XML Service Level Specification and Validation

Pedro Alipio, Solange Lima and Paulo Carvalho
Universidade do Minho, Departamento de Informatica,
4710-057 Braga, Portugal
{*pma,solange,pmc*}*@di.uminho.pt*

Abstract

This paper addresses the problem of formalizing Service Level Specifications (SLSs) as a first step to simplify and automate the configuration and management of multiservice IP networks. A formal representation of SLSs will allow their automatic validation and processing, fostering the dynamic negotiation of SLSs and the interoperability among service management entities. In this way, taking advantage of XML extensibility and portability, a Schema is presented describing XML SLSs sections and their contents. In addition, an XML validator tool was built to check if SLSs are correctly specified. An XML SLS for an IP telephony service is used to exemplify this proposal expressiveness.

1 Introduction

The offering of multiple QoS-based services on public IP infrastructures demands for new approaches on Internet Service Providers (ISPs) network management, where the ability for self-managing systems and services is of particular relevance. A first step toward autonomic network management involves a change in the way the contracts of service are defined and processed. These contracts are expressed through Service Level Agreements (SLAs), where a technical part, called Service Level Specification (SLS), is devoted to the specification of services. While the SLA includes legal, administrative, and economic information, the SLS includes edge-to-edge IP level information about the offered services quality. Therefore, an SLS is a set of parameters and their values which together define the service offered to a traffic stream by a network domain [8].

As ISP domains are increasing in size and in complexity, there is the urge of new network management systems that may operate without or with a minimum human intervention, capable of mapping the requirements expressed in SLA/SLSs into network configurations [15]. A first step toward developing an autonomic management system is to formalize SLA/SLSs. The main concern and objective of this paper is to provide a formal ground to express the services technical requirements, i.e. SLSs. Through a formal representation of an SLS, the automatic validation, processing and transferring of service requirements is possible, facilitating the configuration and management of network services. In this way, a formal representation of SLSs is presented resorting to Extensible Markup Language (XML), which brings an additional advantage as regards a subsequent SLS validation. This SLS validation is a crucial step to undertake before SLS processing. In fact, although several validation schemas could be applied to validate XML documents, XML Schema was adopted as it is a widely supported and rich grammar specification language, allowing the description of the document structure, elements and types. In this paper, an XML Schema for SLS contents description is presented containing the grammar rules used to validate XML SLS. Although, a DiffServ network is usually used as an example, the proposed SLS specification intends to be generic and might be applied to other multiservice networks such as MPLS. The proposed specification also covers IPv6 specific information such as the flow label present in IPv6 packet headers.

This paper has the following structure: Section 2 presents related work concerning service specification; Section 3 discusses the XML Schema model for XML SLSs focusing on aspects such as its structure, elements and data types; Section 4 presents an XML validator utility developed to validate XML SLSs; Section 5 includes an example to demonstrate the expressiveness of the proposed SLS XML specification; finally, the conclusions and future work are presented in Section 6.

2 Related Work

An SLA is defined as a contract between a customer and a service provider or between service providers, specifying administrative and technical service information. The technical part of an SLA, i.e. the SLS, defines the expected

Table 1. XML Schema graphic model

Graphic	XML Schema
	Element
	Attribute
	All
	Choice
	Optional
	Repeatable
	Optional and repeatable

service level, QoS parameters and traffic control issues[1].

The definition of SLSs, apart from being a key aspect for QoS provisioning, provides a valuable input for network configuration. Therefore, defining a standard set of SLS parameters and semantics is crucial for ensuring edge-to-edge QoS. Several working groups are committed to SLS definition [14, 5, 17, 16, 7] and management [14, 5, 13, 4, 2, 18]. However, these proposals do not consider any particular language to formalize the semantics, parameters or the negotiation requirements. In the web services domain, several proposals have been presented to formally define services [12, 11, 9, 6]. Unfortunately, these languages focus mostly on higher level issues instead of specifying services at IP level. Several of these proposals consist of XML based description languages. However, XML has enough expressive power by itself to describe service specifications. Therefore, there is no need of developing new specification languages based on it. In fact, XML has several characteristics which turn it into a good choice for SLS definition. XML is extensible, if new service requirements are identified, then they can be accommodated easily. XML uses text files and is a *tag* based language, therefore, it may be processed in any platform and be transported over any type of network. Finally, there are several Application Programming Interfaces (APIs) for several programming languages to parse, validate and process XML documents.

3 An XML Schema for SLSs

As XML Schemas tend to be very verbose, a graphical notation is used to represent the structure, types, elements and content models of documents. Table 1 illustrates all graphics used to model the SLS XML Schema.

[1] An SLA may include multiple SLSs, however, an SLS is usually related to a service class usage. In the context of network management, as the relevant part of an SLA is the SLS, from now on we refer uniquely to SLSs and assume that each SLS is embedded in the corresponding service class.

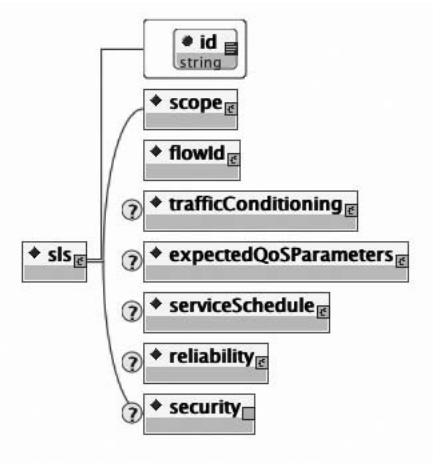

Figure 1. XML SLS Structure

3.1 SLS Structure

An SLS should include the following sections: (i) the scope of the service, defining the boundaries of the region over which the service will be enforced; (ii) the flow identification section, defining the fields which identify an individual or aggregate flow; (iii) the traffic conditioning section, containing rules to identify in or out-of-profile traffic; (iv) the expected QoS performance parameters; (v) the service scheduling section, defining the time period when the service is available; (vi) the service reliability section, defining parameters related to the consistency and reliability of the service to be provided and finally (vii) the security section was left for future work.

Figure 1 illustrates the XML Schema model reflecting these sections. While Scope and FlowId are compulsory sections all the others are optional. All services must be clearly identified as well as the entry and exit points of the ISP domain used by each service. Traffic Conditioning, Expected QoS Parameters, Service Schedule and Reliability sections are optional because they may not be required for all service specifications. For instance, for best-effort services, traffic conformance and expected QoS parameters are usually not defined. If service scheduling is not defined a permanent service is assumed. Reliability and security are optional for all types of network services.

3.2 Scope

The boundaries of topological regions must be specified as they are enforcement locations for the QoS policies. Therefore, the scope of an SLS indicates where the QoS policy has to be enforced when offering a specific service to customers. It is expressed through a set of *ingress* and

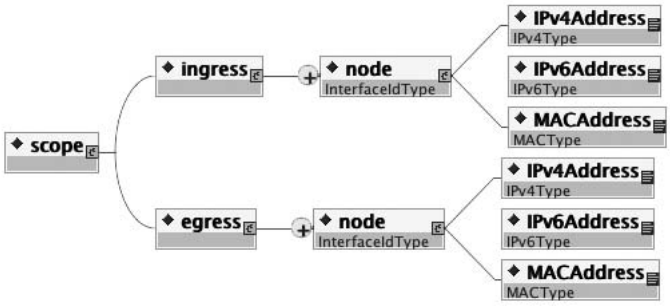

Figure 2. Scope Section of an SLS.

egress nodes, denoting the entry and exit nodes of the network domain, respectively.

The requirements for the scope are expressed through the element *scope* in the XML Schema, containing one *ingress* and one *egress* elements. Each of these elements consist of several *node* elements, containing identifications of different types of interfaces. The element *node* is of type *InterfaceIdType*. This Complex Type may be either an IPv4 address, an IPv6 address or a MAC address. Figure 2 illustrates the scope section of an SLS.

3.3 Flow Identification

Flow Identification is required to identify over which IP packets a QoS policy is due to be enforced regarding the specified service. Therefore, it must be included as an SLS section. Service traffic may consist of a microflow (identified by a combination of a Source Address, a Destination Address, a Source Port and a Destination Port) or a macroflow. A macroflow is an aggregate flow aiming at a specific service and may be specified through information such as: (i) a set of microflows - in this case, pairs of IP (Source Address and Destination Address), transport information (Source Port, Destination Port) and Protocol Id; (ii) one or a set of DiffServ Codepoints (DSCPs) - identifying one or a set of valid marks for traffic entering the domain; (iii) other IP information such as Protocol Id and IPv6 FlowLabel; (iv) any set combining these fields.

In the XML Schema three types of information are considered: DiffServ; generic IP and transport information. If *diffServInfo* element is specified in the FlowId Section of XML SLS, then it contains one or a set of DSCPs elements. If *IPInfo* element is specified, then it must include one or several of the following elements: *source*; *destination*; *protocolNumber* and *IPv6FlowLabel*. Finally, if *transportInfo* element is specified, then it must include one or several *sourcePort* and *destinationPort* elements. Figure 3 illustrates the Flow Id Section in the XML Schema.

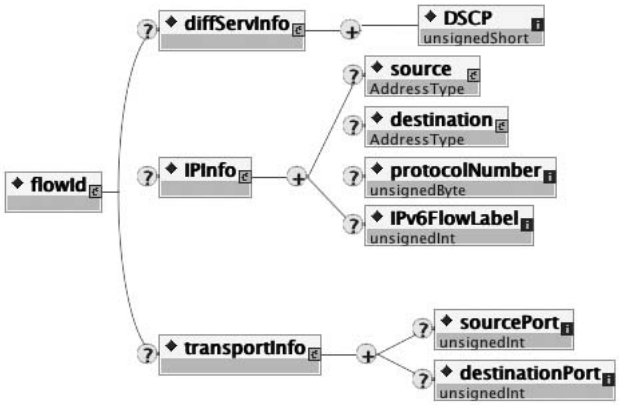

Figure 3. Flow Id Section.

3.4 Traffic Conditioning

The Traffic Conditioning Section of an SLS includes a Conformance Algorithm, Conformance Parameters and the specification of the action to take on excess traffic. This SLS section is illustrated in Figure 4.

Although several algorithms may be applied for service traffic conditioning, following the configuration guidelines in [1], a Token Bucket and a Two Rate Three Color Marker [10] are included in the XML Schema. Other algorithms may be applied by extending it. Conformance parameters may comprise both algorithm and common traffic parameters. Algorithm parameters are specified as elements of the *conformanceAlgorithm* element. For example, if the *tokenBucket* element is specified, then the *tokenBucketRate* and the *tokenBucketDepth* elements must also be included. Traffic parameters such as: *peakRate, meanRate*, *minRate*, *MTU* and *minPacketSize* may be included in the *otherParameters* element.

The Conformance Algorithm allows to identify in-profile or out-of-profile traffic. Traffic tagged as out-of-profile is considered traffic in excess. As consequence, several actions may be applied to excess packets. As an example, the XML schema includes *drop, mark* and *shape* actions. Nevertheless, it may be extended with more actions. The actions *shape* and *mark* must include the shaper and DSCP parameters, respectively.

3.5 Expected QoS Parameters

The expected QoS parameters express the service level guarantees the network offers to the service customer. Four QoS parameters are considered: (i) One-way Delay, measurement period, optional quantile; (ii) Interpacket Delay Variation, measurement period, optional quantile; (iii)

Packet Loss Ratio, measurement period; (iv) Throughput, measurement period. Other performance parameters may be expressed by extending the XML Schema. This SLS Section is illustrated in Figure 5. All these parameters refer to an IP flow as described in the Flow Id Section and to the scope area described in the Scope Section of the SLS. The performance values may be qualitative or quantitative being measured in a time interval basis. QoS performance values may describe the worst case values. Quantiles may also be included to express the probability of exceeding those values.

3.6 Service Schedule

The service schedule indicates the start and end time of a service, i.e., the time period in which the service is due to be offered to the customer. In addition, Service Scheduling information is particularly relevant as regards the medium and long term forecast of network resource planning and provisioning. This section is optional, therefore, when it is not specified a permanent service is assumed. It might be expressed by the following parameters: (i) time of the day range; (ii) day of the week range; (iii) month of the year range; (iii) year range. As several occurrences of each of these elements are allowed, it is possible to specify several start and end times for the service. The Service Schedule SLS Section is illustrated in Figure 6.

3.7 Reliability

The reliability section of an SLS indicates the Maximum mean Down Time (MDT) and the Maximum Time To Repair (MTTR) in case of service breakdown. The MDT value should be related to the service scheduling period. This SLS section is described in Figure 7.

3.8 Simple Types

Several XML Schema Simple Types were created to validate the data which the elements may contain. These data types use regular expressions to define the data patterns allowed in each element. Table 2 includes the Simple Types defined in the XML Schema.

4 Validation

Having defined the XML Schema for QoS oriented network services a validation tool was developed to verify the corresponding specifications. The validation is performed at two levels: (i) if the XML is correct and (ii) if it was written according to the XML Schema. In the second case, both structural and syntactic validation is performed.

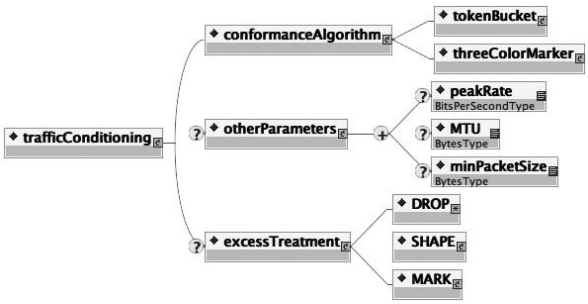

Figure 4. Traffic Conditioning Section.

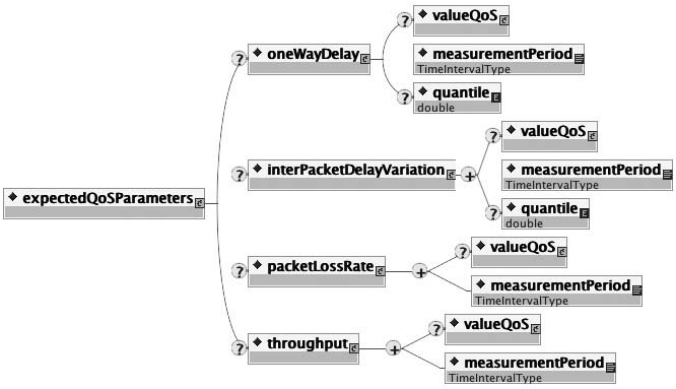

Figure 5. Expected QoS Parameters.

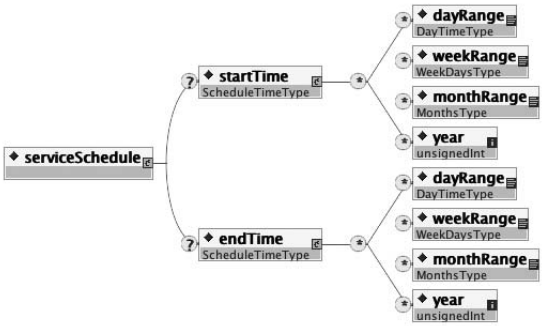

Figure 6. Service Schedule Section.

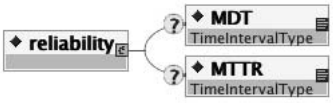

Figure 7. Reliability Section.

Table 2. XML SLS simple data types.

Simple Type	Regular Expression
IPv4Type	\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
IPv6Type	\d{1,3}\.\d{1,3}\.\d{1,3}\. \d{1,3}:\d{1,3}:\d{1,3}
MACType	[a-zA-z0-9]{2}:[a-zA-z0- 9]{2}:[a-zA-z0-9]{2}:[a- zA-z0-9]{2}
NetPrefixIPv4Type	\d{1,3}\.\d{1,3}\.\d{1,3}\. \d{1,3}/\d{1,2}
NetPrefixIPv6Type	\d{1,3}\.\d{1,3}\.\d{1,3}\. \d{1,3}:\d{1,3}:\d{1,3}/\d{1,2}
BitsPerSecondType	\d+[kmgKMG]?bps
BytesType	\d+[kmgKMG]?B
TimeIntervalType	\d+((h)|(m)|(s)|(ms)|(us)){1}
BinByteType	[01]{8}
WeekDaysType	([Ss]sunday)|([Mm]onday) |([Tt]uesday)|([Ww]ednesday) |([Tt]hursday)|([Ff]riday)|([Ss]aturday)
MonthsType	([Jj]anuary)|([Ff]ebruary) |([Mm]arch)|([Aa]pril)|([Mm]ay) |([Jj]une)|([Jj]uly)|([Aa]ugust) |([Ss]etember)|([Oo]ctober) |([Nn]ovember)|([Dd]ecember)
DayTimeType	\d{2}h\d{2}

The Xerces-c library from Apache[2] was used to build a SAX2 [3] based validator and parser which runs from the command line in UNIX based operating systems. The CMAKE utility was used to provide cross-platform compilation [3]. Figure 8 illustrates the validator's architecture.

A file called *main.cpp* has the *c++ main* function which handles several XML related parameters obtained from the command line invocation. Those parameters specify the XML file to be validated, the text encodings used in the XML file and whether the XML file is due to be checked for correctness or validated according to an XML Schema model. Before the command line parameters are processed, the XML Xerces-c platform is initialized. If an error occurs, then the program will stop running. A *SAX XML Reader* instance is then initialized. This class instance is used to read, parse and validate the XML file. Validation errors are handled by a class derived from the *HandlerBase* defined in files *validator.hpp* and *validator.cpp*. This class is then set to be the error handler for the *SAX XML Reader* class instance. The error related methods were overrided to display output messages according to the detected error. Finally, the memory occupied by the created instances is released and the XML file validation output displayed.

[2]Apache Xerces-c `http://xml.apache.org/xerces-c/`
[3]CMake `http://www.cmake.org`

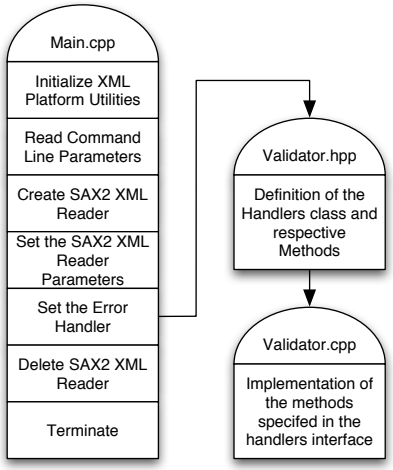

Figure 8. SLS validator utility architecture.

5 An SLS for an IP Telephony Service

This section shows an example of an IP telephony service (VoIP) offering with a guaranteed throughput of 64Kbps, a one-way transit delay of 100ms for a 0.001 quantile and without packet loss. This service applies to traffic marked with DSCP 46 and is provided permanently. The XML SLS is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<sls id="sls-example-VoIP"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="sls.xsd">
    <scope>
        <ingress>
            <node>
               <IPv4Address>100.1.1.1</IPv4Address>
            </node>
        </ingress>
        <egress>
            <node>
                <IPv4Address>100.1.2.254</IPv4Address>
            </node>
        </egress>
    </scope>
    <flowId>
        <diffServInfo>
            <DSCP>46</DSCP>
        </diffServInfo>
    </flowId>
    <trafficConditioning>
        <conformanceAlgorithm>
            <tokenBucket>
                <tokenBucketRate>64Kbps</tokenBucketRate>
                <bucketDepth>5KB</bucketDepth>
            </tokenBucket>
        </conformanceAlgorithm>
        <excessTreatment>
            <DROP />
        </excessTreatment>
    </trafficConditioning>
    <expectedQoSParameters>
        <oneWayDelay>
            <value>
                <quantitative>100ms</quantitative>
            </value>
```

```
            <measurementPeriod>1m</measurementPeriod>
            <quantile>0.001</quantile>
        </oneWayDelay>
        <packetLossRate>
            <value>
                <quantitative>0</quantitative>
            </value>
            <measurementPeriod>1m</measurementPeriod>
        </packetLossRate>
        <throughput>
            <value>
                <quantitative>64Kbps</quantitative>
            </value>
            <measurementPeriod>1m</measurementPeriod>
        </throughput>
    </expectedQoSParameters>
    <reliability>
        <MTTR>30s</MTTR>
    </reliability>
</sls>
```

6 Conclusions

This paper has focused on formalizing SLSs, as a way to foster the self-configuration and management of multi-service IP networks. The formal specification of SLSs is recommended as it allows: (i) automatic validation and processing of SLSs, (ii) dynamic SLS negotiation and (iii) to create a common ground for interoperability among management entities both intra and interdomain. XML was chosen to define the structural and syntactic requirements of SLSs through XML Schemas, allowing a rich description of the elements and element types. An XML Schema grammar was presented specifying the structure and the parameters required for service specification. The example provided (VoIP service) has illustrated the simplicity and expressiveness of this approach. Current work focuses on mapping XML SLS parameters into network service configuration.

References

[1] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for DiffServ Service Classes. Internet Draft (work in progress), October 2004.

[2] P. Bhoj, S. Singhal, and S. Chutani. SLA Management in Federated Environments. *Computer Networks*, 35(1), Jan. 2001.

[3] D. Brownell. *SAX2*. O'Reilly, 2002.

[4] J. Chen, A. McAuley, V. Sarangan, S. Baba, and Y. Ohba. Dynamic Service Negotiation Protocol (DSNP) and Wireless Diffserv. in ICC'02, Apr. 2002.

[5] A. Diaconescu, S. Antonio, M. Esposito, S. Romano, and M. Potts. Cadenus D2.3 - Resource Management in SLA Networks. Cadenus Project IST-1999-11017, May 2003.

[6] S. Frolund and J. Koisten. QML: A Language for Quality of Service Specification. Technical Report HPL-98-10, Hewlett Packard, 1998.

[7] D. Goderis, Y. T'joens, C. Jacquenet, G. Memenious, G. Pavlou, R. Egan, D. Griffin, P. Georgatsos, L. Georgiadis, and P. V. Heuven. Service Level Specification Semantics, Parameters, and Negotiation Requirements. Internet-Draft,draft-tequila-sls-03.txt, work in progress, October 2003.

[8] D. Grossman. New Terminology and Clarifications for Diffserv. IETF RFC 3260, 2002.

[9] X. Gu, K. Nahrstedt, W. Yuan, D. Wichadakul, and D. Xu. An XML-based Quality of Service Enabling Language for the Web. *Journal of Visual Language and Computing (JVLC), special issue on Multimedia Languages for the Web*, 13(1):61–95, February 2002.

[10] J. Heinanen, T. Finland, and R. Guerin. RFC 2698 - A Two Rate Three Color Marker. IETF - RFC2698, September 1999.

[11] A. Keller and H. Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1), 2003.

[12] D. Lamanna, J. Skene, and W. Emmerich. SLAng: A Language for Defining Service Level Agreements. In *9th IEEE Workshop on Future Trends in Distributed Computing Systems - FTDCS 2003 (Puerto Rico, May 2003)*, pages 100–106. IEEE-CS Press, 2003.

[13] M. Mellia, C. Casetti, G. Mardente, and M. Marsan. An Analytical Framework for SLA Admission Control in a Diffserv Domain. In *IEEE INFOCOM'03*, Mar. 2003.

[14] P. Morand, M. Boucadair, P. Levis, R. Egan, H. Asgari, D. Griffin, J. Griem, J. Spencer, P. Trimintzios, M. Howarth, N. Wang, P. Flegkas, K. Ho, S. Georgoulas, G. Pavlou, P. Georgatsos, and T. Damilatis. Mescal D1.2 - Initial Specification of Protocols and Algorithms for Inter-domain SLS Management and Traffic Engineering for QoS-based IP Service Delivery and their Test Requirements. Mescal Project IST-2001-37961, Jan. 2004.

[15] A. G. Prieto and M. Brunner. Sls to diffserv configuration mappings. In *12th International Workshop on Distributed Systems: Operations & Management*, 2001.

[16] S. Salsano, F. Ricciato, M. Winter, G. Eichler, A. Thomas, F. Fuenfstueck, T. Ziegler, and C. Brandauer. Definition and Usage of SLSs in the Aquila consortium. IETF draft: draft-salsano-aquila-sls-00.txt (work in progress), Nov. 2000.

[17] A. Sevasti and M. Campanella. Service Level Agreements Specification for IP Premium Service. Geant and Sequin Projects, Oct. 2001.

[18] P. Trimintzios, I. Andrikopoulos, G. Pavlou, C. Cavalcanti, D. Goderis, Y. T'Joens, P. Georgatsos, L. Georgiadis, D. Griffin, C. Jacquenet, R. Egan, and G. Memenios. An Architectural Framework for Providing QoS in IP Differentiated Services Networks. In *7th IFIP/IEEE International Symposium on Integrated Network Management - IM'01*, 2001.