Proceedings of the 6th International Conference on
Internet and Multimedia Systems and Applications, IMSA' 2002
August 2002, Kauai, Hawaii, pp-145-15.

A FRAMEWORK FOR ACTIVE UBIQUITOUS APPLICATION SERVICES

Javed I. Khan and Yihua He

Media Communications and Networking Research Laboratory
Department of Math & Computer Science
Kent State University, Ohio, USA
javed|yhe5@kent.edu

Abstract

In this paper we present an overlay infrastructure for provisioning content services over the Internet. This provides a uniform framework to develop ubiquitous application services and deploy it on a shared active infrastructure on need basis. This meta-protocol powered framework is intended to support a wide array of emerging application service scenarios, ranging from simple caching and content customization to adaptation for ubiquitous computing, co-operative filtering, content-mining, to high performance multimedia transcoding.

Key words: content services networking, meta-protocol.

1. Introduction

The Internet is increasingly becoming 'active'. We are already seeing the emergence of overlay *content delivery networks* (CDN). Any modern portal will show significant amount of content adaptation, personalization, and location-aware data insertion [1,2]. The information that comes from the state-of-the art sites today are now quite complex. A growing trend is that these documents have high level of information multiplexing. Served pages are dynamically combining information arriving and originating from multiple parties. Though the first generation CDNs emerged as temporary caching proxies of HTTP responses now we are seeing increased array of other services for customized content delivery. There will be an increasing demand for more advanced processing of the incoming information at various intermediate points in the information network. These points will act as the hub for various actions ranging from rich domain knowledge based information steering, filtering, multiplexing, adaptation that will be required by ubiquitous services. Unfortunately, at present there is a serious gap in Internet protocol suit that can provide systematic support for the emerging services. Currently most such 'adaptations' are simulated in content provider's own site typically with arrays of backend servers. Such content servicing is mostly isolated and lacks interoperability or scalability. The overall growth scenario lacks roadmap for sustained evolution of such services.

In this paper we investigate the vision of a generalized content services network framework. We outline a framework that can supports deployment of wide range of services with various specification and initiation dependencies. However, before presenting the framework, first we briefly present some of the very recent and interesting developments in this fast unfolding area.

2. Background and Related Work

The quest for systematic distributed cache's coordination has recently evolved into proposals for *content distribution networks* (CDN). Commercially, we have seen the emergence of global content caching systems such as Akamai [8]. More recently, a number of teams are looking into technology for content adaptation at origin servers or in these proxy caches. Example works include Spyglass [5], IBM Transcoding proxy [10], UC Berkeley TranSend [11], and Mobiware [13]. IETF Working Group has recently proposed the Open Pluggable Edge Services (OPES) [6] and the Internet Content Adapation Protocol (iCAP) [7] defining the extended functions of future caching proxy. iCAP defined how various caching objects can be transported from one cache to another. An OPES proxy can be equipped with message parsers, rule modules, and proxylets library. When messages flow through an OPES proxy, they are not only cached but also can be automatically parsed and processed with these rules. However, the OPES is overly caching proxy centric. Almost all active processing logic in OPES comes from the interplay of the message content and the rules in the proxylets. Though OPES provides significant active capability to caches, but it does not provide enough flexibility in accommodating various service types and arrangements that may arise in the real life situations. Where services are often restricted by where, when and how the service can be performed, redirected, and administered. Also, multimedia content, which is a major component of internet information are very difficult to process by rule based parsers. Ma et. al. [3] suggested an enhanced model of *content services networking* (CSN) pursuing a more powerful view of the application server (or proxy). Ma's CSN separates passive caching proxies from application servers. The application servers can directly communication with the content servers and user-agents. Ma shows indeed this approach can handle more service scenarios. These include post or pre distribution services either on behalf of the user agent or on behalf of the content-provider. Also, it allows for more versatile services

to be placed into the system as the processing is performed entirely in the application server—a separate entity than the caching proxy. Also, a notable generalization of Ma's proposal is that the 'application service' here does not mean only XML-like markup languages processing such as dynamic assembly and delivery of web content. Thus, it is much more suitable for multimedia content processing. Ma's CSN model faces limitations on the infrastructure issue. Ma's CSN requires application proxy to be always located in application service provider's ownership domain. It requires the traffic to be always diverted to the *application service provider's* (ASP) custom servers. Essentially all *application service provides* (ASP) would have to own worldwide infrastructure[1] [8].

In this backdrop we propose a refined *Internet Application Service Networking* (IASN) concept that present a strong *application service* oriented approach rather than *caching proxy* or *application server* centered approach. We further separate the application server from the application service. We envision a sharable processing server overlay network to be incrementally available from the future infrastructure providers. The important distinction is that in this model the **IASN server providers** can concentrate on the service delivery mechanism via provisioning active servers at topologically and geographically strategic locations. On the other hand an **IASN service provider** can focus on developing the service, without worrying about the service delivery infrastructure. We propose the *active service distribution and location* (ASDL) model that will allow these parties to work together and provide application service between the end-user and the content-provider. This will not require traffic to be diverted into application service provider's site rather the application service provisioning modules will flow into the locations in their natural paths which are best with respect to strategic consideration—including even content server's location.

A particular design challenge of a generalized application services framework arises from the fact that such a system involves complex negotiation between multiple ownership parties. Consequently, we take a novel *meta-protocol* based approach to design the ADSL protocol. Potentially not only it can support a wide variety of application scenarios but also it allows them to be managed efficiently from a meta-level. It has the following novel aspects:

- There can be geographically distributed hierarchical (ISP like) CSN server providers. Server providers will be able to choose service from multiple application service providers for their customers, and vice versa.
- The service modules can flow to appropriate network point(s) for co-operative processing, as needed by the service model, rather then being dictated by the origin server's or ASP's location. This will expand the range of application services that can be handled by this scheme.
- The specialized ASP processing servers allow the supported services to work into deeper network layers (such as TCP/IP). This will provide another degree of freedom on the range of applications services that can be deployed. This will also enables more efficient implementation of upper layer services.

In section 3, we first describe the active content servicing scenario that ADSL can handle. Section 4 and 5 present the meta-protocol and ADSL. Finally in section 6 we provide an example of complex content service.

3. Active Content Servicing Model

3.1. *Architecture and components:*

The Active Service Distribution and Locution (ASDL) Model we propose identifies the following entities namely (i) service management servers (SMS), (ii) adaptation routers (AR), (iii) content provider and (iv) end-user-agents (EU) as shown in Figure 1. In this extended CSN infrastructure, the first two components plays novel role. We provide a short description of each:

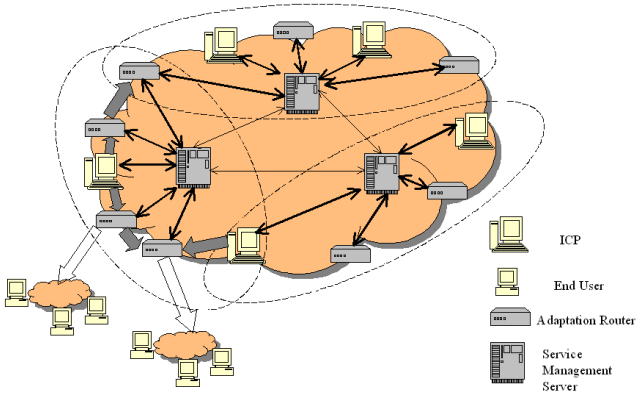

Fig 1: ASDL Architecture and Components

1. Service Management Server (SMS): SMS serve as the principle service provider. They act as the mediation center among the end-users, adaptation router infrastructure providers and the content providers. The SMS own the program modules called *switchlets* which are dynamically deployable to the ARs. These programs form the actual service. SMS are responsible for the following tasks: (1) they maintain static and dynamic information about the service execution environment and the locations of the applications; (2) they receive the service registration or cancellation requests from end-users, adaptation routers or content providers, (3) provide all authentication services, (4) aggregate the information about usage, availability and location of each deployed service, and then provide the information back to the deployment requester, (5) provide dynamic status visualization and monitoring, accounting and billing functionalities to value added service participating parties who use ASDL as an information

[1] . This model has been used in Akamai ®, which currently runs about over 14,000 servers globally.

exchange path, and (6) each SMS is responsible for collecting information about its domain and periodically exchanges the information, such as registration and deployment status, with other cooperating SMSs. These exchanges can be triggered automatically if there is a change in the system.

2. Adaptation-Router (AR): The adaptation routers are sparsely distributed special networked computing platforms, which, typically, will be deployed near the edge of the Internet. These can be setup as a service overlay and be owned by certain ISPs (or overlay ISPs). These ARs can be rented as computational platform to process data streams either on behalf of content providers (CPs), the end users or even on behalf of the overlay ISP. Unlike the CSN's application proxy servers [1], these ARs can have special TCP/IP layers, which can enable them to fast intercept streams. The processing speed can be much higher than in application level, because (1) much less decapsulation, encapsulation work will be needed; (2) and simpler instructions in IP level will let us take advantage of RISC technology; (3) and some of data streams may indexed or marked by the corresponding ICP serverlet, for random access into the data stream. See [4] for architecture of such a system.

3. Content-Provider (CP): CP servers can be typical web servers. However, the protocol allows *servelets* to be deployed at sender's location, if required by any service. For example a serverlet may premark the outgoing data streams, when a particular service is active on the stream. The marker in a data stream can enable random access in adaptation routers (ARs), and therefore dramatically reduces the computation burden of ARs. We will show an active hyperlinking example in section 6 and the saving and the cost will be shown and analyzed in our example.

4. End-User-Agent: are the sinks/terminals of data streams. They may be the normal desktop/laptop computers, or maybe handheld or wireless devices, or wearable computers. These terminals may have some kind of resource limitation, and therefore they need the resource or service provided by the ISP/AR. End-user agents generally maintain a resource-personalization specification, which can be polled by the SMS for determine the type and extend of preprocessing required.

3.2. Information Components

Any service arrangement will require various types of information to be exchanged in various sequences among these parties.

The first form is the program elements (or the servelets and switchlets) those together create the service. A single service may require switchlets and serverlets to be deployed into multiple points.

These modules themselves also require additional parameters to run the service. The model identifies two types of such parameters. The *static* adaptation parameters are those can be received before the service begins. The dynamic adaptation parameters are those required with every request. We call this kind of parameters as *specifications*, and the party who send out the specifications as the specifier.

Example of static information includes *personalization cookie box* that contains a set of tablets containing the user, user-agent, and user-environment specific constraint information.

ADSL also allows *dynamic* custom index based random stream access. A serverlet running on the content provider's site is a program that can be designed to help the service from the content source, such as source file indexing. An active application is a program that provides the service directly to the end-user, and it is designed to run on an adaptation router, which is normally controlled by some Internet service provider (ISP).

3.3. ADSL Contracting Model

The complexity of application service management grows because these information elements can come from variety of parties in various sequences based on the specific application service scenario. Before we introduce the ADSL protocol let us consider the issues: (1) who is going to supply the serverlet running on the side of content provider and the active applications running on the side of service provider? (2) Who may be the service initiator? (3) Who are going to provide the parameter specifications?

Specfier	Destination of specification		
	EU	CP	AR(SP)
EU	No	Yes, by HTTP extensions or web forms	Yes, by HTTP extensions or web forms
CP	Yes, by XML or HTTP meta extensions	No	Yes, by serverlet
AR(SP)	Yes, by HTTP meta extensions	Yes, by active application program	No

Chart 1: Specification methods between different parties.

All the three parties (EU, CP and AR) can be initiators and parameter specifiers of the ASDL services. However, when the initiator and the specifier are the same party, there is no need for extra transmission. For example, if an end-user is requesting a bandwidth adaptation service, he or she can include the bandwidth information inside the initial request. However, transmission for dependent specifications between different parties is necessary. There are several ways to transmit the specifications: (1) by tightly coupled serverlet and active application programs (2) by XMLs or XML-like languages (3) by meta tags. The specifications between a CP and a SP can be expressed by method (1), because they share a couple of servlet and active application programs, both of which derived from SMS.

Information can be exchanged freely between the coupled programs. The specification from content providers to the end-users can be expressed by the XMLs and HTTP meta extensions, while the specification from AR (SP) to the end-users can only be expressed by the HTTP meta extension. The CP and AR can make up web forms for the end-users convenience to provide the specification information. Chart-1 summarizes the discussion.

3.4. Classification of Active Services

From the service requesters' view, we may classify the services into two categories: (1) the single service request and (2) the group service request. A single service is requested by a single user and it will work solely for one user to meet its specific request. For example, a handheld device holder may request the adaptation router to translate all English web pages into German. This cannot be done at the handheld device, since it lacks memory, storage or processing speed to finish that task. In the case, the end-user may "buy" computation resource from the "net". The other type of service is group service, which is initiated either by the service provider or the content provider. For example, a service provider may have some agreement with the third party and advertise for them. The service provider then can analysis the web html files and put the ads at appropriate places. The group service can also be initiated by content providers. For example, a video source server may put special marks in the video stream and help the adaptation routers to downscale the video gracefully and meet the bandwidth requirement for all different users. The service examples and the modes they belong to are listed below in chart 2.

Example of active services	Mode		
	Single Service	Group Services	
	EUI	SPI	CPI
Insertion of Ad Banners		*	
Multimedia adaptation for limited client bandwidth	*	*	*
Multi-language adaptation for different user preference	*	*	*
Active hyperlinking	*	*	
Active re-direction	*	*	
Virus Scannering	*		
Stream data adaptation and optimization	*	*	
Watermarking			*
Insertion of regional data	*	*	
Language translation	*		

Chart 2: A list of example services and their modes

4. Modular Meta-Protocol Model

By now it can be appreciated that the design challenge of a generalized application services framework arises from the fact that, such a system involves complex negotiation between multiple parties. Conversation can take many turns based on the situation. As the task becomes realistically complex and the number of parties increase the potential conversation paths explodes. We have modeled the protocol between the involved parties using a novel **modular meta-protocol model**[2] (M^3) formalism using techniques from natural language type dialogue research.

Below now we explain the M^3 formalism, and then we will describe how within this superstructure it handles the specific service scenario for three different application service models.

In this model every element of conversation has two participants, the *initiator* and the *responder*. The initiator initiates a dialogue and waits for response.

Conversation A_{IR}(*E,*F,*G)

```
INITIATOR
      BEGIN
        ASK A
        EXPECT CONVERSATIONS (E,F,G);
        WAIT for REPLY A;
        END EXPECTATION (E,F,G);
        PROCESS A;
      END

-------------------------------------------------------------------------------------

RESPONDER
        BEGIN
          EXPECT A
          PROCESS A
          HOLD and evaluate if CONVERSATIONS (E,F,G) Is needed reply
          REPLY A

        END
```

Chart 3(a) Initiator and Responder of a Protocol

An elementary conversation A between an Initiator (I) and Responder (R) involves I asking for an answer for a question from R. The responder receives the request and then determines the response. However, in the determination process the responder can initiate recursive reverse dialogue. The initiator thus can receive a question rather than the answer in between. Chart-3(a) shows the template of the actions taken by I and R. Complex conversations can be built using recursive holds in the responders. We will use the following notations to denote various types of HOLDs.

A|B	Either A or B will occur

[2] The word 'protocol' has a origin in the Greek word "kolla' which means gluing together. Webster defines protocol as "a preliminary memorandum often formulated and signed by diplomatic negotiators as a basis for a final convention or treaty".

*A	A may or may not occur.
A→B	A and then B will occur.
A,B	A and B both will occur but in any order.

Let us consider a conversation such as "Can you allocate some space?". The responder can reply "Yes, I have reserve 20K for you" or "Probably, but how much space would you need?", "I will need about 20 K", "Fine! Would you please give me your ID?, "Yes! But is Driver's License is OK?", "Yes", "Here it is", "Ok I am reserving 25K for you". In this paper we will model the conversation, identify methods for automatic verification and probably automatic generation of bug free protocol code. The protocol for this conversation can be modeled as:

GRANT-ME-SPACE (*WHAT-SIZE, *ID-PLEASE (TYPE-CHECK))

We call it *template*. From a conversation template definition the M^3 allows determination of the code in each side following recursive. It also allows performance optimization. For example, if it is known that responder for GRANT-ME-SPACE might initiate WHAT-SIZE or ID-PLEASE protocols before responding GRANT-ME-SPACE protocol then responder threads for these can be pre-staged. However, if initiator moves into answer stage skipping them, then these threads can be terminated. The M^3 meta-protocol can be used to exchange the conversation templates. The template may be changed during conversation by the parties. However, as long as the meta-protocol propagates the changes, prior to their execution the overall operation can be kept consistent.

4.1. M^3 Superstructure of ADSL

The M^3 based ADSL superstructure has the following form. In the top level it has three phases (i) contracting, (ii) service, and (iii) billing. The individual elements can be configured according to situation. Chart-3(b) shows the ADSL structure.

```
ADSL

{SEND-CONVERSATION-TEMPLATE→CONTRACT
   (REQUEST-AUTH→SETUP
         (GET-TASK-SPEC→GET-NET-SPEC (AUTHITYPE)→INSTALL
               (AUTH, SENDMODULE, SEND SPECS)
         )
   )→SERVICE→*BILLING

}
```

Chart 3(b) ADSL Top Level Template

Each side will contain Initiator and Responder modules for each component protocol. Which of these components will actually be used, and in which sequence they will be used are decided by a *conversation template*. The meta protocol can allow a conversation template to be distributed among the concerned parties.

Potentially a number of advantages can be derived from this M^3 formalism: (i) the consistency and accuracy of complex dialogues can be verified. (ii) Potentially, the actual codes for each involved party can be generated automatically. (iii) A part of the conversation can be altered without requiring other parts to be changes. (iv) In case of change the other parties can be informed of the changes in conversation sequences via a compact language. (v) Complex conversations can be mechanically optimized. (vi) Implementation can be optimized by thread control and pre-staging. (vii) The cost of particular protocol sequence can be accounted for power and footprint optimized conversation.

5. ADSL SCENARIOS

5.1. EUI model

In this scenario, the end-user initiates the service. Fig-2 illustrates the communication steps.

Setup Stage:

(1) The EU sends service request to SMS.
(2) SMS sends query to the participating ICP Source (ICPS) and AR to collect necessary configuration data. The query is with the identification of the SMS.
(3) The ICPS and the AR response with digital signature for authentication and other necessary configuration information to SMS.
(4) SMS then delivers the application modules to ICPS and AR, with corresponding security keys, which are required when installing the modules.
(5) The ICPS and the AR send back the acknowledgements.
(6) SMS sends the response back to EU with the certificates that EU may need when sending requests to AR and ICPS.

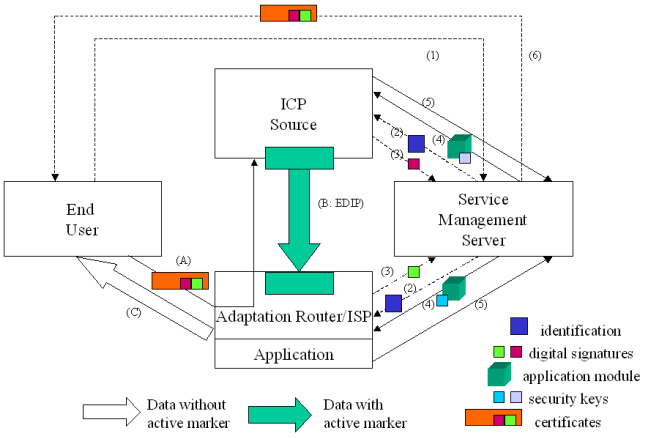

Fig 2: End-user initiates single service

Data Transfer Stage:

(A) EU sends request with certificates provided by SMS.
(B) ICPS sends out data packages with EDIP headers.
(C) AR processes the packages with EDIP headers, performs value-added in service, and sends result to EU with normal IP packages

5.2. CPI Model

In this scenario, the content-provider initiates the service. Fig-3 illustrates the communication steps.

Setup Stage:
(1) The ICP Source (ICPS) sends service request to SMS.
(2) SMS sends query to the participating ICPS and AR to collect necessary configuration data. The query is with the identification of the SMS.
(3) The ICPS and the AR response with digital signature for authentication and other necessary configuration information to SMS.
(4) SMS then delivers the application modules to ICPS and AR, with corresponding security keys, which are required when installing the modules.
(5) The ICPS and the AR send back the acknowledgements.
(6) SMS sends the response back to ICPS with certificates that ICPS may need when sending requests to AR. .

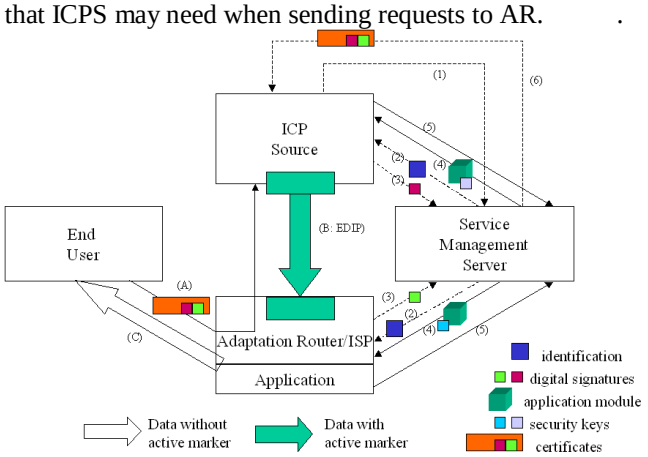

Fig 3: Content Provider initiates group service

Data Transfer Stage:
(A) End-user (EU) sends the data request.
(B) ICPS sends out data packages with EDIP headers.
(C) AR processes the packages with EDIP headers, performs value-added in service, and sends result to EU with normal IP packages.

5.3. SPI Model

In this scenario, the service provider itself initiates the service, and requests contracts from the content provider and adaptation routers. Fig-4 illustrates the communication steps.

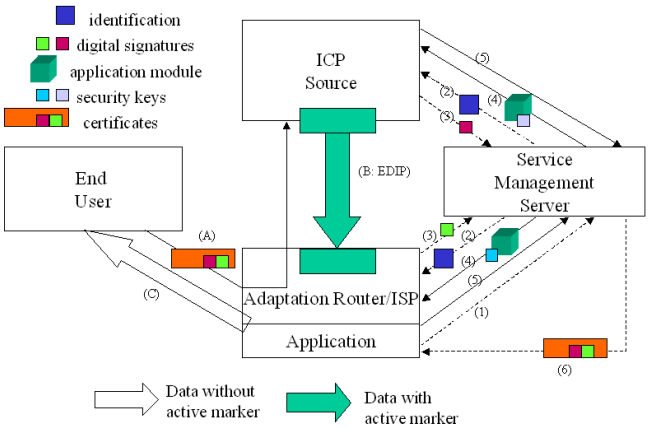

Fig 4: Service provider initiates group service

Setup Stage:
(1) The Service Provider (SP) sends service request to SMS
(2) SMS sends query to the participating ICP Source (ICPS) and AR to collect necessary configuration data. The query is with the identification of the SMS.
(3) The ICPS and the AR response with digital signature for authentication and other necessary configuration information to SMS.
(4) SMS then delivers the application modules to ICPS and AR, with corresponding security keys, which are required when installing the modules.
(5) The ICPS and the AR send back the acknowledgements.
(6) SMS sends response back to ICPS with the certificates that ICPS may need when sending requests to AR.
Data Transfer Stage:
(A) EU sends the data request.
(B) ICPS sends out data packages with EDIP headers.
(C) AR processes the packages with EDIP headers, performs value-added in service, and sends result to EU with normal IP packages.

6. Example and Analysis

Finally, we explain the operation of the system. In the process of we use a novel service called *active hyperlinking filter* [4] as our example Active Hyperlinking Filter is a service that the service provider analyses the HTML web page streams passing-by, and adds appropriate hyper links at appropriate places, upon the request from the end users or advertisement companies. It belongs to EUI or SPI. Also, it demonstrates how multi-point processing can help (in this case one adaptation router and the service specific processing at the content server). This example will also show how this application level processing can be accelerated by low level processing at the adaptation router. The traditional solution will have two major disadvantages: (1) Service providers have to decode the stream in order to analysis the content, (2) or the content provides have to perform all adaptation. By using the ASDL protocol, we can avoid those two disadvantages.

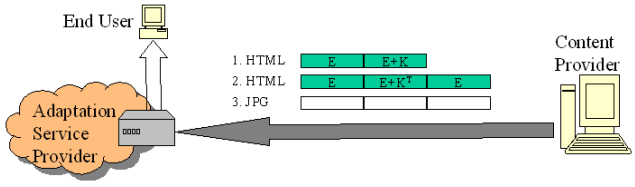

Fig 5: The example active hyperlinking service in ASDL.

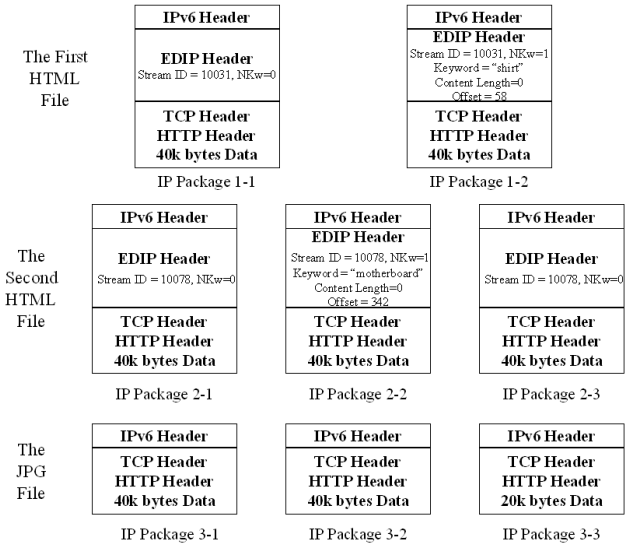

Fig 6: The IP packages after marked by EDIP

Figure 5 shows the example of hyperlinking. The service provider contacts the SMS server and deploys the serverlet on the content server. The serverlet works as an index maker and tells which stream contains the keywords and the offsets about where they are. This information is stored in a special designed header in IP extension header called Embedded Data Indexing Protocol (EDIP) (detail is in [4]). When adaptation routers get the information, they don't need to decode or search. Instead, they can process the modification immediately. We also illustrate the performance improvement by an example operation. Let the end-user request 3 files --- two are html files and the other one is a jpg file. The two HTMLs are bearing the EDIP

Packages	Actions and Destinations (ASDL)				
	Router Level	**Intermediate**	**App Level**	**Intermediate**	**Router Level**
1-1 [E]	To buffer				forward
1-2 [E+K]	To detector To buffer				forward
2-1 [E]	To buffer	decapsulate	Modify add a link	encapsulate	send
2-2 [E+K^T]	To detector To buffer	decapsulate		encapsulate	send
2-3 [E]	To buffer	decapsulate		encapsulate	send
3-1	To forwarder				forward
3-2	To forwarder				forward
3-3	To forwarder				forward

Chart 4: Cost and savings when using ASDL service mode

header but only the second HTML file has the keyword in the service provider's service list. The adaptation router of the service provider will scan the IP header extensions (EDIP header in our example) and invoke proper API to modify the content of the second HTML file. Let the resulting three IP packet streams have respectively 2, 3, and 3 packets. The EDIP headers are shown in Fig-6. In Chart-4 we show the processing stages these packets will face while passing via an ADSL/EDIP powered router/proxy filter. In a traditional full search filtering service mode, all 8 IP packages have to be decoded and sent up to application level for process. However, in our ASDL/EDIP powered service mode, only the IP packages containing the expected keyword will be decoded and processed later. All the other IP packages are process as normal router.

7. Conclusion and Current Work

ADSL is has been aimed for a content servicing internet where internet content service provider does not need to acquire large processing infrastructure but can 'rent' publicly available 'active' infrastructure. An important piece in such location independent ubiquitous Internet application service provisioning is the sharable code servers inside network. In recent years advances have been made in programmable networks. Among them active networks [9,12,14] initiative proposes the generalization of the traditional router concept— where transiting packets can be modified almost in any way with custom embedded program modules in the network elements. Several other architectures are in the design table, where processors are being added with routers/ switching hardware.

Many of the issues of CSN will be associated with the definition of component and content ownership in a complex multiparty information distribution and en-route processing framework within legal and social constraints and implications. This indeed will dictate the scenarios. The objective of our CSN research is to support as many of them as possible. Any CSN model also need to be careful about rapid evolution path. A novel aspect of our approach is the introduction of meta-protocols technique. The pioneering meta-model presented here is simple. More sophisticated meta-protocols techniques will be needed with the growth of multiparty multi-ownership systems.

The work is currently being funded by the DARPA Research Grant F30602-99-1-0515 under its Active Network initiative.

8. References:

[1] Wei-Ying Ma, Bo Shen and Jack Brassil, Content Services Network: The Architecture and Protocols, *International workshop on web caching and content distribution*, June 2001.

[2] M. Hofmann and A. Beck, Example Services for Network Edge Proxies, *Internet-Draft draft-hofmann-esfnep-00.txt*, Sept 2000.

[3] S. J. Lee, W. Y. Ma, and B. Shen, Interactive video caching and delivery using video abstraction and summarization, *Proc. International Workshop on Web caching and Content distribution* (WCW'01), Jun 2001

[4] Javed I. Khan & Yihua He, Fast Intercept of a Passing Stream for a High Performance Filter Appliances, *Proceedings of the IEEE Intl. conf. on High Speed Networking and Multimedia Communication 2002*, Jeju, Korea, July, 2002. [available at http:// bristi.facnet.mcs.kent.edu/medianet]

[5] *Spyglass-Prism*. http://www.spyglass.com.

[6] *Open Pluggable Edge Services (OPES),* http://www.ietf-opes.org/

[7] Jeremy Elson and Alberto Cerpa, Editors, *ICAP, The Internet Content Adaptation Protocol, 2001*

[8] *Akami*. http://www.akami.com.

[9] Javed I. Khan, S. S. Yang, A Framework for Building Complex Netcentric Systems on Active Network, *Proceedings of DARPA Active Network Confrence and Exposition 2002, IEEE Press, San Francisco, May, 2002*. [available at http:// bristi.facnet.mcs.kent.edu/medianet]

[10] J. Smith, R. Mohan, and C. Li, "Scalable multimedia delivery for pervasive computing," ACM Multimedia,1999.

[11] Fox, S. D. Gribble, Y. Chawathe, and E. A. Brewer, Adapting to network and client variation using active proxies: lessons and perspectives, *IEEE Personal Communication, Vol. 5, No. 4,* pp. 10-19, August 1998.

[12] Wetherall, David, Active Network Vision and Reality: Lessons from capsule-based System, *Operating Systems Review, 34(5)*: pages 64-79, December 1999.

[13] O. Angin, A.T. Campbell, M. E. Kounavis, and R. R.-F. Liao, The Mobiware Toolkit: Programmable support for adaptive mobile networking, *IEEE Personal Communications, Vol. 5, No. 4,* August 1998, pp. 32-43.

[14] Jonathan M. Smith, Programmable Networks: Selected Challenges in Computer Networking, *Computer, January 1999 (Vol. 32, No. 1),* pp. 40-42.