RC 22071 (99002) 22 May 2001 Computer Science

IBM Research Report

Electronic Service Level Agreement (eSLA) for Application Hosting

Robert Kearney, Richard King, Martin Sachs, Asit Dan, Daniel Dias
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

Electronic Service Level Agreement (eSLA) for Application Hosting

by

Robert Kearney, Richard King, Martin Sachs,

Asit Dan, Daniel Dias

Abstract

Current service level agreements (SLAs) between service providers and their customers are English-language documents that often deal only with the availability of the contracted services. Formalizing these agreements as electronic service level agreements (eSLA) and extending the scope of SLAs to capture additional IT resource level concerns, make it possible for the system management software at each of the parties to directly monitor and enforce these agreements. We focus on the agreement information to be captured in an eSLA that describes services and their resource requirements, tools and supporting services in forming an eSLA, and a concrete web hosting example to demonstrate the deployment and enforcement of the eSLA for managing network bandwidth. We have prototyped this service example along with the tools for creation and deployment of an eSLA. Finally, we have devised an XML DTD for eSLA definition.

1 Introduction

Currently, agreements between ISPs and their customers regarding levels of service are English-language documents. They most often deal only with availability, although some also provide guarantees of throughput and Web site performance[Chatterton]. This, however, is becoming increasingly unsatisfactory, as more customers make more detailed demands of their providers, leading to increasingly complex service level agreements[Chorley][UUNETSLA][Verma]. The burden then falls on the provider to decipher the agreement actually reached and to relate the specific terms of each agreement to the corresponding system management practices and controls necessary to implement that agreement.

An electronic service level agreement (eSLA) documents the agreement between a resource or service provider (e.g., a Web hosting service provider) and a resource consumer (or customer, e.g., a large online department store) regarding quality of service in the operation of a managed entity hosted by the resource provider. An example of a managed entity is a web server. Here, the resource provider hosts a web site on behalf of the customer in order to allow users to access the customer's site. In addition to specifying the agreement between the parties, the eSLA is used as input to the mechanisms that will enforce and monitor the agreement.

The concept of electronic contract was introduced in[Dan98] where the elements of an interaction protocol for integrating cross-enterprise business applications are captured as an electronic contract. Such an electronic contract has been used in configuring business application and/or B2B gateways[Dan01]. Emerging standards for dealing with cross-enterprise application integration. Examples are Web Services Description Language[WSDL] and ebXML Collaboration Protocol Profiles and Agreements[CPPCPA]. Our current work on electronic service level agreements dealing with IT level resources is complementary to the above work.

1.1 General Principles

The eSLA is an electronic contract between a customer and a resource provider. It consists of a list of services required by the customer (and hence furnished by the resource provider), and for each service a set of parameters which describe a level of service the customer requires, and that the service provider is able to support. This level of support is referred to as the Quality of Service (QoS). A resource provider may support multiple customers which may use different sets of resources. Often, however, the customers will be competing for resources shared among them. The eSLA describes only the resources required by a single customer and the service provider. It is the responsibility of the service provider to consider all eSLAs when guaranteeing a level of service to any individual customer.

Quality of service has two dimensions: the level of service promised by the resource provider and the load imposed by the customer. Service policies expressed in the eSLA specify responsibilities of customers and resource providers. They state both what the customer can request and what level of service the resource provider will guarantee. The former are stated in terms of the service provided by the resources, while the latter are expressed in terms of the resources. Resource consumption may vary by time of day, day of week, or other conditions. The quality of service which the resource provider will guarantee may also vary as a function of conditions such as time of day. Guarantees may include such parameters as service time (i.e., response time within the service provider's domain), throughput, percentiles of these, or aggregations over various time scales. If the resource provider fails to adhere to its side of the eSLA, it might be due to the customer loading the system in excess of what was agreed to rather than to some failure on the part of the resource provider. In this case, it would be the customer, rather than the resource provider, which owes a penalty. Violating a policy (by either the customer or the resource provider) should lead to some action. The violation might simply generate an informational message (which needs to be sufficiently descriptive), or might result in some automatic recourse.

1.2 Outline of Paper

The remainder of this paper is divided into the following outline structure.

Service Level Parameters – discusses workload requirements, quality of service, and types of services and how they are interrelated;

eSLA Life Cycle – examines the life cycle of an eSLA, from negotiation to deployment to monitoring and enforcement;

eSLA Definition – provides an overview of the actual structure of an eSLA;

Prototype - demonstrates use of an eSLA in managing a web-hosting service, especially with respect to network bandwidth. The components of the prototype include an authoring tool for eSLA and a code generation tool that generates configuration files from an eSLA;

Summary and Conclusions - contains some final points about the eSLA;

Appendix – details examples of files described in this document, including the eSLA DTD, an example eSLA, and generated configuration files which are used in the prototype discussed herein.

2 Service Level Parameters

The eSLA is prepared by representatives of the resource provider and customer using an authoring tool. It documents the agreement between the resource provider and the customer on the values of service parameters of various kinds. The types of service parameters include:

- A list of services to be supported by the resource provider, and their associated resource parameters. Service descriptions include how an service can be accessed by end-users;
- Customer workload expectations, which describe how much load the customer expects to place on the service provider's installation; and
- QoS guarantee, specifying a guarantee by the resource provider on providing a level of service as it will be perceived by a user of the service (such as a shopper using the customer's web site).
- The parameters also include prices for guaranteeing certain levels of QoS for the expected workload, policies for dealing with workload exceeding expectations as well as penalties for not meeting the QoS guarantees. Both customer workload expectations and QoS guarantees can be time-varying, i.e., may have different values for different time periods.

Once an agreement is reached, the service provider's installation is configured using the information provided in the eSLA so as to enforce it. Instrumentation is set up to verify that the service levels specified in the eSLA are monitored.

2.1 Service List

A customer specifies a list of services (i.e., applications) and their associated resource parameters to be supported by a resources provider. Each service represents access to a hosted application by the end-users, and hence, a distinguishable user experience. Examples of services include browsing a storefront, purchasing, making reservations, and playing an online video/audio. Note that a customer may want to represent many different transaction types by a single service, when a single QoS guarantee is desired for all transactions. The description of a service includes:

- *Access method for this service*, i.e., protocol (e.g., HTTP, FTP) and associated parameters (e.g., url, port number);
- *Per user access resource requirements* , i.e., transaction resource characteristics (e.g., bandwidth, processor requirements);
- *User access independent resource requirements* , i.e., Database size (e.g., disk space requirements), background processes (e.g., processor requirements, memory).

The exact set of resource parameters that must be specified will depend on the availability of detailed characterization of the applications, and the parameters that are considered critical for providing QoS. Any parameters that are not fixed or can vary on a per transaction basis are expressed as expected workload parameters.

2.2 Quality Of Service

To ensure end-user satisfaction and to ensure available resources for supporting customer workload, especially at critical/peak hours, the customer requires certain guarantees on the QoS. Specification of QoS includes the followings:

- *Per-access QoS specification* includes

    - bandwidth, peak bandwidth, jitter and maximum delay for continuous media workload;
    - maximum service time for interactive workload and/or bandwidth for other data intensive applications;
- *Aggregate specification* may include the statistical distribution of the per-access QoS parameters, and aggregate measures such as throughput;
- *Availability specification* includes guaranteed average up-time for the service (say, per week), and maximum downtime before the service is restored;
- *Contingency specification* lists policies by which the customer workload is managed if and when the monitored workload exceeds the specification of expected values. This may include but not be limited to:
    - prioritization across transaction classes for resource scheduling at the resource provider;
    - increased allocation on a best effort basis;
    - initiating a new level of QoS for the entire workload.

2.3 Customer Workload Expectations

A resource provider can guarantee a customer its specified QoS only if the provider has enough resources for this workload. The resource provider may dedicate a fixed amount of resources for satisfying the peak demand at a higher price or may employ dynamic resource allocation using various optimization algorithms at a lower price. To fulfill this demand, the resource provider needs to know the expected customer workload.

Examples of parameters which govern the customer expected workload are:

- *Variability on transaction resource specification* may include processor requirements, in terms of a standard benchmark value, and bandwidth;
- *Access request arrival specification* may include rate, distribution and peak on the transaction demand.

2.4 Time-Varying Service Requirements

In some scenarios, the services a consumer requires vary due to well-defined factors. These include:

- *Time of day:* For example, on-line shopping is cyclical, with more shopping being done between late morning and early evening, than overnight;
- *Special events or seasons:* For example, the Professional Golf Association Tour web site is much more active during major tournaments than other, non major tournaments, and is heavily influenced by which individuals are entered or leading the tournament;
- *Differentiation of services*: For example, more resources are required for performing purchases than simply browsing. Currently, such differentiation can be indicated by protocol, IP address or port. However, other categorizations may be required for other service types.

The eSLA must be able to categorize services according to time and types.

2.5 Reports to the Customer

The eSLA defines what is reported to the customer and how frequently. An e-mail address of the recipient is specified in the reporting section. There are at least three types of reports:

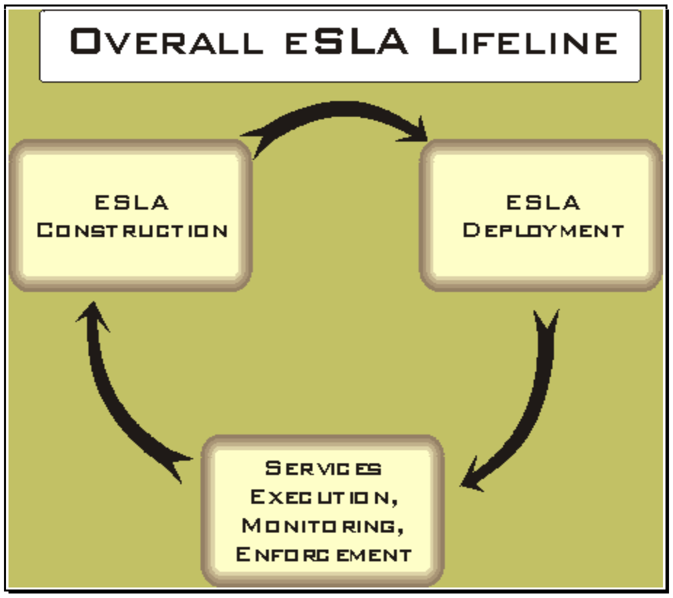
1. a standard report, which describes average usage of each resource being monitored;
2. a modification report, indicating that the service provider has changed the level of service being provided, as allowed in this eSLA;
3. an emergency report, which is issued when a violation of the service level agreement has occurred.

For standard reports, a time interval describes how often the report is issued. This also implies over what interval the average is taken.

For emergency reports, a time interval indicates the minimum time between reports. Since emergencies can occur more often, a report contains the number of violations since the last emergency report.

3 eSLA Life Cycle

A scenario between a customer and a service provider takes place in two steps: creation of the eSLA, and using the eSLA to configure and monitor the installation that provides the services. The overall eSLA life cycle is illustrated in the below.



3.1 Creating the eSLA

An eSLA has to be negotiated between the two parties involved. Once the resources the customer requires have been expressed, the eSLA has to be validated to insure that the service provider can support these resources. Finally, a price for the service is presented back to the customer. The diagram below illustrates the process of eSLA construction.

1. An eSLA is authored, capturing the resources the customer and service provider have tentatively agreed upon;

2. Current eSLAs and the current system configuration are used by the validation process;

3. If the current system configuration cannot provide the resources, then one of two choices must be made:

   a. The service provider can attempt to add resources to his system in order to support the added service load;

   b. The eSLA can be renegotiated to reduce the service load requested by the customer;

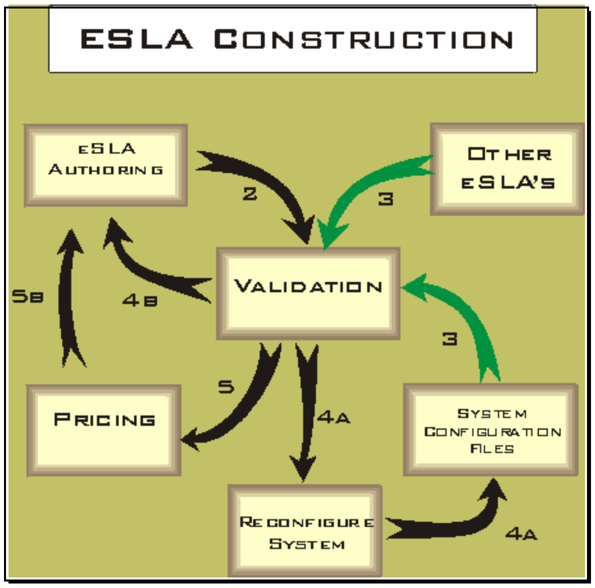
4. Once a system configuration exists which will support the customer's request, a price for the service is obtained. The customer can either:

   a. accept the price;

   b. attempt to renegotiate a better price from the service provider;

   c. renegotiate the resource levels in order to obtain a different price;

5. Once the customer and service provider agree to all resource levels, the eSLA is published and made available to the service provider's system administrator, and to the customer.



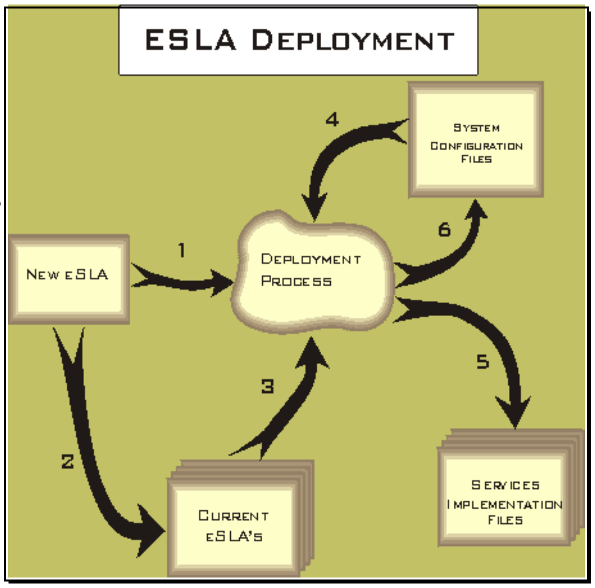
3.2 eSLA Deployment

Once an eSLA has been created, it is available to the deployment apparatus to

- generate files needed for the reconfiguration of the installation hardware; and
- generate files for the services described in the eSLA.

The process of eSLA deployment is illustrated in the following figure.

1. The eSLA is passed to the deployment process;
2. The eSLA is also entered in the library of all eSLAs which are handled by the installation;
3. The deployment process considers both the new services' requirements, as well as all of the services being handled already;
4. The deployment process examines the current installation configuration, and based upon the findings in steps (3) and (4), creates a new installation configuration.
5. Files representing the new installation configuration are generated. These files are available to the system administrators, if hardware



reconfiguration is necessary;

6. A set of files representing the new services is generated. These files may consist of XML, Java or some other representation of the services and resource requirements.

It is anticipated that the entire installation may participate in a "Quality of Service" framework, whereby the files generated in steps (5) and (6) can be used by the framework for automated service configuration and deployment.

3.3 Enforcing and Monitoring the eSLA at Run Time

Additional files representing monitoring and enforcement policies can be generated from the eSLA. While enforcement policies are not part of this paper, one such policy could be implemented to detect whether either the service provider or customer is chronically in violation of the eSLA. Then the system could suggest to the parties that a renegotiation of the eSLA be undertaken.

4 Structure of Electronic Service Level Agreements

The eSLA structure depends upon the service requested by the customer (and provided by the service provider). We make the assumption that the eSLA is an agreement between two parties (customer and service provider); this assumption is based only on experience to date, and may have to be revised if the eSLA is to be applied to a service requiring an agreement by several parties.

An eSLA is formulated as an eXtensible Markup Language[XML] document, with a Document Type Definition (DTD) defined specifically for the eSLA. Other formulations could be applied, using the concepts described here. An authoring tool can be used to build eSLAs; we describe the Watson Authoring Tool for this purpose. See sections 2 and 3 of the appendix for examples of the DTD and an eSLA.

The eSLA consists of the following top level sections:

Contract Information - This section details the name and the duration of the eSLA. The XML element structure for this section of the eSLA is present in all eSLAs, and has the following form:

- **Contract Name** - The value of this element is to be some unique text which describes the purpose of the contract;
- **Duration** - This element consists child elements whose values are a starting and ending date for the contract. It is optional, and if omitted, then the contract's lifetime is open-ended. For example:

```
<Duration>
    <StartDate>2000-03-16</StartDate>
    <EndDate>2001-03-15</EndDate>
</Duration>
```

Contract Participants Section - This section details the organizations which are parties to the eSLA. The XML element structure for this section is invariant. The following are the elements in this section:

- **Role** - This element defines the participant which plays each role (customer or service provider) in the contract. There is one **Role** element for each participant in the contract. The role is defined by an attribute, Name, which has one of the following values: "customer" or "service provider".
- **Role Player** - This child element of the Role element identifies the organization performing the associated role by means of a standardized identifier such as a DUNS number. The Type

attribute defines the type of identifier being used. Other child elements identify the name of the organization, its address, and representatives who act as contacts during both the negotiation of this contract and the execution of the service. An example of the Role element is:

```
<Role Name="Customer">
    <RolePlayer MemberId="0001" IdCodeType="ZZ">
    <OrganizationName>Store 1</OrganizationName>
    <Contact Type="MainContact">
         <Name>Kris Kringle</Name>
         <Address>
              <AddressLine>34th St</AddressLine>
              <City>New York</City>
              <State>NY</State>
              <Zip>10101</Zip>
              <Country>USA</Country>
         </Address>
         <EMail>kris@store-1.com</EMail>
         <Telephone>212-555-1234</Telephone>
         <Fax>212-555-4321</Fax>
    </Contact>
    </RolePlayer>
</Role>
```

Services Section - This section describes the services being provided between the participants. This section is divided into subsections, where each subsection is headed by a **Service** element which in turn has an attribute indicating the service type. The attributes currently defined include**:**

- **HTTP** - the service is a Web Hosting service;
- **FTP** - the service is an FTP site hosting;
- **Audio** - the service is a multimedia source;
- **Backup** - the service is a data archive/backup service.

Within each subsection are elements which identify information which is relevant to the service type:

- **ClusterAddress** - the IP address providing the service;
- **Port** - the port within the above address designated for the service (e.g., port 80 for HTTP, and port 443 for SSL);
- **BenchmarkValue** - a value, in terms of units of a benchmark whose name appears as an attribute of this element. A benchmark is a standardized measurement of the system (e.g., CPU, database) utilization to perform a particular task. A benchmark value indicates the weight of the customer's actual application relative to the weight of the programs described in the benchmark. Any benchmark acceptable to both parties can be used. For example:

```
<BenchmarkValue Name="Webstone">100</BenchmarkValue>
```

- **Report** - zero or more elements identifying report types, recipients and intervals such as:

```
<Report Type="Standard" Recipient="servicewatcher@store-1.com">
     <Interval>600</Interval>
</Report>
```

- **Event** elements which indicate a service resource level which becomes the current level when the event occurs. Within each **Event** element is a **ResourceList** element. It, in turn contains an element for each resource required by the service. The value of the element is the level for that resource. Currently defined resources include:

- Guaranteed Bandwidth, which defines the bandwidth the service provider has assured the customer will be available. Any request for more will be made available on a "best effort" basis;
- Maximum Bandwidth, which defines the greatest bandwidth the service provider will make available to the customer at any cost. Any request for more will not be honored;
- Guaranteed Throughput, which defines the number of service requests the service provider has assured the customer will be available;
- Maximum Throughput, which defines the greatest number of service requests the service provider will allow at any cost. Any requests for more will not be honored;
- Maximum Service Time, which defines the maximum time a service request will take, once the request has begun to be serviced.

An example of an event element is:

```
<Event Value="Initial">
    <ResourceList>
        <GuaranteedBandwidth>
            <GuaranteedValue>10000</GuaranteedValue>
            <Availability>
                <RunningInterval>100</RunningInterval>
                <SampleInterval>2</SampleInterval>
                <Limit>.95</Limit>
            </Availability>
        </GuaranteedBandwidth>
        <MaximumBandwidth>15000</MaximumBandwidth>
    </ResourceList>
</Event>
```

It is possible to define ***availability*** in terms of guaranteed bandwidth and guaranteed throughput, where if certain levels of either is not met, then the corresponding resource is deemed to be unavailable. A possible measure of availablity can be found in section 1 of the appendix.

Finally, the above guarantees and resources can be associated with a ***contract state*** which is set by a well defined **event**. When a particular event occurs, the management framework evaluates the associated constraints and if satisfied, in response switches to the service resource level indicated under this **Event** element there can be zero or more **Constraint** elements which:

- define an expression that evaluates to a boolean *true* or *false*;
- define a service resource level that becomes the current level when the expression evaluates to a set value.

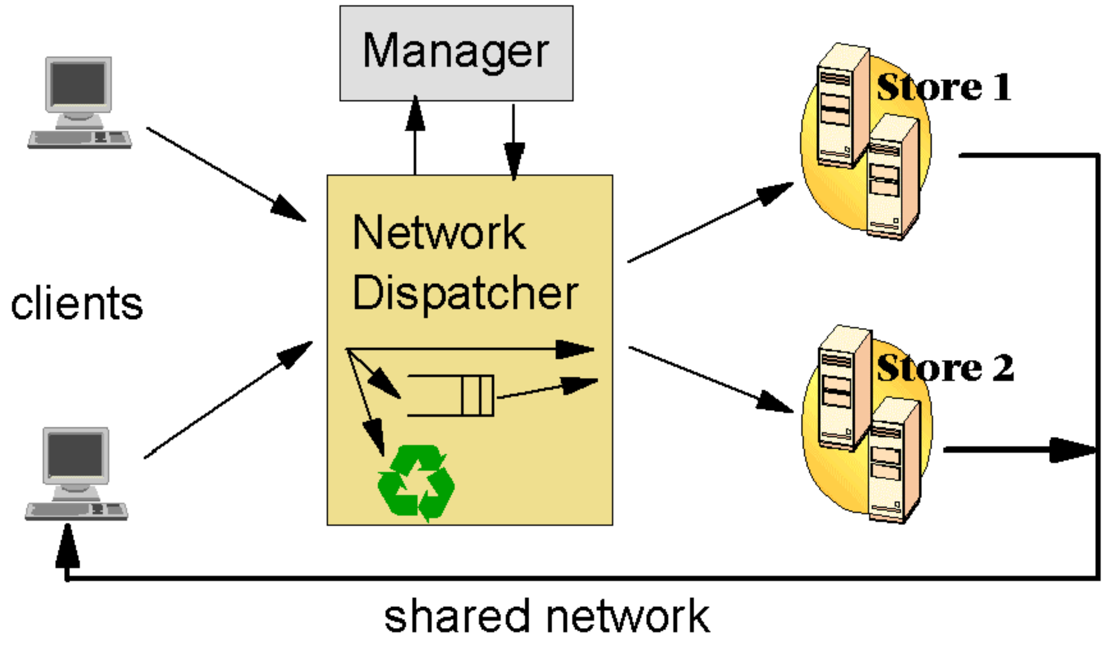
5 Prototype Description

Some document definitions (DTDs), some documents based on those DTDs, and some software to manipulate them have been assembled to demonstrate how some of these pieces that we have discussed would fit together. This prototype also helps us keep our work grounded in reality.

5.1 Build Time

The eSLA prototype builds on a separate research project to enhance the IBM Network

Dispatcher[King]. These enhancements make it possible for Network Dispatcher to control the consumption of network bandwidth by the users of a server farm. Configuration of the Network Dispatcher involves the building of files that describe how much bandwidth is available, what different services are being hosted, how much bandwidth is reserved for each port of that service, and which services (or ports thereof) have priority in the sharing of otherwise unused bandwidth. The structure of the prototype is shown in the following figure.



In addition to the eSLAs, two documents that describe the server farm are needed. These are:

- the **Farm document**, which describes the network capacity of the server farm, and the set of Network Dispatchers that are available;
- the **Layout document**, which describes which eSLAs have been assigned to each Network Dispatcher.

Examples of these XML documents can be found in sections 4 and 5 of the appendix.

The prototype validator takes these documents and combines information in them with information taken from the eSLAs mentioned in the Layout document. It then verifies that the overall load on the server farm would not be excessive. When an excessive load is detected, it is flagged as an error. Otherwise, configuration files for each Network Dispatcher are produced, plus a configuration file for the global manager that communicates with the Network Dispatchers to adjust the bandwidth budgets assigned to each of them based on the actual loads being presented by the services being controlled by them.

In the eSLA example in the appendix, the Services Section describes the bandwidth requirements for hosting this organization's Web site. It indicates the IP address that user's requests will be coming to and, for the two ports that will receiving requests (the normal Web server port 80 plus the SSL port 443), the amount of bandwidth guaranteed to each of them. It also indicates that whenever there is contention between these ports, or the ports of other services being hosted at the same server farm, that the SSL port 443, on which orders would be arriving, will have priority over the browsing port 80, and any other services ports that have priority values greater than 10. This contention would, of course, only be for amounts of bandwidth over and above the amount guaranteed to them, which is always available regardless of the activity levels of any other ports or services. Note that, as they stand, this eSLA is valid relative to the sample farm document, since the bandwidth guarantees add up to less than the aggregate bandwidth of the farm, but also note that it is the sum over all of the eSLAs in force that must obey that constraint, not just each individual eSLA.

5.2 eSLA Authoring Tool

Once the service provider and customer have agreed to the details of an agreement, the document itself must be fashioned. Any authoring tool, such as a text editor, can be used to create an eSLA. However, due to the complex syntax and semantics of the eSLA, it would be difficult to create one which is syntactically and semantically valid without an eSLA-aware authoring tool.

Our authoring tool uses a *Document Model* which contains semantic information about the eSLA in order to assure semantic validity of the final eSLA. The document model provides:

1. defaults for fields which are relatively consistent over eSLAs;
2. a guided process of well-documented choices, with the provision that the only choices which are available are those which are semantically correct;
3. semantic constraints, so that, for example, if inclusion of one field implies that another field must or must not be included, the document model, together with the Authoring Tool, enforces the inclusion or exclusion.

The document model is also constructed using the Authoring Tool. The *model maker* is responsible for embodying the above features into the model. The model maker is generally a person with expert knowledge of the semantics of the eSLA. Once a document model is created, then an individual with a passing knowledge of the structure of an eSLA can, by using the model and the Authoring Tool, easily generate a valid, usable eSLA.

5.3 eSLA deployment

A key feature of the eSLA is that it is machine readable, and therefore it is the basis for configuration documents and programming source code. Our prototype code generator tool takes the eSLA and a number of document templates containing local configuration information as input and generates complete configuration information. The document templates consist of a mix of text, macro-language statements and directives, and replaceable text. The code generator tool uses information extracted from the eSLA (as well as other sources, if necessary), follows the macro-language directives, and generates output documents (generally one for each input template) ready for use in the Quality of Service framework.

5.4 Monitoring

Once it is configured, the Network Dispatcher monitors the bandwidth being consumed by connections to each hosted service port. Although only packets passing from users to servers pass through the Network Dispatcher, this is sufficient to monitor the bandwidth consumption of TCP/IP connections. This is made possible by the presence of ACK sequence numbers in all TCP packets. These indicate the number of bytes (octets) that have passed from the server to the user. The bandwidth consumption data are maintained for each connection individually, and for all connections to each port.

5.5 Enforcement

The bandwidth manager for the Network Dispatcher reads the bandwidth consumption data periodically. Based on the guarantees, the actual consumption, and the limit defined for the entire server farm, the bandwidth manager sets a new bandwidth budget for each communication port being serviced. This budget is chosen such that any service that wants any amount of bandwidth up to its guaranteed amount will certainly get it. Beyond that amount, further demand is fulfilled only if the total capacity of the farm will not be exceeded, and if no service with higher priority has claimed it first.

Once the new budget is set, the Network Dispatcher will force a service to stay within that budget using two different mechanisms: admission control and acknowledgment delay. Acknowledgment delay is applied to existing connections when they attempt to exceed the bandwidth limit for the service. The delays are increased until the actual bandwidth being consumed is within the desired limits. Unfortunately, the server resources consumed will continue to grow without bounds if new connections are always allowed in while the existing connections are more and more slowly completing their requests. Admission control is therefore applied when the connection that are already active are sufficient to consume the available bandwidth.

6 Summary and Conclusions

We have introduced the notion of formalizing the service level agreement as an electronic document that can be used to configure and enforce the behaviour of web hosting, and other, applications. Formalizing these agreements as electronic service level agreements (eSLAs) and extending the scope of an SLA to capture additional IT resource level concerns, make it possible for the system management software at each of the parties to directly monitor and enforce these agreements. This work complements emerging standards on electronic contracts for specifying interaction protocols for enabling cross-enterprise business applications.

We have detailed the agreement information captured in an eSLA that describes services and their resource requirements. We have also described tools and supporting services in forming an eSLA. We have used a concrete web hosting example and a prototype system to demonstrate the deployment and enforcement of an eSLA for managing network bandwidth.

At this point, our eSLA framework consists of the DTD for the XML documents and some automatic tools for interpreting certain elements of such documents. The prototype system can take a collection of eSLAs, combined with definition of the network resources available to a server farm, and produce the configuration files that direct Network Dispatchers in their control of bandwidth consumption.

This work represents only the beginnings of a definition of an eSLA. The DTD for the XML grammar that eSLAs are expressed in will grow and change as we learn what parameters and constraints are of real interest and use. The prototype software deals with just one phase of the processing and management of eSLAs and their relationships with each other and the systems they are defined for. Total integration, where the runtime violation of the eSLA automatically triggers renegotiation of a new eSLA is a likely next step toward a more complete system. Integrating the pricing aspect of agreed services is also part of our future goals. Our long-term focus is to turn this into a complete, working system.

Acknowledgement: The authors gratefully acknowledge the contribution of Joseph Hellerstein in originating some of these ideas.

References

[Chatterton] Chatterton, Ruth and Rebecca Wetzel. "Service Level agreements Key Feature for ISPs." *Interactive Week*, May 4, 1998. Also available at http://www.zdnet.com/intweek/print/980504/313656.html.

[Chorley] Chorleywood Consulting Ltd. *The Manual of Service Level Agreements*, April 1998. Available at http://www.chorleywood.com/manuals/ccb/sla/sla.htm.

[CPPCPA] Collaboratio-Protocol Profile and Agreement Specification, Version 1.0, http://www.ebxml.org/specs/ebCPP.pdf

[Dan98] Dan, A., D. Dias, T. Nguyen, M. Sachs, H. Shaikh, R. King, and S. Duri, "The Coyote Project: Framework for Multi-Party e-commerce," Proceedings of Research and Advanced Technology for Digital Libraries, Second European Conference, ECDL'98, Heraklion, Greece (September 1998), Springer-Verlag, Berlin (1998), pp. 873-889.

[Dan01] Dan, A., D. M. Dias, R. Kearney, T. C. Lau, T. N. Nguyen, F. N. Parr, M. W. Sachs, and H. H. Shaikh, "Business to Business Integration with TpaML and a B2B Protocol Framework (BPF)", IBM Systems Journal, Vol. 40, No. 1, 2001, pp. 68-90.

[King] King, Richard P., Junehwa Song, and Daniel M. Dias, "A System for Resource Sharing and Control of a Cluster-Based Web Server Farm", IBM Research Report, RC 21881, 9 October 2000.

[UUNETSLA] UUNET Service Level Agreement, available at http://www.uu.net/terms/sla.

[Verma] Verman, Dinesh. *Supporting Service Level Agreements on IP Networks*, Indianapolis, IN: Macmillan Technical Publishing, 1999.

[WSDL] Web Services Description Language (WSDL), http://www/w3/org/TR/wsdl

[XML] Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml

Appendix

1. Availability, Throughput, and Bandwidth

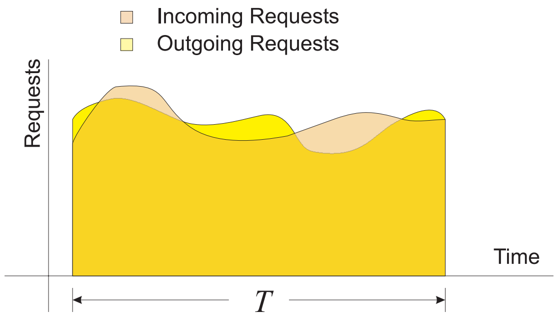
Availability

Availability is a measure of whether the service provider is meeting the requirements specified in the eSLA. Since the eSLA can be violated when any of the requirements specified in the eSLA are not met, each has an element which describes a function used to measure the "availability" of the requirement. The following are defined:

Throughput

Throughput is measured in terms of requests processed versus requests made. See the figure at the right. Consider a running interval, T. Divide the interval into n equal time intervals, Δt_i.

Let r_i be the number of requests made during the interval Δt_i



and s_i be the number of requests processed. The fraction $\frac{r_i}{s_i}$, taken over a long interval, should approximate 1. By choosing an appropriate interval, and then summing over T, we have a measurement of how well the service provider is keeping up with incoming requests. If we choose a minimum acceptable limit, z, we can then say that the availability for throughput fails if

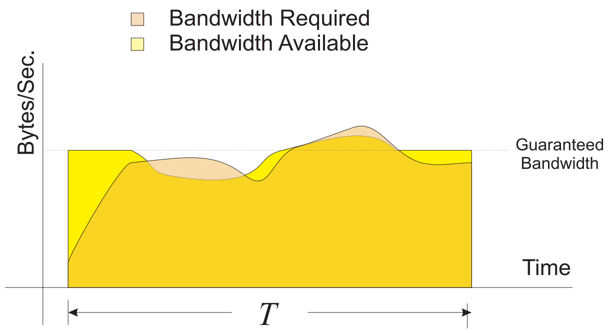
$$\frac{\sum_{i=1}^{n} \frac{r_i}{s_i}}{n} < z$$

Note that Δt and T should be modeled to determine "best" values depending upon the services being requested.

Factors to be recorded in the eSLA are Δt (or n), T and z.

Bandwidth

Bandwidth is a measure of how much bandwidth is required by the customer versus the guaranteed bandwidth. See the figure at the right. Again, take a running interval, T, and divide it into n equal time intervals, Δt_i, and consider the fraction



$$\frac{r_i}{a_i}$$

where r_i is the bandwidth required by the customer, and a_i is the bandwidth available from the service provider. These bandwidth values are measured at a well-defined point in the given interval.
The expression

$$\frac{\sum_{i=1}^{n} \frac{\min(r_i, g)}{\min(a_t, g)}}{T}$$

where g is the guaranteed bandwidth, defines a measure of bandwidth availability taken over the interval T, i.e., what fraction of the requested bandwidth was not delivered by the service provider. If we choose a minimum acceptable limit, z, such that the expression is less than this limit, then the availability of bandwidth fails.

The factors which must be recorded in the eSLA are Δt (or n), T and z.

2. eSLA DTD

The following DTD definition will be modified as services are identified, and the service resources pertinent to them are defined.

<!ELEMENT **ESLA** (InformationSection, RoleSection, ServicesSection)>
<!ELEMENT **InformationSection** (ContractName, Duration?)>
<!ELEMENT **ContractName** (#PCDATA)>
<!ELEMENT **Duration** (StartDate?, EndDate?)>
<!ELEMENT **StartDate** (#PCDATA)>
<!ELEMENT **EndDate** (#PCDATA)>

<!ELEMENT **RoleSection** (Role+)>
<!ELEMENT **Role** (RolePlayer)>
<!ATTLIST Role **Name** (Customer | Server | Arbitrator) "Customer">

<!ELEMENT **RolePlayer** (OrganizationName, Contact+)>
<!ATTLIST RolePlayer **MemberId** CDATA #REQUIRED

```
        IdCodeType CDATA #REQUIRED>

<!ELEMENT OrganizationName (#PCDATA)>
<!ELEMENT Contact (Name, Address, EMail, Telephone, Fax)>
<!ATTLIST Contact Type CDATA #REQUIRED>

<!ELEMENT Name (#PCDATA)>
<!ELEMENT Address (AddressLine+, City, State, Zip, Country)>
<!ELEMENT AddressLine (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Zip (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT EMail (#PCDATA)>
<!ELEMENT Telephone (#PCDATA)>
<!ELEMENT Fax (#PCDATA)>

<!--  This section defines the service covered by this eSLA -->
<!ELEMENT ServicesSection ((HTTPService | FTPService | AudioService | BackupService)+)>

<!--  These are the services defined so far.  Others will be added as needed -->
<!ELEMENT HTTPService (ClusterAddress, Port, Priority, BenchmarkValue?, Report*, Event+)>
<!ATTLIST HTTPService Name CDATA #REQUIRED>
<!ELEMENT FTPService (ClusterAddress, Port, Priority, BenchmarkValue?, Report*, Event+)>
<!ATTLIST FTPService Name CDATA #REQUIRED>
<!ELEMENT AudioService (ClusterAddress, Port, Priority, BenchmarkValue?, Report*, Event+)>
<!ATTLIST AudioService Name CDATA #REQUIRED>
<!ELEMENT BackupService (Frequency, Depth)>

<!--  These elements are members of the Backup Service -->
<!--  This element describes how frequently the backup is taken
<!ELEMENT Frequency (#CDATA)>
<!ATTLIST Frequency Unit (Hours | Days) “Days”)>
<!--  This element describes how many backups are in the rotation
<!ELEMENT Depth (#PCDATA)>

<!ELEMENT Event (ResourceList)>
<!ATTLIST Event Value CDATA #REQUIRED>

<!--  This is the structure of the Service element -->
<!ELEMENT ClusterAddress (#PCDATA)>
<!ELEMENT Port (#PCDATA)>
<!ELEMENT Priority (#PCDATA)>
<!ELEMENT BenchmarkValue (#PCDATA)>
<!ATTLIST BenchmarkValue Name CDATA #REQUIRED>
<!ELEMENT Report (Interval?)>
<!ATTLIST Report Type (Standard | Emergency | Modification) "Standard"
        Recipient CDATA #REQUIRED>
<!ELEMENT Interval (#PCDATA)>

<!ELEMENT ResourceList ANY>

<!--These values are used in evaluating availability -->
<!ELEMENT GuaranteedValue (#PCDATA)>
<!ELEMENT Availability (RunningInterval, SampleInterval | SampleCount, Limit)>
<!ELEMENT RunningInterval (#PCDATA)>
<!ELEMENT SampleCount (#PCDATA)>
```

```
<!ELEMENT SampleInterval (#PCDATA)>
<!ELEMENT Limit (#PCDATA)>

<!-- The following are resource descriptions -->
<!-- They will be members of the ResourceList element -->
<!ELEMENT GuaranteedThroughput (GuaranteedValue, Availability)>
<!ELEMENT MaximumThroughput (#PCDATA)>
<!ELEMENT GuaranteedBandwidth (GuaranteedValue, Availability)>
<!ELEMENT MaximumBandwidth (#PCDATA)>
<!ELEMENT MaximumServiceTime (#PCDATA)>
```

3. eSLA Prototype Example

```
<?xml version="1.0"?>
<!DOCTYPE ESLA SYSTEM "eSLA.dtd">
<ESLA>
    <InformationSection>
    <!--*********************************************************************-->
    <!-- General Contract information                                   -->
    <!--*********************************************************************-->
        <ContractName>IGS + Store 1</ContractName>
        <Duration>
            <StartDate>2000-03-16</StartDate>
            <EndDate>2001-03-15</EndDate>
        </Duration>
    </InformationSection>

    <!--*********************************************************************-->
    <!-- Role section, where participants are identified                  -->
    <!--*********************************************************************-->
    <RoleSection>
        <Role Name="Customer">
            <RolePlayer MemberId="0001" IdCodeType="ZZ">
            <OrganizationName>Store 1</OrganizationName>
            <Contact Type="MainContact">
                <Name>Kris Kringle</Name>
                <Address>
                    <AddressLine>34th St</AddressLine>
                    <City>New York</City>
                    <State>NY</State>
                    <Zip>10101</Zip>
                    <Country>USA</Country>
                </Address>
                <EMail>kris@store-1.com</EMail>
                <Telephone>212-555-1234</Telephone>
                <Fax>212-555-4321</Fax>
            </Contact>
            </RolePlayer>
        </Role>
        <Role Name="Server">
            <RolePlayer MemberId="0001" IdCodeType="ZZ">
            <OrganizationName>IGS</OrganizationName>
            <Contact Type="MainContact">
                <Name>John Doe</Name>
                <Address>
                    <AddressLine>1 IGS Plaza</AddressLine>
                    <City>Schaumburg</City>
                <State>IL</State>
                <Zip>60606</Zip>
                <Country>USA</Country>
            </Address>
            <EMail>johndoe@us.ibm.com</EMail>
            <Telephone>302-555-4567</Telephone>
            <Fax>302-555-9999</Fax>
        </Contact>
        </RolePlayer>
    </Role>
</RoleSection>

<!--*******************************************************************-->
<!-- Service section, where services covered by this eSLA are identified  -->
<!--*******************************************************************-->
<ServicesSection>
    <HTTPService Name="HTTPService1">
        <ClusterAddress>63.73.131.68</ClusterAddress>
        <Port>80</Port>
        <Priority>30</Priority>
        <BenchmarkValue Name="Webstone">100</BenchmarkValue>
        <Report Type="Standard" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Report Type="Emergency" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Report Type="Modification" Recipient="servicewatcher@store-1.com"/>
        <Event Value="LowVolumeTime">
            <ResourceList>
                <GuaranteedBandwidth>
                    <GuaranteedValue>100000</GuaranteedValue>
                    <Availability>
                        <RunningInterval>100</RunningInterval>
                        <SampleInterval>2</SampleInterval>
                        <Limit>.95</Limit>
                    </Availability>
                </GuaranteedBandwidth>
                <MaximumBandwidth>150000</MaximumBandwidth>
                <MaximumServiceTime>3</MaximumServiceTime>
                <GuaranteedThroughput>
                    <GuaranteedValue>10</GuaranteedValue>
                    <Availability>
                        <RunningInterval>100</RunningInterval>
                        <SampleCount>50</SampleCount>
                        <Limit>.95</Limit>
                    </Availability>
                </GuaranteedThroughput>
                <MaximumThroughput>15</MaximumThroughput>
            </ResourceList>
        </Event>
        <Event Value="HighVolumeTime">
            <ResourceList>
                <GuaranteedBandwidth>
                    <GuaranteedValue>200000</GuaranteedValue>
                    <Availability>
                        <RunningInterval>100</RunningInterval>
                        <SampleInterval>2</SampleInterval>
                        <Limit>.95</Limit>
                    </Availability>
                    </GuaranteedBandwidth>
                <MaximumBandwidth>250000</MaximumBandwidth>
                <MaximumServiceTime>10</MaximumServiceTime>
                <GuaranteedThroughput>
                    <GuaranteedValue>1</GuaranteedValue>
                    <Availability>
                        <RunningInterval>100</RunningInterval>
                        <SampleCount>50</SampleCount>
                        <Limit>.95</Limit>
                    </Availability>
                </GuaranteedThroughput>
            </ResourceList>
        </Event>
</HTTPService>
<FTPService Name="FTPService1">
        <ClusterAddress>63.73.131.68</ClusterAddress>
        <Port>20</Port>
        <Priority>30</Priority>
        <Report Type="Standard" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Report Type="Emergency" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Event Value="Initial">
            <ResourceList>
                <GuaranteedBandwidth>
                    <GuaranteedValue>10000</GuaranteedValue>
                    <Availability>
                        <RunningInterval>100</RunningInterval>
                        <SampleInterval>2</SampleInterval>
                        <Limit>.95</Limit>
                    </Availability>
                </GuaranteedBandwidth>
                <MaximumBandwidth>15000</MaximumBandwidth>
            </ResourceList>
        </Event>
</FTPService>
<FTPService Name="FTPService2">
        <ClusterAddress>63.73.131.68</ClusterAddress>
        <Port>21</Port>
        <Priority>40</Priority>
        <Report Type="Standard" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Report Type="Emergency" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Event Value="Initial">
            <ResourceList>
                <GuaranteedBandwidth>
                    <GuaranteedValue>40000</GuaranteedValue>
                    <Availability>
                        <RunningInterval>100</RunningInterval>
                        <SampleInterval>2</SampleInterval>
                        <Limit>.95</Limit>
                    </Availability>
                </GuaranteedBandwidth>
                <MaximumBandwidth>45000</MaximumBandwidth>
            </ResourceList>
        </Event>
    </FTPService>
    <AudioService Name="AudioService">
        <ClusterAddress>63.73.131.68</ClusterAddress>
        <Port>114</Port>
        <Priority>20</Priority>
        <Report Type="Standard" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Report Type="Emergency" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Event Value="Initial">
            <ResourceList>
                <GuaranteedBandwidth>
                    <GuaranteedValue>100000</GuaranteedValue>
                    <Availability>
                        <RunningInterval>100</RunningInterval>
                        <SampleInterval>2</SampleInterval>
                        <Limit>.95</Limit>
                    </Availability>
                </GuaranteedBandwidth>
                <MaximumBandwidth>150000</MaximumBandwidth>
            </ResourceList>
        </Event>
    </AudioService>
    <HTTPService Name="HTTPService2">
        <ClusterAddress>63.73.131.68</ClusterAddress>
        <Port>443</Port>
        <Priority>10</Priority>
        <BenchmarkValue Name="Webstone">125</BenchmarkValue>
        <Report Type="Standard" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Report Type="Emergency" Recipient="servicewatcher@store-1.com">
            <Interval>600</Interval>
        </Report>
        <Event Value="Initial">
            <ResourceList>
                <GuaranteedBandwidth>
                    <GuaranteedValue>20000</GuaranteedValue>
                    <Availability>
                        <RunningInterval>100</RunningInterval>
                        <SampleInterval>2</SampleInterval>
                        <Limit>.95</Limit>
                    </Availability>
                </GuaranteedBandwidth>
                <MaximumBandwidth>250000</MaximumBandwidth>
                <MaximumServiceTime>10</MaximumServiceTime>
                <GuaranteedThroughput>
                    <GuaranteedValue>1</GuaranteedValue>
```

```
                                    <Availability>
                                        <RunningInterval>100</RunningInterval>
                                        <SampleInterval>2</SampleInterval>
                                        <Limit>.95</Limit>
                                    </Availability>
                                </GuaranteedThroughput>
                                <MaximumThroughput>3</MaximumThroughput>
                            </ResourceList>
                        </Event>
                    </HTTPService>
                </ServicesSection>
            </ESLA>
```

4. Sample Farm Document

```
<?xml version="1.0"?>
<!DOCTYPE DOC SYSTEM "farm.dtd">
<DOC>
  <Communications>
    <AggregateBandwidth>1200000</AggregateBandwidth>
    <GlobalPolling>20</GlobalPolling>
    <NetworkDispatchers>
      <ND>
        <NdAddr>1.1.1.1</NdAddr>
        <NdPort>18001</NdPort>
        <NdPoll>5</NdPoll>
        </ND>
      <ND>
        <NdAddr>2.2.2.2</NdAddr>
        <NdPort>18001</NdPort>
        <NdPoll>5</NdPoll>
        </ND>
      </NetworkDispatchers>
    </Communications>
  </DOC>
```

5. Sample Layout Document

```
<?xml version="1.0"?>
<!DOCTYPE DOC SYSTEM "layout.dtd">
<DOC>
  <Communications>
    <ND>
      <NdAddr>1.1.1.1</NdAddr>
      <SLA>sla1.xml</SLA>
      <SLA>sla3.xml</SLA>
      </ND>
    <ND>
      <NdAddr>2.2.2.2</NdAddr>
      <SLA>sla2.xml</SLA>
      </ND>
    </Communications>
```

</**DOC**>