Towards a Distributed Architecture for Value Added Services to Digital Libraries

Maurizio Marchese, Aliaksei Yanchuk, Fausto Giunchiglia
{marchese, yanchuk, fausto }@dit.unitn.it

Department of Information and Communication Technology
University of Trento, Italy

Abstract. Scientific communities are relying more and more on value added services offered by different systems on top of digital document repositories and libraries. Current systems, like CiteSeer and Google Scholar, offer important services to these communities but show their limitations when it comes to scalability, decentralize control and plan for a sustainable model for community support. In this paper we propose and describe a new distributed and service oriented architecture to support the creation of such services. Our architecture aims at lowering the technological barriers to build distributed community environments by providing a small number of key core services and a layered architecture for extension services.

1 Introduction

Current evolution of software tools for managing, searching and navigating scientific literature materials, both on the Web, digital libraries and local repositories, is creating a scenario where many scientific communities are starting to rely more and more on the added value of such services. Different systems are now available, starting from the pioneering work of CiteSeer [7], the premier repository of scientific papers for the computer science community, and arriving to the current beta version of Google Scholar[1], a commercially-managed system that proposes a business model for such academic search engines. Moreover, rumors about new features in the next versions of the most popular operating systems concern the availability of advanced and efficient content search capabilities already embedded in the operating system.

Current systems (with some specific differences) support, either internally or externally, a number of services, among which: crawl and collect documents from the web, convert them in different electronic formats, automated meta-data extraction (such as extract citations from the text, identify citations referred to same papers) and visualize the context of citations in the body of articles. CiteSeer also ranks papers and authors in various ways, and can identify similarity among papers. Google Scholar indexes also commercial digital libraries

[1] Look at: `http://scholar.google.com`

and brings the user to the appropriate portal where she can decides eventually to purchase the document.

These kind of services have proved a vital resource for academic communities. However, despite their community value, the future of such services is uncertain without a sustainable model for community support. As web-based collaborative environments will become more easily accessible, usable and with lower maintenance costs, the number and scale of these kind of web communities will increase. In the current Web, this issue is solved through centralization. Current generic search engines (like Google, Yahoo! and others) are best example of this: they are centralised indexes of generic contents. This is only possible because (a)these centralised search engines provide a one-size-fits-all-users service, and (b) the number of information providers is low compared to the number of information consumers (only 30 million active web sites for an estimated billion people online, estimates of early 2005 [2]). Our basic belief is that such a centralisation in web communities in general (and in particular in scientific communities) is undesirable for multiple reasons:

- as the web becomes more knowledge intensive, avoidance of centralised control of access to information becomes more an important issue, both technically (access bottlenecks) and strategically (society at large);
- as the barriers to build, join and maintain such open collaborative environments will be lowered, the number of information providers will increase and the centralized model will not scale in the new scenario.

In the present paper we propose a distributed P2P architecture for a renew version of a CiteSeer-like system, that we call CiteSeer.EU. Such new system will continue to provide CiteSeer traditional features, but will be capable to support different knowledge domain as well as to allow aggregation of distributed resources, using a P2P service oriented infrastructure. To this end, we are currently working at two levels:

- the first implementation of CiteSeer.EU delivers a centralized community server architecture which offers a set of value added services,in particular content enrichment with semantically well-defined meta-data, to the authenticated community users. We are currently developing and validating a prototype system within a consortium of Universities and research centers that includes the University of Trento, the University of Siena, CINECA, itc-IRST as well as a collaboration with the group at Penn State's School of Information Sciences and Technology that has developed and is maintaining the original CiteSeer system.
- to support scalability we are investigating the evolution of such centralized community server system into a distributed architecture capable to support secured and dynamic collaboration among knowledge intensive web communities.

[2] from `http://news.netcraft.com/archives/webserversurvey.html`

In this paper we focus on the second point and present our vision for such distributed architecture. The remainder of the paper is organized as follows. In Section 2 we briefly discuss services and limitations of current CiteSeer and then introduce the main concepts of our distributed architecture. In Section 3 we present our main design choices. In Section 4 we show how the main operations performed on our system, namely query and search, can be implemented. Conclusions and future work conclude the paper.

2 CiteSeer.EU Distributed Platform Concepts

CiteSeer [7] is a scientific literature digital library and search engine that focuses primarily on the literature in computer and information science. CiteSeer was developed at the NEC Research Institute by Steve Lawrence, Lee Giles and Kurt Bollacker. It is currently hosted at Penn State's School of Information Sciences and Technology under the direction of Professor Lee Giles.

CiteSeer indexes PostScript and PDF research articles on the Web, and provides a number of features. In particular the system support Autonomous Citation Indexing (ACI see also [7]). i.e. it can automatically create a citation index from literature in electronic format. Moreover, CiteSeer also ranks papers and authors in various ways, and can identify similarity among papers. The critical parts of the system are those involved in meta-data extraction.

Although current CiteSeer is an excellent research tool, it faces big challenges to evolve into a digital library tool capable to cope with today growth of information to be processed. In fact, CiteSeer is a centralized system. Scaling up centralized system is possible only extensively, but not intensively.

We are pursuing the opportunity to re-design the CiteSeer system based on current technologies and theoretic research results in order to meet this specific challenge. To this end we propose to use distributed software technologies to achieve scalability and performance on low-profile computer hardware.

The new system is going to be a distributed service oriented platform with up to date tools to cope with the the increased acquisition of sources. The large data volumes to be processed requires special kind of software and hardware to handle them in a time-efficient manner. Traditional engineering approach for such large-scale computation projects was to design a software for massively parallel processing (MPP) systems (e.g., using message-passing interface) to spread the computation load on the CPUs in the supercomputer [3]. Recent years were characterized by important advancements in the personal computer technologies and distributed computation techniques such as Service Oriented Architectures [11] and Grid Computing [4].

Our proposed system wants to leverage from current state of the art distributed technologies and deliver a new architecture for allowing the user a homogenous access to scientific documents available in both digital libraries and open Web resources.

A key concept in the proposed new architecture is the notion of *Global service*: i.e. the consumable service provided by heterogeneous entities on the physically

highly distributed network environment that is perceived by the service consumer as a single, integral service entity with a single endpoint in service consumer's view. In other words, from end-user's point of view, employing entire distributed environment to service user's request is the same as to employ just local computational facility. Thus, global service is both *location-* and *scale-transparent*. The unique feature of the global service is that global service implementation is collaboration-oriented. At any given moment of time, a request to the global service is serviced by the available components located on the network across control domains that are capable of best-serving the request. Such opportunistic behavior of the implementation relates both to the concepts and issues investigated in Grid technology and SOA research[8].

The goal of our system is to define and support a number of dynamic global services to the distributed user audience.

2.1 Core Concepts

Software as a Platform Building decentralized environment of site peers will require providing software solution that will support complete software life cycle. The ultimate challenge of a decentralized environment is that user groups will have contradictory requirements for software. CiteSeer.EU addresses this challenge by adopting the concept of the *platform*. The proposed architecture provides both comprehensive software solution for distributed community environment that is usable out-of-the-box, as well as means to extend and refine particular platform components, including complete re-implementation, should participating site choose to do so. The platform will lower the barriers to build, join and maintain such open collaborative environments.

The Site Concept The *site* term refers to the installed Platform instance with optional components deployed on top of the Platform. The site is characterized by the following properties:

1. Subject to single control domain: site has full control over authentication and authorization of its users;
2. Homogeneous network environment: the site operates on top of LAN or WAN network hardware;
3. User auditorium sharing the same software usage interests.

A site can be also described as the software / hardware infrastructure to support user groups (primarily research-oriented) to securely interact with the remote groups.

The Community Concept Remote user groups interact by uniting into a *community*. A community is a union of numerous sites that unite to pursue a common goal (typically, common research objective): for instance sites $Site_0$, $Site_1$, and $Site_2$ form a community by establishing and joining a *Virtual Community Network*. The Virtual Community Network (referred to further as *VCN*)

concept is an extension to the well-know Virtual Private Networking (VPN) [9] concept. VCN is a dynamic open environment where multiple sites can dynamically join and disjoin, and at the same time securely communicate with each other via public network infrastructure(such as Internet). The VCN infrastructure is built upon widely used network security tools and technologies such as Firewalls, VPNs, encrypted communication (SSL), and connector software provided by the proposed platform.

Communities, Sites, and User Groups Relations between communities and sites may range from simplistic to very complicated. Relationship between two particular sites may be characterized by different degree of *friendliness*. The friendliness describes how much $Site_A$ trusts $Site_B$. Trust may range from *neutral* (reasonable trust) to *friendly* (complete trust). Mechanism to ban particular site are included in the platform.

To structure work of the large user audience, user groups (working groups) can be created to pursue particular (e.g., research) targets.

Value Added Services In the proposed architecture any kind of value added service comes as an extension service to the Platform. We envisage a number of such extension service, among which:

- enhanced support for content enrichment with semantically well-defined metadata
- enhanced support for document similarity, classification, clustering and social network analysis (i.e. analysis of authors social networks, affiliation and founding agencies networks etc.)
- enhanced support for querying functionalities (template querying, similarity querying etc.)

3 CiteSeer.EU Platform Architecture

One main challenge to the practical distributed CiteSeer.EU Platform implementation is the selection of the correct technological paradigm. The platform goal is to provide maximum pluggability, adaptation, reuse, and shrink-to-fit as well as enlarge-to-cope features.

Out of all technologies and methodologies available, the Service-Oriented Architecture [12] (referred to as SOA further on) seams to provide the best applicability for the purposes of the present platform. Compared to other technologies and methodologies, its keen characteristic is a clear guidance to define and build distributed software around well-established technology-neutral functionality re-use. The re-usability trail of the SOA should be distinguished from the distributed object-oriented technologies, such as CORBA or DCOM. The Distributed Object Oriented architectures focus on the way a request from one object is delivered and dispatched on the other object, the mechanics of the

networking. The focus of SOA is mainly platform independence and ease of reusability.

The rule of thumb for service-oriented design is design for reuse from the very beginning [2]. The concept of *service* is by its very nature software- and platform-neutral, whereas with other technologies number of limitations apply on the design, implementation, and software operation. Unlike any other architectures, SOA itself is so generic that a concrete SOA implementation is necessary to deliver a software product. Today SOA offers numerous implementations such as Web Services, Jini, OpenWings, JGroups — these are the widely known ones. At the same time, it is possible to use several of the SOA implementations seamlessly within the same project without introducing considerable technical difficulties.

Moreover, SOA allows designing more reliable self-healing software relatively to most other alternatives for distributed communications. This minimizes integration time, which minimizes deploy time for complex systems, such as CiteSeer.EU.

3.1 Implementing SOA for CiteSeer.EU

Our overall targets may be broken down into the following major components:

1. Infrastructure services and Software Development Kit (SDK) product line designed to provide shrink-to-fit capabilities of the deploying organization and seamless community integration.
2. Security and Data Delivery services providing the functionality baseline.
3. Extended services offering useful services on top of security and data deliver services.

Effectively, the distributed CiteSeer.EU is designed as a "platform" since the out-of-the-box installation would provide little or no utility to the users of the environment the platform is deployed at. The "bare" architecture is only a generic architecture about sharing data with anyone. Moreover, the proposed CiteSeer.EU platform addresses major issues that are to be tackled by the software designers to share data in the distributed decentralized community environment. Figure 1 illustrates the proposed layered architecture composed of a core layer of services on top of which users can build *extension services*, that may be site-specific, group-specific, or community services.

The notion behind this layered architecture is linked to the concept of extended SOA (xSOA), as detailed in [10]. xSOA is a stratified service-based architecture to attempt to streamline, group together and logically structure the functional requirements of complex applications that make use of the service-oriented paradigm. What we propose here is an implementation of xSOA in the specific context of content distribution.

Our core layer will provide the following basic services:

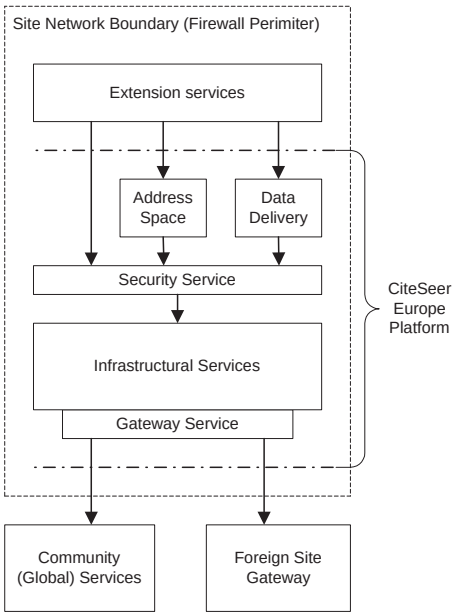

Fig. 1. CiteSeer Europe Vertical Architecture

Address Space Service To get any kind of data served to the user, that piece of data must first be found somewhere. With numerous solutions for metadata repositories available on the market (e.g., from major relational database vendors), our architecture proposes a small yet crucial conceptual difference: CiteSeer.EU records not *where* the data is located, but *who* can provide the data. The implementation of this functionality is the Address Space Service.

The importance of such approach relates to the flexibility this service supports, in particular it provides for considerable decoupling from actual storage solution. To draw a parallel with existing technologies, it is possible to change a hosting provider for a site and move all the content to a physically different computer. At the same time Web search engine would still contain accurate data since the computer will still be identifiable by the DNS name. However, if site was indexed with IP address instead of DNS name, Web search engine's data will be inaccurate. It would not be possible to connect to the Web server based on results of the query. But what if one would like to change the name of its service? Presently, the Web search engine would have to re-index the site, although it is *the same* site the engine already has in its cache. The Address Space Service is built to address precisely this kind of challenges.

There are two kinds of "addresses" in the proposed approach: the "data piece identification" and the "supplier identification".

The data "data piece identification" is the algorithm to calculate the "indexes" of the data in a way that it is meaningful to other entities in the system.

The theory and the application development experience of the database technologies over last decades mandates that a tuple stored in the database is accessed via primary key that is known to the caller prior the database call is placed. We propose a different approach based on a number of assumptions:

- The data piece identification is expressed as a superposition of atomic indexes, usually provided by the supplier of the data. The superposition is *highly probably* globally unique. If several data pieces become ambiguous, additional atomic indexes can be introduced to disambiguate two pieces of data, or the duplication would have to be detected and eventually resolved.
- Individual atomic indexes are *calculable*. At query time, it is possible for a caller to construct an index that would provide good enough results (from the point of view of the caller);
- The caller is not required to supply all atomic indexes to the calculation (see section 4 for more details on the query implementation in this framework)

So the address $A(D_i)$ of the i^{th} data piece D_i can be formalized as pair:

$$A(D_i) \equiv < I_i, \Psi_j >; \tag{1}$$

where I_i is the set of individual atomic indexes $(i_1, i_2 \dots i_n)$, and Ψ_j is the j-th supplier identification.

The "supplier identification" Ψ_j is a *pseudo-address*, i.e. the piece of information that points out to a resolvable service endpoint that is capable to supply a data stream for the data piece identification. The technical entity that provides for this resolution is the Data Delivery Service. The information codified in the pseudo-address is sufficient to perform unambiguous discovery of the service endpoint, establish communication, and request data stream.

Data Delivery Service. The service that takes the pseudo-address and data piece identification and delivers the actual data is the Data Delivery Service. Functionally, it is decoupled from the Address Space Service on this premise: it knows *how* to reach the entities described by the pseudo-address obtained from the Address Space Service.

The benefit for this design is obvious in the example with the re-allocated site we presented above. In this case, the Address Space Service will provide that data piece DP_ϕ may be obtained from the file $file_i$ on the web site $WebSite_A$. The site may move to a completely different location, and the Address Space will still be accurate. It is the data delivery configuration that needs to be updated on this major change. Moreover, this design simplifies the creation of fail over copies for disastrous situations, such ISP service disruption on one of the sites of the community. To achieve this, the address space would just need to know that more than one web site can provide the data for DP_ϕ.

Data Deliver Network: The benefit of having the above separate services becomes even clearer when these two services are considered in the community context. Provided that the pseudo-addresses are organized in such a way that

the Data Delivery Service can distinguish between local and external locations of the data, sharing data between sites participating in the community is equivalent to sharing the snapshots of the local data with all the participating sites. Given that the Data Delivery Service knows on the connectivity details, all the data needed on $Site_B$ that originated on $Site_A$ will be copied locally "on demand", with users not even knowing it comes from different site.

The implementation of the proposed approach leads to a new paradigm: the Data Delivery Network (DNN). DDN is the facility of the overall platform that should be considered a Virtual Community Network that seamlessly provides secure data integration and exchange between participating sites and communities. The Data Delivery Network concept extends the Content Delivery Network[3] (CDN) concept. In CDN scenario, there is a centralized content provider and a large end-user audience that is geographically distributed. However, in our proposed architecture based on VCN there's no centralized content provider. Along with actually delivering community-shared data from one site to other sites, the DDN network should implement P2P routing mechanism (like in [1]) as well as authenticity mechanism. DDN conceptually consists of two interoperable services: *distributed address space*, that stores meta-information about available data sources and their pseudo-addresses, and *pseudo-address resolution and delivery* service.

This breakdown relates to the information/data source classification in the following manner:

- Distributed Address Space service maintains data sources;
- Pseudo-address resolution and delivery service allocates physical data source from the pseudo-address provided and delivers data from it to the user requesting it. The service is designed to provide standardized data transmission interface among different communication end points, thus shielding users and service providers from technical issues such as routing and interoperability.

Distributed Address Space service is *scoped* service. The concept of the scope means that user may easily select the subset of the whole address space where the search is performed. The end user can easily scope her search: limiting search on selected digital libraries, sites, or entire community and / or its particular sub-communities.

The Pseudo-address resolution and delivery service combines the functionality of both Fa cade, Adaptor or Proxy software design patterns [5], depending on the nature of the entity addressed by the pseudo-address. Practically the concrete version of the service can understand only the predefined set of systems and protocols and transfer mechanisms. The "bare" service is an intelligent caching framework for different systems plus plug-ins, and communication protocols, including non-TCP traffic. In this way the service may be programmed in a manner that supports re-configuration on-the-fly and addition / removal of the plug-ins.

[3] CDN is essentially an overlay network of customer content, distributed geographically to enable rapid, reliable retrieval from any end-user location. See http://www.isp-planet.com/technology/cdn_connection.html

Other services, namely Security service, Infrastructural Services and Gateway Service, are part of the core layer, but due to lack of space we will not detail them here.

4 Querying Functionality

Any kind of data querying functionality (as well as more sophisticated value added service) comes as an extension service in the proposed architecture. Although it is possible to some degree to compute the address of the requested data piece, in some cases it is better to offer users other services, such as full-text search or meta-data search. Such extension services would provide all data piece addresses that would need to be supplied to the data delivery service.

A number of issues need to be solved to support such functionalities in the proposed distributed environment:

Data Accounting To understand, how, when, and if the data could be queried for, we need to describe how the data accounting is implemented. Out-of-the-box Data Delivery Network installation contains no information at all. There are five entities involved in the accounting of the data piece:

- The data piece origin, the O_{D_i};
- The data piece itself, the D_i;
- The storage service where the D_i is stored: S;
- Address Space service, A_S;
- Data Delivery Service D_S.

In our architecture, the storage service , where the D_i are located, is an extension service to the platform. In order for the Address Space Service to know the D_i, and the Data Delivery Service to be capable to deliver it, the D_i should be registered so that:

- Address pair for D_i, i.e. $A(D_i) =< I_i, \Psi_S >$, should be allocated and stored in the address space,
- the Data Delivery Service should be able to compute D_i, i.e. the service should be able to compute D_i from Ψ_S and I_i.

The Address Space service is a *passive* service, i.e. it doesn't search actively for new data. An example could be that the system installation may be provided with autonomous agents (such as Web crawlers) that scan the storages available and inform Address Space and Data Delivery services of the existence of data pieces.

To search the data in such context, three possible scenarios are possible:

- Knowledge of $A(D_i)$ is available prior to query time, and feed it directly to the Data Delivery Network services;
- Construction of a template of the address is done at query time and used to perform the lookup query;

– Extension services are called to perform advanced queries and then feed the resulting address to the Data Delivery Network Services

Template Querying Let us concentrate on the template querying. This is the mechanism of the Address Space Service to implement the query-by-example semantics. The qualifying condition for such implementation is that the environment where data pieces are registered in the Data Delivery Network allows computing the number of individual indexes of the data piece addresses capable to resolve the $A(D_i)$ address almost unambiguously (if ambiguities arise we might get duplicates or not relevant results).

The template $T^* =< I^*, \Psi^* >$ is a pair of two sets I^* and Ψ^*, where I^* is the set of example indexes the caller assumes the D_i would have, and Ψ^* is the set of example pseudo-addresses the caller assumes the data pieces may be served by. The matching function M is a function that checks if template T^* is actually reflected on the data piece address $A(D_i)$. We express it formally, as follows:

$$M : T^* \rightarrow A(D_i) \equiv \begin{cases} \hat{I}_i = (M_I : I^* \rightarrow I_i), \hat{I}_i \in I_i \\ \hat{\Psi} = (M_\Psi : \Psi^* \rightarrow \Psi_S), \hat{\Psi} \in \Psi_S \end{cases} \quad (2)$$

Where $\hat{I}_i$ and $\hat{\Psi}$ are respectively the subsets of the individual atomic indices and of the pseudo-addresses that satisfy a given matching criteria. For instance , we can implement M so that the template T^* is said to match the address $A(D_i)$ if ALL example indexes AND pseudo addresses can be reflected on the indexes and pseudo-addresses of a particular data piece address. Less restricting criteria can also be applied.

Searching data The method described in the previous section has the big disadvantage that the user should know the way the individual indexes are calculated to be able to query the service. This may work in small communities. Larger communities will need to establish facilities where the knowledge of the individual indexes that make up the data piece identity is not required.

Such services are pure extension to the proposed architecture. Since the Address Space knows of the existence of data piece only on the initiative of an external entity (i.e. indication of URLs where to start the crawling process), extension service may provide associations between query-answering information and the data piece address or a template that could be used to query later the data.

In the example of full-text search, the full text index would be created as an extension service to the the system. To allow full-text searching, the data pieces would first be fed to the full-text search service for processing. The latter would eventually register the data piece with the Address Space Service. Queries based on full-text will be executed on the full text service. This service would return addresses or templates that match the results of the query.

5 Conclusions and Future Work

In this paper we have presented a distributed architecture for the management and use of digital content. The architectural framework is based on P2P-style

architecture to bring the dynamism, extensibility and decentralize control from P2P system. A P2P approach is also pursued by other groups [6] in the same context of scholarly publications. Unlike such proposals we based the data exchange and delivery mechanisms on Service Oriented Architectures leveraging platform-independence and the dynamic discovery/bind/invoke mechanism of both core and extension services. The proposed implementation of a decoupled Address Space and Data Delivery services is actually the core of the proposed platform. The purpose of the Address Space service is to track the information assets of sites and community, while the Data Delivery service provides universal application-level access point to the data stored in the community. Internally, the Data Delivery service can use a number of available techniques (P2P routing, SOA, CORBA, even classic "client/server" etc.) useful to get the data from source to requesting user.

The proposed architecture together with the querying functionalities described in section 4 provides a number of benefits to the extensibility and transparency of the system:

- Addresses of data pieces may be considered immutable. This means that search engines may be built on the premise that once the data piece has been indexed, there's no need to re-index it again;
- Increased results accuracy. Instead of having references to copies of the same data piece scattered around community, a single address is returned and the caller may be selective on the entity that can provide the data piece (for instance considering nearest geographic location in a Grid environment);
- Template mechanism allows searching similar data pieces based on their content and their location: e.g., it is possible to use global full-text search service to get addresses of data pieces, and then construct templates based on the addresses that could be used to search data on particular sites;
- Inherent specialization. Sites providing extended query services are not required to have local copies of data they index. This ultimately means that numerous search services may be offered by sites in the community independently from each other that could index entire community data without actually having a local data copy;

However much more work need to be done to realize and validate our proposed architecture. Our future work includes the implementation of the distributed architecture using as starting point the current community server architecture version of CiteSeer.EU.

6 Acknowledgments

The Citeseer.Eu project is being lead by Marco Gori and Fausto Giunchiglia and involves the collaboration of many people, including Lee Giles, Marco Maggini, Ernesto Di Iorio e Augusto Pucci. All these people are thanked for the continuous support and contributions to the project.

References

1. B. Liskov A. Gupta and R. Rodrigues. Efficient routing for peer-to-peer overlays. In *NSDI '04: Proceedings of the First First Symposium on Networked Systems Design and Implementation*, San Francisco, CA, 2004., 2004.
2. Vincenzo D'Andrea Maurizio Marchese Alexander Ivanyukovich, G. R. Gangadharan. Towards a service-oriented development methodology. In *IDPT '05: Proceedings of the Eighth International Conference on Integrated Design and Process Technology*, page 8, 2005.
3. Ralph Duncan. A survey of parallel computer architectures. *Computer*, 23(2):5–16, 1990.
4. Ian Foster and Carl Kesselman, editors. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
5. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
6. Jinyang Li M. Frans Kaashoek David R. Karger Robert Morris Scott Shenker Jeremy Stribling, Isaac G. Councill. Overcite: A cooperative digital research library. In *4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, Ithaca, New York, February 2005.
7. Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *Computer*, 32(6):67–71, 1999.
8. Frank Leymann and Kai Gunztel. The business grid: Providing transactional business processes via grid services. In *ICSOC '03: Proceedings of the First International Conference on Service Oriented Computing*. Lectures Notes of Computer Science, Springer Verlang, 2003.
9. G Huston P Ferguson. What is a vpn? In *Open Signaling for ATM, INTERNET and Mobile Networks (OPENSIG'98)*. ACM Press, 1998.
10. M. P. Papazoglou and Dubray. Technical report $dit - 04 - 058$: A survey of web service technologies, 01 Feb., 2001.
11. M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Commun. ACM*, 46(10), 2003.
12. Mike P. Papazoglou. Service -oriented computing: Concepts, characteristics and directions. In *WISE '03: Proceedings of the Fourth International Conference on Web Information Systems Engineering*, page 3, Washington, DC, USA, 2003. IEEE Computer Society.