 36 (2003) 41–50.
[2] IBM Corporation, An architectural blueprint for autonomic computing, White Paper 2003.
[3] D. Xiangdong et al., Autonomia: an autonomic computing environment, Proceedings of the IEEE International Performance, Computing, and Communications Conference (2003) 61–68.
[4] M. Agarwal et al., AutoMate: enabling autonomic applications on the grid, Proceedings of the Autonomic Computing Workshop (2003) 48–57.
[5] IBM and University of Berkeley, Oceano Project, <http://www.research.ibm.com/oceanoproject/>.
[6] S. Graupner, A. Andrzejak, V. Kotov, H. Trinks, Adaptive control overlay for service management, Proceedings of the Workshop on the Design of Self-Managing Systems, International Conference on Dependable Systems and Networks (DSN) (2003).
[7] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol label switching architecture, IETF RFC 3031 (2001).
[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An architecture for differentiated services, Internet RFC 2475 (1998).

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

[9] F. Le Faucheur et al., Multi-protocol label switching (MPLS) support of differentiated services, IETF RFC 3270 (2002).
[10] A. Leon-Garcia, L.G. Mason, Virtual network resource management for next-generation networks, IEEE Commun. Mag. 41 (2003) 102–109.
[11] P. Trimintzios et al., A management and control architecture for providing IP differentiated services in MPLS-based networks, IEEE Commun. Mag. 39 (2001) 80–88.
[12] S. Krause, T. Magedanz, Mobile service agents enabling intelligence on demand in telecommunications, Proceedings of IEEE GLOBECOM'96 (1996) 18–22.
[13] M. Feridun, W. Kasteleijn, J. Krause, Distributed management with mobile components, Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (1999) 857–870.
[14] H. Liu, M. Parashar, Accord: a programming framework for autonomic applications, IEEE Trans. Syst., Man. Cybern. C. 36 (2006) 341–352.
[15] D. Talia, The open grid services architecture: where the grid meets the web, IEEE Internet Comput. 6 (2002) 67–71.
[16] K. Appleby et al., Oceano: SLA based management of a computing utility, Proceedings of the IEEE/IFIP International Symposium on Integrated Network Management (2001) 855–868.
[17] HP Laboratories, The HP vision for the Adaptive Enterprise: achieving business agility, <http://h71028.www7.hp.com/enterprise/downloads/ae_business_white_paper_final0703.pdf/>.
[18] Microsoft, Dynamic Systems Initiative Overview, <http://www.microsoft.com/windowsserversystem/dsi/dsioverview.mspx/>.
[19] F. Krief, Self-aware management of IP networks with QoS guarantees, Int. J. Network Mgmt. 14 (2004) 351–354.
[20] D. Agrawal, K.-W. Lee, J. Lobo, Policy-based management of networked computing systems, IEEE Commun. Mag. 43 (2005) 69–75.
[21] Distributed Management Task Force, Common Information Model (CIM) Standards, <http://www.dmtf.org/standards/cim//>.
[22] Distributed Management Task Force, CIM Policy0 Model, v. 2.8, <http://www.dmtf.org/standards/cim/cim_schema_v281/CIM_Policy28-Final.pdf/>, 2004.
[23] B. Moore, E. Ellesson, J. Strassner, A. Westerinen, Policy core information model – Version 1 specification, IETF RFC 3060 (2001).
[24] B. Moore, Policy core information model (PCIM) extensions, IETF RFC 3460 (2003).
[25] Z. Duan, Z.-L. Zhang, Y.T. Hou, Service overlay networks: SLAs, QoS, and bandwidth provisioning, IEEE/ACM Tran. Networking 11 (2003) 870–883.
[26] J. Ash, L. Chung, K. D'Souza, W.S. Lai, H. Van der Linde, Y. Yu, AT&T's MPLS OAM architecture, experience, and evolution, IEEE Commun. Mag. 42 (2004) 100–111.
[27] E. Bouillet, D. Mitra, K.G. Ramakrishnan, The structure and management of service level agreements in networks, IEEE J. Select. Areas Commun. 20 (2002) 691–699.
[28] S.C. Borst, D. Mitra, Virtual partitioning for robust resource sharing: computational techniques for heterogeneous traffic, IEEE J. Select. Areas Commun. 16 (1998) 668–678.
[29] A. Leon-Garcia, I. Widjaja, Communication Networks: Fundamental Concepts and Key Architectures, Mc Graw Hill, 2004.
[30] The WS-Resource Framework, <http://www.globus.org/wsrf//>.
[31] OASIS Standard, Web Services Resource Framework (WSRF) v1.2, <http://www.oasis-open.org/specs/index.php#wsrfv1.2/>.
[32] H. Zhang, M. Savoie, J. Wu, S. Campbell, G.v.Bochmann and W.St. Arnaud, Service-oriented layer 1 virtual private network for grid applications, Proceedings of the International Conference on Grid Computing and Applications, Las Vegas, USA, 2005.
[33] W3C Web Services Architecture Working Group, Web Services Architecture, 2004, <http://www.w3.org/TR/ws-arch/wsa.pdf/>.
[34] J. Schonwalder, A. Pras, J. Martin-Flatin, On the future of Internet management techniques, IEEE Commun. Mag. 41 (2003) 90–97.
[35] E.C. Rosen, Y. Rekhter, BGP/MPLS IP Virtual Private Networks (VPNs), IETF RFC 4364 (2006).
[36] Y. Cheng, Efficient resource allocation in differentiated services networks, PhD thesis, Unviersity of Waterloo, 2003.
[37] F.P. Kelly, Notes on Effective Bandwidth, Stochastic Networks: Theory and Applications, Oxford University Press, Oxford, 1996, pp. 141–168.
[38] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, Assured forwarding PHB group, IETF RFC 2597 (1999).

Yu Cheng received the B.E. and M.E. degrees in Electrical Engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree in Electrical and Computer Engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003.

From September 2003 to August 2004 he was a postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Waterloo. From September 2004 to August 2006, he was a postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Toronto, ON, Canada. He joins the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, USA as an Assistant Professor in September 2006. His research interests include service-oriented networking, autonomic network management, Internet performance analysis, quality of service provisioning, wireless networks, and wireless/wireline interworking. He received a Postdoctoral Fellowship Award from the Natural Sciences and Engineering Research Council of Canada (NSERC) in 2004. He is a Member of IEEE and ACM.

Ramy Farha is a Ph.D. candidate in the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada. He received his B.Eng degree from the American University of Beirut in 2001, and his M.A.Sc degree from the University of Toronto in 2003. He received the Natural Sciences and Engineering Research Council (NSERC) PGS award and the Ontario Graduate Scholarship (OGS) award, both in 2003. In addition, he holds the Distinguished Graduate Award from the American University of Beirut in 2001. His research interests include IP mobility, passive optical networks, peer-to-peer and overlay networks, and autonomic service management.

Myung Sup Kim received the BS, MS and PhD degrees in computer science and engineering from Pohang University of Science and Technology (POSTECH), Korea, in 1998, 2000 and 2004, respectively. From September 2004 to July 2006, he was a post-doctoral fellow in the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joins the Department of Computer and Information Science, Korea University, Jochiwon, Korea, as an Assistant Professor in August 2006. His research interests include service and network management, Internet traffic monitoring and analysis, application-level networking, network-security attack detection and prevention, and autonomic network resource management. He is a member of KNOM.

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.

Alberto Leon-Garcia received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Southern California, in 1973, 1974, and 1976 respectively. He is a Full Professor in the Department of Electrical and Computer Engineering, University of Toronto, ON, Canada, and he currently holds the Nortel Institute Chair in Network Architecture and Services. In 1999 he became an IEEE fellow for "For contributions to multiplexing and switching of integrated services traffic".

Dr. Leon-Garcia was Editor for Voice/Data Networks for the IEEE Transactions on Communications from 1983 to 1988 and Editor for the IEEE Information Theory Newsletter from 1982 to 1984. He was Guest Editor of the September 1986 Special Issue on Performance Evaluation of Communications Networks of the IEEE Selected Areas on Communications. He is also author of the textbooks Probability and Random Processes for Electrical Engineering (Reading, MA: Addison-Wesley), and Communication Networks: Fundamental Concepts and Key Architectures (McGraw-Hill), coauthored with Dr. Indra Widjaja.

James Won-Ki Hong is an associate professor in the Dept. of Computer Science and Engineering, POSTECH, Pohang, Korea. He received a Ph.D. degree from the University of Waterloo, Canada in 1991 and an M.S. degree from the University of Western Ontario in 1985. He has worked on various research projects on network and systems management, with a special interest in Web, Java, CORBA, and XML technologies. His research interests include network and systems management, distributed computing, and network monitoring and planning. He has served as Technical Chair (1998–2000), Vice Chair (2003–2005) and Chair (2005-present) for IEEE Comsoc CNOM. He is also serving as Director of Online Content for the IEEE Comsoc (Jan. 2004-Dec. 2005). He is a NOMS/IM Steering Committee Member and a Standing Committee Member of APNOMS. He was technical co-chair of NOMS 2000 and APNOMS '99. He was Finance Chair for IM 2005 and Finance Chair and Chair of Local Planning Committee for NOMS 2004. He is an editorial advisory board member of JNSM and IJNM. He is also editor-in-chief of KNOM Review Journal. He is a member of IEEE, KICS, KNOM, and KISS.

Please cite this article as: Yu Cheng et al., A generic architecture for autonomic service and ..., Computer Communications (2006), doi:10.1016/j.comcom.2006.06.017.