Intermediaries for the World-Wide Web Overview and Classification[*]

Marios Dikaiakos
Department of Computer Science
University of Cyprus
P.O. Box 20537, 1678 Nicosia, Cyprus
mdd@ucy.ac.cy

Abstract

Intermediaries are software entities deployed on Internet hosts of the wireline and wireless Web that intervene to the flow of information from clients to origin servers at the application level of the WWW. Intermediaries represent a useful abstraction for the design and study of emerging software infrastructures for "next-generation" Web services. Their importance is increasing with the increasing demand for personalization, localization, and support for ubiquitous access over different physical media and protocols. In this paper, we present an overview of a wide range of systems that can be described as intermediaries, classifying them in a number of broad categories according to their basic functionalities. Going beyond simple WWW proxies, we examine the requirements arising from the need to support personalization, mobility and ubiquity under high loads. We identify and refine a set of important properties and characteristics of intermediary systems. Based on these properties, we introduce a detailed taxonomy of characteristic systems and identify a number of key component of emerging intermediary infrastructures.

1 Introduction

The World-Wide Web has become the prevailing paradigm for information dissemination on Internet and the driving force behind the popularity of Internet services. The tremendous success of the Web is changing the basic model of Web-service provision. Typically, a Web service is established upon a *Web-server*, i.e., a process running on a well-defined host and providing information to *Web clients* over a specific communication protocol like HTTP. This client-server model is being subsumed by a model entailing a fully distributed and dynamic web of interacting servers and software entities, deployed throughout the Internet infrastructure [7]. Already, a large number of tools, systems, services and infrastructures have been proposed in this context. These systems can be collectively described as *enhanced proxies* or *intermediaries*.

We define intermediaries as *software entities that intervene to the flow of information from clients to origin servers at the application level of the WWW. Intermediaries are deployed on Internet hosts of the wireline and wireless Web, between origin servers and client systems* (for a similar definition see [2]). Intervention varies from simple relaying and caching performed by Web proxies and proxy caches, to complicated transformations, such as filtering, indexing, and transcoding. Intermediary systems fall under the general term of "middleware," as they provide a "reusable and expandable set of services and functions, commonly needed by many applications to function well in a networked environment" [1].

The importance of intermediary systems is becoming evident with the expansion of the Web and the emergence of mobile and thin clients as alternative devices for Web access. Furthermore, intermediaries represent a useful abstraction for designing, developing, analyzing and comparing the emerging software infrastructures for "next-generation" Web services, which provide personalization, customization, localization, and support ubiquitous access from various terminal devices over different physical media and protocols.

In this paper, we examine intermediaries for Internet and the World-Wide Web, going beyond "traditional" systems with minimal functionality and very wide adoption, i.e., proxy servers for the WWW like Harvest and Squid. We present an overview of characteristic intermediary systems introduced to cope with end-user information overloading, support for mobile Web access, personalization, and infrastructural support for ubiquitous services. Furthermore, we identify families of parameters that can be used in the classification of current and future systems. Due to lack of space, we do not examine intermediaries located at the

[*]Work supported in part by the Research Promotion Foundation of Cyprus, under the PENEK 23/2000 program.

origin-server or at the client-side. Instead, we focus on intermediaries located in the network infrastructure between the origin server and the client system.

The remaining of this paper is organized as follows: Section 2 presents “notification systems” on the World-Wide Web. Section 3 examines the challenges that mobile devices and wireless connections place upon intermediary systems, and discusses emerging intermediary infrastructures for ubiquitous Internet services, which incorporate ideas and approaches from proxy caches, notification systems, and wireless intermediaries. In Section 4 we present a taxonomy of intermediary systems and classify characteristic systems accordingly. We conclude in Section 5.

2 Notification Systems

Notification systems are intermediaries that monitor changes in information sources on behalf of subscribed users. Whenever an update in the content of a monitored source is observed, the notification service evaluates the relevance of this change with respect to stored user profiles, and notifies accordingly interested subscribers.

One of the first examples of an intermediary notification system for Internet is SIFT, the *Stanford Information Filtering Tool* (SIFT) [10]. SIFT has been developed to provide large-scale information dissemination services to users that subscribe their interests to SIFT servers. At subscription, a user provides the system with an information-retrieval-style profile and additional parameters that declare the desired frequency of updates and expected amount of information to be received. SIFT employs the NNTP protocol to collect news articles published over USENET News. Collected content is indexed and filtered according to profiles registered in a SIFT database. Based on filtering results, SIFT produces notifications, which are delivered to interested subscribers via email.

Another example of a notification system for WWW resources is AIDE, the *AT&T Internet Difference Engine*. AIDE was designed to archive and handle multiple versions of changing WWW resources [6]. AIDE supports recursive tracking and differencing of Web pages and their descendants. Special emphasis is placed on the graphical viewing of changes taking place between subsequent versions.

AIDE is comprised of a centralized notification server, a version archive and a difference engine to find and display changes of pages on the WWW. Subscribers register in AIDE the list of URLs they wish to track, and a few parameters configuring the degree of desired notification. Alerts about changes in tracked pages reach the users via email; additional access is provided via the Web.

Both SIFT and AIDE are “server-based” and centralized tracking tools, since they rely on monitoring software running on a centralized server and not on a user’s machine.

An approach for building decentralized, distributed intermediaries has been explored with the *Financial Information Gathering Infrastructure* (FIGI), which is a notification system that retrieves, caches, filters and serves financial data [4].

Similarly to SIFT [10], a FIGI user profile is a set of long-term, continuously evaluated queries. These queries may include typical queries to Web databases, HTTP requests for World-Wide Web resources, access to general-purpose Search Engines or Subject Cataloging Sites, subscription to Usenet News, etc.

3 Intermediaries for Mobility and Ubiquity

As PDAs, thin clients, mobile phones and the like become more popular, the heterogeneity of client devices and the capacity mismatch between clients and servers are expected to grow. To cope with these trends, software infrastructures for Internet services have to: (a) Support seamless access from a variety devices; (b) Customize content to adapt to different terminal devices. (c) Support both synchronous (on-demand) and asynchronous modes of interaction with users, thus coping with frequent disconnections of wireless connections and user mobility. (d) Optimize the amount of useful content that reaches users through client devices with limited resources and restricted interfaces, by enabling service personalization, localization and filtering of information. (e) Guarantee high availability and robustness, as well as incremental performance and capacity scalability with an expanding user base. Clearly, these requirements cannot be met by “traditional” proxy infrastructures.

A number of projects have addressed related issues of wireless connectivity by deploying simple proxy-agents on the wireline network and/or the mobile hosts. Such agents mediate between origin servers and mobile applications, optimizing communication, dealing with disconnections, etc. A characteristic example of this approach is IBM’s *Web-Express* [8]. WebExpress is a *client/intercept*-based system that optimizes Web browsing on resource-poor clients connected over wireless connections, running a stripped-down version of the HTTP/1.0 protocol optimized for wireless communication. Both agents provide caching of content, and run a simple differencing protocol between the client-side and the server-side cache to optimize the provision of dynamic content.

Performance scalability problems are expected to grow with the wide-spread of Internet use. Service infrastructures will have to accommodate millions of simultaneous end-users connecting from a large heterogeneity of end-user devices and resulting to highly bursty workloads. At the same time, service infrastructures are required to sustain a very high throughput of service requests, support ubiquitous access through different client-terminals, exhibit 24x7 availability and robustness, be scalable in terms of capacity and performance. These requirements suggest the shift of computation, storage and complexity from the mobile devices and mobile support-stations into the networking infrastructure, in order to achieve performance scalability, better sharing of resources, higher cost efficiency and a streamlining of new service provision [3].

Such an approach would result to the deployment throughout the network of distributed, programmable and possibly mobile *intermediary servers*, mediating between primary information sources and various client systems. In this section we describe a number of systems seeking to comply with these characteristics.

WBI: A programmable framework for building various intermediary services is the *Web Browser Intelligence* or *WeB Intermediaries* (WBI) by IBM Almaden [2]. WBI is a programmable proxy server designed for the development and deployment of intermediary applications. The design of WBI is based on the notion of "information streams" that convey data from information providers (e.g., a Web server) to information consumers (e.g., a Web browser), and can be classified into unidirectional and bidirectional messages or transaction streams [2].

WBI provides an approach to develop intermediaries for the unidirectional transaction streams of the WWW. To this end, it provides five basic building blocks: request editors, generators, document editors, monitors and autonomous functions [2]. *Request editors* receive and modify requests before forwarding them towards their destination. *Generators* are abstractions of information sources that produce documents in response to requests. *Editors* receive and modify responses before passing them down to their destination. *Monitors* observe transactions without interfering. *Autonomous functions* run independently of information processing transactions and perform background tasks.

These blocks are called collectively *MEG*'s (Monitor/Editor/Generator) and can be assembled together into *WBI plugins*, which are used to construct data paths that transform information flowing from origin servers to users and implement different application scenarios. WBI constructs data paths dynamically, using a rule-based approach that determines which MEGs must be invoked and in what sequence.

iMobile: Building mobile services from proxy components is the main goal of the *iMobile* project of AT&T Research [9]. The iMobile proxy maintains user and device profiles, accesses and processes Internet resources on behalf of the user, keeps track of user interaction and performs content transformations according to device and user profiles. The architecture of iMobile is established upon the infrastructure of *iProxy*, a programmable proxy server designed to host agents and personalized services developed in Java. iMobile consists of three main abstractions: *devlets*, *infolets* and *applets*.

A devlet is an agent-abstraction for supporting the provision of iMobile services to different types of mobile devices connected through various access networks. The infolet abstraction provides a common way of expressing the interaction between the iMobile server and various information sources or information spaces (in iMobile terminology) at a level higher than the HTTP protocol and the URI specification (JDBC and ODBC, X10, IMAP, etc.). Finally, an applet is a module that processes and aggregates content retrieved by different sources and relays results to various destination devices.

At the core of an iMobile server resides the "let engine," which registers all devlets, infolets and applets, receives commands from devlets, forwards them to the right infolet or applet, transcodes the result to an appropriate terminal-device format and forwards it to the terminal device via the proper devlet.

TACC and Ninja: TACC is a general model for intermediaries, that provide transformation, aggregation, caching and customization of content [3]. TACC was proposed in the context of the BARWAN project at UC/Berkeley. The *transformation* functionality deals with various changes done by the proxy to the content of a single data object, to achieve filtering, format conversion and compression. *Aggregation* enables the collection of data from different origin servers, and the combination thereof in a pre-specified way that adds value to the collected information. *Caching* provided by TACC allows for the caching of original Internet content, post-transformation data, aggregated information and intermediate results. Finally, *customization* is achieved via the parameterization of services according to user preferences.

An evolution of the TACC model is provided in the context of the Ninja project from UC/Berkeley [7]. This project seeks to develop a robust infrastructure for Internet-scale systems and services in Java. The architecture of Ninja consists of *bases*, *active proxies*, *units* and *paths*. Bases are scalable platforms designed to host Internet services. They consist of a programming model and I/O substrate designed to provide high-concurrency, robustness, and transparent distribution of data to cluster-nodes. Moreover, they include a cluster-based execution environment (*vSpace*) that provides facilities for service component replication, load-balancing and fault-tolerance [7]. The programming model of Ninja consists of four *design patterns* that service programmers can use to compose different stages of a single service: wrap, pipeline, combine and replicate [7]. Active proxies perform data distillation, protocol adaptation, caching, encryption, etc. Examples of active proxies include wireless base-stations, network gateways, firewalls, and caching proxies. In Ninja terminology, a path is a flow of typed data through multiple proxies across a wide-area network; each proxy performs transformational operations to adapt the data into a form acceptable by the next service or device along the path. Similarly to WBI plugins, Ninja-paths can be established dynamically. Finally, units are abstractions for the client devices attached to the Ninja infrastructure, which range from PC's and laptops to mobile devices, sensors and actuators.

eRACE: The *extensible Retrieval, Annotation and Caching Engine* (eRACE) is a middleware infrastructure designed to support the development and deployment of intermediaries on Internet [5]. eRACE is a modular, programmable and distributed proxy infrastructure that collects information from heterogeneous Internet sources and protocols according to XML-encoded *eRACE profiles*, registered within the infrastructure, and end-user requests. Collected information is stored in a software cache for further processing, personalized dissemination to subscribed users, and wide-area dissemination on the wireline or wireless Internet.

eRACE supports personalization by managing personal profiles representing the interests of individual users. Furthermore, its structure allows the easy customization of service provision according to parameters, such as information-access models (pull or push), client-proxy communication (wireline or wireless; email, HTTP, WAP), and client-device capabilities (PC, PDA, mobile phone, thin clients). Ubiquitous service-provision is supported by eRACE thanks to the decoupling of content publishing and distribution from information retrieval, storage and filtering. The eRACE infrastructure can also incorporate easily mechanisms for providing subscribed users with differentiated service-levels at the middleware level.

eRACE is organized as a two-tier architecture. The first tier includes modules that manage services provided to users: the Service Manager, Content-Distribution Agents, and Personal Information Roadmap (PIR) Servlets. The second tier of eRACE consists of a number of protocol-specific *Agent-Proxies* like WebRACE, mailRACE, newsRACE and dbRACE that retrieve and cache information from the WWW, POP3 email-accounts, USENET NNTP-news, and Web-database queries respectively.

At the core of the second tier lies WebRACE, the Agent-Proxy that deals with information sources on the WWW, accessible through the HTTP protocols (HTTP/1.0, HTTP/1.1). WebRACE is developed in Java and consists of a Distributed Crawler [5], an Object Cache storing multiple versions of retrieved resources, and an Annotation Engine that indexes collected resources, executes user-queries, and produces "user alerts," encoded in XML. Other proxies have the same general architecture with WebRACE, differing only in the implementation of their protocol-specific proxy engines.

4 A Taxonomy of Intermediaries

In the previous sections, we presented an overview of typical intermediary systems developed to provide various kinds of Web services over fixed and wireless connections. Besides their differences, all these systems exhibit many common features, which can be analyzed along three main axes: (i) The architecte of the intermediary; (ii) The communication of the intermediary with clients and origin servers; (iii) The functionality provided by the intermediary over and above simple client proxying. Classification of intermediaries along these three axes can be refined further according to a set of characteristics specifying these axes in more detail. These traits can be taken into consideration when classifying existing and emerging systems. Below, we examine the particular features used to describe and classify different intermediaries.

Architecture: An important aspect that characterizes the structure of an intermediary system is whether it is composed of a "centralized" software module running on one host, or of a number of distributed, software modules residing at different hosts and communicating via message exchanges, distributed events, or shared memory. Distributed design has obvious advantages as it supports load distribution to multiple processors, scalability of performance, and improves system robustness and availability. Another characteristic of the structure of an intermediary system is the location of its software modules, which can reside in front of origin servers, in intermediate hosts residing on the network, between origin servers and clients, and on the client side. Finally, another aspect that characterizes the architecture of recent intermediary systems is their support for configurability and programmability. This is necessary for using intermediaries as an infrastructure for deploying various services. Programmability can be provided at different levels of abstraction and flexibility: (i) Through configuration parameters, which are defined in the information architecture of an intermediary system and guide the information retrieval, transformation and dissemination that the system makes. Different configurations can result to new services provided via the intermediary. (ii) With the employment of generic middleware platforms, such as Jini and mobile agents, which provide the substratum for extending intermediary systems. (iii) Through API's, software libraries and design patterns developed for programming new services in the intermediary's context [7]. (iv) With components that can be used as building blocks according to a higher-level programming model, combined in different ways and extended to define new services [2]. (v) With different combinations of the options described above.

Communication features: Another important issue in examining different intermediary systems is the way they interact with origin servers and client systems. Different

	Squid	**WebExpress**	**SIFT**	**AIDE**	**WBI**	**TACC**	**eRACE**
Architecture							
Structure	centralized	distr.	centr.	centr.	distr.	distr.	distr.
Location	network	network client	netw.	netw.	netw. client, server	network server	network
Programmability	-	-	-	-	√	√	√
Communication							
Proxy-Server Protocol	HTTP	HTTP	NNTP	HTTP	HTTP	HTTP	HTTP, NNTP SMTP
Client-Proxy Protocol	HTTP	"Reduced" HTTP	SMTP HTTP	HTTP SMTP	HTTP	wireless protocols	HTTP, SMTP GMS/SMS
Medium	wireline	wireless	wireline	wireline	wireline wireless	wireless wireline	wireline wireless
Access Model	pull	pull	pull/push	pull/push	pull/push	pull	pull/push
Timing	synch.	synch.	asynch.	asynch.	synch. asynch.	synch.	asynch. synch.
Functionality							
Customization	-	-	-	-	√	√	√
Filtering	-	-	√	√	√	-	√
Annotation	-	-	-	√	√	√	√
Transcoding	-	√	-	-	√	√	√
Aggregation	-	-	√	√	√	√	√
Caching	√	√	√	√	√	√	√
Crawling support	-	-	-	√	-	-	√
Versioned Caching	-	-	-	√	-	-	√

Table 1. Important Features of Intermediary Systems.

interaction approaches can be characterized according to the suite of supported *application-level communication protocols* and the *medium* employed to carry the interaction (wireline or wireless). Most intermediary servers on the Web "speak" the HTTP protocol and connect to Web servers and download information. Given, however, the existence of a variety of other information sources on Internet (email-lists, newsgroups, Web databases, WML sites), it can be useful for an intermediary system to support other popular application-level protocols, such as SMTP, IMAP, NNTP, WAP, and to have an extensible architecture that could easily incorporate new protocols [5]. The support for a wider variety of protocols is necessary in intermediary systems seeking to provide services to mobile clients, which receive information from the network typically through protocols streamlined for low-resource devices and wireless connectivity. Therefore, some intermediaries implement customary protocols for communication with particular mobile terminals [3], customize existing application-level protocols according to the requirements of the wireless channel [8], or interface with modules that "speak" wireless protocols (such as WAP/GSM, SMS/GSM), customizing their content accordingly [7].

The interaction between an intermediary and its clients and servers can be characterized further by the *access model* (push or pull) and the *timing of interaction* (on-demand or asynchronous). Interaction on the WWW is essentially pull-based. A number of research and commercial systems, however, have investigated the advantages of push-based approaches in wide-area networks. Most intermediaries work as user-agents, being constantly connected on the fixed network, collecting and filtering information on behalf of users. Therefore, intelligent content-push from intermediaries to their users can be combined with on-demand content provision, employing various protocols enabling information push, such as SMS/GSM and Java RMI over TCP/IP.

Proxy servers and transcoding intermediaries typically perform their intermediation activities in a synchronous (on demand) manner: the intermediary is activated upon receipt of a user request, interacts with origin servers and returns a reply "synchronously," while the user remains connected to the system. There is also a large number of "asynchronous" intermediaries, which perform operations on behalf of users on a longer-term basis, or perform complicated operations on-demand, providing users with a result at a later time.

Functionality: Finaly, different intermediaries can be classified according to the functionality they provide. A number of important intermediary functions have been identified in the literature [7]: customization, filtering, annotation, transcoding, aggregation, caching, versioning, etc. *Customization* refers to the restructuring of content-presentation according to end-user preferences, context, location, etc. *Filtering* is defined as the pruning of the collected information according to the "semantic" interests of individual end-users, before dissemination. *Annotation* is the processing of collected and filtered content, in order to provide users with additional, useful meta-information, such as summaries, keywords, highlights, etc. *Transcoding* is the transformation of the content from one format to another, to make it deliverable to terminal devices that support different formats or to optimize its transportation via wireless channels. *Aggregation* means the capability to integrate content from multiple origin servers into a single new service. *Caching* refers to the incorporation of a software-cache in an intermediary, in order to store part of its information flow and enhance information processing and sharing of intermediary resources among different end-users. Caching can be extended to *versioned caching*, so as to maintain multiple versions of an information resource, even after its expiration at the origin server [6]. Last, but not least, a functionality that will be considered commonplace in emerging and future intermediary systems, is that of crawlers. The importance of crawlers is increasing with the tremendous increase of origin servers and the need to achieve resource sharing, cost efficiency and high-performance while reaping content from the Web on behalf of large user communities [5].

A detailed classification of intermediary systems described earlier is given in Table 1.

5 Conclusions

The archetypal client-server model of the Web is being sumsumed by intermediaries that intervene between origin servers and client systems, as information flows from one end to the other during a simple Web interaction. The common goal of intermediaries is to improve the quality of end-user's Web experience by improving the performance of Web requests, by coping with information overloading, and by supporting seamless access to Web services via different terminal devices and physical connections. Their intervention ranges from very simple chores, like relaying requests and replies and transcoding content-formats, to more complicated tasks such as caching, filtering, personalization, and crawling. Intermediaries represent a useful abstraction for designing, developing, analyzing and comparing emerging software infrastructures for "next-generation" Web services.

In this paper we presented an overview of a wide range of systems that can be described as intermediaries, classifying them in a number of broad categories according to their basic functionality: Web proxies, notification systems, wireless-Web proxies, infrastructural intermediaries. We examined the requirements arising from the need to support personalization, mobility and ubiquity under high loads. We identified and refined a set of important properties and characteristics, which can be used for: (i) the classification of existing systems and the analysis of their capabilities; (ii) the comparative study of different systems; (iii) the design of new intermediary systems. Based on this set of properties, we introduced a detailed taxonomy of characteristic intermediary systems (Table 1), identifying and investigating important features. From this taxonomy, it becomes evident that more recent systems typically consist of distributed software modules, which support a wider variety of client devices and protocols. Furthermore, that emerging intermediary systems have infrastructural characteristics as they provide abstractions and modules for the development and deployment of new applications and services.

References

[1] R. Aiken, M. Carey, B. Carpenter, I. Foster, C. Lynch, J. Mambreti, R. Moore, J. Strasnner, and B. Teitelbaum. Network Policy and Services: A Report of a Workshop on Middleware. Technical Report RFC 2768, IETF, 2000. http://www.ietf.org/rfc/rfc2768.txt.

[2] R. Barrett and P. Maglio. Intermediaries: New Places for Producing and Manipulating Web Content. *Computer Networks and ISDN Systems*, 30(1–7):509–518, April 1998.

[3] E. Brewer, R. Katz, E. Amir, H. Balakrishnan, Y. Chawathe, A. Fox, S. Gribble, T. Hodes, G. Nguyen, V. Padmanabhan, M. Stemm, S. Seshan, and T. Henderson. A Network Architecture for Heterogeneous Mobile Computing. *IEEE Personal Communications Magazine*, 5(5):8–24, October 1998.

[4] M. Dikaiakos and D. Gunopulos. FIGI: The Architecture of an Internet-based Financial Information Gathering Infrastructure. In *Proceedings of the International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems*, pages 91–94. IEEE-Computer Society, April 1999.

[5] M. Dikaiakos and D. Zeinalipour-Yazti. A Distributed Middleware Infrastructure for Personalized Services. Technical Report TR-01-4, Department of Computer Science, University of Cyprus, December 2001.

[6] F. Douglis, T. Ball, Y.-F. Chen, and E. Koutsofios. The AT&T Internet Difference Engine: Tracking and Viewing Changes on the Web. *World Wide Web*, 1(1):27–44, January 1998.

[7] S. Gribble, M. Welsh, R. von Behren, E. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R.H. Katz, Z.M. Mao, S. Ross, and B. Zhao. The Ninja Architecture for Robust Internet-scale Systems and Services. *Computer Networks*, 35:473–497, 2001.

[8] B.C. Housel, G. Samaras, and D.B. Lindquist. WebExpress: A Client/Intercept Based System for Optimizing Web Browsing. *ACM/Baltzer Journal of Mobile Neworking and Applications (MONET)*, 3:419–431, 1998.

[9] H. Rao, Y. Chen, D. Chang, and M. Chen. iMobile: A Proxy-based Platform for Mobile Services. In *The First ACM Workshop on Wireless Mobile Internet (WMI 2001)*, 2001.

[10] T. W. Yan and H. Garcia-Molina. SIFT - A Tool for Wide-Area Information Dissemination. In *Proceedings of the 1995 USENIX Technical Conference*, pages 177–186, 1995.