

QoS and Delivery Context in Rule-Based Edge Services

Ng, Chan Wah

Tan, Pek Yew

*Panasonic Singapore Laboratories Pte Ltd
Blk 1022 Tai Seng Ave #04-3530 Singapore 534415*

cwng@psl.com.sg

pytan@psl.com.sg

Abstract

As the Internet grows, more and more content distribution networks are deploying services at the network edge to effectively perform content adaptation. In a wireless web environment where bandwidth resource is expensive, edge services requires information such as the Quality of Service (QoS) to perform optimal bandwidth adaptation. In addition, due to the diversification of wireless device accessing the Internet, it is necessary to provide an extensive set of information on the wireless device so that edge services can be carried out efficiently. A complete collection of such information is known as Delivery Context. We describe the use of sub-systems in an Open Pluggable Edge Services (OPES) intermediary to provide edge service agents the needed QoS and delivery context information.

1. Introduction

With the proliferation of the Internet, application designed for the World Wide Web (WWW) are facing serious extension and scaling problem. To alleviate the problem of longer load time for WWW page request and lost of quality in real-time audio-visual playback, more and more Content Distribution Networks (CDN) are deploying intelligent network elements at the network edge, nearer to the end users.

The most common use of such intermediate network elements are to function as caches, such as hyper text transfer protocol (HTTP) caching proxies described in [1]. These have been successful in reducing the network load at the content server and accelerating content delivery to the end users. However, as the WWW gets more pervasive, the range of web contents is also broadening. Not only do we see contents in Hyper-Text Markup Language (HTML), but also contents created for the Wireless Application Protocol (WAP) in Wireless Markup Language (WML), audio-visual contents such as Macromedia Flash, and even contents that are dynamically generated in real time. Simply replicating static web contents cannot hope to sustain the ever-increasing demands from the end users.

Furthermore, as the Internet grows, so does the range of devices that are used to access contents from the Web. This diversification of browser types has been accelerated with recent advancements in wireless Internet technology, whereby tiny handheld devices such as digital personal assistants (PDA) and mobile phones have micro-browsers built in that browse the

web, or playback audio/visual streams. No longer can content authors develop contents with the assumption that end users will only view the content created using traditional desktop computers. *Device independence* is now a critical consideration [2].

In an effort to resolve these issues, CDNs are increasingly introducing edge services to dynamically adapt content at the network edge. The Open Pluggable Edge Service (OPES) Working Group of the Internet Engineering Task Force (IETF) has defined a framework for deploying edge services [3].

In this paper, we will outline an OPES intermediary designed to provide edge services for applications in wireless web and audio-visual streaming, in addition to the traditional functionality of content caching. When delivering web contents and audio-visual streams to the wireless clients, Quality of Service (QoS) requirement is an important consideration. In addition, it will be beneficial for any edge service designed for the wireless clients to have the information about type, platform and capabilities of the wireless client agent. A complete set of such information is known as *Delivery Context* [2].

In the remaining portion of this paper, we will first briefly describe the OPES framework and how it is used to provide rule-based edge services. Next we outline how the OPES framework can be enhanced so that QoS parameters and delivery context are available to the edge service agents. This is followed by an illustration of the prototype implemented and its deployment scenario in a wireless network environment.

2. OPES Framework

2.1. Overview of OPES

The OPES framework was first proposed by the IETF with the idea to extend the functionality of traditional HTTP proxy beyond simple caching and request relaying. Examples of additional services that can be provided by the intermediary are presented in [4], which includes advertisement insertion, HTML content adaptation, and limited bandwidth adaptation. A complete OPES infrastructure is currently being developed.

In essence, the OPES infrastructure consists of a rule engine, an adaptation server, and a HTTP proxy, as illustrated in Figure 1. The rule engine intercepts HTTP requests and responses flowing through the proxy and triggers services provided by the adaptation server to act on the HTTP contents. Conditions are specified using Intermediary Rule Markup Language (IRML) [5], a rule specification language that is an application of the eXtensible Markup Language (XML). These conditions, called “property” in IRML, are expressed in terms of a set of parameters. Ordinarily, these parameters based their values from the HTTP headers. For example, the line `<property name=“Content-type” matches=“jpg”>` specifies the condition of the “Content-Type” HTTP header containing the string “jpg”. These “property” expressions can be nested to implement a wider range of condition evaluations. When a condition is true, one or more adaptation services can be triggered. These adaptation services can be invoked locally or redirected to remote callout servers. Parameters interpreted by the OPES rule

engine can be passed to the invoked adaptation service so that the adaptation service need not retrieve these information again.

It must be noted that the IRML specification allows a rule to be specified for evaluation at different processing points: (1) evaluated for each client request before the cache, (2) evaluated for each client request after the cache, (3) evaluated for each server response before the cache, and (4) evaluated for each server response after the cache. Such flexibility allows rules to be specifically designed for dynamic contents. Adaptation services that altered a content in such a way that the altered content is cacheable can be triggered at processing point (3); whereas an adaptation service which altered the content in a way that is non-cacheable can be triggered at processing point (4).

2.2. Limitations of IRML

Although similar frameworks for dynamically adapting web contents have been proposed elsewhere [6,7], the standardization effort initiated by the IETF is one of the more generalized approaches. However, since IRML defines only “properties” that are based on HTTP headers, the conditions that can be evaluated are limited. For edge services in a wireless environment, this limitation is especially crippling. To understand why this is the case, we first note that adaptation of contents for wireless clients typically has two objectives in mind.

The first objective is to adapt the contents such that it can be more suitably rendered and presented on a wireless device. To do so effectively, the edge service may require more information on the client agent than is

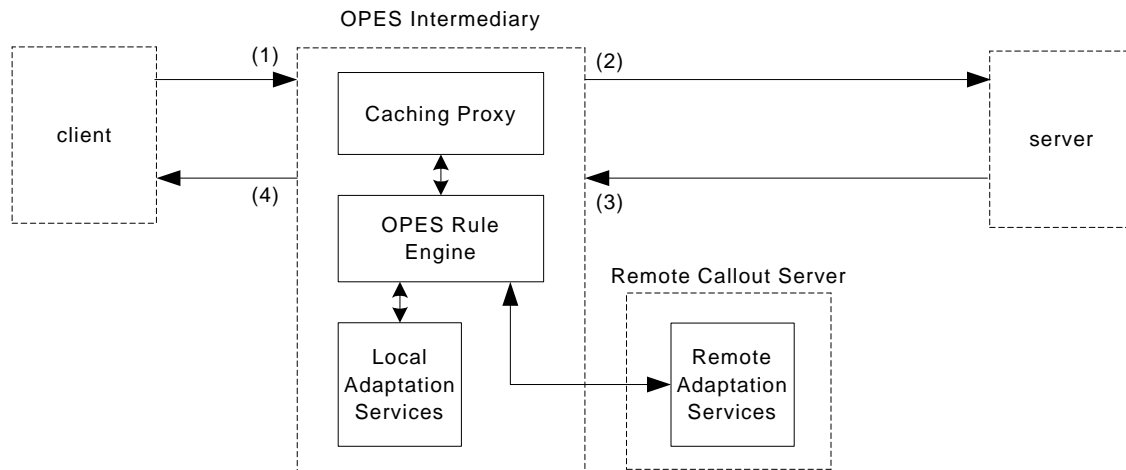


Figure 1 - OPES Architecture. An OPES architecture typically consist of a rule engine sitting on top of a caching proxy. The rule engine will trigger adaptation services locally or remotely based on the requests and responses flowing through the proxy. Note that rules can be evaluated before the cache (i.e. 1 and 3) or after the cache (i.e. 2 and 4) so that different rules can be applied to static contents, dynamically generated contents, and cached contents.

available from the “User-Agent” header. Either there must exist an alternative channel from which the edge service can obtain such delivery context information, or the edge service simply has to provide a sub-optimal adaptation due to limited information.

The second objective is to adapt the contents so that the expensive, limited resource of wireless bandwidth can be used more efficiently. Examples of such services include down sampling of images and removal of enhancement layers in MPEG-4 streaming. In order to effectively perform bandwidth adaptation, QoS parameters must be made available to the edge service. However, HTTP has no provision for transfer of such information. Thus, edge services must obtain the QoS information independently.

As described above, edge services must find alternative methods to obtain vital information in order to provide efficient content adaptation for wireless clients. The lack of standard method of obtaining such information will result in vendors offering incompatible edge services, thereby undermining the advantages a standardized OPES architecture can bring.

2.3. Sub-System Extensions

In an attempt to overcome such limitation, recent efforts from Ng *et. al.* have resulted in the OPES rule engine having the capability of sub-systems extension [8]. This allows the rule specification to extend beyond the limitations of HTTP headers, since sub-system can be defined to interpret parameters that are out of the scope of HTTP. Sub-system can be viewed as an abstraction layer that defines a common set of parameters, which edge service developers and rule authors can use without having to consider the implementation of the actual mechanism that retrieve the values of the parameters. In addition, since sub-systems are optional in the IRML specification, service providers can install different sub-systems for different application needs. Thus, an OPES implementation, which includes sub-systems capable of interpreting QoS parameters and delivery context, can be employed to complement wireless edge services.

3. QoS-Awareness and Delivery Context

3.1. QoS Sub-System

In [9], Ng *et. al.* proposed a QoS sub-system designed to allow the IRML rule engine to interpret conditions that are expressed using QoS parameters. The proposed QoS parameters supported by the said sub-system can be classified into 4 categories, as shown in Figure 2:

static, dynamic, difference in dynamic, and system. The *static QoS parameters* are those that do not change once the connection is established. These are usually values set by QoS policy. The *dynamic QoS parameters* are the statistics of the content transmissions, which reflect the current network condition. These are updated at a known, specific interval. The *difference of dynamic QoS parameters* are the difference between the current values of the dynamic QoS parameters and the values before they are last updated. The *difference in dynamic QoS parameters* allow the rule engine to have a stateless implementation. The *system parameters*, though not strictly QoS parameters, are provided for the rule engine to query the system load, which may affect the QoS of the content delivered. In addition to defining QoS parameters, QoS sub-system for the OPES rule engine allows conditions to be evaluated based on arithmetic relationship (e.g. bigger than, less than), as opposed to the regular-expression-only matching in the original IRML specification [5], since most of the QoS parameters have numerical values.

The introduction of the QoS sub-system does not define how the actual values of the QoS parameters are obtained. It merely defines the parameters, and vendors are free to use any mechanism to obtain the actual values, such as a traffic-monitoring agent using the Simple Network Management Protocol (SNMP). This relief author of IRML rules and edge services the responsibility of obtaining such values independently.

3.2. Delivery Context Sub-System

Using an approach similar to the QoS sub-system, a Delivery Context sub-system can be proposed to allow the IRML engines to interpret parameters from a delivery context. Figure 3 shows four major classes of delivery context that are defined: *User Preferences*, *Agent Capabilities*, *Device Capabilities*, and *Natural Environment*.

User Preferences refers to information about the human user, including browsing preference, language preferences, display preferences, QoS preferences, age group and gender. *Agent Capabilities* provide information on the software agent, such as the agent type, supported formats, supported languages, and supported transport protocols. *Device Capabilities* refer to the information about the hardware device, which include the device type, processor speed and type, memory capacity, screen resolution and depth, and operating systems. *Natural Environment* provide information about the natural environment surrounding the end user, including whether the end user is indoor or outdoor, the end user’s velocity, location of the end user, and illumination properties.

Parameter Name	Parameter Value
Static QoS Parameters	
allocated-bandwidth	allocated bandwidth for the end-user
requested-bandwidth	requested bandwidth for this connection
available-bandwidth	amount of bandwidth available for the end-user
delay-bound	maximum delay requested
Dynamic QoS Parameters	
r-octets-count	accumulated number of octets received by the end-user
s-octets-count	accumulated number of octets received by the intermediary
r-packets-count	accumulated number of packets received by the end-user
s-packets-count	accumulated number of packets received by the intermediary
r-packets-lost	total number of packets not received by the end-user
s-packets-lost	total number of packets not received by the intermediary
r-fraction-lost	fraction of packets lost reported by the end-user since the previous report
s-fraction-lost	fraction of packets lost reported by the intermediary since the previous report
r-jitter	inter-arrival jitter reported by the end-user
s-jitter	inter-arrival jitter reported by the intermediary
Difference in Dynamic QoS Parameters	
r-octets-diff	difference in accumulated number of octets received by the end-user
s-octets-diff	difference in the accumulated number of octets received by the intermediary
r-packets-diff	difference in the accumulated number of packets received by the end-user
s-packets-diff	difference in the accumulated number of packets received by the intermediary
r-packets-lost-diff	difference in the total number of packets not received by the end-user
s-packets-lost-diff	difference in the total number of packets not received by the intermediary
r-jitter-diff	difference in the inter-arrival jitter reported by the end-user
s-jitter-diff	difference in the inter-arrival jitter reported by the intermediary
System Parameters	
system-processing-load	current processing load in percentage of the intermediary
system-connections	current number of established end-user connections

Figure 2 – QoS Parameters. Parameters ranging from static QoS parameters such as allocated bandwidth to dynamic network conditions such as the difference in octets received between two consecutive report are extended to the IRML rule engine with the QoS sub-system.

Parameters	Values
User Preferences	
browsing-preference	the user's browsing preferences, such as text only, image sizes, handicaps accessibilities options, searching and filtering preferences
language-preference	descriptive text about user's order of language preferences
display-preference	descriptive text about user's color preferences, full screen or window
age-group	descriptive text about the user's age group
gender	descriptive text about the user's gender
employment	descriptive text about the user's job nature
Agent Capabilities	
agent-type	descriptive text about the software agent
supported-formats	content formats supported, and content encoding supports
supported-languages	language supported by the software agent
supported-protocols	transmission protocols supported by the software agent, and whether it has multicasting, broadcasting capabilities
Device Capabilities	
device-type	descriptive text about the device type
processor	descriptive text about processor speed and family
memory-capacity	descriptive text about memory capacity of the physical and secondary memory
screen	descriptive text about resolution and depth
operating-system	descriptive text about the operating system type
Natural Environment	
location	descriptive text about the user's location, such as indoor or outdoor, and the locale
mobility	descriptive text about the user's mobility, whether fixed or moving, and the velocity if moving
illuminations	descriptive text about illuminations surrounding the end user

Figure 3 – Delivery Context Parameters. These parameters involving user preferences, browser capabilities, and natural environment are extensions to the IRML rule engine introduced by the Delivery Context sub-system.

As with the QoS sub-system, the actual retrieval of delivery context information is out of scope of the sub-system specification, and is up to individual vendor implementations. For instance, user profile information can be obtained from a (remote or local) database through the User Profile Information Protocol [10], or the Application Configuration Access Protocol [11]. The sub-system in IRML engine merely presents IRML authors a standardized set of delivery context parameters to construct rules and conditions.

4. Implementation

4.1. Prototype

To demonstrate the use of the proposed QoS and Delivery Context sub-systems, we implemented an OPES intermediary prototype with the two sub-systems integrated. Figure 4 depicts its architecture. The prototype consisted of a HTTP proxy with caching capabilities. An IRML rule engine was hooked to the proxy at points before and after the cache to process HTTP requests and responses. QoS and Delivery Context sub-systems were implemented as loadable libraries to extend the rule engine.

For the QoS sub-system, it interfaced to an external module, known as a Traffic Control Framework (TCF), which monitors and conditions traffic flowing to the

network [12]. TCF can classify data packets into different flows (such as packets with the same destination-source pair), and independently monitors and control traffic of each flow. The QoS sub-system queries traffic information about each active HTTP session using proprietary application programming interface (API), and translate the information to the corresponding IRML properties.

Currently, the Delivery Context sub-system extracts delivery context information from HTTP headers. We modified the client-side software to insert delivery context information into HTTP headers. In future, a delivery context database will be implemented, and delivery context information can then be retrieved from the database.

4.2. Deployment Scenario

The prototype described was implemented on a gateway that allows wireless terminals to access the wired Internet. We successfully tested a few adaptation services that can make use of information on user profile and network conditions to dynamically adapt contents or redirect the request to a more appropriate source. For instance, when processing a HTML page, the rule engine, based on the network conditions and device capabilities of the client, can trigger an adaptation service that converts all the tags in the

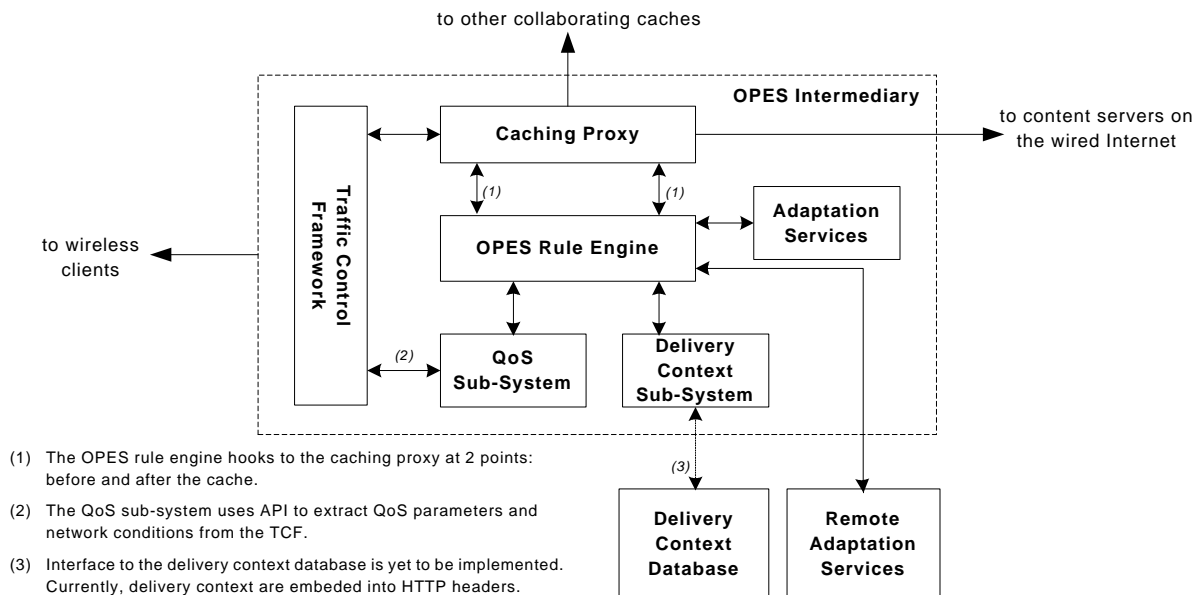


Figure 4 - Prototype. The OPES prototype implemented follows the OPES architecture, with the insertion of two sub-systems: QoS and Delivery Context. A Traffic Control Framework (TCF) was used to monitor and condition incoming and outgoing traffic in the wireless network. The QoS sub-system extracts QoS parameters in the IRML via an API interface with the TCF. Delivery Context sub-system currently extracts information from the HTTP headers. When the prototype is fully developed, it will obtain most information from an external delivery context database.

HTML page to their ALT text equivalent. With more advance adaptation services, it is possible to realize the usage scenario described below with our OPES prototype:

A user is accessing a news portal site using his/her WAP-enabled PDA, while simultaneously listening to an audio stream. A few edge services may have been kicked into action. For one, an edge service may be employed to fetch different news headlines based on the preference profile of the user, and assembled them into a single page. These headlines may be fetched from cached copies of collaborating intermediaries or from the original content sites. A second edge service is used to convert the assembled HTML page into WML that the browser on the PDA understood. Such a conversion will have to take into consideration the QoS requirements of the wireless networks, such as the bandwidth allocated and the congestion condition of the channel. Typically, the bandwidth allocated to the wireless PDA would be limited, thus irrelevant images will have to be stripped off. In addition, because the real estate of a PDA screen is limited as well, the conversion edge service will have to consider the delivery context of the browser so that the converted page can be displayed sensibly on the PDA screen. Audio streaming is handled by a third edge service to automatically adapt the audio bit rate to suit the fluctuating wireless channel conditions.

The user may later choose to switch to a wireless laptop. Web services can be configured so that such switching can take place seamlessly (i.e. the audio stream are being re-directed to the laptop, and the news page the user was initially browsing with the PDA is re-rendered according to the laptop capabilities).

The above scenario portrayed seamless re-configuration and adaptation of contents delivery. This will require up-to-date information on the QoS policy and requirements, in addition to delivery context such as user preferences and browser capabilities. The sub-systems described in this paper are designed to provide such information.

4. Conclusion

Strategic deployment of edge services is vital in improving the Internet access experience in wireless web. To achieve this, edge services often require extensive information on the wireless network and client agents. In this paper, an architecture for providing edge services in wireless web was described.

We explained the OPES infrastructure and discussed how it can be enhanced to provide edge service agents the needed information on quality of service and delivery context. Finally, a working prototype and its deployment scenario were also presented.

References

- [1] Fielding, R., et. al., "Hypertext Transfer Protocol -- HTTP/1.1", IETF RFC 2616, <http://www.ietf.org/rfc/rfc2616.txt>, June 1999.
- [2] Gimson, R., et. al., "Device Independence Principles", W3C Working Draft, <http://www.w3.org/TR/di-princ/>, September 2001.
- [3] Tommlinson, G., et. al., "A Model for Open Pluggable Edge Services", IETF Internet Draft, Work In Progress, <http://www.ietf.org/internet-drafts/draft-tomlinson-opes-model-01.txt>, November 2001.
- [4] McHenry, S., Condry, M. and G. Tomlinson, "Open Pluggable Edge Services Use Cases and Deployment Scenarios", IETF Internet Draft, Work In Progress, <http://www.ietf.org/internet-drafts/draft-mchenry-opes-deployment-scenarios.txt>, November 2001.
- [5] Beck, A. and Hofmann, M., "IRML: A Rule Specification Language for Intermediary Services", IETF Internet Draft, Work In Progress, <http://www.ietf.org/internet-drafts/draft-beck-opes-irml-02.txt>, November 2001.
- [6] Challenger, J., Iyengar, A., Witting, K., Ferstat, C., and Reed, P., "A Publishing System for Efficiently Creating Dynamic Web Content", *INFOCOM 2000*.
- [7] Nottingham, M., and Liu, X., "Edge Architecture Specification", http://www.edge-delivery.org/architecture_spec_1-0.html.
- [8] Ng, C.W., Tan, P.Y., and Cheng, H., "Sub-System Extension to IRML", IETF Internet Draft, Work In Progress, <http://www.ietf.org/internet-drafts/draft-ng-opes-irmlsubsys-00.txt>, July 2001.
- [9] Ng, C.W., Tan, P.Y., and Cheng, H., "QoS Extension to IRML", IETF Internet Draft, Work In Progress, <http://www.ietf.org/internet-drafts/draft-ng-opes-irmlqos-01.txt>, February 2002.
- [10] Penno, R., and Pham, H. T., "User Profile Information Protocol", IETF Internet Draft, Work In Progress, <http://www.ietf.org/internet-drafts/draft-penno-cdnac-userid-05.txt>, July 2001.
- [11] Newman, C., and Myers, J. G., "ACAP – Application Configuration Access Protocol", IETF RFC 2244, <http://www.ietf.org/rfc/rfc2244.txt>, November 1997.
- [12] Tan, P.Y., and Cheng, H., "Micro-Flow Control Framework for Home Gateway", *International Symposium on Consumer Electronics*, p. 94-98, December 2000.