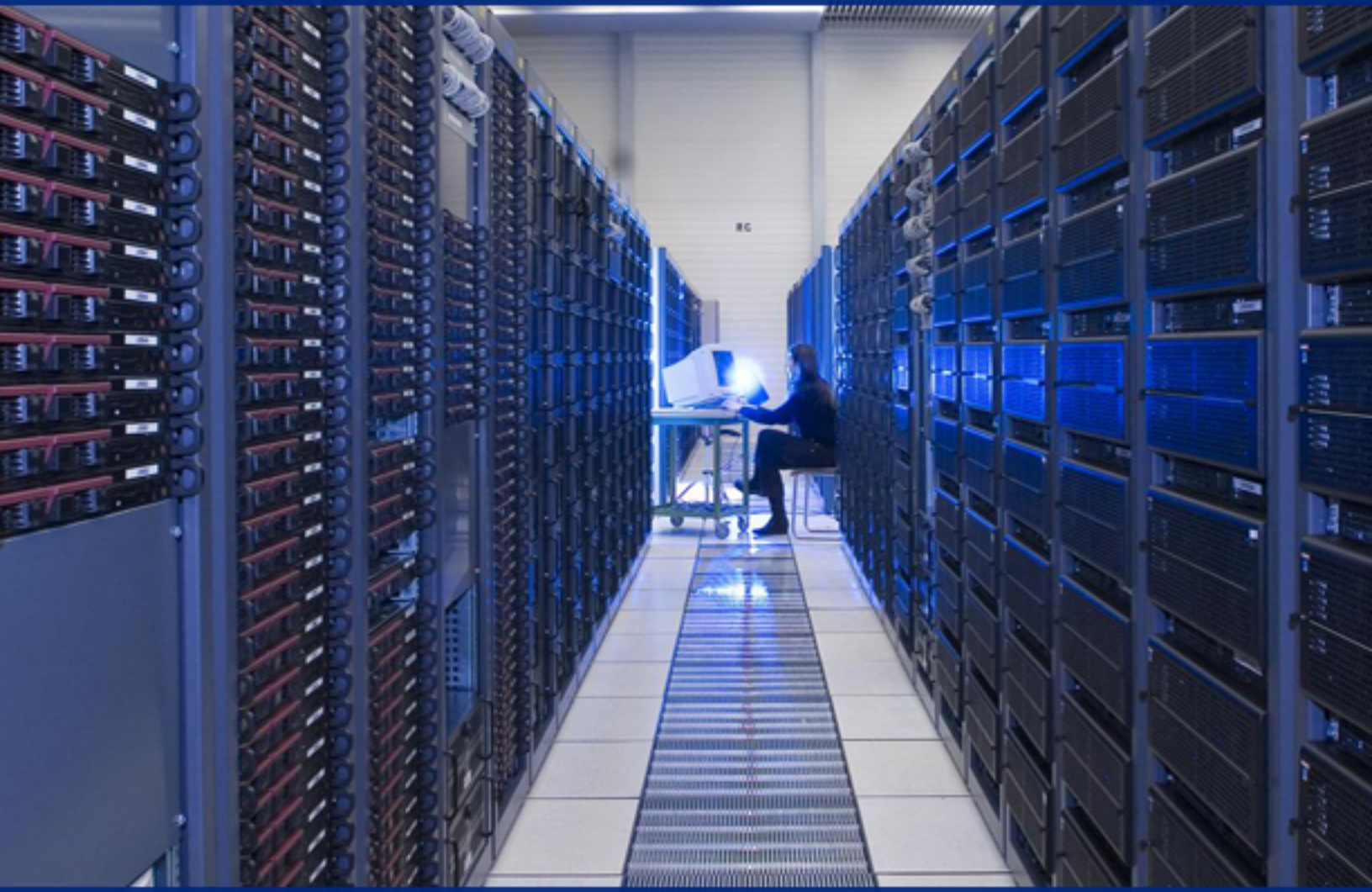
Peer-to-Peer and Grid Computing



Troy Carr

Ilana Sherry

First Edition, 2012

ISBN 978-81-323-0964-2

Published by:
Academic Studio
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1

Peer-to-Peer

A peer-to-peer system of nodes without central infrastructure.

Centralized server-based service model.

Peer-to-peer (**P2P**) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts. Peers are both suppliers and consumers of resources, in contrast to the traditional client–server model where only servers supply, and clients consume.

The peer-to-peer application structure was popularized by file sharing systems like Napster. The concept has inspired new structures and philosophies in many areas of human interaction. Peer-to-peer networking is not restricted to technology, but covers also social processes with a peer-to-peer dynamic. In such context, social peer-to-peer processes are currently emerging throughout society.

Architecture of P2P systems

Peer-to-peer systems often implement an abstract overlay network, built at Application Layer, on top of the native or physical network topology. Such overlays are used for indexing and peer discovery and make the P2P system independent from the physical network topology. Content is typically exchanged directly over the underlying Internet Protocol (IP) network. Anonymous peer-to-peer systems are an exception, and implement extra routing layers to obscure the identity of the source or destination of queries.

In *structured* peer-to-peer networks, peers (and, sometimes, resources) are organized following specific criteria and algorithms, which lead to overlays with specific topologies and properties. They typically use distributed hash table-based (DHT) indexing, such as in the Chord system (MIT).

Unstructured peer-to-peer networks do not provide any algorithm for organization or optimization of network connections.. In particular, three models of unstructured architecture are defined. In *pure peer-to-peer* systems the entire network consists solely of equipotent peers. There is only one routing layer, as there are no preferred nodes with any special infrastructure function. *Hybrid peer-to-peer* systems allow such infrastructure nodes to exist, often called *supernodes*. In *centralized peer-to-peer* systems, a central server is used for indexing functions and to bootstrap the entire system.. Although this has similarities with a structured architecture, the connections between peers are not determined by any algorithm. The first prominent and popular peer-to-peer file sharing system, Napster, was an example of the centralized model. Gnutella and Freenet, on the other hand, are examples of the decentralized model. Kazaa is an example of the hybrid model.

P2P networks are typically used for connecting nodes via largely *ad hoc* connections. Data, including digital formats such as audio files, and real time data such as telephony traffic, is passed using P2P technology.

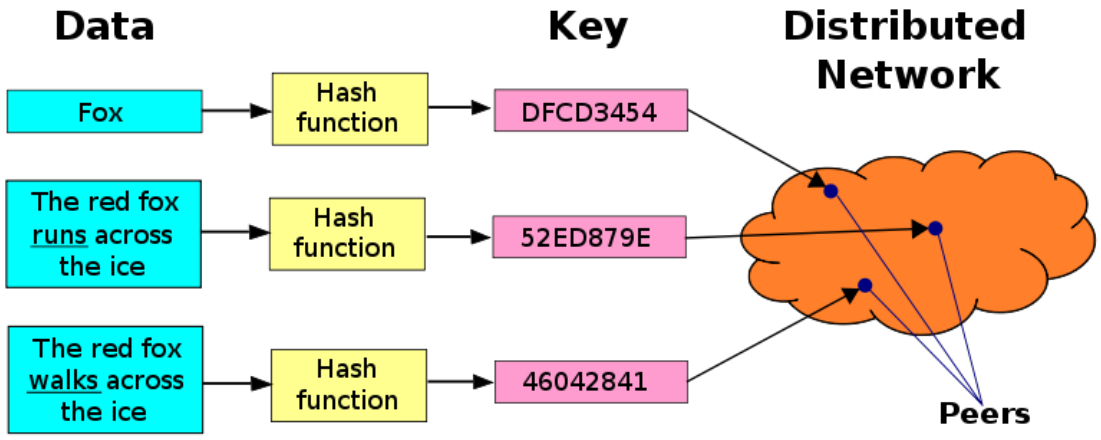
A pure P2P network does not have the notion of clients or servers but only equal *peer* nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network. This model of network arrangement differs from the client–server model where communication is usually to and from a central server. A typical example of a file transfer that does not use the P2P model is the File Transfer Protocol (FTP) service in which the client and server programs are distinct: the clients initiate the transfer, and the servers satisfy these requests.

The P2P overlay network consists of all the participating peers as network nodes. There are links between any two nodes that know each other: i.e. if a participating peer knows the location of another peer in the P2P network, then there is a directed edge from the former node to the latter in the overlay network. Based on how the nodes in the overlay network are linked to each other, we can classify the P2P networks as unstructured or structured.

Structured systems

Structured P2P networks employ a globally consistent protocol to ensure that any node can efficiently route a search to some peer that has the desired file, even if the file is extremely rare. Such a guarantee necessitates a more structured pattern of overlay links. By far the most common type of structured P2P network is the distributed hash table (DHT), in which a variant of consistent hashing is used to assign ownership of each file to a particular peer, in a way analogous to a traditional hash table's assignment of each key to a particular array slot.

Distributed hash tables



Distributed hash tables

Distributed hash tables (DHTs) are a class of decentralized distributed systems that provide a lookup service similar to a hash table: (*key*, *value*) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows DHTs to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.

DHTs form an infrastructure that can be used to build peer-to-peer networks. Notable distributed networks that use DHTs include BitTorrent's distributed tracker, the Kad network, the Storm botnet, YaCy, and the Coral Content Distribution Network.

Some prominent research projects include the Chord project, the PAST storage utility, the P-Grid, a self-organized and emerging overlay network and the CoopNet content distribution system.

DHT-based networks have been widely utilized for accomplishing efficient resource discovery for grid computing systems, as it aids in resource management and scheduling of applications. Resource discovery activity involves searching for the appropriate

resource types that match the user's application requirements. Recent advances in the domain of decentralized resource discovery have been based on extending the existing DHTs with the capability of multi-dimensional data organization and query routing. Majority of the efforts have looked at embedding spatial database indices such as the Space Filling Curves (SFCs) including the Hilbert curves, Z-curves, k-d tree, MX-CIF Quad tree and R*-tree for managing, routing, and indexing of complex Grid resource query objects over DHT networks. Spatial indices are well suited for handling the complexity of Grid resource queries. Although some spatial indices can have issues as regards to routing load-balance in case of a skewed data set, all the spatial indices are more scalable in terms of the number of hops traversed and messages generated while searching and routing Grid resource queries.

Unstructured systems

An unstructured P2P network is formed when the overlay links are established arbitrarily. Such networks can be easily constructed as a new peer that wants to join the network can copy existing links of another node and then form its own links over time. In an unstructured P2P network, if a peer wants to find a desired piece of data in the network, the query has to be flooded through the network to find as many peers as possible that share the data. The main disadvantage with such networks is that the queries may not always be resolved. Popular content is likely to be available at several peers and any peer searching for it is likely to find the same thing. But if a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful. Since there is no correlation between a peer and the content managed by it, there is no guarantee that flooding will find a peer that has the desired data. Flooding also causes a high amount of signaling traffic in the network and hence such networks typically have very poor search efficiency. Many of the popular P2P networks are unstructured.

In *pure* P2P networks: Peers act as equals, merging the roles of clients and server. In such networks, there is no central server managing the network, neither is there a central router. Some examples of pure P2P Application Layer networks designed for peer-to-peer file sharing are Gnutella (pre v0.4) and Freenet.

There also exist *hybrid* P2P systems, which distribute their clients into two groups: client nodes and overlay nodes. Typically, each client is able to act according to the momentary need of the network and can become part of the respective overlay network used to coordinate the P2P structure. This division between normal and 'better' nodes is done in order to address the scaling problems on early pure P2P networks. Examples for such networks are for example Gnutella (after v0.4) or G2.

Another type of hybrid P2P network are networks using on the one hand central server(s) or bootstrapping mechanisms, on the other hand P2P for their data transfers. These networks are in general called 'centralized networks' because of their lack of ability to work without their central server(s). An example for such a network is the eDonkey network (eD2k).

Indexing and resource discovery

Older peer-to-peer networks duplicate resources across each node in the network configured to carry that type of information. This allows local searching, but requires much traffic.

Modern networks use central coordinating servers and directed search requests. Central servers are typically used for listing potential peers (Tor), coordinating their activities (Folding@home), and searching (Napster, eMule). Decentralized searching was first done by flooding search requests out across peers. More efficient directed search strategies, including supernodes and distributed hash tables, are now used.

Many P2P systems use stronger peers (super-peers, super-nodes) as servers and client-peers are connected in a star-like fashion to a single super-peer.

Peer-to-peer-like systems

In modern definitions of peer-to-peer technology, the term implies the general architectural concepts outlined here. However, the basic concept of peer-to-peer computing was envisioned in earlier software systems and networking discussions, reaching back to principles stated in the first Request for Comments, RFC 1.

A distributed messaging system that is often likened as an early peer-to-peer architecture is the USENET network news system that is in principle a client–server model from the user or client perspective, when they read or post news articles. However, news servers communicate with one another as peers to propagate Usenet news articles over the entire group of network servers. The same consideration applies to SMTP email in the sense that the core email relaying network of Mail transfer agents has a peer-to-peer character, while the periphery of e-mail clients and their direct connections is strictly a client–server relationship. Tim Berners-Lee's vision for the World Wide Web, as evidenced by his WorldWideWeb editor/browser, was close to a peer-to-peer design in that it assumed each user of the web would be an active editor and contributor creating and linking content to form an interlinked *web* of links. This contrasts to the broadcasting-like structure of the web as it has developed over the years.

Advantages and weaknesses

In P2P networks, clients provide resources, which may include bandwidth, storage space, and computing power. As nodes arrive and demand on the system increases, the total capacity of the system also increases. In contrast, in a typical client–server architecture, clients share only their demands with the system, but not their resources. In this case, as more clients join the system, less resource are available to serve each client.

The distributed nature of P2P networks also increases robustness, by—in pure P2P systems—enabling peers to find the data without relying on a centralized index server. In the latter case, there is no single point of failure in the system.

As with most network systems, unsecure and unsigned codes may allow remote access to files on a victim's computer or even compromise the entire network. In the past this has happened for example to the FastTrack network when anti P2P companies managed to introduce faked chunks into downloads and downloaded files (mostly MP3 files) were unusable afterwards or even contained malicious code. Consequently, the P2P networks of today have seen an enormous increase of their security and file verification mechanisms. Modern hashing, chunk verification and different encryption methods have made most networks resistant to almost any type of attack, even when major parts of the respective network have been replaced by faked or nonfunctional hosts.

Internet service providers (ISPs) have been known to throttle P2P file-sharing traffic due to the high-bandwidth usage. Compared to Web browsing, e-mail or many other uses of the internet, where data is only transferred in short intervals and relative small quantities, P2P file-sharing often consists of relatively heavy bandwidth usage due to ongoing file transfers and swarm/network coordination packets. As a reaction to this bandwidth throttling several P2P applications started implementing protocol obfuscation, such as the BitTorrent protocol encryption. Techniques for achieving "protocol obfuscation" involves removing otherwise easily identifiable properties of protocols, such as deterministic byte sequences and packet sizes, by making the data look as if it were random.

A possible solution to this is called P2P caching, where a ISP stores the part of files most accessed by P2P clients in order to save access to the Internet.

Social and economic impact

The concept of P2P is increasingly evolving to an expanded usage as the relational dynamic active in distributed networks, *i.e.*, not just computer to computer, but human to human. Yochai Benkler has coined the term commons-based peer production to denote collaborative projects such as free and open source software. Associated with peer production are the concepts of:

- peer governance (referring to the manner in which peer production projects are managed)
- peer property (referring to the new type of licenses which recognize individual authorship but not exclusive property rights, such as the GNU General Public License and the Creative Commons licenses)
- peer distribution (or the manner in which products, particularly peer-produced products, are distributed)

Some researchers have explored the benefits of enabling virtual communities to self-organize and introduce incentives for resource sharing and cooperation, arguing that the social aspect missing from today's peer-to-peer systems should be seen both as a goal and a means for self-organized virtual communities to be built and fostered. Ongoing research efforts for designing effective incentive mechanisms in P2P systems, based on principles from game theory are beginning to take on a more psychological and information-processing direction.

Applications

There are numerous applications of peer-to-peer networks. The most commonly known is for content distribution

Content delivery

- Many file sharing networks, such as Gnutella, G2 and FastTrack popularized peer-to-peer technologies. From 2004, it is the largest contributor of network traffic on the Internet.
- Peer-to-peer content delivery networks (P2P-CDN) (Giraffic, Kontiki, Ignite, RedSwoosh).
- Peer-to-peer content services, e.g. caches for improved performance such as Correli Caches
- Software publication and distribution (Linux, several games); via file sharing networks.
- Streaming media. P2PTV and PDTP. Applications include TVUPlayer, Joost, CoolStreaming, Cybersky-TV, PPLive, LiveStation and Didiom.
- Spotify uses a peer-to-peer network along with streaming servers to stream music to its desktop music player.
- Peercasting for multicasting streams.
- Pennsylvania State University, MIT and Simon Fraser University are carrying on a project called LionShare designed for facilitating file sharing among educational institutions globally.
- Osiris (Serverless Portal System) allows its users to create anonymous and autonomous web portals distributed via P2P network.

Networking

- Domain Name System, for Internet information retrieval.
- cloud computing
- Dalesa a peer-to-peer web cache for LANs (based on IP multicasting).

Science

- In bioinformatics, drug candidate identification. The first such program was begun in 2001 the Centre for Computational Drug Discovery at the University of Oxford in cooperation with the National Foundation for Cancer Research. There are now several similar programs running under the United Devices Cancer Research Project.
- The sciencenet P2P search engine.
- BOINC

Search

- YaCy, a free distributed search engine, built on principles of peer-to-peer networks.

Communications networks

- Skype, one of the most widely used internet phone applications is using P2P technology.
- VoIP (using application layer protocols such as SIP)
- Instant messaging and online chat
- Completely decentralized networks of peers: Usenet (1979) and WWIVnet (1987).

General

- Research like the Chord project, the PAST storage utility, the P-Grid, and the CoopNet content distribution system.
- JXTA, for Peer applications.

Miscellaneous

- The U.S. Department of Defense has started research on P2P networks as part of its modern network warfare strategy. In May, 2003 Dr. Tether. Director of Defense Advanced Research Project Agency testified that U.S. Military is using P2P networks.
- Kato et al.'s studies indicate over 200 companies with approximately $400 million USD are investing in P2P network. Besides File Sharing, companies are also interested in Distributing Computing, Content Distribution.
- Wireless community network, Netsukuku
- An earlier generation of peer-to-peer systems were called "metacomputing" or were classed as "middleware". These include: Legion, Globus
- Bitcoin is a peer-to-peer based digital currency.

Historical perspective

Tim Berners-Lee's vision for the World Wide Web was close to a P2P network in that it assumed each user of the web would be an active editor and contributor, creating and linking content to form an interlinked "web" of links. This contrasts to the current broadcasting-like structure of the web.

Some networks and channels such as Napster, OpenNAP and IRC serving channels use a client–server structure for some tasks (e.g., searching) and a P2P structure for others. Networks such as Gnutella or Freenet use a P2P structure for nearly all tasks, with the exception of finding peers to connect to when first setting up.

P2P architecture embodies one of the key technical concepts of the Internet, described in the first Internet Request for Comments, RFC 1, "Host Software" dated April 7, 1969. More recently, the concept has achieved recognition in the general public in the context of the absence of central indexing servers in architectures used for exchanging multimedia files.

Network neutrality controversy

Peer-to-peer applications present one of the core issues in the network neutrality controversy. In October 2007, Comcast, one of the largest broadband Internet providers in the USA, started blocking P2P applications such as BitTorrent. Their rationale was that P2P is mostly used to share illegal content, and their infrastructure is not designed for continuous, high-bandwidth traffic. Critics point out that P2P networking has legitimate uses, and that this is another way that large providers are trying to control use and content on the Internet, and direct people towards a client-server-based application architecture. The client-server model provides financial barriers-to-entry to small publishers and individuals, and is quite inefficient for sharing large files.

Chapter 2

Peer-to-Peer File Sharing

Peer-to-Peer file sharing is a form of file sharing using peer-to-peer networking.

The widespread adoption and facilitation of Peer to Peer file sharing was helped by several factors. These include increasing Internet bandwidth, the widespread digitization of physical media files, and the capabilities of home PC's increasing to better handle playing and storing digitized audio and video files. This in turn made it relatively easy to transfer either one or more files from one computer to another across the Internet through various file transfers and file-sharing networks.

Peer to Peer file sharing can be done by free file sharing applications.

Legal aspects

Public perception and usage

In 2004, an estimated 70 million people participated in online file sharing. According to a CBS News poll, nearly 70 percent of 18 to 29 year olds thought file sharing was acceptable in some circumstances and 58 percent of all Americans who followed the file sharing issue considered it acceptable in at least some circumstances.

In January 2006, 32 million Americans over the age of 12 had downloaded at least one feature length movie from the Internet, 80 percent of whom had done so exclusively over P2P. Of the population sampled, 40 percent felt that downloading copyrighted movies off the Internet constituted a very serious offense, however 78 percent believed taking a DVD from a store without paying for it constituted a very serious offense.

In July 2008, 20 percent of Europeans used file sharing networks to obtain music, while 10 percent used paid-for digital music services such as iTunes.

In February 2009, a Tiscali UK survey found that 75 percent of the English public polled were aware of what was legal and illegal in relation to file sharing, however there was a divide as to where they felt the legal burden should be placed: 49 percent of people believed P2P companies should be held responsible for illegal file sharing on their

networks, 18 percent viewed individual file sharers as the culprits, while 18 percent either didn't know or chose not to answer.

According to an earlier poll, 75 percent of young voters in Sweden (18-20) supported file sharing when presented with the statement: "I think it is OK to download files from the Net, even if it is illegal." Of the respondents, 38 percent said they "adamantly agreed" while 39 percent said they "partly agreed".

Economic impact

In the book The Wealth of Networks, Yochai Benkler says that peer-to-peer file sharing is economically efficient and that the users pay the full transaction cost and marginal cost of such sharing even if it "throws a monkey wrench into the particular way in which our society has chosen to pay musicians and re-cording executives. This, trades off efficiency for longer-term incentive effects for the recording industry. However, it is efficient within the normal meaning of the term in economics in a way that it would not have been had Jack and Jane used subsidized computers or net- work connections".

According to Benkler:

"What is truly unique about peer-to-peer networks as a signal of what is to come is the fact that with ridiculously low financial investment, a few teenagers and twenty-something-year-olds were able to write software and protocols that allowed tens of millions of computer users around the world to cooperate in producing the most efficient and robust file storage and retrieval system in the world. No major investment was necessary in creating a server farm to store and make available the vast quantities of data repre-sented by the media files. The users' computers are themselves the "server farm." No massive investment in dedicated distribution channels made of high-quality fiber optics was necessary. The standard Internet connections of users, with some very intelligent file transfer protocols, sufficed. Archi-tecture oriented toward enabling users to cooperate with each other in stor-age, search, retrieval, and delivery of files was all that was necessary to build a content distribution network that dwarfed anything that existed before."

Economic impact on the music industry

The economic effect of copyright infringement through peer-to-peer filesharing on music revenue has been controversial and difficult to determine. Music sales dropped globally from approximately $38 billion in 1999 to $32 billion in 2003, and an increasing number of studies found that file sharing had a negative impact on record sales. It has proven difficult to untangle the cause and effect relationships among a number of different trends, including an increase in legal online purchases of music; illegal file-sharing; drops in the prices of CDs; and the extinction of many independent music stores with a concommitant shift to sales by big-box retailers. According to David Glenn, writing in *The Chronicle of Higher Education*, "A majority of economic studies have concluded that

file sharing hurts sales", though not always to the precise degree "the record industry would like the public to believe."

A study by Felix Oberholzer-Gee and Koleman Strumpf in 2004, analyzing logs of downloads on file sharing networks, found that file sharing had no negative effect on CD sales, and would possibly slightly improve the sales of top albums. This work was challenged by Professor Stan Liebowitz, who accused Oberholzer-Gee and Strumpf of making multiple assumptions about the music industry "that are just not correct." Professor Liebowitz, whose work is funded by the record industry, has not published any of these claims in a peer-reviewed journal.

The MPAA reported that American studios lost $2.3 billion to Internet piracy in 2005, representing approximately one third of the total cost of film piracy in the United States. The MPAA's estimate was doubted by commentators since it was based on the assumption that one download was equivalent to one lost sale, and downloaders might not purchase the movie if illegal downloading was not an option. Due to the private nature of the study, the figures could not be publicly checked for methodology or validity, and on January 22, 2008, as the MPAA was lobbying for a bill which would compel universities to crack down on piracy, it was admitted that MPAA figures on piracy in colleges had been inflated by up to 300%.

A 2010 study, commissioned by the International Chamber of Commerce and conducted by independent Paris-based economics firm TERA, estimated that unlawful downloading of music, film and software cost Europe's creative industries several billion in revenue each year. Furthermore, the TERA study entitled “Building a Digital Economy: The Importance of Saving Jobs in the EU's Creative Industries” predicted losses due to piracy reaching as much as 1.2 million jobs and €240 billion in retail revenue by 2015 if the trend continued. Researchers applied a substitution rate of ten percent to the volume of copyright infringements per year. This rate corresponded to the number of units potentially traded if unlawful file sharing were eliminated and did not occur. Piracy rates of one-quarter or more for popular software and operating systems have been common, even in countries and regions with strong intellectual property enforcement, such as the US or the EU.

The independent label Lion Music has stated that copyright infringement through peer-topeer filesharing has a negative economic impact on them and their grass roots artists cannot be denied as it is difficult to compete with unauthorized free distribution of their copyrighted music.

Risks

Researchers have examined potential security risks including the release of personal information, bundled spyware, and viruses downloaded from the network. Some proprietary file sharing clients have been known to bundle malware, though open source programs typically have not. Some open source file sharing packages have even provided integrated anti-virus scanning. A drastic increase in inadvertent P2P file sharing of

personal and sensitive information became evident in 2009 at the beginning of President Obama's administration when the blueprints to the helicopter Marine One were made available to the public through a breach in security via a P2P file sharing site. Access to this information has the potential of being detrimental to US security.

Furthermore, shortly before this security breach, the *Today* show had reported that more than 150,000 tax returns, 25,800 student loan applications and 626,000 credit reports had been inadvertently made available through file sharing.

Since approximately 2004 identity theft has become more prevalent, and in July 2008 there was another inadvertent revealing of vast amounts of personal information through careless use of a P2P site. The "names, dates of birth, and Social Security numbers of about 2,000 of (an investment) firm's clients" were exposed, "including [those of] Supreme Court Justice Stephen Breyer."

Researchers have discovered thousands of documents containing sensitive patient information on popular peer-to-peer (P2P) networks, including insurance details, personally identifying information, physician names and diagnosis codes on more than 28,000 individuals. Many of the documents contained sensitive patient communications, treatment data, medical diagnoses and psychiatric evaluations.

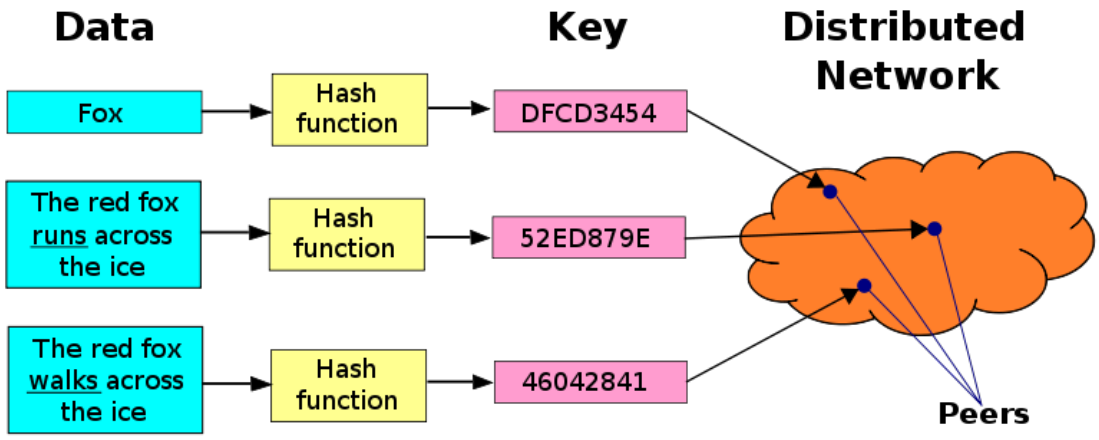
The United States government has attempted to make users more aware of the potential risks involved with P2P file sharing programs through legislation such as H.R. 1319, the Informed P2P User Act. According to this act, it would be mandatory for individuals to be aware of the risks associated with peer-to-peer file sharing before purchasing software with informed consent of the user required prior to use of such programs. In addition, the act would allow users to block and remove P2P file sharing software from their computers at any time, with the Federal Trade Commission enforcing regulations.

Chapter 3

Distributed Hash Table

A **distributed hash table** (**DHT**) is a class of a decentralized distributed system that provides a lookup service similar to a hash table; (*key*, *value*) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.

DHTs form an infrastructure that can be used to build more complex services, such as anycast, cooperative Web caching, distributed file systems, domain name services, instant messaging, multicast, and also peer-to-peer file sharing and content distribution systems. Notable distributed networks that use DHTs include BitTorrent's distributed tracker, the Coral Content Distribution Network, the Kad network, the Storm botnet, and YaCy.



Distributed hash tables

History

DHT research was originally motivated, in part, by peer-to-peer systems such as Freenet, gnutella, and Napster, which took advantage of resources distributed across the Internet to provide a single useful application. In particular, they took advantage of increased bandwidth and hard disk capacity to provide a file-sharing service.

These systems differed in how they *found* the data their peers contained:

- Napster, the first large-scale P2P content delivery system to exist, had a central index server: each node, upon joining, would send a list of locally held files to the server, which would perform searches and refer the querier to the nodes that held the results. This central component left the system vulnerable to attacks and lawsuits.
- Gnutella and similar networks moved to a flooding query model—in essence, each search would result in a message being broadcast to every other machine in the network. While avoiding a single point of failure, this method was significantly less efficient than Napster.
- Finally, Freenet is fully distributed, but employs a heuristic key-based routing in which each file is associated with a key, and files with similar keys tend to cluster on a similar set of nodes. Queries are likely to be routed through the network to such a cluster without needing to visit many peers. However, Freenet does not guarantee that data will be found.

Distributed hash tables use a more structured key-based routing in order to attain both the decentralization of Freenet and gnutella, and the efficiency and guaranteed results of Napster. One drawback is that, like Freenet, DHTs only directly support exact-match search, rather than keyword search, although that functionality can be layered on top of a DHT.

In 2001, four systems—CAN, Chord, Pastry, and Tapestry—ignited DHTs as a popular research topic, and this area of research remains active. Outside academia, DHT technology has been adopted as a component of BitTorrent and in the Coral Content Distribution Network.

Properties

DHTs characteristically emphasize the following properties:

- Decentralization: the nodes collectively form the system without any central coordination.
- Fault tolerance: the system should be reliable (in some sense) even with nodes continuously joining, leaving, and failing.
- Scalability: the system should function efficiently even with thousands or millions of nodes.

A key technique used to achieve these goals is that any one node needs to coordinate with only a few other nodes in the system – most commonly, O(log n) of the n participants (see below) – so that only a limited amount of work needs to be done for each change in membership.

Some DHT designs seek to be secure against malicious participants and to allow participants to remain anonymous, though this is less common than in many other peer-to-peer (especially file sharing) systems.

Finally, DHTs must deal with more traditional distributed systems issues such as load balancing, data integrity, and performance (in particular, ensuring that operations such as routing and data storage or retrieval complete quickly).

Structure

The structure of a DHT can be decomposed into several main components. The foundation is an abstract **keyspace**, such as the set of 160-bit strings. A **keyspace partitioning** scheme splits ownership of this keyspace among the participating nodes. An **overlay network** then connects the nodes, allowing them to find the owner of any given key in the keyspace.

Once these components are in place, a typical use of the DHT for storage and retrieval might proceed as follows. Suppose the keyspace is the set of 160-bit strings. To store a file with given *filename* and *data* in the DHT, the SHA-1 hash of *filename* is generated, producing a 160-bit key k, and a message *put*(k,*data*) is sent to any node participating in the DHT. The message is forwarded from node to node through the overlay network until it reaches the single node responsible for key k as specified by the keyspace partitioning. That node then stores the key and the data. Any other client can then retrieve the contents of the file by again hashing *filename* to produce k and asking any DHT node to find the data associated with k with a message *get*(k). The message will again be routed through the overlay to the node responsible for k, which will reply with the stored *data*.

The keyspace partitioning and overlay network components are described below with the goal of capturing the principal ideas common to most DHTs; many designs differ in the details.

Keyspace partitioning

Most DHTs use some variant of consistent hashing to map keys to nodes. This technique employs a function $\delta(k_1,k_2)$ that defines an abstract notion of the *distance* between the keys k_1 and k_2, which is unrelated to geographical distance or network latency. Each node is assigned a single key called its *identifier* (ID). A node with ID i_x owns all the keys k_m for which i_x is the closest ID, measured according to $\delta(k_m,i_x)$.

Example. The Chord DHT treats keys as points on a circle, and $\delta(k_1,k_2)$ is the distance traveling clockwise around the circle from k_1 to k_2. Thus, the circular keyspace is split

into contiguous segments whose endpoints are the node identifiers. If i_1 and i_2 are two adjacent IDs, then the node with ID i_2 owns all the keys that fall between i_1 and i_2.

Consistent hashing has the essential property that removal or addition of one node changes only the set of keys owned by the nodes with adjacent IDs, and leaves all other nodes unaffected. Contrast this with a traditional hash table in which addition or removal of one bucket causes nearly the entire keyspace to be remapped. Since any change in ownership typically corresponds to bandwidth-intensive movement of objects stored in the DHT from one node to another, minimizing such reorganization is required to efficiently support high rates of churn (node arrival and failure).

Locality-preserving hashing ensures that similar keys are assigned to similar objects. This can enable a more efficient execution of range queries. Self-Chord decouples object keys from peer IDs and sort keys along the ring with a statistical approach based on the swarm intelligence paradigm. Sorting ensures that similar keys are stored by neighbour nodes and that discovery procedures, including range queries, can be performed in logarithmic time.

Overlay network

Each node maintains a set of links to other nodes (its *neighbors* or routing table). Together, these links form the overlay network. A node picks its neighbors according to a certain structure, called the network's topology.

All DHT topologies share some variant of the most essential property: for any key *k*, each node either has a node ID that owns *k* or has a link to a node whose node ID is *closer* to *k*, in terms of the keyspace distance defined above. It is then easy to route a message to the owner of any key *k* using the following greedy algorithm (that is not necessarily globally optimal): at each step, forward the message to the neighbor whose ID is closest to *k*. When there is no such neighbor, then we must have arrived at the closest node, which is the owner of *k* as defined above. This style of routing is sometimes called key-based routing.

Beyond basic routing correctness, two important constraints on the topology are to guarantee that the maximum number of hops in any route (route length) is low, so that requests complete quickly; and that the maximum number of neighbors of any node (maximum node degree) is low, so that maintenance overhead is not excessive. Of course, having shorter routes requires higher maximum degree. Some common choices for maximum degree and route length are as follows, where *n* is the number of nodes in the DHT, using Big O notation:

Degree	Route length	Notice
$O(1)$	$O(n)$	
$O(\log n)$	$O(\log n / \log(\log n))$	
$O(\log n)$	$O(\log n)$	most common, but not optimal (degree/route length)
$O(\sqrt{n})$	$O(1)$	

The most common third choice is not optimal in terms of degree/route length tradeoff, as such topologies typically allow more flexibility in choice of neighbors. Many DHTs use that flexibility to pick neighbors that are close in terms of latency in the physical underlying network.

Maximum route length is closely related to diameter: the maximum number of hops in any shortest path between nodes. Clearly, the network's worst case route length is at least as large as its diameter, so DHTs are limited by the degree/diameter tradeoff that is fundamental in graph theory. Route length can be greater than diameter, since the greedy routing algorithm may not find shortest paths.

Algorithms for overlay networks

Aside from routing, there exist many algorithms that exploit the structure of the overlay network for sending a message to all nodes, or a subset of nodes, in a DHT. These algorithms are used by applications to do overlay multicast, range queries, or to collect statistics. Two systems that are based on this approach are Structella, which implements flooding and random walks on a Pastry overlay, and DQ-DHT, which implements a dynamic querying search algorithm over a Chord network.

DHT implementations

Most notable differences encountered in practical instances of DHT implementations include at least the following:

- The address space is a parameter of DHT. Several real world DHTs use 128-bit or 160-bit key space
- Some real-world DHTs use hash functions other than SHA-1.
- In the real world the key *k* could be a hash of a file's *content* rather than a hash of a file's *name* to provide content-addressable storage, so that renaming of the file does not prevent users from finding it.
- Some DHTs may also publish objects of different types. For example, key *k* could be the node *ID* and associated data could describe how to contact this node. This allows publication-of-presence information and often used in IM applications, etc. In the simplest case, *ID* is just a random number that is directly used as key *k* (so in a 160-bit DHT *ID* will be a 160-bit number, usually randomly chosen). In some DHTs, publishing of nodes IDs is also used to optimize DHT operations.

- Redundancy can be added to improve reliability. The (*k*,*data*) key pair can be stored in more than one node corresponding to the key. Usually, rather than selecting just one node, real world DHT algorithms select *i* suitable nodes, with *i* being an implementation-specific parameter of the DHT. In some DHT designs, nodes agree to handle a certain keyspace range, the size of which may be chosen dynamically, rather than hard-coded.
- Some advanced DHTs like Kademlia perform iterative lookups through the DHT first in order to select a set of suitable nodes and send *put*(*k*,*data*) messages only to those nodes, thus drastically reducing useless traffic, since published messages are only sent to nodes that seem suitable for storing the key *k*; and iterative lookups cover just a small set of nodes rather than the entire DHT, reducing useless forwarding. In such DHTs, forwarding of *put*(*k*,*data*) messages may only occur as part of a self-healing algorithm: if a target node receives a *put*(*k*,*data*) message, but believes that *k* is out of its handled range and a closer node (in terms of DHT keyspace) is known, the message is forwarded to that node. Otherwise, data are indexed locally. This leads to a somewhat self-balancing DHT behavior. Of course, such an algorithm requires nodes to publish their presence data in the DHT so the iterative lookups can be performed.

Examples

DHT protocols and implementations

- Apache Cassandra
- BitTorrent DHT - based on Kademlia as provided by Khashmir.
- CAN (Content Addressable Network)
- Chord
- Kademlia
- Pastry
- P-Grid
- Tapestry

Applications employing DHTs

- Codeen: Web caching
- Coral Content Distribution Network
- Dijjer: Freenet-like distribution network
- FAROO: Peer-to-peer Web search engine
- Freenet: A censorship-resistant anonymous network
- GNUnet: Freenet-like distribution network including a DHT implementation
- JXTA: Opensource P2P platform
- maidsafe: C++ implementation of Kademlia, with NAT traversal and crypto libraries. On its home page listed as "Available as a technology licence and a software solution written in cross platform C++."
- WebSphere eXtreme Scale: proprietary DHT implementation by IBM, used for object caching

- YaCy: distributed search engine

Chapter 4

Distributed Data Store and P2PTV

Distributed data store

A **distributed data store** is a blurred concept and means either a distributed database where users store their information on a *number of nodes*, or a network in which a user stores their information on a *number of peer network nodes*.

Distributed databases

Distributed data store are non-relational databases that make a quick access to data over a large number of nodes possible. Examples for this kind of data stores are Google's BigTable, which is much more than a distributed file system or a peer-to-peer network, or Amazon's Dynamo.

As the ability of arbitrary querying is not as important as the availability, designers of distributed data stores have increased the latter at an expense of consistency. But the high-speed read/write access results in reduced consistency, as it is not possible to have both consistency, availability, and partition tolerance of the network, as it has been proven by the CAP theorem.

Peer network node data stores

In peer network data stores, the user can usually reciprocate and allow other users to use their computer as a storage node as well. Information may or may not be accessible to other users depending on the design of the network.

Most of the peer-to-peer networks do not have distributed data stores in that the user's data is only available when their node is on the network. However, this distinction is somewhat blurred in a system such as BitTorrent, where it is possible for the originating node to go offline but the content to continue to be served. Still, this is only the case for individual files requested by the redistributors, as contrasted with a network such as Freenet where all computers are made available to serve all files.

Distributed data stores typically use an error detection and correction technique. Some distributed data stores (such as Parchive over NNTP) use forward error correction techniques to recover the original file when parts of that file are damaged or unavailable. Others try again to download that file from a different mirror.

Examples

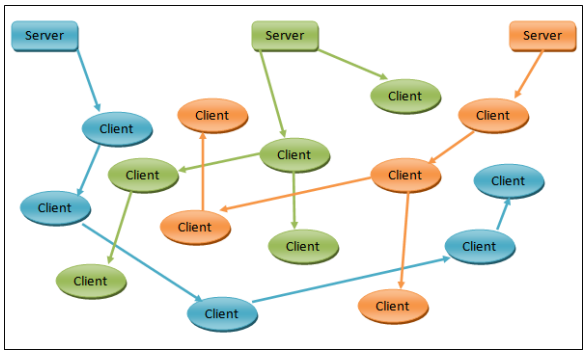
Distributed non-relational databases

- Apache Cassandra, the data store of Facebook
- BigTable, the data store of Google
- Dynamo of Amazon
- Voldemort

Peer network node data stores

- BitTorrent
- Chord project
- GNUnet
- Freenet
- NNTP (the distributed data storage protocol used for Usenet news)
- Mnet
- Storage@home
- Wuala

P2PTV



P2PTV overlay network serving three streams.

The term **P2PTV** refers to peer-to-peer (P2P) software applications designed to redistribute video streams in real time on a P2P network; the distributed video streams are typically TV channels from all over the world but may also come from other sources. The

draw to these applications is significant because they have the potential to make any TV channel globally available by any individual feeding the stream into the network where each peer joining to watch the video is a relay to other peer viewers, allowing a scalable distribution among a large audience with no incremental cost for the source.

Technology and use

In a P2PTV system, each user, while downloading a video stream, is simultaneously also uploading that stream to other users, thus contributing to the overall available bandwidth. The arriving streams are typically a few minutes time-delayed compared to the original sources. The video quality of the channels usually depends on how many users are watching; the video quality is better if there are more users. The architecture of many P2PTV networks can be thought of as real-time versions of BitTorrent: if a user wishes to view a certain channel, the P2PTV software contacts a "tracker server" for that channel in order to obtain addresses of peers who distribute that channel; it then contacts these peers to receive the feed. The tracker records the user's address, so that it can be given to other users who wish to view the same channel. In effect, this creates an overlay network on top of the regular internet for the distribution of real-time video content.

The need for a tracker can also be eliminated by the use of distributed hash table technology.

Some applications allow users to broadcast their own streams, whether self-produced, obtained from a video file, or through a TV tuner card or video capture card.

Many of the commercial P2PTV applications were developed in China (TVUPlayer, PPLive, QQLive, PPStream). The majority of available applications broadcast mainly Asian TV stations, with the exception of TVUPlayer, which carries a number of North American stations including CBS, Spike TV, and Fox News. Some applications distribute TV channels without a legal license to do so; this utilization of P2P technology is particularly popular to view channels that are either not available locally, or only available by paid subscription, as is the case for some sports channels. By January 2009, there were about 14,000 P2P channels on PPStream.

Other commercial P2PTV applications outside China are Abroadcasting (USA), Zattoo (Switzerland/USA), Octoshape (Denmark), LiveStation (UK).

Issues for broadcasters

- Broadcasting via a P2PTV system is usually much cheaper than the alternatives and can be done by private individuals.
- No quality of service (QoS). Compared to unicasting (the standard server-client architecture used in streaming media) no one can guarantee a reliable stream, since every user is a rebroadcaster. Each viewer is a part of a chain of viewers which all can have a negative influence on the reliability of the stream (by having

a slow PC, a filled downlink or uplink or an unreliable consumer grade DSL or cable connection).
- Less control. If a broadcaster prefers to limit access to their content based on regions, and would like good data on viewer behaviour, such as volume, trends and viewing time, then a traditional broadcasting solution offers more control.
- Professional broadcasters and distributors have used a hybrid solution for many years. Distribution servers are not centrally installed, but are rolled out in a smart, decentralized way. A central management facility manages content distribution over multiple peer servers (also known as edge servers, or caches), strategically located near user swarms (generally popular access ISP networks), manages load balancing, redirection of users, view reporting and QoS. An example is Akamai.

Notable applications

Branded webtv service for end-users

- Babelgum.com (non-live, used peer-to-peer technology until March 2009)
- BBC iPlayer (live and non-live, used peer-to-peer technology until December 2008)
- Joost.com (non-live, live trials)
- LiveStation.com (Windows, Linux, Mac) – based in United Kingdom
- Miro (non-live)
- ReelTime.com (non-live)
- Zattoo.com (Windows, Mac)
- Hypp.TV (live and non-live) – based in Malaysia
- pldtwatchpad.com (live and non-live) – based in the Philippines

Commercial solutions for broadcasters

- Alluvium – based in Texas, USA
- Octoshape (Windows, Linux, Mac)
- Pando
- Rawflow
- SyQic (PC and Over-The-Top STB)
- RayV – Windows P2PTV Software and Network

Unclassified (yet)

- Afreeca – based in South Korea
- CDNetworks (CDN service)
- CoolStreaming (discontinued service)
- Cybersky-TV – software
- PeerCast (Windows, Linux, Mac)
- PPLive – based in China mainland, Chinese only program.
- PPStream – based in China mainland

- Pulse – (Windows, Linux) LGPL P2PTV engine with announcement portal and unrestricted access
- SopCast – Windows and Linux P2PTV Software and Network
- Tribler – linked to P2P-Next, relies on BitTorrent protocol
- TVUnetworks – Windows and MacOSX P2PTV Software and Network
- TvAnts – software developed by Zhejiang University.

Chapter 5

Streaming Media

Streaming media is multimedia that is constantly received by and presented to an end-user while being delivered by a streaming provider. The name refers to the delivery method of the medium rather than to the medium itself. The distinction is usually applied to media that are distributed over telecommunications networks, as most other delivery systems are either inherently streaming (e.g., radio, television) or inherently non-streaming (e.g., books, video cassettes, audio CDs). The verb 'to stream' is also derived from this term, meaning to deliver media in this manner. Internet television is a commonly streamed medium.

Live streaming, more specifically, means taking the media and broadcasting it live over the Internet. The process involves a camera for the media, an encoder to digitize the content, a media publisher where the streams are made available to potential end-users and a content delivery network to distribute and deliver the content. The media can then be viewed by end-users live.

Security remains one of the main challenges with this new methodology. Digital rights management (DRM) systems are an example of a solution to keep this content secure.

History

Attempts to display media on computers date back to the earliest days of computing in the mid-20th century. However, little progress was made for several decades, primarily due to the high cost and limited capabilities of computer hardware.

From the late 1980s through the 1990s, consumer-grade personal computers became powerful enough to display various media. The primary technical issues related to streaming were:

- having enough CPU power and bus bandwidth to support the required data rates
- creating low-latency interrupt paths in the operating system (OS) to prevent buffer underrun.

However, computer networks were still limited, and media was usually delivered over non-streaming channels, such as by downloading a digital file from a remote server and

then saving it to a local drive on the end user's computer or storing it as a digital file and playing it back from CD-ROMs.

During the late 1990s and early 2000s, Internet users saw:

- greater network bandwidth, especially in the last mile
- increased access to networks, especially the Internet
- use of standard protocols and formats, such as TCP/IP, HTTP, and HTML
- commercialization of the Internet.

Real Networks pioneered the streaming media markets and broadcast the first audio event over the Internet - a baseball game between the Yankees and Seattle Mariners - in 1995. They went on to launch the first streaming video technology in 1997. According to some accounts, by 2000, more than 85% of streaming content on the Internet was in the Real format.

Despite this success, problems arose because Real's primary business model depended upon the sale of servers, and Microsoft and Apple were giving those products away. As servers from Microsoft and Apple became more capable, Real's sales inevitably eroded on the business side.

Also damaging to Real, consumers started balking at the intrusiveness of the free Real Player, which installed multiple extraneous programs and made itself the default player for all multimedia content while constantly nagging the user to upgrade to the latest version. Despite its initial popularity, consumers uninstalled the software and PC manufacturers stopped pre-installing on new computers.

These advances in computer networking combined with powerful home computers and modern operating systems made streaming media practical and affordable for ordinary consumers. Stand-alone Internet radio devices emerged to offer listeners a no-computer option for listening to audio streams.

In general, multimedia content has a large volume, so media storage and transmission costs are still significant; to offset this somewhat, media are generally compressed for both storage and streaming.

Increasing consumer demand for streaming of high definition (HD) content to different devices in the home has led the industry to develop a number of technologies, such as Wireless HD or ITU-T G.hn, which are optimized for streaming HD content without forcing the user to install new networking cables.

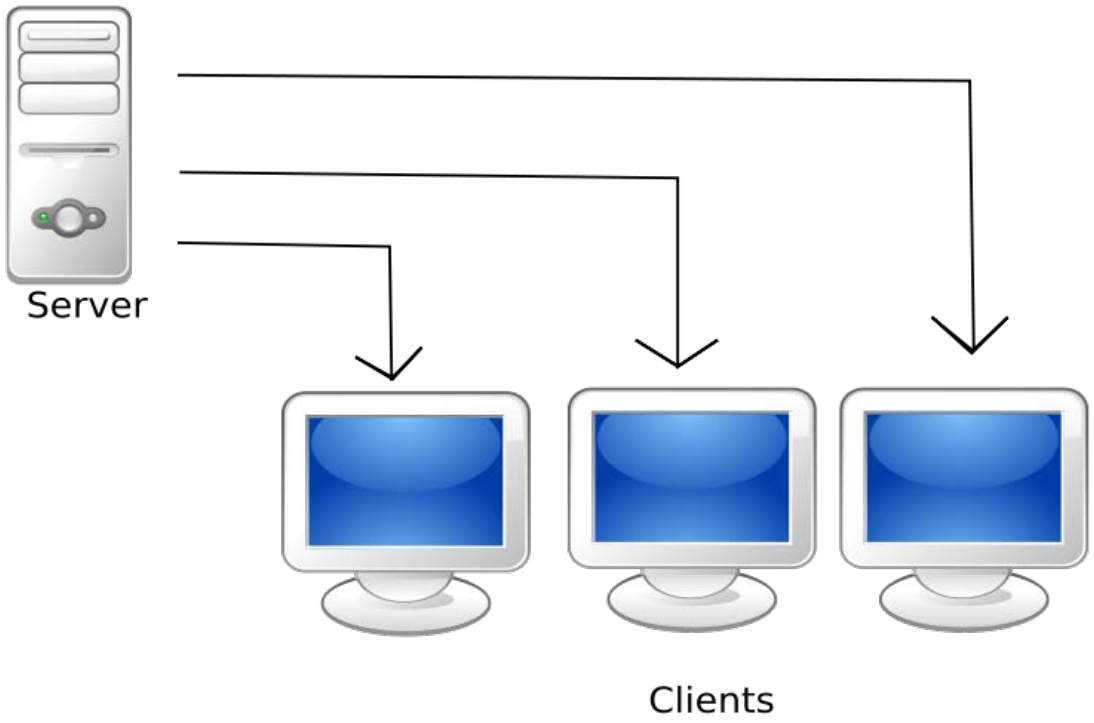
Increasing consumer demand for live streaming has prompted YouTube to implement their new Live Streaming service to users.

A media stream can be streamed either live or on demand. Live streams are generally provided by a means called true streaming. True streaming sends the information straight

to the computer or device without saving the file to a hard disk. On Demand streaming is provided by a means called *progressive streaming* or *progressive download.* Progressive streaming saves the file to a hard disk and then is played from that location. On Demand streams are often saved to hard disks and servers for extended amounts of time; while the live streams are only available at one time only (e.g. during the Football game).

Streaming bandwidth and storage

A broadband speed of 2.5 Mbps or more is recommended for streaming movies, for example to an Apple TV, Google TV or a Sony TV Blu-ray Disc Player, 10 Mbps for High Definition content.



Unicast connections require multiple connections from the same streaming server even when it streams the same content

Streaming media storage size is calculated from the streaming bandwidth and length of the media using the following formula (for a single user and file):

> storage size (in megabytes) = length (in seconds) × bit rate (in bit/s) / (8 × 1024 × 1024)

Real world example:

One hour of video encoded at 300 kbit/s (this is a typical broadband video as of 2005 and it is usually encoded in a 320 × 240 pixels window size) will be:

(3,600 s × 300,000 bit/s) / (8×1024×1024) requires around 128 MiB of storage.

If the file is stored on a server for on-demand streaming and this stream is viewed by 1,000 people at the same time using a Unicast protocol, the requirement is:

300 kbit/s × 1,000 = 300,000 kbit/s = 300 Mbit/s of bandwidth

This is equivalent to around 135 GB per hour. Of course, using a multicast protocol the server sends out only a single stream that is common to all users. Hence, such a stream would only use 300 kbit/s of serving bandwidth.

The calculation for Live streaming is similar.

Assumptions: speed at the encoder, is 500 kbit/s.

If the show last for 3 hours, with 3000 viewers then the calculation is:

Number of MB transferred = encoder speed (in bps) × number of seconds × number of viewer / (8*1024*1024)
Number of MB transferred = 500.000 (bps) × 3 × 3600 (= 3 hours) × 3000 (nbr of viewers) / (8*1024*1024) = 1931190 MB

Codec, bitstream, transport, control

The audio stream is compressed using an audio codec such as MP3, Vorbis or AAC.

The video stream is compressed using a video codec such as H.264 or V8.

Encoded audio and video streams are assembled in a container bitstream such as FLV, WebM, ASF or ISMA.

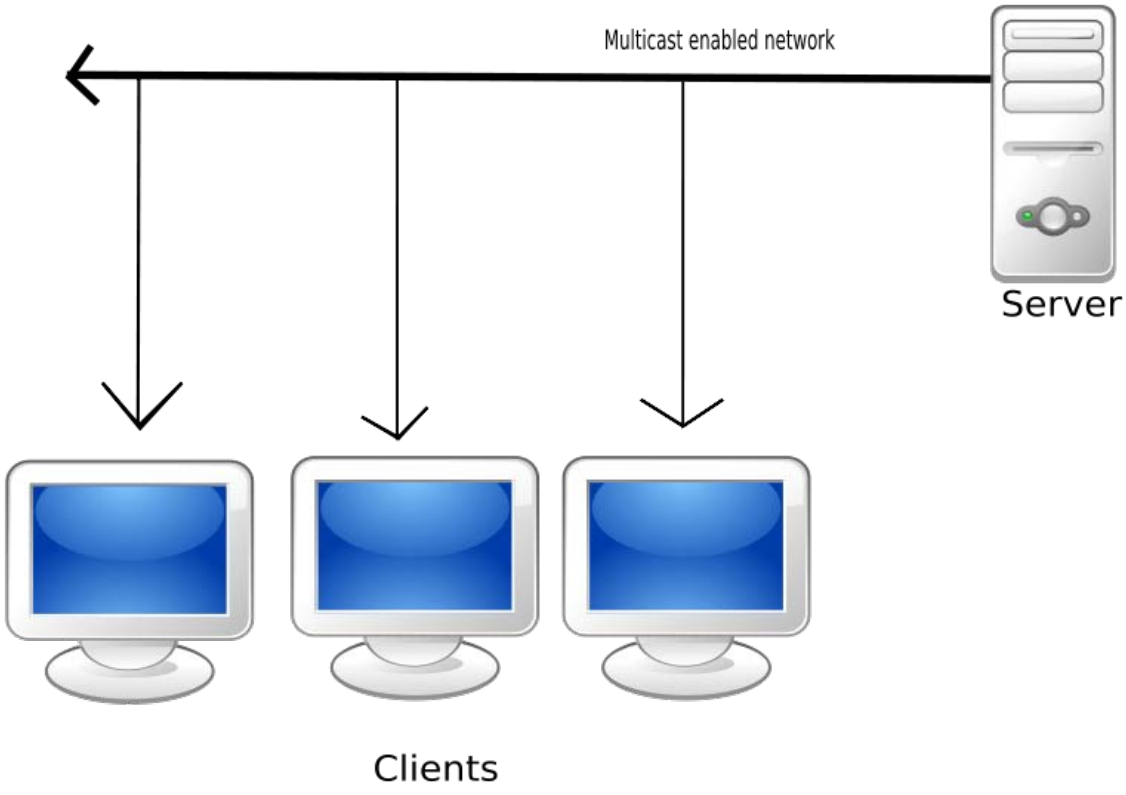
The bitstream is delivered from a streaming server to a streaming client using a transport protocol, such as MMS or RTP.

The streaming client may interact with the streaming server using a control protocol, such as MMS or RTSP.

Protocol issues

Designing a network protocol to support streaming media raises many issues, such as:

- Datagram protocols, such as the User Datagram Protocol (UDP), send the media stream as a series of small packets. This is simple and efficient; however, there is no mechanism within the protocol to guarantee delivery. It is up to the receiving application to detect loss or corruption and recover data using error correction techniques. If data is lost, the stream may suffer a dropout.
- The Real-time Streaming Protocol (RTSP), Real-time Transport Protocol (RTP) and the Real-time Transport Control Protocol (RTCP) were specifically designed to stream media over networks. RTSP runs over a variety of transport protocols, while the latter two are built on top of UDP.
- Another approach that seems to incorporate both the advantages of using a standard web protocol and the ability to be used for streaming even live content is the HTTP adaptive bitrate streaming. HTTP adaptive bitrate streaming is based on HTTP progressive download, but contrary to the previous approach, here the files are very small, so that they can be compared to the streaming of packets, much like the case of using RTSP and RTP.
- Reliable protocols, such as the Transmission Control Protocol (TCP), guarantee correct delivery of each bit in the media stream. However, they accomplish this with a system of timeouts and retries, which makes them more complex to implement. It also means that when there is data loss on the network, the media stream stalls while the protocol handlers detect the loss and retransmit the missing data. Clients can minimize this effect by buffering data for display. While delay due to buffering is acceptable in video on demand scenarios, users of interactive applications such as video conferencing will experience a loss of fidelity if the delay that buffering contributes to exceeds 200 ms.
- Unicast protocols send a separate copy of the media stream from the server to each recipient. Unicast is the norm for most Internet connections, but does not scale well when many users want to view the same program concurrently.



Multicasting broadcasts the same copy of the multimedia over the entire network to a group of clients

- Multicast protocols were developed to reduce the data replication (and consequent server/network loads) that occurs when many recipients receive unicast content streams independently. These protocols send a single stream from the source to a group of recipients. Depending on the network infrastructure and type, multicast transmission may or may not be feasible. One potential disadvantage of multicasting is the loss of video on demand functionality. Continuous streaming of radio or television material usually precludes the recipient's ability to control playback. However, this problem can be mitigated by elements such as caching servers, digital set-top boxes, and buffered media players.
- IP Multicast provides a means to send a single media stream to a group of recipients on a computer network. A multicast protocol, usually Internet Group Management Protocol, is used to manage delivery of multicast streams to the groups of recipients on a LAN. One of the challenges in deploying IP multicast is that routers and firewalls between LANs must allow the passage of packets destined to multicast groups. If the organization that is serving the content has control over the network between server and recipients (i.e., educational, government, and corporate intranets), then routing protocols such as Protocol

Independent Multicast can be used to deliver stream content to multiple Local Area Network segments.

- Peer-to-peer (P2P) protocols arrange for prerecorded streams to be sent between computers. This prevents the server and its network connections from becoming a bottleneck. However, it raises technical, performance, quality, and business issues.

Chapter 6

FAROO and Content Addressable Network

FAROO

FAROO

File:Faroo screenshot small.gif
FAROO screenshot

Developer(s)	FAROO Limited
Written in	C#
Operating system	Microsoft Windows
Platform	.NET Framework and Mono
Type	Web search engine
License	Freeware

FAROO is a universal web search engine based on peer-to-peer technology. It uses a distributed crawler that stores search data on users' computers instead of a central server. Whenever a user visits a website, it is automatically indexed and distributed to the network. Ranking is done by comparing usage statistics of users, such as web pages visited, amount of time spent on each page, and whether the pages were bookmarked or printed.

Benefits

- FAROO takes user behavior into account when calculating a website's rank, resulting in more relevant search results than traditional search engines can provide.
- Because pages are automatically indexed upon being visited, updates to the index are nearly instant.
- No central servers are required, drastically reducing infrastructure costs and allowing the service to scale infinitely.
- FAROO plans to share up to 50 % of its advertising revenue with its users.

Content addressable network

The **Content Addressable Network** (**CAN**) is a distributed, decentralized P2P infrastructure that provides hash table functionality on an Internet-like scale. CAN was one of the original four distributed hash table proposals, introduced concurrently with Chord, Pastry, and Tapestry.

Overview

Like other distributed hash tables, CAN is designed to be scalable, fault tolerant, and self-organizing. The architectural design is a virtual multi-dimensional Cartesian coordinate space, a type of overlay network, on a multi-torus. This d-dimensional coordinate space is a virtual logical address, completely independent of the physical location and physical connectivity of the nodes. Points within the space are identified with coordinates. The entire coordinate space is dynamically partitioned among all the nodes in the system such that every node possesses at least one distinct zone within the overall space.

Routing

A CAN node maintains a routing table that holds the IP address and virtual coordinate zone of each of its neighbors. A node routes a message towards a destination point in the coordinate space. The node first determines which neighboring zone is closest to the destination point, and then looks up that zone's node's IP address via the routing table.

Node joining

To join a CAN, a joining node must:

1. Find a node already in the overlay network.
2. Identify a zone that can be split
3. Update the routing tables of nodes neighboring the newly split zone.

To find a node already in the overlay network, bootstrapping nodes may be used to inform the joining node of IP addresses of nodes currently in the overlay network.

After the joining node receives an IP address of a node already in the CAN, it can attempt to identify a zone for itself. The joining node randomly picks a point in the coordinate space and sends a join request, directed to the random point, to one of the received IP addresses. The nodes already in the overlay network route the join request to the correct device via their zone-to-IP routing tables. Once the node managing the destination point's zone receives the join request, it may honor the join request by splitting its zone in half, allocating itself the first half, and allocating the joining node the second half. If it does not honor the join request, the joining node keeps picking random points in the coordinate space and sending join requests directed to these random points until it successfully joins the network.

After the zone split and allocation is complete, the neighboring nodes are updated with the coordinates of the two new zones and the corresponding IP addresses. Routing tables are updated and updates are propagated across the network.

Node departing

To handle a node departing, the CAN must i) identify a node is departing, ii) have the departing node's zone merged or taken-over by a neighboring node, and iii) update the routing tables across the network.

Detecting a node's departure can be done, for instance, via heartbeat messages that periodically broadcast routing table information between neighbors. After a predetermined period of silence from a neighbor, that neighboring node is determined as failed and is considered a departing node. Alternatively, a node that is willingly departing may broadcast such a notice to its neighbors.

After a departing node is identified, its zone must be either merged or taken-over. First the departed node's zone is analyzed to determine whether a neighboring node's zone can merge with the departed node's zone to form a valid zone. For example, a zone in a 2d coordinate space must be either a square or rectangle and cannot be L-shaped. The validation test may cycle through all neighboring zones to determine if a successful merge can occur. If one of the potential merges is deemed a valid merge, the zones are then merged. If none of the potential merges are deemed valid, then the neighboring node with the smallest zone takes over control of the departing node's zone. After a take-over, the take-over node may periodically attempt to merge its additionally controlled zones with respective neighboring zones.

If the merge is successful, routing tables of neighboring zones' nodes are updated to reflect the merge. The network will see the subsection of the overlay network as one, single zone after a merge and treat all routing processing with this mindset. To effectuate a take-over, the take-over node updates neighboring zones' nodes' routing tables, so that requests to either zone resolve to the take-over node. And, as such, the network still sees the subsection of the overlay network as two separate zones and treats all routing processing with this mindset.

Developers

Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker

Chapter 7

Content Delivery Network

A **content delivery network** or **content distribution network** (**CDN**) is a system of computers containing copies of data, placed at various points in a network so as to maximize bandwidth for access to the data from clients throughout the network. A client accesses a copy of the data near to the client, as opposed to all clients accessing the same central server, so as to avoid bottlenecks near that server.

Content types include web objects, downloadable objects (media files, software, documents), applications, real time media streams, and other components of internet delivery (DNS, routes, and database queries).

CDN benefits

The capacity sum of strategically placed servers can be higher than the network backbone capacity. This can result in an impressive increase in the number of concurrent users. For instance, when there is a 10 Gbit/s network backbone and 200 Gbit/s central server capacity, only 10 Gbit/s can be delivered. But when 10 servers are moved to 10 edge locations, total capacity can be 10×10 Gbit/s.

Strategically placed edge servers decrease the load on interconnects, public peers, private peers and backbones, freeing up capacity and lowering delivery costs. It uses the same principle as above. Instead of loading all traffic on a backbone or peer link, a CDN can offload these by redirecting traffic to edge servers.

CDNs generally deliver content over TCP and UDP connections. TCP throughput over a network is impacted by both latency and packet loss. In order to reduce both of these parameters, CDNs traditionally place servers as close to the edge networks that users are on as possible. Theoretically the closer the content the faster the delivery, although network distance may not be the factor that leads to best performance. End users will likely experience less jitter, fewer network peaks and surges, and improved stream quality - especially in remote areas. The increased reliability allows a CDN operator to deliver HD quality content with high Quality of Service, low costs and low network load.

CDNs can dynamically distribute assets to strategically placed redundant core, fallback and edge servers. CDNs can have automatic server availability sensing with instant user redirection. A CDN can offer 100% availability, even with large power, network or hardware outages.

CDN technologies give more control of asset delivery and network load. They can optimize capacity per customer, provide views of real-time load and statistics, reveal which assets are popular, show active regions and report exact viewing details to the customers. These usage details are an important feature that a CDN provider must provide, since the usage logs are no longer available at the content source server after it has been plugged into the CDN, because the connections of end-users are now served by the CDN edges instead of the content source.

ASP versus on-net

Most CDNs are operated as an application service provider (ASP) on the Internet, although an increasing number of internet network owners, such as AT&T and Level3, have built their own CDN to improve on-net content delivery and to generate revenues from content customers. Some develop internal CDN software; others use commercially available software.

Technology

CDN nodes are usually deployed in multiple locations, often over multiple backbones. These nodes cooperate with each other to satisfy requests for content by end users, transparently moving content to optimize the delivery process. Optimization can take the form of reducing bandwidth costs, improving end-user performance (reducing page load times and improving user experience), or increasing global availability of content.

The number of nodes and servers making up CDN varies, depending on the architecture, some reaching thousands of nodes with tens of thousands of servers on many remote PoPs. Others build a global network and have a small number of geographical PoPs.

Requests for content are typically algorithmically directed to nodes that are optimal in some way. When optimizing for performance, locations that are best for serving content to the user may be chosen. This may be measured by choosing locations that are the fewest hops, the fewest number of network seconds away from the requesting client, or the highest availability in terms of server performance (both current and historical), so as to optimize delivery across local networks. When optimizing for cost, locations that are least expensive may be chosen instead.

In an optimal scenario, these two goals tend to align, as servers that are close to the end user at the edge of the network may have an advantage in performance or cost. The Edge Network is grown outward from the origin/s by further acquiring (via purchase, peering, or exchange) co-locations facilities, bandwidth and servers.

Content networking techniques

The Internet was designed according to the end-to-end principle. This principle keeps the core network relatively simple and moves the intelligence as much as possible to the network end-points: the hosts and clients. As a result the core network is specialized, simplified, and optimized to only forward data packets.

Content Delivery Networks augment the end-to-end transport network by distributing on it a variety of intelligent applications employing techniques designed to optimize content delivery. The resulting tightly integrated overlay uses web caching, server-load balancing, request routing, and content services. These techniques are briefly described below.

Web caches store popular content on servers that have the greatest demand for the content requested. These shared network appliances reduce bandwidth requirements, reduce server load, and improve the client response times for content stored in the cache.

Server-load balancing uses one or more techniques including service based (global load balancing) or hardware based, i.e. layer 4–7 switches, also known as a web switch, content switch, or multilayer switch to share traffic among a number of servers or web caches. Here the switch is assigned a single virtual IP address. Traffic arriving at the switch is then directed to one of the real web servers attached to the switch. This has the advantage of balancing load, increasing total capacity, improving scalability, and providing increased reliability by redistributing the load of a failed web server and providing server health checks.

A content cluster or service node can be formed using a layer 4–7 switch to balance load across a number of servers or a number of web caches within the network.

Request routing directs client requests to the content source best able to serve the request. This may involve directing a client request to the service node that is closest to the client, or to the one with the most capacity. A variety of algorithms are used to route the request. These include Global Server Load Balancing, DNS-based request routing, Dynamic metafile generation, HTML rewriting, and anycasting. Proximity—choosing the closest service node—is estimated using a variety of techniques including reactive probing, proactive probing, and connection monitoring.

CDNs use a variety of methods of content delivery including, but not limited to, manual asset copying, active web caches, and global hardware load balancers.

Content service protocols

Several protocol suites are designed to provide access to a wide variety of content services distributed throughout a content network. The Internet Content Adaptation Protocol (ICAP) was developed in the late 1990s to provide an open standard for connecting application servers. A more recently defined and robust solution is provided

by the Open Pluggable Edge Services (OPES) protocol. This architecture defines OPES service applications that can reside on the OPES processor itself or be executed remotely on a Callout Server. Edge Side Includes or ESI is a small markup language for edge level dynamic web content assembly. It is fairly common for websites to have generated content. It could be because of changing content like catalogs or forums, or because of personalization. This creates a problem for caching systems. To overcome this problem a group of companies created ESI.

Peer-to-peer CDNs

Although peer-to-peer (P2P) is not traditional CDN technology, it is increasingly used to deliver content to end users. P2P claims low cost and efficient distribution. Even though P2P actually generates more traffic than traditional client-server CDNs for the edge provider (because a peer also uploads data instead of just downloading it) it's welcomed by parties running content delivery/distribution services. The real strength of P2P shows when one has to distribute data in high demand, like the latest episode of a television show or some sort of software patch/update in short period of time. One of the advantages of this is that the more people who download the (same) data, the more efficient P2P is for the provider, slashing the cost of the transit fees that a CDN provider has to pay to their upstream IP transit providers.

On the other hand, the “long tail” type material does not benefit much from P2P delivery schema, since, to gain advantage over traditional distribution models, a P2P-enabled CDN must force storing (caching) data on peers—something that is usually not desired by users and which is rarely enabled.

Contrary to popular belief P2P is not limited to low-bandwidth audio-video signal distribution. There is no technical boundary, built-in inefficiency, or flaw-by-design in peer-to-peer technology to prevent distribution of full HD audio+video signal at, for example, 8 Mbit/s. It's just environmental factors, like low (upload) bandwidth or inadequate computing power in CE devices, that prevent HD material being publicly available in P2P CDNs. (Low bandwidth problems also apply to traditional CDN, though.)

There are some concerns about lack of Quality of Service control over P2P distribution, but these are being addressed by the P2P-Next consortium. Other concerns include security (e.g. modification of content to include malware) and DRM.

Chapter 8

BitTorrent (Protocol)

BitTorrent is a peer-to-peer file sharing protocol used for distributing large amounts of data. BitTorrent is one of the most common protocols for transferring large files, and it has been estimated that it accounted for roughly 27% to 55% of all Internet traffic (depending on geographical location) as of February 2009.

Programmer Bram Cohen designed the protocol in April 2001 and released a first implementation on July 2, 2001. It is now maintained by Cohen's company BitTorrent, Inc. There are numerous BitTorrent clients available for a variety of computing platforms.

Description

The BitTorrent protocol can distribute a large file without the heavy load on the source computer and network. Rather than downloading a file from a single source, the BitTorrent protocol allows users to join a "swarm" of hosts to download and upload from each other simultaneously. The protocol works as an alternative method to distribute data and can work over networks with low bandwidth so even small computers, like mobile phones, are able to distribute files to many recipients.

A user who wants to upload a file first creates a small *torrent* descriptor file that he distributes by conventional means (web, email, etc.). He then makes the file itself available through a BitTorrent node acting as a *seed*. Those with the torrent descriptor file can give it to their own BitTorrent nodes which, acting as *peers* or *leechers*, download it by connecting to the seed and/or other peers.

The file being distributed is divided into segments called *pieces*. As each peer receives a new piece of the file it becomes a source of that piece to other peers, relieving the seed from having to send a copy to every peer. With BitTorrent, the task of distributing the file is shared by those who want it; it is entirely possible for the seed to send only a single copy of the file itself to an unlimited number of peers.

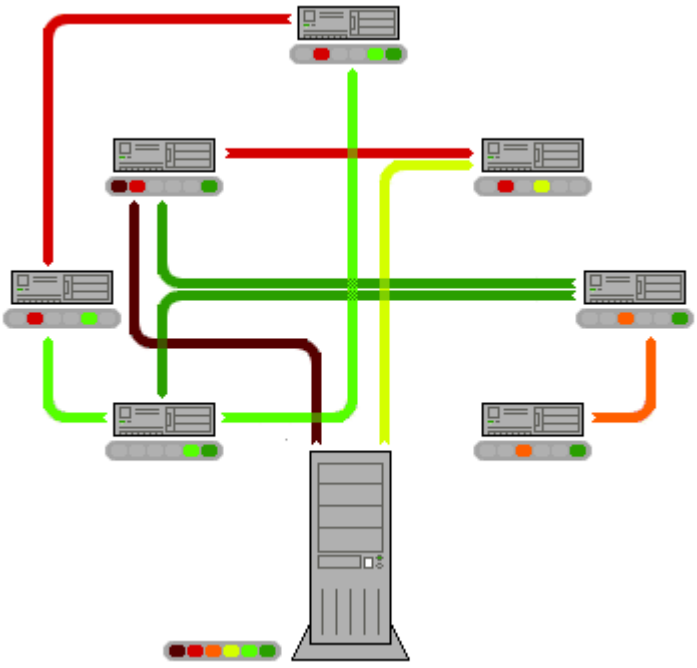
Each piece is protected by a cryptographic hash contained in the torrent descriptor. This prevents nodes from maliciously modifying the pieces they pass on to other nodes. If a

node starts with an authentic copy of the torrent descriptor, it can verify the authenticity of the actual file it has received.

When a peer completely downloads a file, it becomes an additional seed. This eventual shift from peers to seeders determines the overall "health" of the file (as determined by the number of times a file is available in its complete form).

This distributed nature of BitTorrent leads to a flood like spreading of a file throughout peers. As more peers join the swarm, the likelihood of a successful download increases. Relative to standard Internet hosting, this provides a significant reduction in the original distributor's hardware and bandwidth resource costs. It also provides redundancy against system problems, reduces dependence on the original distributor and provides a source for the file which is generally temporary and therefore harder to trace than when provided by the enduring availability of a host in standard file distribution techniques.

Operation



In this image, the colored bars beneath all of the 7 clients in the upper region above represent the file, with each color representing a individual piece of the file. After the initial pieces transfer from the seed (large system at the bottom), the pieces are individually transferred from client to client. The original *seeder* only needs to send out one copy of the file for all the clients to receive a copy.

A BitTorrent client is any program that implements the BitTorrent protocol. Each client is capable of preparing, requesting, and transmitting any type of computer file over a network, using the protocol. A peer is any computer running an instance of a client.

To share a file or group of files, a peer first creates a small file called a "torrent" (e.g. MyFile.torrent). This file contains metadata about the files to be shared and about the tracker, the computer that coordinates the file distribution. Peers that want to download the file must first obtain a torrent file for it and connect to the specified tracker, which tells them from which other peers to download the pieces of the file.

Though both ultimately transfer files over a network, a BitTorrent download differs from a classic download (as is typical with an HTTP or FTP request, for example) in several fundamental ways:

- BitTorrent makes many small data requests over different TCP connections to different machines, while classic downloading is typically made via a single TCP connection to a single machine.
- BitTorrent downloads in a random or in a "rarest-first" approach that ensures high availability, while classic downloads are sequential.

Taken together, these differences allow BitTorrent to achieve much lower cost to the content provider, much higher redundancy, and much greater resistance to abuse or to "flash crowds" than regular server software. However, this protection, theoretically, comes at a cost: downloads can take time to rise to full speed because it may take time for enough peer connections to be established, and it may take time for a node to receive sufficient data to become an effective uploader. This contrasts with regular downloads (such as from an HTTP server, for example) that, while more vulnerable to overload and abuse, rise to full speed very quickly and maintain this speed throughout.

In general, BitTorrent's non-contiguous download methods have prevented it from supporting "progressive downloads" or "streaming playback". However, comments made by Bram Cohen in January 2007 suggest that streaming torrent downloads will soon be commonplace and ad supported streaming appears to be the result of those comments. In January 2011 Cohen demonstrated an early version of BitTorrent streaming, saying the feature will be available by summer 2011.

Creating and publishing torrents

The peer distributing a data file treats the file as a number of identically sized pieces, usually with byte sizes of a power of 2, and typically between 32 kB and 4 MB each. The peer creates a hash for each piece, using the SHA-1 hash function, and records it in the torrent file. Pieces with sizes greater than 512 kB will reduce the size of a torrent file for a very large payload, but is claimed to reduce the efficiency of the protocol. When another peer later receives a particular piece, the hash of the piece is compared to the recorded hash to test that the piece is error-free. Peers that provide a complete file are called seeders, and the peer providing the initial copy is called the initial seeder.

The exact information contained in the torrent file depends on the version of the BitTorrent protocol. By convention, the name of a torrent file has the suffix `.torrent`. Torrent files have an "announce" section, which specifies the URL of the tracker, and an

"info" section, containing (suggested) names for the files, their lengths, the piece length used, and a SHA-1 hash code for each piece, all of which are used by clients to verify the integrity of the data they receive.

Torrent files are typically published on websites or elsewhere, and registered with at least one tracker. The tracker maintains lists of the clients currently participating in the torrent. Alternatively, in a *trackerless system* (decentralized tracking) every peer acts as a tracker. Azureus was the first BitTorrent client to implement such a system through the distributed hash table (DHT) method. An alternative and incompatible DHT system, known as Mainline DHT, was later developed and adopted by the BitTorrent (Mainline), µTorrent, Transmission, rTorrent, KTorrent, BitComet, and Deluge clients.

After the DHT was adopted, a "private" flag — analogous to the broadcast flag — was unofficially introduced, telling clients to restrict the use of decentralized tracking regardless of the user's desires. The flag is intentionally placed in the info section of the torrent so that it cannot be disabled or removed without changing the identity of the torrent. The purpose of the flag is to prevent torrents from being shared with clients that do not have access to the tracker. The flag was requested for inclusion in the official specification in August, 2008, but has not been accepted. Clients that have ignored the private flag were banned by many trackers, discouraging the practice.

Downloading torrents and sharing files

Users browse the web to find a torrent of interest, download it, and open it with a BitTorrent client. The client connects to the tracker(s) specified in the torrent file, from which it receives a list of peers currently transferring pieces of the file(s) specified in the torrent. The client connects to those peers to obtain the various pieces. If the swarm contains only the initial seeder, the client connects directly to it and begins to request pieces.

Clients incorporate mechanisms to optimize their download and upload rates; for example they download pieces in a random order to increase the opportunity to exchange data, which is only possible if two peers have different pieces of the file.

The effectiveness of this data exchange depends largely on the policies that clients use to determine to whom to send data. Clients may prefer to send data to peers that send data back to them (a tit for tat scheme), which encourages fair trading. But strict policies often result in suboptimal situations, such as when newly joined peers are unable to receive any data because they don't have any pieces yet to trade themselves or when two peers with a good connection between them do not exchange data simply because neither of them takes the initiative. To counter these effects, the official BitTorrent client program uses a mechanism called "optimistic unchoking", whereby the client reserves a portion of its available bandwidth for sending pieces to random peers (not necessarily known good partners, so called preferred peers) in hopes of discovering even better partners and to ensure that newcomers get a chance to join the swarm.

Although swarming scales well to tolerate flash crowds for popular content, it is less useful for unpopular content. Peers arriving after the initial rush might find the content unavailable and need to wait for the arrival of a seed in order to complete their downloads. The seed arrival, in turn, may take long to happen (this is termed the seeder promotion problem). Since maintaining seeds for unpopular content entails high bandwidth and administrative costs, this runs counter to the goals of publishers that value BitTorrent as a cheap alternative to a client-server approach. This occurs on a huge scale; measurements have shown that 38% of all new torrents become unavailable within the first month. A strategy adopted by many publishers which significantly increases availability of unpopular content consists of bundling multiple files in a single swarm. More sophisticated solutions have also been proposed; generally, these use cross-torrent mechanisms through which multiple torrents can cooperate to better share content.

BitTorrent does not offer its users anonymity. It is possible to obtain the IP addresses of all current and possibly previous participants in a swarm from the tracker. This may expose users with insecure systems to attacks. It may also expose users to the risk of being sued, if they are distributing files without permission from the copyright holder(s). However, there are ways to promote anonymity; for example, the OneSwarm project layers privacy-preserving sharing mechanisms on top of the original BitTorrent protocol.

Adoption

A growing number of individuals and organizations are using BitTorrent to distribute their own or licensed material. Independent adopters report that without using BitTorrent technology and its dramatically reduced demands on their private networking hardware and bandwidth, they could not afford to distribute their files.

Film, video and music

- BitTorrent Inc. has amassed a number of licenses from Hollywood studios for distributing popular content from their websites.
- Sub Pop Records releases tracks and videos via BitTorrent Inc. to distribute its 1000+ albums. Babyshambles and The Libertines (both bands associated with Pete Doherty) have extensively used torrents to distribute hundreds of demos and live videos. US industrial rock band Nine Inch Nails frequently distributes albums via BitTorrent.
- Podcasting software is starting to integrate BitTorrent to help podcasters deal with the download demands of their MP3 "radio" programs. Specifically, Juice and Miro (formerly known as Democracy Player) support automatic processing of .torrent files from RSS feeds. Similarly, some BitTorrent clients, such as µTorrent, are able to process web feeds and automatically download content found within them.
- DGM Live! purchases are provided via BitTorrent.

Broadcasters

- In 2008, the CBC became the first public broadcaster in North America to make a full show (*Canada's Next Great Prime Minister*) available for download using BitTorrent.
- The Norwegian Broadcasting Corporation (NRK) has since March 2008 experimented with bittorrent distribution, available online. Only selected material in which NRK owns all royalties are published. Responses have been very positive, and NRK is planning to offer more content.
- The Dutch VPRO broadcasting organization released three documentaries under a Creative Commons license using the content distribution feature of the Mininova tracker.

Personal material

- The Amazon S3 "Simple Storage Service" is a scalable Internet-based storage service with a simple web service interface, equipped with built-in BitTorrent support.
- Blog Torrent offers a simplified BitTorrent tracker to enable bloggers and non-technical users to host a tracker on their site. Blog Torrent also allows visitors to download a "stub" loader, which acts as a BitTorrent client to download the desired file, allowing users without BitTorrent software to use the protocol. This is similar to the concept of a self-extracting archive.

Software

- Blizzard Entertainment uses BitTorrent (via a proprietary client called the "Blizzard Downloader") to distribute most content for StarCraft II and World of Warcraft, including the games themselves.
- Many software games, especially those whose large size makes them difficult to host due to bandwidth limits, extremely frequent downloads, and unpredictable changes in network traffic, will distribute instead a specialized, stripped down bittorrent client with enough functionality to download the game from the other running clients and the primary server (which is maintained in case not enough peers are available).
- Many major open source and free software projects encourage BitTorrent as well as conventional downloads of their products (via HTTP, FTP etc.) to increase availability and to reduce load on their own servers, especially when dealing with larger files.
- Entropia Universe also begun distributing the client file(s) through BitTorrent.

Government

- The UK government used BitTorrent to distribute details about how the tax money of UK citizens was spent.

Others

- Facebook uses BitTorrent to distribute updates to Facebook servers.
- Twitter uses BitTorrent to distribute updates to Twitter servers.

Network impact

CableLabs, the research organization of the North American cable industry, estimates that BitTorrent represents 18% of all broadband traffic. In 2004, CacheLogic put that number at roughly 35% of all traffic on the Internet. The discrepancies in these numbers are caused by differences in the method used to measure P2P traffic on the Internet.

Routers that use network address translation (NAT) must maintain tables of source and destination IP addresses and ports. Typical home routers are limited to about 2000 table entries while some more expensive routers have larger table capacities. BitTorrent frequently contacts 300–500 servers per second rapidly filling the NAT tables. This is a common cause of home routers locking up.

Indexing

The BitTorrent protocol provides no way to index torrent files. As a result, a comparatively small number of websites have hosted a large majority of torrents, many linking to copyrighted material without the authorization of copyright holders, rendering those sites especially vulnerable to lawsuits. Several types of websites support the discovery and distribution of data on the BitTorrent network.

Public torrent hosting sites such as The Pirate Bay allow users to search and download from their collection of torrent files. Users can typically also upload torrent files for content they wish to distribute. Often, these sites also run BitTorrent trackers for their hosted torrent files, but these two functions are not mutually dependent: a torrent file could be hosted on one site and tracked by another, unrelated site.

Private host/tracker sites operate like public ones except that they restrict access to registered users and keep track of the amount of data each user uploads and downloads, in an attempt to reduce leeching.

Search engines allow the discovery of torrent files that are hosted and tracked on other sites; examples include Mininova, BTJunkie, Torrentz, The Pirate Bay, Eztorrent and isoHunt. These sites allow the user to ask for content meeting specific criteria (such as containing a given word or phrase) and retrieve a list of links to torrent files matching those criteria. This list can often be sorted with respect to several criteria, relevance (seeders-leechers ratio) being one of the most popular and useful (due to the way the protocol behaves, the download bandwidth achievable is very sensitive to this value). Metasearch engines allow one to search several BitTorrent indices and search engines at once.

Technologies built on BitTorrent

The BitTorrent protocol is still under development and therefore may still acquire new features and other enhancements such as improved efficiency.

Distributed trackers

On May 2, 2005, Azureus 2.3.0.0 (now known as Vuze) was released, introducing support for "trackerless" torrents through a system called the "distributed database." This system is a DHT implementation which allows the client to use torrents that do not have a working BitTorrent tracker. The following month, BitTorrent, Inc. released version 4.2.0 of the Mainline BitTorrent client, which supported an alternative DHT implementation (popularly known as "Mainline DHT") that is incompatible with that of Azureus. Current versions of the official BitTorrent client, µTorrent, BitComet, and BitSpirit all share compatibility with Mainline DHT. Both DHT implementations are based on Kademlia. As of version 3.0.5.0, Azureus also supports Mainline DHT in addition to its own distributed database through use of an optional application plugin. This potentially allows the Azureus client to reach a bigger swarm.

Another idea that has surfaced in Vuze is that of *virtual torrents*. This idea is based on the distributed tracker approach and is used to describe some web resource. Currently, it is used for instant messaging. It is implemented using a special messaging protocol and requires an appropriate plugin. Anatomic P2P is another approach, which uses a decentralized network of nodes that route traffic to dynamic trackers.

Most BitTorrent clients also use Peer exchange (PEX) to gather peers in addition to trackers and DHT. Peer exchange checks with known peers to see if they know of any other peers. With the 3.0.5.0 release of Vuze, all major BitTorrent clients now have compatible peer exchange.

Web seeding

Web seeding was implemented in 2006 as the ability of BitTorrent clients to download torrent pieces from an HTTP source in addition to the swarm. The advantage of this feature is that a website may distribute a torrent for a particular file or batch of files and make those files available for download from that same web server; this can simplify long-term seeding and load balancing through the use of existing, cheap, web hosting setups. In theory, this would make using BitTorrent almost as easy for a web publisher as creating a direct HTTP download. In addition, it would allow the "web seed" to be disabled if the swarm becomes too popular while still allowing the file to be readily available.

This feature has two distinct and incompatible specifications.

The first was created by John "TheSHAD0W" Hoffman, who created BitTornado. From version 5.0 onward, the Mainline BitTorrent client also supports web seeds, and the

BitTorrent web site had a simple publishing tool that creates web seeded torrents. µTorrent added support for web seeds in version 1.7. BitComet added support for web seeds in version 1.14. This first specification requires running a web service that serves content by info-hash and piece number, rather than filename.

The other specification is created by GetRight authors and can rely on a basic HTTP download space (using byte serving).

In September 2010, a new service named Burnbit was launched which generates a torrent from any URL using webseeding.

There exist server-side solutions that provide initial seeding of the file from the webserver via standard Bittorrent protocol and when the number of external seeders reach a limit, they stop serving the file from the original source.

RSS feeds

A technique called Broadcatching combines RSS with the BitTorrent protocol to create a content delivery system, further simplifying and automating content distribution. Steve Gillmor explained the concept in a column for Ziff-Davis in December, 2003. The discussion spread quickly among bloggers (Ernest Miller, Chris Pirillo, etc.). In an article entitled *Broadcatching with BitTorrent*, Scott Raymond explained:

I want RSS feeds of BitTorrent files. A script would periodically check the feed for new items, and use them to start the download. Then, I could find a trusted publisher of an Alias RSS feed, and "subscribe" to all new episodes of the show, which would then start downloading automatically — like the "season pass" feature of the TiVo.
—Scott Raymond, *scottraymond.net*

The RSS feed will track the content, while BitTorrent ensures content integrity with cryptographic hashing of all data, so feed subscribers will receive uncorrupted content.

One of the first and popular software clients (free and open source) for *broadcatching* is Miro. Other free software clients such as PenguinTV and KatchTV are also now supporting broadcatching.

The BitTorrent web-service MoveDigital has the ability to make torrents available to any web application capable of parsing XML through its standard REST-based interface. Additionally, Torrenthut is developing a similar torrent API that will provide the same features, as well as further intuition to help bring the torrent community to Web 2.0 standards. Alongside this release is a first PHP application built using the API called PEP, which will parse any Really Simple Syndication (RSS 2.0) feed and automatically create and seed a torrent for each enclosure found in that feed.

Throttling and encryption

Since BitTorrent makes up a large proportion of total traffic, some ISPs have chosen to throttle (slow down) BitTorrent transfers to ensure network capacity remains available for other uses. For this reason, methods have been developed to disguise BitTorrent traffic in an attempt to thwart these efforts.

Protocol header encrypt (PHE) and Message stream encryption/Protocol encryption (MSE/PE) are features of some BitTorrent clients that attempt to make BitTorrent hard to detect and throttle. At the moment Vuze, Bitcomet, KTorrent, Transmission, Deluge, µTorrent, MooPolice, Halite, rTorrent and the latest official BitTorrent client (v6) support MSE/PE encryption.

In September 2006 it was reported that some software could detect and throttle BitTorrent traffic masquerading as HTTP traffic.

Reports in August 2007 indicated that Comcast was preventing BitTorrent seeding by monitoring and interfering with the communication between peers. Protection against these efforts is provided by proxying the client-tracker traffic via an encrypted tunnel to a point outside of the Comcast network. Comcast has more recently called a "truce" with BitTorrent, Inc. with the intention of shaping traffic in a protocol-agnostic manner. Questions about the ethics and legality of Comcast's behavior have led to renewed debate about Net neutrality in the United States.

In general, although encryption can make it difficult to determine *what* is being shared, BitTorrent is vulnerable to traffic analysis. Thus even with MSE/PE, it may be possible for an ISP to recognize BitTorrent and also to determine that a system is no longer downloading but only uploading data, and terminate its connection by injecting TCP RST (reset flag) packets.

Multitracker

Another unofficial feature is an extension to the BitTorrent metadata format proposed by John Hoffman and implemented by several indexing websites. It allows the use of multiple trackers per file, so if one tracker fails, others can continue to support file transfer. It is implemented in several clients, such as BitComet, BitTornado, BitTorrent, KTorrent, Transmission, Deluge, µTorrent, rtorrent, and Vuze. Trackers are placed in groups, or tiers, with a tracker randomly chosen from the top tier and tried, moving to the next tier if all the trackers in the top tier fail.

Torrents with multiple trackers can decrease the time it takes to download a file, but also has a few consequences:

- Poorly implemented clients may contact multiple trackers, leading to more overhead-traffic.

- Torrents from closed trackers suddenly become downloadable by non-members, as they can connect to a seed via an open tracker.

Decentralized keyword search

Even with distributed trackers, a third party is still required to find a specific torrent. This is usually done in the form of a hyperlink from the website of the content owner or through indexing websites like The Pirate Bay or Torrentz.

The Tribler BitTorrent client is the first to incorporate decentralized search capabilities. With Tribler, users can find .torrent files that are hosted among other peers, instead of on a centralized index sites. It adds such an ability to the BitTorrent protocol using a gossip protocol, somewhat similar to the eXeem network which was shut down in 2005. The software includes the ability to recommend content as well. After a dozen downloads the Tribler software can roughly estimate the download taste of the user and recommend additional content.

In May 2007 Cornell University published a paper proposing a new approach to searching a peer-to-peer network for inexact strings, which could replace the functionality of a central indexing site. A year later, the same team implemented the system as a plugin for Vuze called Cubit and published a follow-up paper reporting its success.

A somewhat similar facility but with a slightly different approach is provided by the BitComet client through its "Torrent Exchange" feature. Whenever two peers using BitComet (with Torrent Exchange enabled) connect to each other they exchange lists of all the torrents (name and info-hash) they have in the Torrent Share storage (torrent files which were previously downloaded and for which the user chose to enable sharing by Torrent Exchange).

Thus each client builds up a list of all the torrents shared by the peers it connected to in the current session (or it can even maintain the list between sessions if instructed). At any time the user can search into that Torrent Collection list for a certain torrent and sort the list by categories. When the user chooses to download a torrent from that list, the .torrent file is automatically searched for (by info-hash value) in the DHT Network and when found it is downloaded by the querying client which can after that create and initiate a downloading task.

Implementations

The BitTorrent specification is free to use and many clients are open source, so BitTorrent clients have been created for all common operating systems using a variety of programming languages. The official BitTorrent client, µTorrent, Vuze, Transmission, and BitComet are some of the most popular clients.

Some BitTorrent implementations such as MLDonkey and Torrentflux are designed to run as servers. For example, this can be used to centralize file sharing on a single dedicated server which users share access to on the network. Server-oriented BitTorrent implementations can also be hosted by hosting providers at co-located facilities with high bandwidth Internet connectivity (e.g., a datacenter) which can provide dramatic speed benefits over using BitTorrent from a regular home broadband connection.

Services such as ImageShack can download files on BitTorrent for the user, allowing them to download the entire file by HTTP once it is finished.

The Opera web browser supports BitTorrent, as does Wyzo. BitLet allows users to download Torrents directly from their browser using a Java applet. Sites such as xFiles and DuShare allow to transfer big files directly using bittorrent inside adobe Flash.

An increasing number of hardware devices are being made to support BitTorrent. These include routers and NAS devices containing BitTorrent-capable firmware like OpenWrt.

Proprietary versions of the protocol which implement DRM, encryption, and authentication are found within managed clients such as Pando.

Development

An unimplemented (as of February 2008) unofficial feature is Similarity Enhanced Transfer (SET), a technique for improving the speed at which peer-to-peer file sharing and content distribution systems can share data. SET, proposed by researchers Pucha, Andersen, and Kaminsky, works by spotting chunks of identical data in files that are an exact or near match to the one needed and transferring these data to the client if the "exact" data are not present. Their experiments suggested that SET will help greatly with less popular files, but not as much for popular data, where many peers are already downloading it. Andersen believes that this technique could be immediately used by developers with the BitTorrent file sharing system.

As of December 2008, BitTorrent, Inc. is working with Oversi on new Policy Discover Protocols that query the ISP for capabilities and network architecture information. Oversi's ISP hosted NetEnhancer box is designed to "improve peer selection" by helping peers find local nodes, improving download speeds while reducing the loads into and out of the ISP's network.

Legal issues

There has been much controversy over the use of BitTorrent trackers. BitTorrent metafiles themselves do not store file contents. Whether the publishers of BitTorrent metafiles violate copyrights by linking to copyrighted material without the authorization of copyright holders is controversial.

Various jurisdictions have pursued legal action against websites that host BitTorrent trackers. High-profile examples include the closing of Suprnova.org, Torrentspy, LokiTorrent, Mininova and OiNK.cd. The Pirate Bay torrent website, formed by a Swedish group, is noted for the "legal" section of its website in which letters and replies on the subject of alleged copyright infringements are publicly displayed. On 31 May 2006, The Pirate Bay's servers in Sweden were raided by Swedish police on allegations by the MPAA of copyright infringement; however, the tracker was up and running again three days later.

BitTorrent and malware

Several studies on BitTorrent have indicated that a large portion of files available for download via BitTorrent contain malware. In particular, one small sample indicated that 18% of all executable programs available for download contained malware. Another study claims that as much as 14.5% of BitTorrent downloads contain zero-day malware, and that BitTorrent was used as the distribution mechanism for 47% of all zero-day malware they have found.

Chapter 9

Gnutella

Gnutella is a large peer-to-peer network which, at the time of its creation, was the first decentralized peer-to-peer network of its kind, leading to other, later networks adopting the model. It celebrated a decade of existence on March 14, 2010 and has a user base in the millions for peer-to-peer file sharing.

In June 2005, gnutella's population was 1.81 million computers increasing to over three million nodes by January 2006. In late 2007, it was the most popular file sharing network on the Internet with an estimated market share of more than 40%.

History

The first client was developed by Justin Frankel and Tom Pepper of Nullsoft in early 2000, soon after the company's acquisition by AOL. On March 14, the program was made available for download on Nullsoft's servers. The event was prematurely announced on Slashdot, and thousands downloaded the program that day. The source code was to be released later, under the GNU General Public License (GPL).

The next day, AOL stopped the availability of the program over legal concerns and restrained Nullsoft from doing any further work on the project. This did not stop gnutella; after a few days, the protocol had been reverse engineered, and compatible free and open source clones began to appear. This parallel development of different clients by different groups remains the *modus operandi* of gnutella development today.

The gnutella network is a fully distributed alternative to such semi-centralized systems as FastTrack (KaZaA) and the original Napster. Initial popularity of the network was spurred on by Napster's threatened legal demise in early 2001. This growing surge in popularity revealed the limits of the initial protocol's scalability. In early 2001, variations on the protocol (first implemented in proprietary and closed source clients) allowed an improvement in scalability. Instead of treating every user as client and server, some users were now treated as "ultrapeers", routing search requests and responses for users connected to them.

This allowed the network to grow in popularity. In late 2001, the gnutella client Limewire Basic became free and open source. In February 2002, Morpheus, a commercial file sharing group, abandoned its FastTrack-based peer-to-peer software and released a new client based on the free and open source gnutella client Gnucleus.

The word "gnutella" today refers not to any one project or piece of software, but to the open protocol used by the various clients.

The name is a portmanteau of *GNU* and *Nutella*, the brand name of a hazelnut flavored spread: supposedly, Frankel and Pepper ate a lot of Nutella working on the original project, and intended to license their finished program under the GNU General Public License. Gnutella is not associated with the GNU project or its own peer-to-peer network, GNUnet.

On October 26, 2010, popular gnutella servent LimeWire was ordered shut down by Judge Kimba Wood of the United States District Court for the Southern District of New York when she signed a Consent Injunction which LimeWire and recording industry plaintiffs had agreed upon. This event may have caused a notable drop in the size of the network because, while negotiating the injunction, LimeWire had staff add remote shutdown of gnutella access and file transfers as some users installed affected versions. When the injunction came into force versions later than 5.5.10 left the network or were disabled as users started them.

On November 9, 2010, LimeWire was resurrected by a secret team of developers and dubbed Limewire Pirate Edition. It was based on LimeWire 5.6 BETA. This version had its server dependencies removed and all the PRO features enabled for free.

Design

To envision how gnutella originally worked, imagine a large circle of users *(called nodes),* each of whom have gnutella client software. On initial startup, the client software must bootstrap and find at least one other node. Various methods have been used for this, including a pre-existing address list of possibly working nodes shipped with the software, using updated web caches of known nodes (called Gnutella Web Caches), UDP host caches and, rarely, even IRC. Once connected, the client requests a list of working addresses. The client tries to connect to the nodes it was shipped, as well as nodes it receives from other clients, until it reaches a certain quota. It connects to only that many nodes, locally caches the addresses it has not yet tried, and discards the addresses it tried that were invalid.

When the user wants to do a search, the client sends the request to each actively connected node. In version 0.4 of the protocol, the number of actively connected nodes for a client was quite small (around 5), so each node then forwarded the request to all its actively connected nodes, and they in turn forwarded the request, and so on, until the packet reached a predetermined number of "hops" from the sender (maximum 7).

Since version 0.6, gnutella is a composite network made of leaf nodes and ultra nodes (also called ultrapeers). The leaf nodes are connected to a small number of ultrapeers (typically 3) while each ultrapeer is connected to more than 32 other ultrapeers. With this higher outdegree, the maximum number of "hops" a query can travel was lowered to 4.

Leaves and ultrapeers use the Query Routing Protocol to exchange a Query Routing Table (QRT), a table of 64 Ki-slots and up to 2 Mi-slots consisting of hashed keywords. A leaf node sends its QRT to each of the ultrapeers it is connected to, and ultrapeers merge the QRT of all their leaves (downsized to 128 Ki-slots) plus their own QRT (if they share files) and exchange that with their own neighbours. Query routing is then done by hashing the words of the query and seeing whether all of them match in the QRT. Ultrapeers do that check before forwarding a query to a leaf node, and also before forwarding the query to a peer ultra node provided this is the last hop the query can travel.

If a search request turns up a result, the node that has the result contacts the searcher. In the classic gnutella protocol, response messages were sent back along the route the query came through, as the query itself did not contain identifying information of the node. This scheme was later revised, so that search results now are delivered over User Datagram Protocol (UDP) directly to the node that initiated the search, usually an ultrapeer of the node. Thus, in the current protocol, the queries carry the IP address and port number of either node. This lowers the amount of traffic routed through the gnutella network, making it significantly more scalable.

If the user decides to download the file, they negotiate the file transfer. If the node which has the requested file is not firewalled, the querying node can connect to it directly. However, if the node is firewalled, stopping the source node from receiving incoming connections, the client wanting to download a file sends it a so called "push request" to the server for the remote client to initiate the connection instead (to "push" the file). At first, these push requests were routed along the original chain it used to send the query. This was rather unreliable because routes would often break and routed packets are always subject to flow control. Therefore so called "push proxies" were introduced. These are usually the ultrapeers of a leaf node and they are announced in search results. The client connects to one of these "push proxies" using a HTTP request and the proxy sends a "push request" to leaf on behalf of the client. Normally, it is also possible to send a push request over UDP to the push proxy which is more efficient than using TCP. Push proxies have two advantages: First, ultrapeer-leaf connections are more stable than routes which makes push requests much more reliable. Second, it reduces the amount of traffic routed through the gnutella network.

Finally, when a user disconnects, the client software saves the list of nodes that it was actively connected to and those collected from pong packets for use the next time it attempts to connect so that it becomes independent from any kind of bootstrap services.

In practice, this method of searching on the gnutella network was often unreliable. Each node is a regular computer user; as such, they are constantly connecting and

disconnecting, so the network is never completely stable. Also, the bandwidth cost of searching on gnutella grew exponentially to the number of connected users, often saturating connections and rendering slower nodes useless. Therefore, search requests would often be dropped, and most queries reached only a very small part of the network. This observation identified the gnutella network as an unscalable distributed system, and inspired the development of distributed hash tables, which are much more scalable but support only exact-match, rather than keyword, search.

To address the problems of bottlenecks, gnutella developers implemented a tiered system of **ultrapeers** and **leaves**. Instead of all nodes being considered equal, nodes entering into the network were kept at the 'edge' of the network as a leaf, not responsible for any routing, and nodes which were capable of routing messages were promoted to ultrapeers, which would accept leaf connections and route searches and network maintenance messages. This allowed searches to propagate further through the network, and allowed for numerous alterations in the topology which have improved the efficiency and scalability greatly.

Additionally gnutella adopted a number of other techniques to reduce traffic overhead and make searches more efficient. Most notable are Query Routing Protocol (QRP) and Dynamic Querying (DQ). With QRP a search reaches only those clients which are likely to have the files, so rare files searches grow vastly more efficient, and with DQ the search stops as soon as the program has acquired enough search results, which vastly reduces the amount of traffic caused by popular searches. Gnutella For Users has a vast amount of information about these and other improvements to gnutella in user-friendly style.

One of the benefits of having gnutella so decentralized is to make it very difficult to shut the network down and to make it a network in which the users are the only ones who can decide which content will be available. Unlike Napster, where the entire network relied on the central server, gnutella cannot be shut down by shutting down any one node and it is impossible for any company to control the contents of the network, which is also due to the many free and open source gnutella clients which share the network.

Protocol features and extensions

Gnutella did once operate on a purely query flooding-based protocol. The outdated gnutella version 0.4 network protocol employs five different packet types, namely

- ping: discover hosts on network
- pong: reply to ping
- query: search for a file
- query hit: reply to query
- push: download request for firewalled servents

These are mainly concerned with searching the gnutella network. File transfers are handled using HTTP.

The development of the gnutella protocol is currently led by the Gnutella Developers Forum ("The GDF"). Many protocol extensions have been and are being developed by the software vendors and free gnutella developers of the GDF. These extensions include intelligent query routing, SHA-1 checksums, query hit transmission via UDP, querying via UDP, dynamic queries via TCP, file transfers via UDP, XML meta data, source exchange (also known as "the download mesh") and parallel downloading in slices (swarming).

There are efforts to finalize these protocol extensions in the gnutella 0.6 specification at the gnutella protocol development website. The gnutella 0.4 standard, although still being the latest protocol specification since all extensions only exist as proposals so far, is outdated. In fact, it is hard or impossible to connect today with the 0.4 handshake and according to developers in the GDF, version 0.6 is what new developers should pursue using the work-in-progress specifications.

The gnutella protocol remains under development and in spite of attempts to make a clean break with the complexity inherited from the old gnutella 0.4 and to design a clean new message architecture, it is still one of the most successful file-sharing protocols to date.

Gnutella2

The Gnutella2 protocol, often referred to as G2, is, despite its name, not a successor protocol of gnutella, but rather a fork. A sore point with many gnutella developers is that the "Gnutella2" name conveys an upgrade or superiority, which led to a "Gnutella2 flame war". Other criticism included the use of the gnutella network to bootstrap G2 peers and poor documentation of the G2 protocol. Additionally, the search retries of the Shareaza client, which was one of the initial G2 clients, could unnecessarily burden the gnutella network.

The fork took place in 2002 and both protocols have undergone significant iterations since that time. G2 has both advantages and disadvantages compared to gnutella. An advantage often cited is Gnutella2's hybrid search is more efficient than the original gnutella query flooding, which was used in 2002. An advantage for gnutella is its user population numbers in the millions, whereas the G2 network is approximately an order of magnitude smaller. It is difficult to compare the protocols in their current form; the individual client choice will probably have as much an effect to an end user on either network.

Software

The following tables compare general and technical information for a number of applications supporting the gnutella network. The tables do **not** attempt to give a complete list of gnutella clients. The tables are limited to clients that can participate in the current gnutella network.

General specifications

Name	Platform	License	Latest Release	Heritage
Acquisition	Mac OS X	Proprietary	2.2 (v223) (November 19, 2010; 4 months ago)	LimeWire
BearFlix	Microsoft Windows	Proprietary	6.2.2.521	BearShare
BearShare (Before Version 6)	Microsoft Windows	Proprietary	5.2.5.6	Original work
Cabos	Java	GNU GPL	0.8.2 (February 9, 2010; 13 months ago)	LimeWire
FilesWire (P2P)	Java	Proprietary	Beta 1.1 (2007)	Original Work
FrostWire	Java	GNU GPL	4.21.5 (March 23, 2011; 14 days ago)	LimeWire
giFT (Gnutella plug-in)	Cross-platform	GNU GPL	0.0.11 (2006-08-06)	Original Work
Gnucleus/GnucDNA	Microsoft Windows	GNU GPL, GNU LGPL	2.2.0.0 (2005-06-17)	Original Work
gtk-gnutella	Cross-platform	GNU GPL	0.96.9 (March 14, 2011; 23 days ago)	Original Work
iMesh (Before Version 6)	Microsoft Windows		Unknown	GnucDNA
KCeasy	Microsoft Windows	GNU GPL	0.19-rc1 (2008-02-03)	giFT
Kiwi Alpha	Microsoft Windows		Unknown	GnucDNA
LimeWire	Java	GNU GPL	5.5.16 (September 30, 2010; 6 months ago)	Original Work
Morpheus	Microsoft Windows	Proprietary	5.55.1 (November 15, 2007; 3 years ago)	GnucDNA
Phex	Java	GNU GPL	3.4.2.116 (2009-02-01)	Original Work
Poisoned	Mac OS X	GNU GPL	0.5191 (August 8, 2006)	giFT

Shareaza	Microsoft Windows	GNU GPL	2.5.4.0 (February 12, 2011; 53 days ago)	Original Work	
Symella	Symbian OS	GNU GPL	1.41 (2009-12-11)	Original Work	
Zultrax	Microsoft Windows	Proprietary	4.33 (April 2009)	Original Work	

Gnutella features

Client	Hash search	Chat[›]	Buddy list	Handles large files (> 4 GiB)	Unicode-compatible Query Routing	UPnP port mapping[›]	NAT traversal	NAT port mapping	RUDP[›]	TCP Push proxy	UDP Push proxy	Ultrapeer	GWebCache[›]	UDP Host Cache	THEX	TLS	Other
BearShare	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Yes	No	-
giFT (core & plug-ins)	Yes	N/A	N/A	No	No	*?*	*?*	*?*	No	Yes[a][›]	No	No[b][›]	Yes	No	No	No	-
GnucDNA[c][›]	Yes	N/A	N/A	No	No	No	No	No	No	Yes	No	No[b][›]	Yes	No	No	No	-
gtk-gnutella	Yes[d][›]	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No (Dropped)	Yes	Yes	Yes	IPv6, DHT
LimeWire[h][›]	Yes[d][›]	Yes	GMail or XMPP	Yes	Yes	Yes	Yes[e][›]	Yes[g][›]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	DHT
Phex	Yes	Yes	No	Yes	Yes	No	Yes[socks][›]	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes	I2P
Shareaza	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes[f][›]	Yes	No	G2, BT, eD2k, IRC

Notes

^ **Chat:** It refers to client-to-client chat.
^ **UPnP port mapping:** Automatically configure port forwarding (requires Router with UPnP support)
^ **RUDP:** Reliable UDP protocol used for NAT-to-NAT transfers; sometimes called Firewall-to-Firewall
^ **GWebCache:** The UDP host cache is the preferred bootstrap method.
^ **a:** Client only
^ **b:** Not high out degree, so unusable in current form.
^ **c:** Version 0.9.2.7
^ **d:** Via the Kademlia based Mojito DHT network only supported by LimeWire and gtk-gnutella (starting version r15750); completely different from SHA-1 searches supported by all other gnutella clients.
^ **e:** Port triggering or *firewall to firewall* (FW2FW).
^ **socks:** Via SOCKS proxy which can tunnel over SSH.
^ **f:** Since version 2.2.4.0
^ **g:** Automatic with UPnP, or manual configuration in LimeWire firewall options
^ **h:** As the LimeWire client isn't officially available any more at the moment, clients like FrostWire that share most of LimeWires code base can be used as an alternative.

- Morpheus differs significantly and may have completely independent code from the GnucDNA engine. Morpheus can function as a *modern* ultrapeer whereas other GnucDNA clients can not.
- Gnucleus and Kiwi Alpha use the GnucDNA engine.
- BearFlix should be similar to BearShare.
- giFTcurs, Apollon, FilePipe, giFToxic, giFTui, giFTwin32, KCeasy, Poisoned, and Xfactor are GUI front-ends for the giFT engine.
- etomi uses outdated Shareaza networking code.
- 360Share, LemonWire, MP3Torpedo, and DexterWire are the LimeWire package.
- FrostWire is near identical to LimeWire; Acquisition and Cabos have custom front-ends but use LimeWire as an engine.

Chapter 10

Gnutella2

Gnutella2, often referred to as **G2**, is a peer-to-peer protocol developed mainly by Michael Stokes and released in 2002. While inspired by the gnutella protocol, G2 shares little of its design with the exception of its connection handshake and download mechanics. It adopts an extensible binary packet format and an entirely new search algorithm. Furthermore, it has a slightly different network topology and an improved metadata system which helps effectively to reduce fake files, viruses (etc.) on the network.

History

In November 2002, Michael Stokes announced the Gnutella2 protocol to the Gnutella Developers Forum. While some thought the goals stated for Gnutella2, primarily to make a clean break with the gnutella 0.6 protocol and start over so that some of gnutella's less clean parts would be done more elegantly, to be impressive and desirable, other developers, primarily those of LimeWire and BearShare, thought it a "cheap publicity stunt" and discounted technical merits. Many still refuse to refer to the network as "Gnutella2" and instead refer to it as "Mike's Protocol" ("MP").

The Gnutella2 protocol still uses the old "GNUTELLA CONNECT/0.6" handshake string for its connections as defined in the gnutella 0.6 specifications, which was criticized by the GDF as an attempt to use the gnutella network for bootstrapping the new, unrelated network, while proponents of the network claimed that its intent was to remain backwards-compatible with gnutella to allow current gnutella clients to add Gnutella2 at their leisure.

With the developers entrenched in their positions, a flame war soon erupted, further cementing both sides' resolve.

The draft specifications were released on March 26, 2003, and more detailed specifications soon followed. G2 is not supported by many of the "old" gnutella network clients, however many Gnutella2 clients still also connect to gnutella. Many Gnutella2 proponents claim that this is because of political reasons, while gnutella supporters claim that the drastic changes don't have enough merit to outweigh the cost of deep rewrites.

Design

Gnutella2 divides nodes into two groups: Leaves and Hubs. Most Leaves maintain two connections to Hubs, while Hubs accept hundreds of Leaf connections, and an average of 7 connections to other Hubs. When a search is initiated, the node obtains a list of Hubs if needed, and contacts the Hubs in the list, noting which have been searched, until the list is exhausted, or a predefined search limit has been reached. This allows a user to find a popular file easily without loading the network, while theoretically maintaining the ability for a user to find a single file located anywhere on the network.

Hubs index what files a Leaf has by means of a Query Routing Table, which is filled with single bit entries of hashes of keywords which the Leaf uploads to the Hub, and which the Hub then combines with all the hash tables its Leaves have sent it in order to create a version to send to their neighbouring Hubs. This allows for Hubs to reduce bandwidth greatly by simply not forwarding queries to Leaves and neighbouring Hubs if the entries which match the search are not found in the routing tables.

Gnutella2 relies extensively on UDP, rather than TCP, for searches. The overhead of setting up a TCP connection would make a random walk search system, requiring the contacting of large numbers of nodes with small volumes of data, unworkable. However, UDP is not without its own drawbacks. Because UDP is connectionless, there is no standard method to inform the sending client that a message was received, and so if the packet is lost, there is no way to know. Because of this, UDP packets in Gnutella2 have a flag to enable a reliability setting. When an UDP packet with enabled reliability flag is received, the client will respond with an acknowledge packet to inform the sending client that their packet arrived at its destination. If the acknowledge packet is not sent, the reliable packet will be retransmitted in an attempt to ensure delivery. Low importance packets which do not have the flag enabled do not require an acknowledge packet, reducing reliability, but also reducing overhead as no acknowledge packet needs to be sent.

Protocol features

Gnutella2 has an extensible binary packet format, comparable to an XML document tree, which was conceived as an answer for some of gnutella's less elegant parts. The packet format was designed so that future network improvements and individual vendor features could be added without worry of causing bugs in other clients on the network.

For file identification and secure integrity check of files it employs SHA-1 hashes. To allow for a file to be reliably downloaded in parallel from multiple sources as well as to allow for the reliable uploading of parts as the file is being downloaded (swarming), Tiger tree hashes are used.

To create a more robust and complete system for searching, Gnutella2 also has a metadata system for more complete labeling, rating, and quality information to be given in the search results than would simply be gathered by the file names. Nodes can even

share this information after they have deleted the file, allowing users to mark viruses and worms on the network without requiring them to keep a copy.

Gnutella2 also utilizes compression in its network connections to reduce the bandwidth used by the network.

Shareaza has the additional feature to request previews of images and videos, though currently no additional clients take advantage of this.

Differences from gnutella

Overall, the two networks are fairly similar, with the primary differences being in the packet format and the search methodology.

Protocol

Gnutella's packet format has been criticized because it was not originally designed with extensibility in mind, and has had many additions over the years, leaving the packet structure cluttered and inefficient. Gnutella2 learned from this, and aside from having many of the added features of gnutella standard in Gnutella2, designed in future extensibility from the start.

Search algorithm

While gnutella uses a query flooding method of searching, Gnutella2 uses a walk system where a searching node gathers a list of Hubs and contacts them directly, one at a time. However, as Hub organize themselves in so called "Hub clusters", where each Hub mirrors the information stored by its neighbours, the Leaf is returned the information of the entire Hub cluster (usually 7 Hubs). This has several advantages over the gnutella's query flooding system. It is more efficient, as continuing a search does not increase the network traffic exponentially, queries are not routed through as many nodes, and it increases the granularity of a search, allowing a client to stop once a pre-defined threshold of results has been obtained more effectively than in gnutella. However, the walk system also increases the complexity of the network and the network maintenance required, as well as requiring safeguards to prevent a malicious attacker from using the network for denial-of-service attacks.

Terminology

There is also a difference in terminology: while the more capable nodes which are used to condense the network are referred to as *Ultrapeers* in gnutella, they are called *Hubs* in Gnutella2, and they are also used slightly differently in topology. In gnutella, the Ultrapeers generally maintain as many leaves as peer connections, while Gnutella2 Hubs maintain far more leaves, and fewer peer (Hub-to-Hub) connections. The reason for this is that the search methods of the various networks have different optimum topologies.

Clients

List

Free software Gnutella2 clients include:

- Adagio (Cross Platform), written in Ada, under the GPL.
- Gnucleus (Windows), written in C/C++, under the LGPL
- MLDonkey (Cross Platform), written in Ocaml, under the GPL, however as of version 2.9.0, support is officially unmaintained and disabled in the binaries.
- Shareaza (Windows), multi-network, written in C++, under the GPL, currently +/- 93% network share
- Sharelin (Cross platform), written in C++, web-GUI
- G2CD (Linux/Unix/BSD) Hub mode only implementation of the Gnutella2 network.
- Quazaa (Cross platform) written in C++/QT4, under GPLv3. New client inspired by Shareaza.

Proprietary software implementations include:

- Foxy (Windows) Chinese GnucDNA-derived program, no interaction with any other G2 clients possible, free.
- Kiwi Alpha (Windows)
- Morpheus (Windows)
- TrustyFiles (Windows)

Comparison

The following table compares general and technical information for a number of available applications supporting the G2 network.

client	Chat	Handles big files (>4 GB)	UKHL	Unicode	UPnP port mapping	NAT traversal	Remote preview	Ability to search with hashes	Hub modus	Spyware/ Adware/ Malware-free	Other networks	Based on	OS	Other
Adagio	No	No	No	No	No	No	No	Yes	No	Yes	N/A	-	Cross-platform	-
Foxy	Yes	No	No	Yes	Yes	No	No	Yes	Foxy only	No	N/A	GnucDNA	Cross-platform	GPL violator
FileScope	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	gnutella, eD2k,	-	Cross-platform	-

											OpenNap			
Gnucleus	No	No	No	No	No	No	No	Yes	No	Yes	gnutella	GnucDNA	Windows	-
Kiwi Alpha	No	No	No	No	No	No	No	Yes	No	No	gnutella	GnucDNA	Windows	-
MLdonkey	Currently no official support for the Gnutella2 plugin. Searching for new maintainer for this plugin.										-	Cross-platform		
Morpheus	Yes	No	No	No	Yes	No	No	Yes	No	No	gnutella, NEOnet	GnucDNA	Windows	Development and hosting of the client has been stopped
Shareaza	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	gnutella, eD2k, BitTorrent	-	Windows	Includes IRC support
Shareli n	No	Yes	Yes	No	No	No	No	Yes	No	Yes	N/A	-	Unix/Linux	Console application with WEB-GUI
Trusty Files	No	No	No	No	No	No	No	Yes	No	removable in commercial version ($29)	eD2k, Overnet, BitTorrent, gnutella	GnucDNA (partial)	Windows	-

Chapter 11

Netsukuku

Netsukuku is the name of an experimental peer-to-peer routing system, developed by the FreakNet MediaLab **(Italian)**, created to build up a distributed network, anonymous and censorship-free, fully independent but not necessarily separated from Internet, without the support of any server, ISP and no central authority. It does not rely on a backbone router, or on any routing equipment other than normal network interface cards.

Basic idea

Netsukuku aims to build a fully distributed network that does not rely on single points of failure as the actual Internet. The main idea is to build a system that can be built and be maintained autonomously. It is designed to handle a very large number of nodes with minimal CPU and memory resources. This mesh network can be built using existing network infrastructure components such as Wi-Fi.

Netsukuku routing protocol builds the appropriate routes that connects all the computer on the mesh, replacing the level 3 of the OSI model with another routing protocol called QSPN (Quantum Shortest Path Netsukuku).

Also, the domain name system (DNS) is replaced by a decentralised and distributed system called ANDNA (A Netsukuku Domain Name Architecture).

Netsukuku was born from the idea to create a pure net that takes advantage of being distributed for creating and maintaining itself autonomously. The network itself can survive node loss without interrupting the service in the network. Note that for a completely dynamic network, it would require a constant update to the routes and this goes against the scalability and stability requirements of Netsukuku.

Since Netsukuku is not a P2P network built upon the Internet it handles routes differently. It is a physical network and it is a dynamic routing system designed to handle 2^{128} nodes without any servers or central systems.

How it works

Netsukuku is aimed to be able to run with minimal effort and resources, so instead of solving heavy computational problems with routing calculation and yet be scalable up to 2^{128} nodes (using IPv6) it uses a routing protocol called QSPN.

When a node joins the mesh network, Netsukuku automatically suits and all other nodes come to know the fastest and most efficient routes to communicate with the newcomer. The nodes have no privileges or restrictions than other nodes; they are all part of the network and contribute to its expansion and efficiency, being all equal.

Whenever the number of its nodes grows, the network changes its shape and the routes keep on improving, automatically removing unnecessary links using disjoint routes. Redundant routes are removed in order to free memory for non-redundant ones. Keeping redundant routes in the kernel routing table isn't optimal, because if one of the routes fail there is a high probability that all the other redundant routes will fail too.

The usual protocols and algorithms for the dynamic routing are usually used to create small and medium networks, like OSPF, RIP or BGP, and use different algorithms to find the best way to reach a node in a network. These protocols require a relevant CPU and memory consumption, and it is for this reason that the Internet routers are often specially dedicated computers. It would be impossible to adopt one of these protocols to create and maintain a very big mesh network.

Hierarchical topology

Routing tables tend to grow big on huge networks. Even if we store just one route to reach one node and even if this route costs one byte, we would need 1GB of memory for a network composed by 10^9 nodes, the current Internet.

For this reason, is necessary to structure the network in a convenient topology. Netsukuku adopts a hierarchical structure. 256 nodes are grouped inside a gnode (group node), 256 gnodes are grouped in a single ggnode (group of group nodes), 256 ggnodes are grouped in a single gggnode, and so on.

Advantages on hierarchical topology are described in the main documentation of the Netsukuku theory.

QSPN

The current version of the protocol is QSPNv2.

QSPN assumes two things:

- Mobile nodes aren't supported by the current theory. This is based on the fact that WiFi mesh routers don't change their location very often. This is a valid

assumption however; usually the routes are plugged to external antennas or mounted on the roofs.
- The network isn't updated quickly. Several minutes may be required before all the nodes become aware of a change in the network, such as new joined nodes, more efficient routes have become available. However, when a node joins the network it can reach all the other nodes from the very first instant using the routes of his neighbours.

The routing algorithm must be capable of finding the routes without overloading the network of the resources of the node. But since QSPN alone wouldn't be capable of handling the whole network because it would still require too much memory, it is necessary to structure the network in a convenient topology.

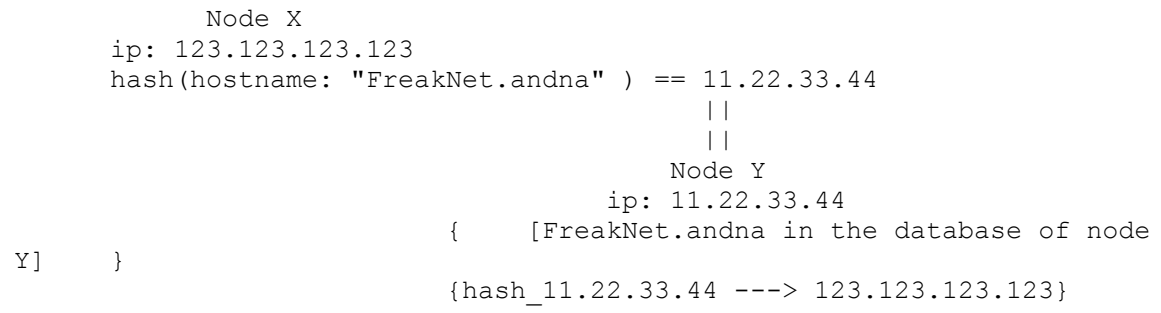
Since in each level there is a maximum of 256 (g)nodes, QSPN will always operate on a maximum of 256 (g)nodes. We just need to be sure that the algorithm works as expected on every case of a graph composed by less or equal to 256 nodes.

A Netsukuku Domain Name Architecture

The **A Netsukuku Domain Name Architecture** (**ANDNA**) is a distributed system of naming and management of names, that plays the same role the DNS does. The ANDNA database is included in the Netsukuku system, so each node includes such database that, at worst scenario may occupy 355 kilobytes of memory.

Simplifying, ANDNA works as follows:

To resolve a symbolic name the host applies a function Hash on behalf. The Hash function returns an address that the host contacts asking for the resolution generated by the hash. The contacted node receives a request, searches in its ANDNA database for the address associated with the name and returns to the applicant host. Recording is in a similar way. For example, let's suppose that the node X should record the address *FreakNet.andna*, X calculates the hash name and obtains the address 11.22.33.44 associated with node Y. The node X contacts Y requiring registration for 11.22.33.44 hash as its own. Y stores the request in its database and any request for resolution of 11.22.33.44 hash, will answer with the X address.



The protocol is obviously more complex as the system provides a public/private key to authenticate the hosts and prevent unauthorized changes to ANDNA database. Furthermore, the protocol provides redundancy database to make the protocol resistant to any failures and also provides for the migration of the database if the network topology changes. The protocol does not provide for the possibility of revoking a symbolic name, this after a certain period of inactivity (currently 3 days) is simply deleted from the database. The protocol also prevents a single host to record an excessive number of symbolic names (at present 256 names) in order to prevent spammers to store a high number of terms commonly used or to perform actions of cybersquatting.

Use

The goal of Netsukuku network is the realization of an infrastructure that not having to rely on the usual Internet infrastructure is more economical and independent, allowing access to network users unable to pay a regular fee to ISPs. The designers of the network think they can achieve this parallel network relying heavily on wireless networks that have a range of several kilometers.

Another possible use of the Netsukuku network would be the realization of cellular networks without the input of telephone operators. This application starts from the consideration that the network algorithms require reduced resources and therefore can easily run on existing phones, though for real applications in this regard are at present only theoretical.

Related items

- Wireless community network
- Mesh Network
- I2P - The Anonymous Network
- Tor (anonymity network)
- Digital divide
- Anonymous P2P
- Freenet

Chapter 12

Grid Computing

Grid computing is a term referring to the combination of computer resources from multiple administrative domains to reach a common goal. The **grid** can be thought of as a distributed system with non-interactive workloads that involve a large number of files. What distinguishes grid computing from conventional high performance computing systems such as cluster computing is that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed. Although a grid can be dedicated to a specialized application, it is more common that a single grid will be used for a variety of different purposes. Grids are often constructed with the aid of general-purpose grid software libraries known as middleware.

Grid size can vary by a considerable amount. Grids are a form of distributed computing whereby a “super virtual computer” is composed of many networked loosely coupled computers acting together to perform very large tasks. Furthermore, “distributed” or “grid” computing, in general, is a special type of parallel computing that relies on complete computers (with onboard CPUs, storage, power supplies, network interfaces, etc.) connected to a network (private, public or the Internet) by a conventional network interface, such as Ethernet. This is in contrast to the traditional notion of a supercomputer, which has many processors connected by a local high-speed computer bus.

Overview

Grid computing combines computers from multiple administrative domains to reach a common goal, to solve a single task, and may then disappear just as quickly.

One of the main strategies of grid computing is to use middleware to divide and apportion pieces of a program among several computers, sometimes up to many thousands. Grid computing involves computation in a distributed fashion, which may also involve the aggregation of large-scale cluster computing-based systems.

The size of a grid may vary from small—confined to a network of computer workstations within a corporation, for example—to large, public collaborations across many companies and networks. "The notion of a confined grid may also be known as an intra-

nodes cooperation whilst the notion of a larger, wider grid may thus refer to an inter-nodes cooperation".

Grids are a form of distributed computing whereby a “super virtual computer” is composed of many networked loosely coupled computers acting together to perform very large tasks. This technology has been applied to computationally intensive scientific, mathematical, and academic problems through volunteer computing, and it is used in commercial enterprises for such diverse applications as drug discovery, economic forecasting, seismic analysis, and back office data processing in support for e-commerce and Web services.

Comparison of grids and conventional supercomputers

“Distributed” or “grid” computing in general is a special type of parallel computing that relies on complete computers (with onboard CPUs, storage, power supplies, network interfaces, etc.) connected to a network (private, public or the Internet) by a conventional network interface, such as Ethernet. This is in contrast to the traditional notion of a supercomputer, which has many processors connected by a local high-speed computer bus.

The primary advantage of distributed computing is that each node can be purchased as commodity hardware, which, when combined, can produce a similar computing resource as multiprocessor supercomputer, but at a lower cost. This is due to the economies of scale of producing commodity hardware, compared to the lower efficiency of designing and constructing a small number of custom supercomputers. The primary performance disadvantage is that the various processors and local storage areas do not have high-speed connections. This arrangement is thus well-suited to applications in which multiple parallel computations can take place independently, without the need to communicate intermediate results between processors. The high-end scalability of geographically dispersed grids is generally favorable, due to the low need for connectivity between nodes relative to the capacity of the public Internet.

There are also some differences in programming and deployment. It can be costly and difficult to write programs that can run in the environment of a supercomputer, which may have a custom operating system, or require the program to address concurrency issues. If a problem can be adequately parallelized, a “thin” layer of “grid” infrastructure can allow conventional, standalone programs, given a different part of the same problem, to run on multiple machines. This makes it possible to write and debug on a single conventional machine, and eliminates complications due to multiple instances of the same program running in the same shared memory and storage space at the same time.

Design considerations and variations

One feature of distributed grids is that they can be formed from computing resources belonging to multiple individuals or organizations (known as multiple administrative

domains). This can facilitate commercial transactions, as in utility computing, or make it easier to assemble volunteer computing networks.

One disadvantage of this feature is that the computers which are actually performing the calculations might not be entirely trustworthy. The designers of the system must thus introduce measures to prevent malfunctions or malicious participants from producing false, misleading, or erroneous results, and from using the system as an attack vector. This often involves assigning work randomly to different nodes (presumably with different owners) and checking that at least two different nodes report the same answer for a given work unit. Discrepancies would identify malfunctioning and malicious nodes.

Due to the lack of central control over the hardware, there is no way to guarantee that nodes will not drop out of the network at random times. Some nodes (like laptops or dialup Internet customers) may also be available for computation but not network communications for unpredictable periods. These variations can be accommodated by assigning large work units (thus reducing the need for continuous network connectivity) and reassigning work units when a given node fails to report its results in expected time.

The impacts of trust and availability on performance and development difficulty can influence the choice of whether to deploy onto a dedicated computer cluster, to idle machines internal to the developing organization, or to an open external network of volunteers or contractors. In many cases, the participating nodes must trust the central system not to abuse the access that is being granted, by interfering with the operation of other programs, mangling stored information, transmitting private data, or creating new security holes. Other systems employ measures to reduce the amount of trust "client" nodes must place in the central system such as placing applications in virtual machines.
___ Public systems or those crossing administrative domains (including different departments in the same organization) often result in the need to run on heterogeneous systems, using different operating systems and hardware architectures. With many languages, there is a trade off between investment in software development and the number of platforms that can be supported (and thus the size of the resulting network). Cross-platform languages can reduce the need to make this trade off, though potentially at the expense of high performance on any given node (due to run-time interpretation or lack of optimization for the particular platform).

Various middleware projects have created generic infrastructure to allow diverse scientific and commercial projects to harness a particular associated grid or for the purpose of setting up new grids.

In fact, the middleware can be seen as a layer between the hardware and the software. On top of the middleware, a number of technical areas have to be considered, and these may or may not be middleware independent. Example areas include SLA management, Trust and Security, Virtual organization management, License Management, Portals and Data Management. These technical areas may be taken care of in a commercial solution, though the cutting edge of each area is often found within specific research projects examining the field.

Market segmentation of the grid computing market

For the segmentation of the grid computing market, two perspectives need to be considered: the provider side and the user side:

The provider side

The overall grid market comprises several specific markets. These are the grid middleware market, the market for grid-enabled applications, the utility computing market, and the software-as-a-service (SaaS) market.

Grid middleware is a specific software product, which enables the sharing of heterogeneous resources, and Virtual Organizations. It is installed and integrated into the existing infrastructure of the involved company or companies, and provides a special layer placed among the heterogeneous infrastructure and the specific user applications. Major grid middlewares are Globus Toolkit, gLite, and UNICORE.

Utility computing is referred to as the provision of grid computing and applications as service either as an open grid utility or as a hosting solution for one organization or a VO. Major players in the utility computing market are Sun Microsystems, IBM, and HP.

Grid-enabled applications are specific software applications that can utilize grid infrastructure. This is made possible by the use of grid middleware, as pointed out above.

Software as a service (SaaS) is "software that is owned, delivered and managed remotely by one or more providers." (Gartner 2007) Additionally, SaaS applications are based on a single set of common code and data definitions. They are consumed in a one-to-many model, and SaaS uses a Pay As You Go (PAYG) model or a subscription model that is based on usage. Providers of SaaS do not necessarily own the computing resources themselves, which are required to run their SaaS. Therefore, SaaS providers may draw upon the utility computing market. The utility computing market provides computing resources for SaaS providers.

The user side

For companies on the demand or user side of the grid computing market, the different segments have significant implications for their IT deployment strategy. The IT deployment strategy as well as the type of IT investments made are relevant aspects for potential grid users and play an important role for grid adoption.

CPU scavenging

CPU-scavenging, **cycle-scavenging**, **cycle stealing**, or **shared computing** creates a "grid" from the unused resources in a network of participants (whether worldwide or internal to an organization). Typically this technique uses desktop computer instruction

cycles that would otherwise be wasted at night, during lunch, or even in the scattered seconds throughout the day when the computer is waiting for user input or slow devices.

Many Volunteer computing projects, such as BOINC, use the CPU scavenging model.

In practice, participating computers also donate some supporting amount of disk storage space, RAM, and network bandwidth, in addition to raw CPU power. Heat produced by CPU power in rooms with many computers can be used for fine heating premises. Since nodes are likely to go "offline" from time to time, as their owners use their resources for their primary purpose, this model must be designed to handle such contingencies.

History

The term *grid computing* originated in the early 1990s as a metaphor for making computer power as easy to access as an electric power grid in Ian Foster's and Carl Kesselman's seminal work, "The Grid: Blueprint for a new computing infrastructure" (2004).

CPU scavenging and volunteer computing were popularized beginning in 1997 by distributed.net and later in 1999 by SETI@home to harness the power of networked PCs worldwide, in order to solve CPU-intensive research problems.

The ideas of the grid (including those from distributed computing, object-oriented programming, and Web services) were brought together by Ian Foster, Carl Kesselman, and Steve Tuecke, widely regarded as the "fathers of the grid". They led the effort to create the Globus Toolkit incorporating not just computation management but also storage management, security provisioning, data movement, monitoring, and a toolkit for developing additional services based on the same infrastructure, including agreement negotiation, notification mechanisms, trigger services, and information aggregation. While the Globus Toolkit remains the de facto standard for building grid solutions, a number of other tools have been built that answer some subset of services needed to create an enterprise or global grid.

In 2007 the term cloud computing came into popularity, which is conceptually similar to the canonical Foster definition of grid computing (in terms of computing resources being consumed as electricity is from the power grid). Indeed, grid computing is often (but not always) associated with the delivery of cloud computing systems as exemplified by the AppLogic system from 3tera.

Fastest virtual supercomputers

- BOINC – 5.634 PFLOPS as of April 4th 2011.
- Folding@Home – 5 PFLOPS, as of March 17, 2009
- As of April 2010, MilkyWay@Home computes at over 1.6 PFLOPS, with a large amount of this work coming from GPUs.
- As of April 2010, SETI@Home computes data averages more than 730 TFLOPS.

- As of April 2010, Einstein@Home is crunching more than 210 TFLOPS.
- As of April 2010, GIMPS is sustaining 44 TFLOPS.

Current projects and applications

Grids computing offer a way to solve Grand Challenge problems such as protein folding, financial modeling, earthquake simulation, and climate/weather modeling. Grids offer a way of using the information technology resources optimally inside an organization. They also provide a means for offering information technology as a utility for commercial and noncommercial clients, with those clients paying only for what they use, as with electricity or water.

Grid computing is being applied by the National Science Foundation's National Technology Grid, NASA's Information Power Grid, Pratt & Whitney, Bristol-Myers Squibb Co., and American Express.

One of the most famous cycle-scavenging networks is SETI@home, which was using more than 3 million computers to achieve 23.37 sustained teraflops (979 lifetime teraflops) as of September 2001.

As of August 2009 Folding@home achieves more than 4 petaflops on over 350,000 machines.

The European Union has been a major proponent of grid computing. Many projects have been funded through the framework programme of the European Commission. Many of the projects are highlighted below, but two deserve special mention: BEinGRID and Enabling Grids for E-sciencE.

BEinGRID (Business Experiments in Grid) is a research project partly funded by the European commission as an Integrated Project under the Sixth Framework Programme (FP6) sponsorship program. Started on June 1, 2006, the project will run 42 months, until November 2009. The project is coordinated by Atos Origin. According to the project fact sheet, their mission is “to establish effective routes to foster the adoption of grid computing across the EU and to stimulate research into innovative business models using Grid technologies”. To extract best practice and common themes from the experimental implementations, two groups of consultants are analyzing a series of pilots, one technical, one business. The project is significant not only for its long duration, but also for its budget, which at 24.8 million Euros, is the largest of any FP6 integrated project. Of this, 15.7 million is provided by the European commission and the remainder by its 98 contributing partner companies.

The Enabling Grids for E-sciencE project, which is based in the European Union and includes sites in Asia and the United States, is a follow-up project to the European DataGrid (EDG) and is arguably the largest computing grid on the planet. This, along with the LHC Computing Grid (LCG), has been developed to support the experiments using the CERN Large Hadron Collider. The LCG project is driven by CERN's need to

handle huge amounts of data, where storage rates of several gigabytes per second (10 petabytes per year) are required. A list of active sites participating within LCG can be found online as can real time monitoring of the EGEE infrastructure. The relevant software and documentation is also publicly accessible. There is speculation that dedicated fiber optic links, such as those installed by CERN to address the LCG's data-intensive needs, may one day be available to home users thereby providing internet services at speeds up to 10,000 times faster than a traditional broadband connection.

Another well-known project is distributed.net, which was started in 1997 and has run a number of successful projects in its history.

The NASA Advanced Supercomputing facility (NAS) has run genetic algorithms using the Condor cycle scavenger running on about 350 Sun and SGI workstations.

In 2001, United Devices operated the United Devices Cancer Research Project based on its Grid MP product, which cycle-scavenges on volunteer PCs connected to the Internet. The project ran on about 3.1 million machines before its close in 2007.

As of 2011, over 6.2 million machines running the open-source Berkeley Open Infrastructure for Network Computing (BOINC) platform are members of the World Community Grid, which tops the processing power of the current fastest supercomputer system (China's Tianhe-I).

Definitions

Today there are many definitions of *grid computing*:

- In his article “What is the Grid? A Three Point Checklist”, Ian Foster lists these primary attributes:
    - Computing resources are not administered centrally.
    - Open standards are used.
    - Nontrivial quality of service is achieved.

- Plaszczak/Wellner define grid technology as "the technology that enables resource virtualization, on-demand provisioning, and service (resource) sharing between organizations."
- IBM defines grid computing as “the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and a vast array of other computing resources over the Internet. A grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across ‘multiple’ administrative domains based on their (resources) availability, capacity, performance, cost and users' quality-of-service requirements”.
- An earlier example of the notion of computing as utility was in 1965 by MIT's Fernando Corbató. Corbató and the other designers of the Multics operating

system envisioned a computer facility operating “like a power company or water company”.

- Buyya/Venugopal define grid as "a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements".
- CERN, one of the largest users of grid technology, talk of **The Grid**: “a service for sharing computer power and data storage capacity over the Internet.”

Grids can be categorized with a three stage model of departmental grids, enterprise grids and global grids. These correspond to a firm initially utilising resources within a single group i.e. an engineering department connecting desktop machines, clusters and equipment. This progresses to enterprise grids where nontechnical staff's computing resources can be used for cycle-stealing and storage. A global grid is a connection of enterprise and departmental grids that can be used in a commercial or collaborative manner.

Chapter 13

D-Grid

The **D-Grid Initiative** (German Grid Initiative) builds a sustainable grid infrastructure for education and research (e-Science) in Germany. The D-Grid infrastructure will help to establish methods of e-Science in three core areas:

- Grid Computing
- Knowledge Management
- e-Learning

D-Grid started September 1, 2005 and consists of six Community Projects and an integration project (DGI) as well as several partner projects.

Integration Project

The D-Grid integration project is a part of the D-Grid Initiative and has to integrate all developments from the different community projects in one common D-Grid platform. The D-Grid integration project will act as a Grid resource and service provider for the science community in Germany. The project office is located at the Institute for Scientific Computing (IWR) at Forschungszentrum Karlsruhe. The resources to ensure a sustainable Grid infrastructure are provided by four work packages:

- Work Package 1: **D-Grid Base-Software**. The major task of this work package is to provide several different middleware packages. These are the Globus Toolkit, UNICORE, LCG/gLite, GridSphere and the Grid Application Toolkit (GAT). The community projects linked together in the D-Grid integration project are supported during the installation, operation and if needed and possible the customisation of the Base-Software.
- Work Package 2: **Deployment and operation of the D-Grid infrastructure**. Work package 2 builds up a Core-D-Grid. This will be used as a prototype to test the operational functionality of the system. This work package also deals with the challenges of Monitoring, Accounting and Billing of the Grid resources.
- Work Package 3: **Networks and Security**. The network infrastructure in D-Grid is based on the DFN Wissenschaftsnetz X-WiN. Work package 3 will provide extensions to the existing network infrastructure according to the needs of Grid

middleware used in D-Grid. Further tasks are to build an AA-Infrastructure in D-Grid, develop firewall concepts for Grid environments and set up Grid specific CERT services.

- Work Package 4: **D-Grid project office**. The work package is responsible for the integration of the deliverables from the Grid integration project and the deliverables from the different community projects in one common D-Grid platform. Work package 4 also deals with the challenge of archiving sustainability in D-Grid and grid-based e-Science systems generally.

Communities

Currently there are six community projects participating the D-Grid Initiative:

AstroGrid-D

AstroGrid-D, also referred to as the German Astronomy Community Grid (GACG), is a joint research project of thirteen astronomical institutes and grid-oriented computer science groups, supported by supercomputing centers. The main objective of AstroGrid-D is the integration of German research facilities into a unified nationwide research infrastructure in the field of astronomy. The goal is to improve the efficiency and usability of hardware and software resources including computer clusters, astronomical data archives, and observational facilities such as robotic telescopes. AstroGrid-D supports the standards of the International Virtual Observatory Alliance (IVOA) and cooperates closely with international projects on grid development.

AstroGrid-D is managed by the Astrophysical Institute Potsdam (AIP).

C3-Grid

At the Collaborative Climate Community Data and Processing Grid (C3-Grid) scientific researchers are trying to understand the earth system including their subsystems like oceans, atmosphere and biosphere. For the last decades the amount of data has increased enormously in the field of climate research. On the one hand, due to rapid rise in computing power scientists are now able to use models with higher resolution and perform long term simulations. The scientists are able to couple models for the mentioned subsystems in complex cumulative simulations producing petabytes of output which is collected in distributed data archives. On the other hand, monitoring the earth with satellites results in a second huge data stream for climate research. Up to now, no uniform access to these distributed data is available what creates a bottleneck for the scientific research. The C3-Grid proposes to link these distributed data archives.

The management of C3-Grid has the Alfred Wegener Institute for Polar and Marine Research (AWI) in Bremerhaven.

GDI-Grid

The GDI-Grid ("Geodateninfrastrukturen-Grid" - "Spatial Data Infrastructure Grid") project focuses on solutions for efficient integration and processing of geodata based on GIS and SDI technologies. The project will integrate GDI and Grid technologies in a working GDI-Grid infrastructure and thus demonstrate the complementarity of both fields of science. Distributed geospatial data—currently accessed via standardized GIS and SDI services—will build the basis for this endeavour. This data basis can be put to more use by processing it and merging it with other data, creating standards-based, multi-functional generic SDI services.

The project focuses on data, models, services and workflows for spatial data infrastructures. Services for integration, processing and management of spatial data are to be developed and implemented within the D-Grid infrastructure. A proof of concept will be given using a number of representative scenarios such as emergency routing for disaster management, flood simulation and sound propagation simulation.

The project is managed by the University of Hanover at the Regional Computing Center for Lower Saxony (RRZN).

HEP-Grid

The HEP-Grid Project has its focus on High Energy, Nuclear as well as Astroparticle Physics. The main task of HEP-Grid is to optimize the data analysis using distributed computing and storage resources. The project developments are important extensions to the grid middleware from the Enabling Grids for E-scienceE (EGEE) and LHC Computing Grid (LCG) projects. They provide significant improvements for data analysis of experiments currently taking data and for those planned in the future at the Large Hadron Collider (LHC) or the proposed International Linear Collider (ILC).

For the HEP-Grid Deutsches Elektronen Synchrotron (DESY), "German Electron Synchrotron") in Hamburg collaborates with eight German research facilities and universities and a set of associate partners.

InGrid

InGrid is a community project in the field of Grid computing in engineering sciences. InGrid aims to enable engineering projects for grid-based applications and allow for the common, efficient use of common compute and software resources. The flexible use of grid technologies will combine the competences in modeling, simulation and optimization.

Five typical applications (foundry technologies, metal forming technologies, groundwater flow and transport, turbine simulation and fluid-structure interaction) are considered as showcases in order to cover the three central areas of computationally intensive engineering applications, that are coupled multi-scale problems, coupled multi-discipline

problems, and distributed simulation-based optimization. In particular adaptive and scalable process models and Grid based runtime environments for these tasks are developed.

Engineering research is inherently application and industry oriented. The support of virtual prototyping and the optimization of scientific engineering operational sequences is therefore an emphasis of the project. The project management for InGrid is provided by the High Performance Computing Center (Höchstleistungsrechenzentrum) Stuttgart (HLRS) of the University of Stuttgart.

MediGRID

The joint project MediGRID unifies well known research institutes in the area of **medicine, biomedical informatics and life sciences** into a consortium. Numerous associated partners from industry, healthcare and research facilities ensure a broad representation of these communities.

The main goal of MediGRID is the development of a Grid middleware integration platform enabling eScience services for biomedical life science. Therefore the consortium allocated the tasks in different modules. The four methodological modules (**middleware, ontology, resource fusion and eScience**) plan to incrementally develop and provide a Grid infrastructure while taking into account the need of the biomedical users. The user communities are represented in three research modules for **biomedical informatics, image processing and clinical research**.

SuGI

SuGI - Sustainable Grid Infrastructure - is a gap project of the German Grid Initiative. Its major task is to disseminate the knowledge of grid technology and to enhance its use. Thus, SuGI addresses all academic computing centers as well as enterprises, which still have not adopted grid technology. They will be supported in providing grid resources and services.

During this project, research experiences gained in the D-Grid projects will be made available to these institutions. Thus, SuGI offers own training courses; attends to external courses, create video and audio recordings and provide these online to the D-Grid communities via a scaling training infrastructure (SuGI-Portal). Further on, SuGI develops training systems for grid middleware, contributes to a simplification of installation and servicing procedures, and works on the development and evaluation of legal and organizational structures.

TextGrid

While grid technologies have first been developed for the natural and the life sciences, there are exciting opportunities for deploying grid technologies and e-Science concepts in

other areas as well. TextGrid is a grid project in the humanities, and thereby contributes to the emerging e-Humanities.

It aims to create a grid-based infrastructure for the collaborative editing, annotation, analysis and publication of specialist texts for researchers in philology, linguistics, and related fields. In addition to providing a comprehensive toolset, the project establishes an open platform for other projects to plug into the TextGrid.

ValueGrids

ValueGrids develops an integrated concept and a set of tools for service level management in service value networks. This will enable providers of Software-as-a-Service solutions to utilize grid infrastructures and leverage the German national grid infrastructure.

The ValueGrids project partners are: SAP AG (Coordinator), Conemis AG, IBM Deutschland Research & Development GmbH, University of Freiburg and the Karlsruhe Institute of Technology.

Partner Projects

Several Partner Projects are involved in the D-Grid Initiative.

WISENT

The e-Science project WISENT ("Wissensnetz Energiemeteorologie") has the aim to optimize the cooperation of scientific organizations in the field of Energy Meteorology employing Grid technologies. The main focus of scientific research within Energy Meteorology is the investigation of the influence of weather and climate on transformation, transport, and utilisation of energy.

Funding

On the basis of the D-Grid Initiative more than 100 German research facilities are funded with about 20 billion Euros for a period of 3 years by the German Federal Ministry of Education and Research.

Chapter 14

Dynamic Infrastructure

Dynamic Infrastructure is an information technology paradigm concerning the design of data centers so that the underlying hardware and software can respond dynamically to changing levels of demand in more fundamental and efficient ways than before. The paradigm is also known as *Infrastructure 2.0* and *Next Generation Data Center*.

Top tier vendors promoting dynamic infrastructures include IBM, Microsoft, Sun, Fujitsu, HP and Dell.

The basic premise of Dynamic Infrastructures is to leverage pooled IT resources to provide flexible IT capacity, enabling the seamless, real-time allocation of IT resources in line with demand from business processes. This is achieved by using server virtualization technology to pool computing resources wherever possible, and allocating these resources on-demand using automated tools. This allows for load balancing and is a more efficient approach than keeping massive computing resources in reserve to run tasks that take place, for example, once a month, but are otherwise under-utilized.

Early examples of server-level Dynamic Infrastructures are the FlexFrame for SAP and FlexFrame for Oracle solutions introduced by Fujitsu Siemens Computers (now Fujitsu) in 2003. The FlexFrame approach is to dynamically assign servers to applications on demand, leveling peaks and enabling organizations to maximize the benefit from their IT investments.

Enterprises switching to Dynamic Infrastructures can also reduce costs, improve quality-of-service and make more efficient use of energy through reducing the number of standby or under-utilized machines in their data centers. Instead of the hot spare principle of keeping second servers on standby to replace all production machines in contingencies for hardware- and software-related failures, Dynamic Infrastructures provide for failover from a smaller pool of spare machines. By reducing redundant capacity, organizations are enabled to make more efficient use of their IT budgets and devote greater proportions of their budget to physical and virtual production servers.

Dynamic Infrastructures may also be used to provide security and data protection when workloads are moved during migrations, provisioning, enhancing performance or building co-location facilities.

Potential benefits of Dynamic Infrastructures include enhancing performance, scalability, system availability and uptime, increasing server utilization and the ability to perform routine maintenance on either physical or virtual systems all while minimizing interruption to business operations and reducing cost for IT. Dynamic Infrastructures also provide the fundamental business continuity and high availability requirements to facilitate cloud or grid computing.

Fujitsu's definition: "Dynamic Infrastructures enable customers to assign IT resources dynamically to services as required and to choose sourcing models which best fit their businesses. This brings IT flexibility and efficiency to the next level."

IBM's definition: “A dynamic infrastructure integrates business and IT assets and aligns them with the overall goals of the business while taking a smarter, new and more streamlined approach to helping improve service, reduce cost, and manage risk.”

For networking companies, Infrastructure 2.0 refers to the ability of networks to keep up with the movement and scale requirements of new enterprise IT initiatives, especially virtualization and cloud computing. According to companies like Cisco, F5 Networks and Infoblox, network automation and connectivity intelligence between networks, applications and endpoints will be required to reap the full benefits of virtualization and many types of cloud computing. This will require network management and infrastructure to be consolidated, enabling higher levels of dynamic control and connectivity between networks, systems and endpoints.

Need for a holistic approach

Even in the face of global uncertainty, it is the infrastructure that continues to enable commerce and communications – the roads, networks, utilities, and technologies connecting and differentiating organizations, competitors and customers. The need therefore, is for a new type of infrastructure that:

- Enables visibility, control and automation across all business and IT assets
- Is highly optimized to achieve more with less
- Addresses the information challenge
- Leverages flexible sourcing like clouds
- Manages and mitigates risks

Organizations need an infrastructure that can propel them forward — not hold them back. Until now, many organizations have thought of physical infrastructure and IT infrastructure as separate. This meant, for example, that airports, roadways, buildings, power plants, and oil wells were managed in one way, while datacenters, PCs, cell phones, routers, and broadband devices were managed quite differently.

To succeed in today's world of instrumented, interconnected, and intelligent assets, a new approach is needed. Now, the infrastructure of atoms and the infrastructure of bits are merging into an intelligent, global, dynamic infrastructure. This convergence of business and IT assets requires an infrastructure that can measure and manage the lifecycle of assets that exist beyond the data center, throughout an organization's entire facilities as well as between one organization and another. The range of this approach is broader than ever before, and its effect on organizations is equally far-reaching.

Benefits of having dynamic infrastructures

Dynamic infrastructures take advantage of intelligence gained across the network. By design, every dynamic infrastructure is service-oriented and focused on supporting and enabling the end users in a highly responsive way. It can utilize alternative sourcing approaches, like cloud computing to deliver new services with agility and speed.

Global organizations already have the foundation for a dynamic infrastructure that will bring together the business and IT infrastructure to create new possibilities. For example:

- Transportation companies can optimize their vehicles' routes leveraging GPS and traffic information.
- Facilities organizations can secure access to locations and track the movement of assets by leveraging RFID technology.
- Production environments can monitor and manage presses, valves and assembly equipment through embedded electronics.
- Technology systems can be optimized for energy efficiency, managing spikes in demand, and ensuring disaster recovery readiness.
- Communications companies can better monitor usage by location, user or function, and optimize routing to enhance user experience.
- Utility companies can reduce energy usage with a "smart grid."

Virtualized applications can reduce the cost of testing, packaging and supporting an application by 60%, and they reduced overall TCO by 5% to 7% in our model. – Source: Gartner – "TCO of Traditional Software Distribution vs. Application Virtualization" / Michael A Silver, Terrence Cosgrove, Mark A Margevicious, Brian Gammage / 16 April 2008

While green issues are a primary driver in 10% of current data center outsourcing and hosting initiatives, cost reductions initiatives are a driver 47% of the time and are now aligned well with green goals. Combining the two means that at least 57% of data center outsourcing and hosting initiatives are driven by green. – Source: Gartner – "Green IT Services as a Catalyst for Cost Optimization." / Kurt Potter / 4 December 2008

"By 2013, more than 50% of midsize organizations and more than 75% of large enterprises will implement layered recovery architectures." – Source: Gartner – "Predicts 2009: Business Continuity Management Juggles Standardization, Cost and Outsourcing

Risk"). / Roberta J Witty, John P Morency, Dave Russell, Donna Scott, Rober Desisto / 28 January 2009

The key to a business and IT infrastructure that is "dynamic" is leveraging technologies, service delivery and acquisition models that optimize the infrastructure for efficiency and flexibility while transforming management to an automated service delivery and management model.

Chapter 15

Data Format Description Language

Data Format Description Language (DFDL, often pronounced *daff-o-dil*) is a modeling language from the Open Grid Forum for describing general text and binary data. A DFDL model or schema allows any text or binary data to be read (or "parsed") from its native format and to be presented as an instance of an information set. The same DFDL schema also allows data to be taken from an instance of an information set and written out (or "serialized") to its native format.

DFDL achieves this by building upon the facilities of W3C XML Schema 1.0. A subset of XML Schema is used, enough to enable the modeling of non-XML data. One of the results of this is that is very easy to use DFDL to convert general text and binary data, via a DFDL information set, into a corresponding XML document.

It is important to note that DFDL is *descriptive* and not *prescriptive*. DFDL is not a data format, nor does it impose the use of any particular data format. DFDL allows an application to design an appropriate data representation according to its requirements, and for that format to be described in a standard way so that multiple programs can directly interchange the data.

History

DFDL was created in response to a need for grid APIs to be able to understand data regardless of source. A language was needed capable of modeling a wide variety of existing text and binary data formats. A working group was established at the Global Grid Forum (which later became the Open Grid Forum) in 2003 to create a specification for such a language.

A decision was made early on to base the language on a subset of W3C XML Schema, using <xs:appinfo> annotations to carry the extra information necessary to describe non-XML physical representations. This is an established approach that is already being used today in commercial systems. DFDL takes this approach and evolves it into an open standard capable of describing many text or binary data formats.

Work continued on the specification, culminating in the publication of DFDL 1.0 as an

OGF Proposed Recommendation in January 2011. A summary of DFDL and its features is available at the OGF site.

Implementations of DFDL processors that can parse and serialize data using DFDL schemas are in progress.

Example

Take as an example the following text data stream which gives the name, age and location of a person:

```
Joe Bloggs,46,Hampshire,England
```

The logical model for this data can be described by the following fragment of an XML Schema document. The order, names, types and cardinality of the fields are expressed by the XML schema model.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" ...>

<xs:complexType name="person_type">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="age" type="xs:short"/>
    <xs:element name="county" type="xs:string"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:schema>
```

To additionally model the physical representation of the data stream, DFDL augments the XML schema fragment with annotations on the xs:element and xs:sequence objects, as follows:

```
<xs:schema xmlns:dfdl="http://www.ogf.org/dfdl/dfdl-1.0/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" ...>

<xs:complexType name="person_type">
  <xs:sequence>
    <xs:annotation><xs:appinfo source="http://www.ogf.org/dfdl/">
        <dfdl:sequence encoding="ASCII" sequenceKind="ordered"
                       separator="," separatorType="infix"
separatorPolicy="required"/>
    </xs:appinfo></xs:annotation>
    <xs:element name="name" type="xs:string">
      <xs:annotation><xs:appinfo source="http://www.ogf.org/dfdl/">
        <dfdl:element lengthKind="delimited" encoding="ASCII"/>
      </xs:appinfo></xs:annotation>
    </xs:element>
    <xs:element name="age" type="xs:short">
      <xs:annotation><xs:appinfo source="http://www.ogf.org/dfdl/">
```

```
        <dfdl:element representation="text" lengthKind="delimited"
encoding="ASCII"
                      textNumberRep="standard" textNumberPattern="#0"
textNumberBase="10"/>
      </xs:appinfo></xs:annotation>
    </xs:element>
    <xs:element name="county" type="xs:string">
      <xs:annotation><xs:appinfo source="http://www.ogf.org/dfdl/">
        <dfdl:element lengthKind="delimited" encoding="ASCII"/>
      </xs:appinfo></xs:annotation>
    </xs:element>
    <xs:element name="country" type="xs:string">
      <xs:annotation><xs:appinfo source="http://www.ogf.org/dfdl/">
        <dfdl:element lengthKind="delimited" encoding="ASCII"/>
      </xs:appinfo></xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

</xs:schema>
```

The property attributes on these DFDL annotations express that the data are represented in an ASCII text format with fields being of variable length and delimited by commas.

An alternative, more compact syntax is also provided, where DFDL properties are carried as non-native attributes on the XML Schema objects themselves.

```
<xs:schema xmlns:dfdl="http://www.ogf.org/dfdl/dfdl-1.0/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" ...>

<xs:complexType name="person_type">
  <xs:sequence dfdl:encoding="ASCII" dfdl:sequenceKind="ordered"
               dfdl:separator="," dfdl:separatorType="infix"
dfdl:separatorPolicy="required">
    <xs:element name="name" type="xs:string"
                dfdl:lengthKind="delimited" dfdl:encoding="ASCII"/>
    <xs:element name="age" type="xs:short"
                dfdl:representation="text" dfdl:lengthKind="delimited"
dfdl:encoding="ASCII"
                dfdl:textNumberRep="standard"
dfdl:textNumberPattern="##0" dfdl:textNumberBase="10"/>
    <xs:element name="county" type="xs:string"
                dfdl:lengthKind="delimited" dfdl:encoding="ASCII"/>
    <xs:element name="country" type="xs:string"
                dfdl:lengthKind="delimited" dfdl:encoding="ASCII"/>
  </xs:sequence>
</xs:complexType>

</xs:schema>
```

Features

The goal of DFDL is to provide a rich modeling language capable of representing any text or binary data format. The 1.0 release is a major step towards this goal. The capability includes support for:

- Language structures such as COBOL, C and PL/1
- Industry standards such as CSV, SWIFT, FIX, HL7, X12, HL7, HIPAA, EDIFACT, ISO8583
- Data delimited by text or binary markup
- Physical data types including text strings, text numbers, binary two's complement integers, BCD, mainframe zoned and packed decimals, IEEE and mainframe floats, text and binary calendars, text and binary Booleans
- Any encoding and endian-ness
- Bi-directional text
- Bit data of arbitrary length
- Pattern languages for text numbers and calendars
- Ordered and unordered content
- Default values on parsing and serializing
- Nil values capability for handling out-of-band data
- A built-in expression language including variables to model dynamic data
- Mechanisms to resolve choices and optionality
- Fixed and variable arrays
- Hiding elements in the data from the information set
- Calculating element values for the information set
- Validation to XML Schema 1.0 rules
- A scoping mechanism that allows common property values to be applied at multiple annotation points

Future releases are anticipated in which it is hoped to include support for:

- Direct access by offset
- True multi-dimensional arrays
- Embedded comments
- Custom language extensions

Chapter 16

Global Information Grid-Bandwidth Expansion

The Global Information Grid Bandwidth Expansion (GIG-BE) Program was a major United States Department of Defense (DOD) net-centric transformational initiative executed by DISA. GIG-BE created a ubiquitous "bandwidth-available" environment to improve national security intelligence, surveillance and reconnaissance, information assurance, as well as command and control. Through GIG-BE, DISA leveraged DOD's existing end-to-end information transport capabilities, significantly expanding capacity and reliability to select Joint Staff-approved locations worldwide. GIG-BE achieved Full Operational Capability (FOC) on Dec. 20, 2005.

Scope

This program provided increased bandwidth and diverse physical access to approximately 87 critical sites in the continental United States (CONUS), Pacific Theater, and European Theater. These locations are interconnected via an expanded GIG core.

Capabilities and Services

GIG-BE provides a secure, robust, optical terrestrial network that delivers very high-speed classified and unclassified Internet Protocol (IP) services to key operating locations worldwide. The Assistant Secretary of Defense for Networks and Information Integration's (ASD/NII) vision is a "color to every base," physically diverse network access, optical mesh upgrades for the backbone network, and regional/MAN upgrades, where needed. "A color to every base" implies that every site has an OC-192 (10 gigabits per second) of usable IP dedicated to that site.

Implementation

After extensive component integration and operational testing, implementation began in the middle of the 2004 fiscal year and extended through calendar year 2005. The initial implementation concentrated on six sites used during the proof of Initial Operational Capability (IOC), achieved on September 30, 2004. The GIG-BE Program Office conducted detailed site surveys at all of the approximately 87 Joint Staff-approved

locations and parallel implementation in CONUS and overseas. The GIG-BE Program Office completed the Final Operational Test and Evaluation (FOT&E) at 54 operational sites on October 7, 2005. On December 20, 2005, the GIG-BE program achieved the milestone of Full Operational Capability.

Contract

GIG-BE was awarded to SAIC in 2001 for $877 million. This contract was for the development, instantiation, and maintenance of the GIG-BE network. SAIC instantly divided the equipment and tasks into subcontracts. These subcontracts are as follows:

- Ciena Corporation- optical transport segment worth $200-$300 million over two years
- Sycamore Networks- optical cross-connect segment worth $100-$150 million
- Cisco Systems- multiservice provisioning platform segment worth $150-$200 million
- Juniper Networks- core IP router portion worth $150-$200 million
- By Light- installation and maintenance worth $100-$150 million

Congressional critic

Representatives Marty Meehan (D-Mass) and Jim Saxton (R-NJ) expressed concern in the selection process for a contractor to lead the GIG-BE effort. They say it was handled "irresponsibly." The reason for this as expressed by Meehan's spokesperson Kimberly Abbott is the "whole bidding process wasn't as fair and open as it could have been", because a contractor and not the government led the decision process. SAIC won the contract who eventually handed a large subset of the engineering to By Light. Meehan did stress that he is **not** questioning the competence of the winners. These two congressman delivered a letter to the DoD regarding the contract but the issue was slowly forgotten.

Chapter 17

Web Services Resource Framework and WEB2GRID

Web Services Resource Framework

Web Services Resource Framework (WSRF) is a family of OASIS-published specifications for web services. Major contributors include the Globus Alliance and IBM.

A web service by itself is nominally stateless, i.e., it retains no data between invocations. This limits the things that can be done with web services, although workarounds exist – such as having the web service read from a database, for example, or using session state by way of cookies or WS-Session.

WSRF provides a set of operations that web services may implement to become stateful; web service clients communicate with *resource* services which allow data to be stored and retrieved. When clients talk to the web service they include the identifier of the specific resource that should be used inside the request, encapsulated within the WS-Addressing endpoint reference. This may be a simple URI address, or it may be complex XML content that helps identify or even fully describe the specific resource in question.

Alongside the notion of an explicit resource reference comes a standardized set of web service operations to get/set resource properties. These can be used to read and perhaps write resource state, in a manner somewhat similar to having member variables of an object alongside its methods. The primary beneficiary of such a model are management tools, which can enumerate and view resources, even if they have no other knowledge of them. This is the basis for WSDM.

Issues with WSRF

WSRF is not without controversy. Most fundamental is architectural: are distributed objects with state and operations the best way to represent remote resources? It is almost a port into XML of the **distributed objects** pattern, of which CORBA and DCOM are examples. A WSRF resource may be a stateful entity to which multiple clients have resource references and the WSRF specification itself does not deal with concerns such as isolation and availability, deferring to the composable nature of web service

specifications to deal with these. Many WSRF stacks appear to avoid these concerns by being low-availability, mapping 1:1 from a WSRF resource reference to a local object instance, which in C++ and Java is usually not at all persistent (with the exception of those bound to a database through some persistence mechanism). There are, however, implementations of WSRF that support persistence, clustering and high-availability of resources (for example, in WebSphere Application Server).

With a distributed objects view of the network, WSRF is also at loggerheads with the REST model of the network, in which everything is a resource, but in which all actions are enabled through a limited and standardized set of operations. In some ways, the two models are closer than pure SOAP and REST, because they both have stateful resources at the far end. However, REST, as implemented on HTTP, assumes that the URL is all that is needed to address the resource – there is no need for the complexity of the WS-Addressing ReferenceParameters. The idea of managing the lifetime of remote content through renewable leasing comes in for particular criticism. The other issue with the architecture from the REST community is that callbacks/notifications, as described in WS-Notification, do not go through firewalls. This is why REST designs prefer polling, such as in RSS and Atom (standard) feeds. WSRF has done nothing to make SOAP more acceptable to the REST community.

The introduction of WSRF also caused splits in the WS-* world. It was first announced to the World at a Global Grid Forum event in February 2004, as a successor to the Open Grid Services Infrastructure. Its limited compatibility with the mainstream WS-I architecture created dissent from the UK grid community. The Global Grid Forum ultimately isolated their dependencies on WSRF in a **WSRF profile** for their Open Grid Services Architecture. WSRF protocols were also used by WSDM as the means to interacts with **manageable resources** described in WSDM. The WS-* world, however, was not united on a single standard for Web services management with Microsoft, Sun and others choosing to pursue WS-Management, with its dependency on WS-Transfer as the means to describe manageable resources.

Eventually, in spring 2006, an announcement was made of a planned future convergence between WSDM and WS-Management. This may or may not include all of WSRF. Most likely, many of the more controversial aspects of the technology will either be omitted or made optional.

Component specifications

- **WS-Resource** defines a *WS-Resource* as the composition of a resource and a Web service through which the resource can be accessed.
- **WS-ResourceProperties** describes an interface to associate a set of typed values with a WS-Resource that may be read and manipulated in a standard way.
- **WS-ResourceLifetime** describes an interface to manage the lifetime of a WS-Resource.
- **WS-BaseFaults** describes an extensible mechanism for rich SOAPFaults.

- **WS-ServiceGroup** describes an interface for operating on collections of WS-Resources.

Also of relevance is WS-Notification which says how to push information to other web-services about what is going on.

Implementations

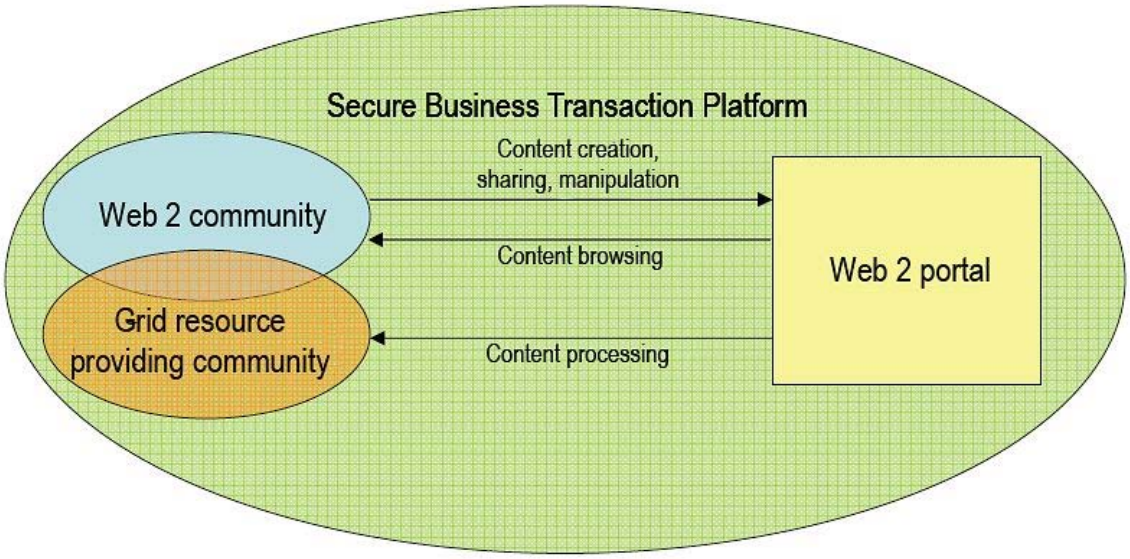
Implementing the basic property get/set semantics of WSRF resources is relatively simple. The hardest problem is probably returning faults as WSRF Base Faults where the specification requires it, because SOAP stacks themselves prefer to raise SOAPFault faults. Managing resource lifetimes is harder, but this is optional, as is WS-Notification, which is the hardest to test.

- The Globus Toolkit version 4 contains Java and C implementations of WSRF; many other Globus tools have been rebuilt around WSRF.
- WebSphere Application Server version 6.1 provides a WSRF environment which supports both simple and clustered, highly available WSRF endpoints.
- The Apache Foundation have a Muse 2.0 project which is a Java-based implementation of the WSRF, WS-Notification, and WSDM specifications.
- WSRF::Lite is a perl-based implementation that makes exclusive use of the *Address* element of the endpoint reference, thus making WS-Resources identifiable via URIs. In addition, WSRF::Lite provides a mapping of HTTP verbs to WSRF operations, making it possible to use WS-Resources in a REST architectural style.
- WSRF.NET is a .NET based project about WSRF specs from a research team of the University of Virginia.
- The latest version 6.0 of UNICORE is built on a Java implementation of the WSRF 1.2 standard including WS-ResourceLifetime and a partial implementation of WS-Notification.

WEB2GRID

The **WEB2GRID** project aims to facilitate the commercial and non-profit exploitation of the EU FP7 EDGeS (Enabling Desktop Grids for e-Science) project and the HAGRID project focusing on the Desktop Grid and WEB2 technologies.

The project, started in 2009, is coordinated by the Laboratory of Parallel and Distributed Systems (LPDS) at MTA-SZTAKI, Hungary.

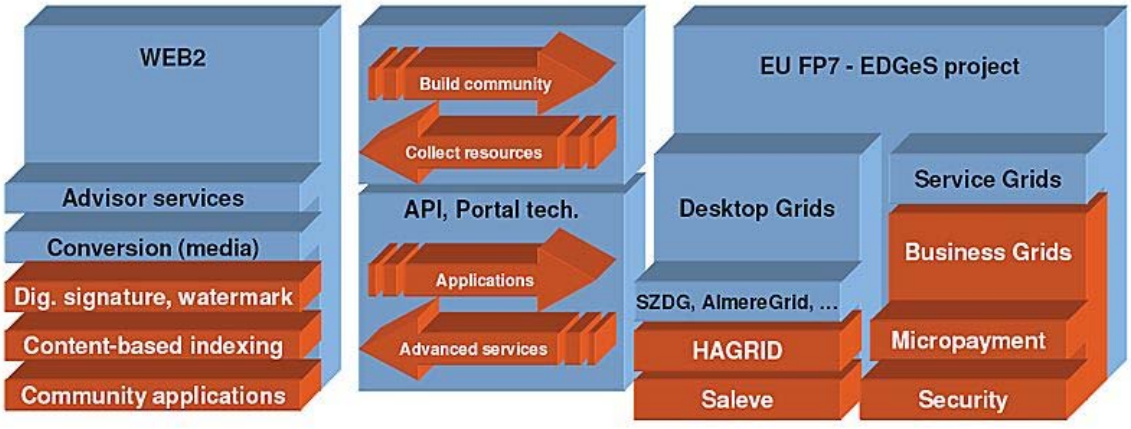


WEB2 and Grid relations

Overview

While WEB2 technologies assist to assure the resources in the desktop grid community on voluntary or settlement bases, the developed platform extended with desktop grid systems offers back-end infrastructure for the operational requirements of WEB2 portals and systems. By combining WEB2 with Desktop Grid technologies, WEB2 can extend its capabilities from the community contents towards to shared services with the help of grid technologies. The project aims to develop the tools, interfaces and methodologies through which the above mentioned targeted services can be established both in closed (local desktop grid), and in open (global desktop grid) environments.

Structure



Connections between the main units of the project

The three main blocks of the project are the Grid, the WEB2 and the connections between the two technologies.

In the last decades Grid systems offered large-scale distributed computing and storage services mainly for academic and university research. In this area the focus is shifting towards services and knowledge sharing, and the extension of the application domain with standardized access methods eliminating system heterogeneity. From Grid point of view, the project relies on the results of the EU FP7 supported EDGeS (Enabling Desktop Grids for e-Science) infrastructure project. In Desktop Grids the system is often based on community offerings, which is very similar to the WEB2 content building solution. While the EDGeS project is important in Grid/WEB2 integration, the WEB2GRID project also extends the EDGeS technologies towards the direction of commercial Grids.

In WEB2 services the owner of the server only provides the service framework, and the real content is shared and maintained by the users. Generally due to the large number of connections and users, WEB2 systems should handle heavy data traffic and complex relations, which may needs large computational power in many cases. From the WEB2 side, numerous application topics with large IT capacity can be solved by Grid technologies.

The connection between WEB2 and Grid is realized with well defined, low-level application development interface (API), and high-level graphical application development solutions of the project partners. WEB2 applications come from various fields, which justifies the need for the development of multi-level interfaces, and the project benefits from numerous EU FP6 and EU FP7 project achievements.

Social and Economical Goals

By connecting WEB2 communities with Grid technologies, the project aims to achieve various social and economical goals including:

- To strengthen the role and recognition of the national research community and knowledge base on international fields
- To demonstrate the wide range of possibilities of the industrial and business usage of Grid and WEB
- To increase the intensity of the connection between state financed research institutes and businesses
- To facilitate the permeation of knowledge-intensive technologies
- The application of methods being developed during the projects contributes to the balanced distribution of research tasks between regions
- The project encourages the cooperation between research institutes and provides academic opportunities for young researchers and PhD students.

Project Partners

The following partners participate in the project:

- econet.hu Media, Telecommunications and Holding Plc.
- Budapest University of Technology and Economics
- E-Group ICT Software Informatics Inc.

Chapter 18

Xgrid and TeraGrid

Xgrid

Xgrid is a proprietary software program and distributed computing protocol developed by the Advanced Computation Group subdivision of Apple Inc that allows networked computers to contribute to a single task.

It provides network administrators a method of creating a computing cluster, which allows them to exploit previously unused computational power for calculations that can be divided easily into smaller operations, such as Mandelbrot maps. The setup of an Xgrid cluster can be achieved at next to no cost, as Xgrid client is pre-installed on all computers running Mac OS X 10.4 or later. The Xgrid controller, the job scheduler of the Xgrid operation, is also included within Mac OS X Server and as a free download from Apple. Apple has kept the command-line job control mechanism minimalist while providing an API to develop more sophisticated tools built around it.

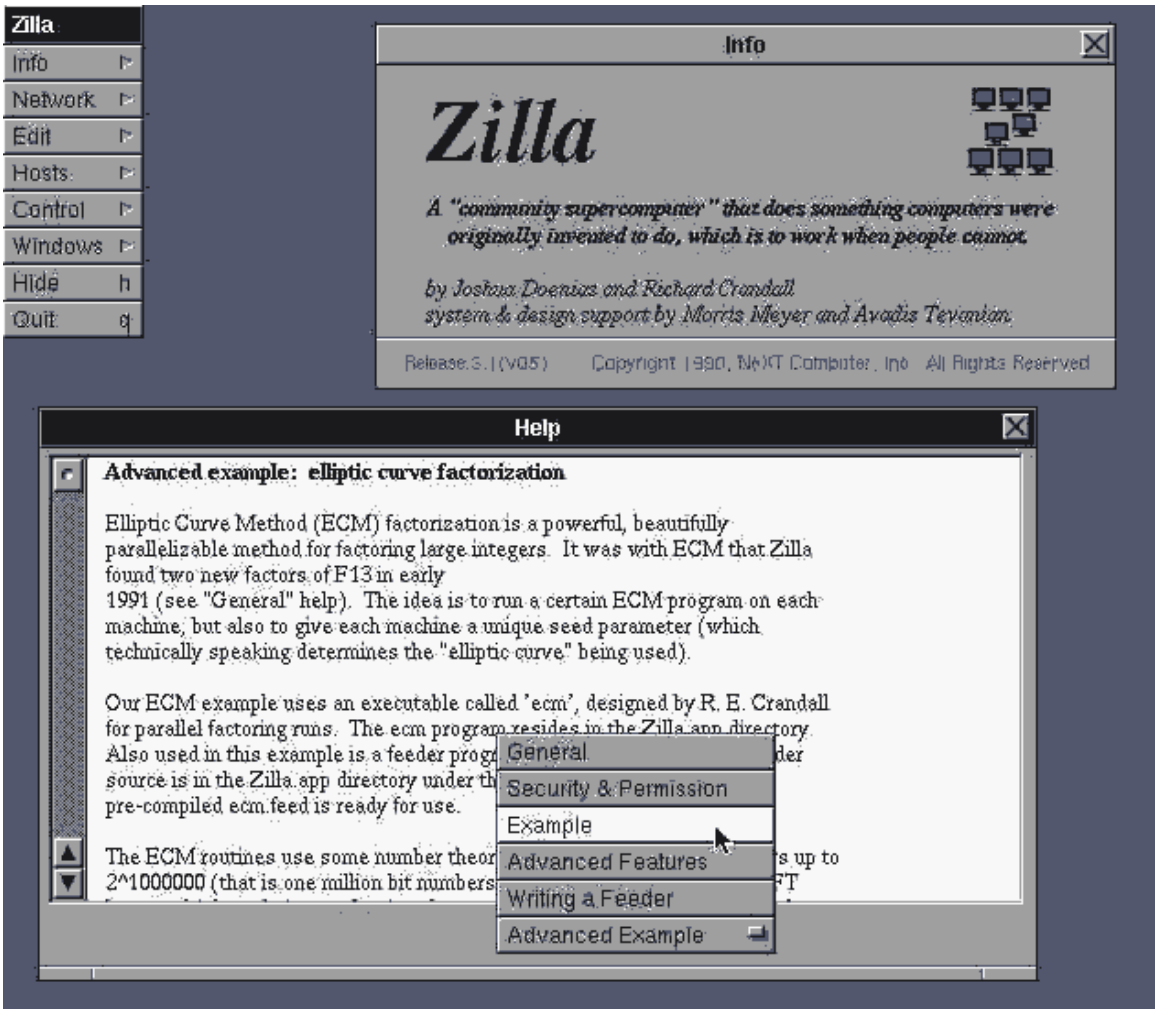
The program employs its own communication protocol layered on top of a schema to communicate to other nodes. This communication protocol interfaces with the BEEP infrastructure, a network application protocol framework. Computers discovered by the Xgrid system, that is computers with Mac OS X's Xgrid service enabled, are automatically added to the list of available computers to use for processing tasks.

When the initiating computer sends the complete instructions, or job, for processing to the controller, the controller splits the task up into these small instruction packets, known as tasks. The design of the Xgrid system consists of these small packets being transferred to all the Xgrid-enabled computers on the network. These computers, or nodes, execute the instructions provided by the controller and then return the results. The controller assembles the individual task results into the whole job results and returns them to the initiating computer.

Apple modeled the design of Xgrid on the Zilla program, distributed with NeXT's OPENSTEP operating system application programming interface (API), which Apple owned the rights to. The company also opted to provide the client version of Mac OS X

with only command-line functions and little flexibility, while giving the Mac OS X Server version of Xgrid a GUI control panel and a full set of features.

History



Zilla

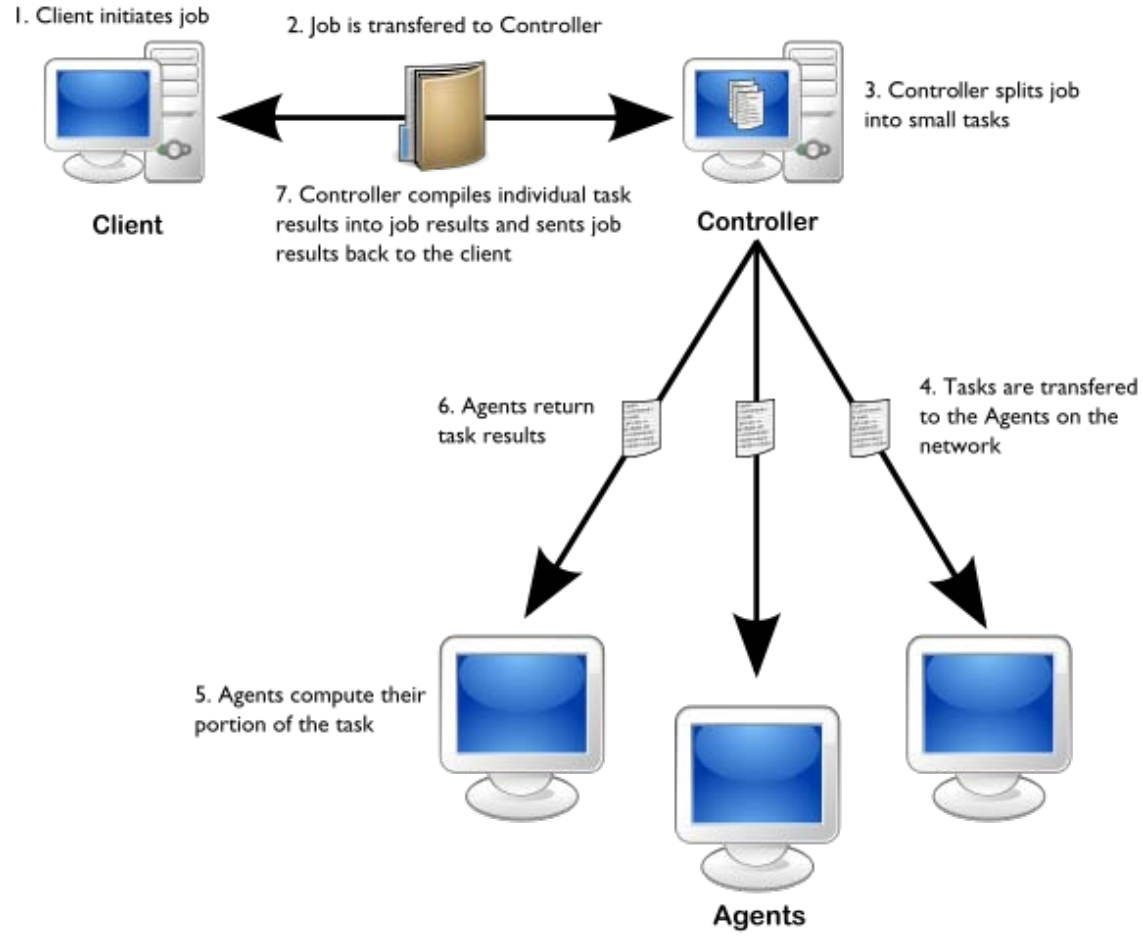
Xgrid's original concept can be traced back to Zilla.app, found in the OPENSTEP operating system API, created by NeXT in the late 1980s. Zilla was the first distributed computing program released on an end-user operating system and which used the idle screen-saver motif, a design feature found in widely used projects such as Seti@Home and Distributed.net. Zilla won the national ComputerWorld Smithsonian Award (Science Category) in 1991 for ease of use and good design. Apple acquired Zilla, along with the rest of NeXT, in 1997 and later used Zilla as inspiration for Xgrid. The first beta version of Xgrid was released in January 2004.

Several organizations have adopted Xgrid in large international computing networks. One example of an Xgrid cluster is MacResearch's OpenMacGrid, where scientists can request access to large amounts of processing power to run tasks related to their research.

Another was the now defunct Xgrid@Stanford project, which used a range of computers on the Stanford University campus and around the world to perform biochemical research.

In a pre-release promotional piece, *MacWorld* cited Xgrid among the Unix features in "10 Things to Know about TIGER", calling it "handy if you work with huge amounts of experimental data or render complex animations". After Xgrid's introduction in 2004, *InfoWorld* noted that it was a "'preview' grade technology" which would directly benefit from the Xserve G5's launch later that year. *InfoWorld* commentator Ephraim Schwartz also predicted that Xgrid was an opening move in Apple's entry into the enterprise computing market.

Protocol



Xgrid Protocol

The Xgrid protocol uses the BEEP network framework to communicate with nodes on the network. The system's infrastructure includes three types of computers which communicate over the protocol. One is the client, which communicates the calculation.

Next is the controller, which starts and segregates the calculation. Finally, the agents process their own allocated part of the calculation.

A computer can act as one or all three of these components at the same time. The Xgrid protocol provides the basic infrastructure for computers to communicate, but is not involved in the processing of the specified calculation. Xgrid is targeted towards time consuming computations that can be easily segregated into smaller tasks, sometimes called *embarrassingly parallel* tasks. This includes Monte Carlo calculations, 3D rendering and Mandelbrot maps.

Within the Xgrid protocol, three types of messages can be passed to other computers on the same cluster: requests, notifications and replies. Requests must be responded to by the recipient with a reply, notifications do not require a reply, and replies are responses to sent messages. They are identified by their name, type (request/notification/reply) and contents. Each message is encapsulated in a BEEP message (BEEP MSG) and is acknowledged on receipt by an empty reply (RPY). Xgrid does not leverage BEEPs message/reply infrastructure. Any received message which requires a response merely generates an independent BEEP message containing the reply. The Xgrid messages are encoded as dictionaries of key/value pairs which are converted to XML before being sent across the BEEP network.

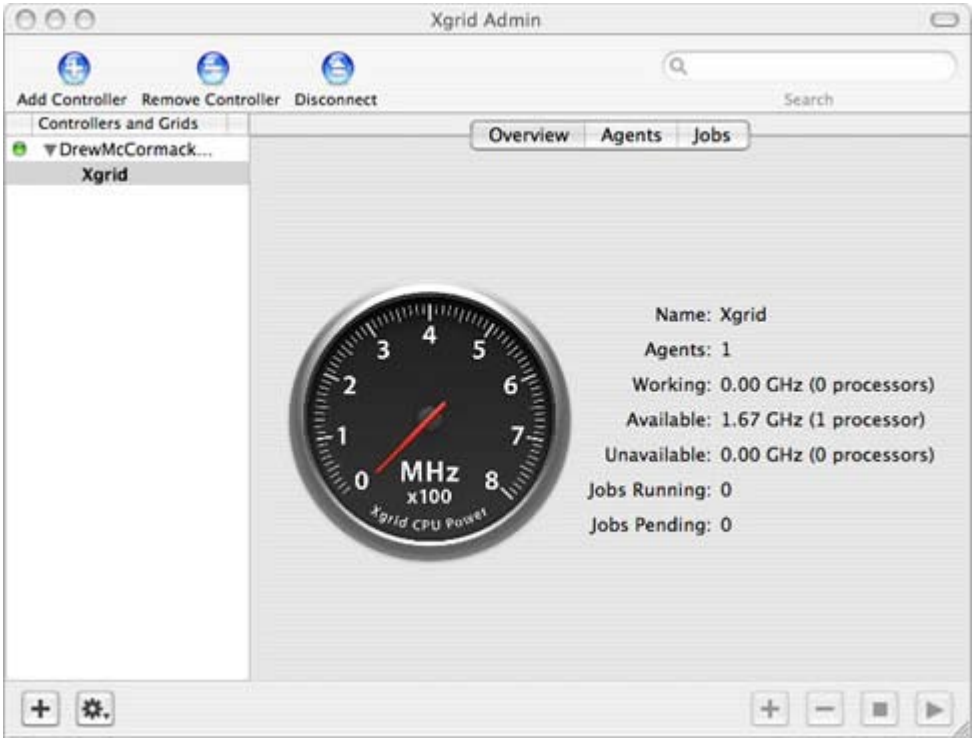
Architecture

The architecture of the Xgrid system is designed around a job based system; the controller sends agents jobs, and the agents return the responses. The actual computation that the controller executes in an Xgrid system is known as a job. The job contains all the files required to complete the task successfully, such as the input parameters, data files, directories, executables and/or shell scripts, the files included in an Xgrid job must be able to be executed either simultaneously or asynchronously, or any benefits of running such a job on an Xgrid is lost. Once the job completes, the controller can be set to notify the client of the task's completion or failure, for example by email. The client can leave the network while the tasks are running. It can also monitor the job status on demand by querying the controller, although it cannot track the ongoing progress of individual tasks.

The controller is central to the correct function of an Xgrid, as this node is responsible for the distribution, supervision and coordination of tasks on agents. The program running on the controller can assign and reassign tasks to handle individual agent failures on demand. The number of tasks assigned to an agent depend on two factors: the number of agents on an Xgrid and the number of processors in each node. The number of agents on an Xgrid determines how the controller will assign tasks. The tasks may be assigned simultaneously for a large number of agents, or queued for a small number of agents. When a node with more than one processor is detected on an Xgrid, the controller may assign one task per processor; this only occurs if the number of agents on the network is lower than the number of tasks the controller has to complete.

Xgrid is layered upon the Blocks Extensible Exchange Protocol (BEEP), an IETF standard comparable to HTTP, but with a focus on two-way multiplexed communication, such as that found in peer-to-peer networks. BEEP, in turn, uses XML to define profiles for communicating between multiple agents over a single network or internet connection.

Interface



Xgrid administration tool

While it is possible to access Xgrid from the command line, the Xgrid graphical user interface, a program bundled with Mac OS X Server and, as of March 2009, available online, is a much more efficient way of administering an Xgrid system. Originally, the Xgrid agent was included in all Mac OS X version 10.4 installations but the GUI was reserved for users of Mac OS X Server. This decision limited the efforts of the computer community to embrace the platform. Eventually, Apple released the Mac OS X Server Administration Tools to the public, which included the Xgrid administration application bundled with Mac OS X Server.

Despite the lack of a graphical controller interface in the standard (non-server) Mac OS X distribution, it is possible to set up an Xgrid controller via the command line tools `xgridctl` and `xgrid`. Once the Xgrid controller daemon is running, administration of the grid with Apple's Xgrid Admin tool is possible. Some applications, such as VisualHub, provided Xgrid controller capability through their user interfaces.

TeraGrid

TeraGrid is an open scientific discovery grid-computing infrastructure combining leadership class resources at eleven partner sites to create an integrated, persistent computational resource.

Using high-performance network connections, the TeraGrid integrates high-performance computers, data resources and tools, and high-end experimental facilities around the country. Currently, TeraGrid resources include more than a petaflop of computing capability and more than 30 petabytes of online and archival data storage, with rapid access and retrieval over high-performance networks. Researchers can also access more than 100 discipline-specific databases. With this combination of resources, the TeraGrid is the world's largest, most comprehensive distributed cyberinfrastructure for open scientific research.

TeraGrid is coordinated through the Grid Infrastructure Group (GIG) at the University of Chicago, working in partnership with the Resource Provider sites: Indiana University, the Louisiana Optical Network Initiative, National Center for Supercomputing Applications, the National Institute for Computational Sciences, Oak Ridge National Laboratory, Pittsburgh Supercomputing Center, Purdue University, San Diego Supercomputer Center, Texas Advanced Computing Center, and University of Chicago/Argonne National Laboratory, and the National Center for Atmospheric Research.

History

The TeraGrid project was launched by the National Science Foundation in August 2001 with $53 million in funding to four sites: the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign, the San Diego Supercomputer Center (SDSC) at the University of California, San Diego, University of Chicago Argonne National Laboratory, and Center for Advanced Computing Research (CACR) at the California Institute of Technology in Pasadena, California.

UCSD TeraGrid

In October 2002, the Pittsburgh Supercomputing Center (PSC) at Carnegie Mellon University and the University of Pittsburgh joined the TeraGrid as major new partners when NSF announced $35 million in supplementary funding. The TeraGrid network was transformed through the ETF project from a 4-site mesh to a dual-hub backbone network with connection points in Los Angeles and at the Starlight facilities in Chicago.

In October 2003, NSF awarded $10 million to add four sites to TeraGrid as well as to establish a third network hub, in Atlanta. These new sites were Oak Ridge National Laboratory (ORNL), Purdue University, Indiana University, and the Texas Advanced Computing Center (TACC) at The University of Texas at Austin.

TeraGrid construction was also made possible through key corporate partnerships with Sun Microsystems, IBM, Intel Corporation, Qwest Communications, Juniper, Myricom, Hewlett-Packard Company, and Oracle Corporation.

TeraGrid construction was completed in October 2004, at which time the TeraGrid facility began full production.

In August 2005, NSF awarded $148M for a five-year program to operate and enhance the TeraGrid facility, with eight resource provider awards and a system integration award (the Grid Infrastructure Group at the University of Chicago).

The TeraGrid: 2005-2011

In August 2005, NSF's newly created Office of Cyberinfrastructure extended support for the TeraGrid with a $150 million set of awards for operation, user support and enhancement of the TeraGrid facility over the next five years. Using high-performance network connections, the TeraGrid featured high-performance computers, data resources and tools, and high-end experimental facilities around the country. In May 2007, TeraGrid integrated resources included more than 250 teraflops of computing capability and more than 30 petabytes (quadrillions of bytes) of online and archival data storage with rapid access and retrieval over high-performance networks. Researchers could access more than 100 discipline-specific databases. In late 2009, The TeraGrid resources had grown to 2 petaflops of computing capability and more than 60 petabytes storage. In mid 2009, NSF extended the operation of TeraGrid to 2011. TeraGrid competes with EGEE to be the world's largest, most comprehensive distributed cyberinfrastructure for open scientific research.

Architecture

TeraGrid resources are integrated through a *service-oriented architecture* in that each resource provides a "service" that is defined in terms of interface and operation. Computational resources run a set of software packages called "Coordinated TeraGrid Software and Services" (CTSS). CTSS provides a familiar user environment on all TeraGrid systems, allowing scientists to more easily port code from one system to another. CTSS also provides integrative functions such as single-signon, remote job submission, workflow support, data movement tools, etc. CTSS includes the Globus Toolkit, Condor, distributed accounting and account management software, verification and validation software, and a set of compilers, programming tools, and environment variables.

TeraGrid uses a 10 Gigabits per second dedicated optical backbone network, with hubs in Chicago, Denver, and Los Angeles. All resource provider sites connect to a backbone node at 10 Gigabits per second. TeraGrid users access the facility through national research networks such as the Internet2 Abilene backbone and National LambdaRail.

Usage

TeraGrid users primarily come from U.S. universities. There are roughly 4,000 users at over 200 universities. Academic researchers in the United States can obtain exploratory, or *development* allocations (roughly, in "CPU hours") based on an abstract describing the work to be done. More extensive allocations involve a proposal that is reviewed during a quarterly peer-review process. All allocation proposals are handled through the TeraGrid website. Proposers select a scientific discipline that most closely describes their work,

and this enables reporting on the allocation of, and use of, TeraGrid by scientific discipline. As of July 2006 the scientific profile of TeraGrid allocations and usage is shown in the following table.

Allocated (%)	Used (%)	Scientific Discipline
19	23	Molecular Biosciences
17	23	Physics
14	10	Astronomical Sciences
12	21	Chemistry
10	4	Materials Research
8	6	Chemical, Thermal Systems
7	7	Atmospheric Sciences
3	2	Advanced Scientific Computing
2	0.5	Earth Sciences
2	0.5	Biological and Critical Systems
1	0.5	Ocean Sciences
1	0.5	Cross-Disciplinary Activities
1	0.5	Computer and Computation Research
0.5	0.25	Integrative Biology and Neuroscience
0.5	0.25	Mechanical and Structural Systems
0.5	0.25	Mathematical Sciences
0.5	0.25	Electrical and Communication Systems
0.5	0.25	Design and Manufacturing Systems
0.5	0.25	Environmental Biology

Each of these discipline categories correspond to a specific program area of the National Science Foundation (thus more detail can be found at the NSF website).

During 2006, TeraGrid has begun to provide application-specific services to *Science Gateway* partners, who serve (generally via a web portal) discipline-specific scientific and education communities. Through the Science Gateways program TeraGrid aims to broaden access by at least an order of magnitude in terms of the number of scientists, students, and educators who are able to use TeraGrid.

TeraGrid Resource Providers

- Argonne National Laboratory (ANL) operated by the University of Chicago and the Department of Energy
- Indiana University
- Louisiana Optical Network Initiative (LONI)
- National Center for Atmospheric Research (NCAR)
- National Center for Supercomputing Applications (NCSA)

- National Institute for Computational Sciences (NICS) operated by University of Tennessee at Oak Ridge National Laboratory.
- Oak Ridge National Laboratory (ORNL)
- Pittsburgh Supercomputing Center (PSC) operated by University of Pittsburgh and Carnegie Mellon University.
- Purdue University
- San Diego Supercomputer Center (SDSC)
- Texas Advanced Computing Center (TACC)

Similar projects

- DEISA Distributed European Infrastructure for Supercomputing Applications, a facility integrating eleven European supercomputing centers.
- EGEE Enabling Grids for E-sciencE
- NAREGI Japanese NAtional REsearch Grid Initiative involving several supercomputer centers
- Open Science Grid - a distributed computing infrastructure for scientific research

Chapter 19

Space-Based Architecture and SAGA C++ Reference Implementation

Space-Based Architecture

Space-Based Architecture (**SBA**) is a software architecture pattern for achieving linear scalability of stateful, high-performance applications using the tuple space paradigm. It follows many of the principles of Representational State Transfer (REST), service-oriented architecture (SOA) and Event-driven architecture (EDA), as well as elements of grid computing. With a space-based architecture, applications are built out of a set of self-sufficient units, known as processing-units (PU). These units are independent of each other, so that the application can scale by adding more units.

The SBA model is closely related to other patterns that have been proved successful in addressing the application scalability challenge, such as Shared-Nothing Architecture, used by Google, Amazon.com and other well-known companies. The model has also been applied by many firms in the securities industry for implementing scalable electronic securities trading applications.

Components of Space-Based Architecture

An application built on the principles of space-based architecture typically has the following components:

- **Processing Unit** — the unit of scalability and fail-over. Normally, a processing unit is built out of a POJO (Plain Old Java Object) container, such as that provided by the Spring Framework.

- **Virtual Middleware** — a common runtime and clustering model, used across the entire middleware stack. The core middleware components in a typical SBA architecture are:

Component	Description
Messaging Grid	Handles the flow of incoming transaction as well as the communication between services

Data Grid	Manages the data in distributed memory with options for synchronizing that data with an underlying database
Processing Grid	Parallel processing component based on the master/worker pattern (also known as a blackboard pattern) that enables parallel processing of events among different services

- **POJO-Driven Services Model** — A lightweight services model that can take any standard Java implementation and turn it into a loosely coupled distributed service. The model is ideal for interaction with services that run within the same processing-unit.

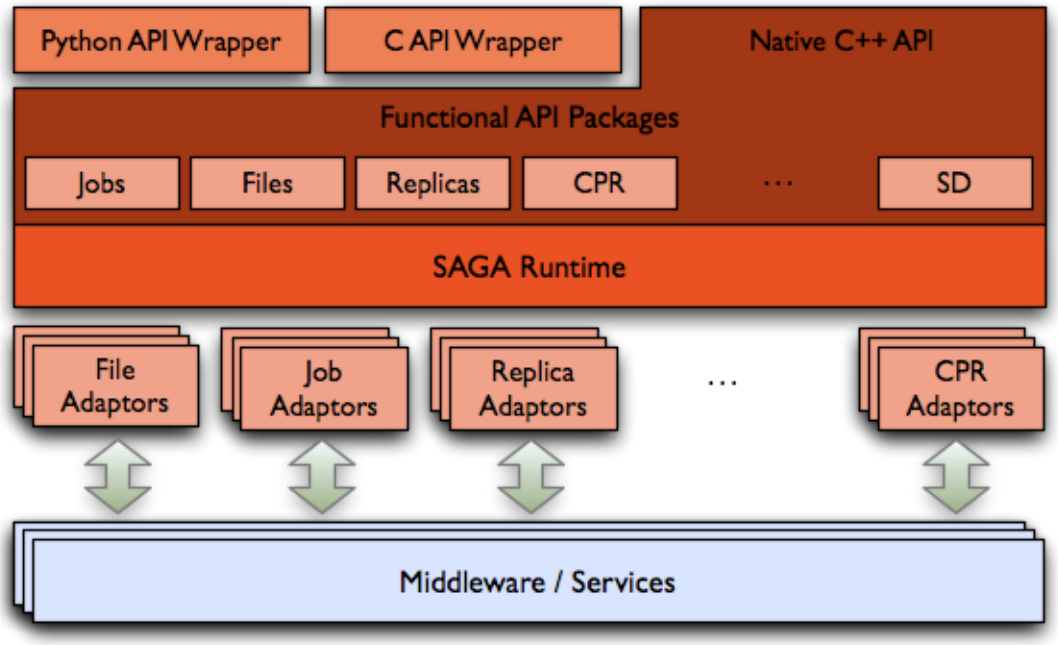
- **SLA-Driven Container** — The SLA-driven container enables the deployment of the application on a dynamic pool of machines based on Service Level Agreements. SLA definitions include the number of instances that need to run in order to comply with the application scaling and fail-over policies, as well as other policies.

SAGA C++ Reference Implementation

The **SAGA C++ Reference Implementation** is a set of free cross-platform libraries written in C++ and Python which provide a set of high-level interfaces and runtime components that allow the development of distributed computing and grid computing applications, frameworks and tools. SAGA is the first complete implementation of the Open Grid Forum Simple API for Grid Applications standard GFD-R-P.90. SAGA is available for all major operating systems, including Linux and other Unix-like systems, Microsoft Windows and Mac OS X. SAGA is open source and licensed under the Boost Software License.

SAGA can be used to develop scalable and portable large-scale distributed applications, frameworks and tools. SAGA supports many of the widely used distributed grid middleware systems, like Globus, Condor, UNICORE and gLite as well as cloud computing services like Amazon EC2 and Eucalyptus.

Architecture



The SAGA C++/Python architecture: a light-weight runtime system dispatches API calls from the application to the middleware through a set of plug-ins or *adaptors*.

SAGA is designed as an object oriented interface. It encapsulates related functionality in a set of objects, that are grouped in functional namespaces, which are called *packages* in SAGA. The SAGA core implementation defines the following packages:

- saga::advert - interface for advert service access
- saga::filesystem - interface for file and directory access
- saga::job - interface for job definition, management and control
- saga::namespace - abstract interface (used by advert, filesystem and replica interfaces)
- saga::replica - interface for replica management
- saga::rpc - interface for remote procedure calls client and servers
- saga::sd- interface for service discovery in distributed environments
- saga::stream - interface for data stream client and servers

The overal architecture of SAGA follows the adaptor pattern, a software design pattern which is used for translating one interface into another. In SAGA it translates the calls from the API packages to the interfaces of the underlying middleware. The SAGA run-time system uses late-binding to decide at run-time which plug-in (*middleware adaptor*) to load and bind.

Supported Middleware

The following table lists the distributed middleware systems that are currently supported by SAGA. The column labeled *Adaptor Suite* names the collection (release package) of the (set of) middleware adaptors that provides support for the middleware system.

Middleware System	SAGA Adaptor Suite	SAGA API Namespace
Amazon EC2	saga-adaptors-aws	saga::job
Condor	saga-adaptors-condor	saga::job
Eucalyptus	saga-adaptors-aws	saga::job
Globus GRAM (2 and 5)	saga-adaptors-globus	saga::job
Globus GridFTP	saga-adaptors-globus	saga::filesystem
Globus RLS	saga-adaptors-globus	saga::replica
HDFS	saga-adaptors-hdfs	saga::file
Local File system	*part of saga-core*	saga::file
Local Fork	*part of saga-core*	saga::job
Nimbus	saga-adaptors-aws	saga::job
PBS (Pro)	saga-adaptors-pbs	saga::job
Platform LSF	saga-adaptors-lsf	saga::job
SQL Advert Service	*part of saga-core*	saga::advert
SQL Replica Service	*part of saga-core*	saga::replica
SSHFS	saga-adaptors-ssh	saga::file
SSH	saga-adaptors-ssh	saga::job
TORQUE	saga-adaptors-torque	saga::job

Examples

Job Submission

A typical task in a distributed application is to submit a *job* to a local or remote distributed resource manager. SAGA provides a high-level API called the *job package* for this. The following two simple examples show how the SAGA job package API can be used to submit an MPI job to a remote Globus GRAM resource manager.

C++:

```
#include <saga/saga.hpp>

int main (int argc, char** argv)
{
  namespace sa  = saga::attributes;
  namespace sja = saga::job::attributes;

  try
```

```
  {
    saga::job::description jd;

    jd.set_attribute (sja::description_executable, "/home/user/hello-
mpi");
    jd.set_attribute (sja::description_output, "/home/user/hello.out");
    jd.set_attribute (sja::description_error, "/home/user/hello.err");

    // Declare this as an MPI-style job
    jd.set_attribute (sja::description_spmd_variation, "mpi");

    // Name of the queue we want to use
    jd.set_attribute (sja::description_queue, "checkpt");
    jd.set_attribute (sja::description_spmd_variation, "mpi");
    // Number of processors to request
    jd.set_attribute (sja::description_number_of_processes, "32");

    saga::job::service js("gram://my.globus.host/jobmanager-pbs");
    saga::job::job j = js.create_job(jd);

    j.run()
  }
  catch(saga::exception const & e)
  {
    std::cerr << "SAGA exception caught: " << e.what() << std::endl;
  }
}
```

Python:

```
import saga

try:
  jd = saga.job.description()

  jd.executable = "/home/user/hello-mpi"
  jd.error  = "/home/user/hello.err"
  jd.output = "/home/user/hello.out"

  # Declar this as an MPI-style job
  jd.spmd_variation = "mpi"

  # Name of the queue we want to use
  jd.queue = "checkpt"
  # Number of processors to request
  jd.number_of_processes = "32"

  # URL of the resource manager. In this case Globus GRAM
  js = saga.job.service("gram://my.globus.host/jobmanager-pbs")

  job = js.create_job(jd)
  job.run()

except saga.exception, e:
  print e.get_all_messages()
```

Chapter 20

Oracle Grid Engine and OurGrid

Oracle Grid Engine

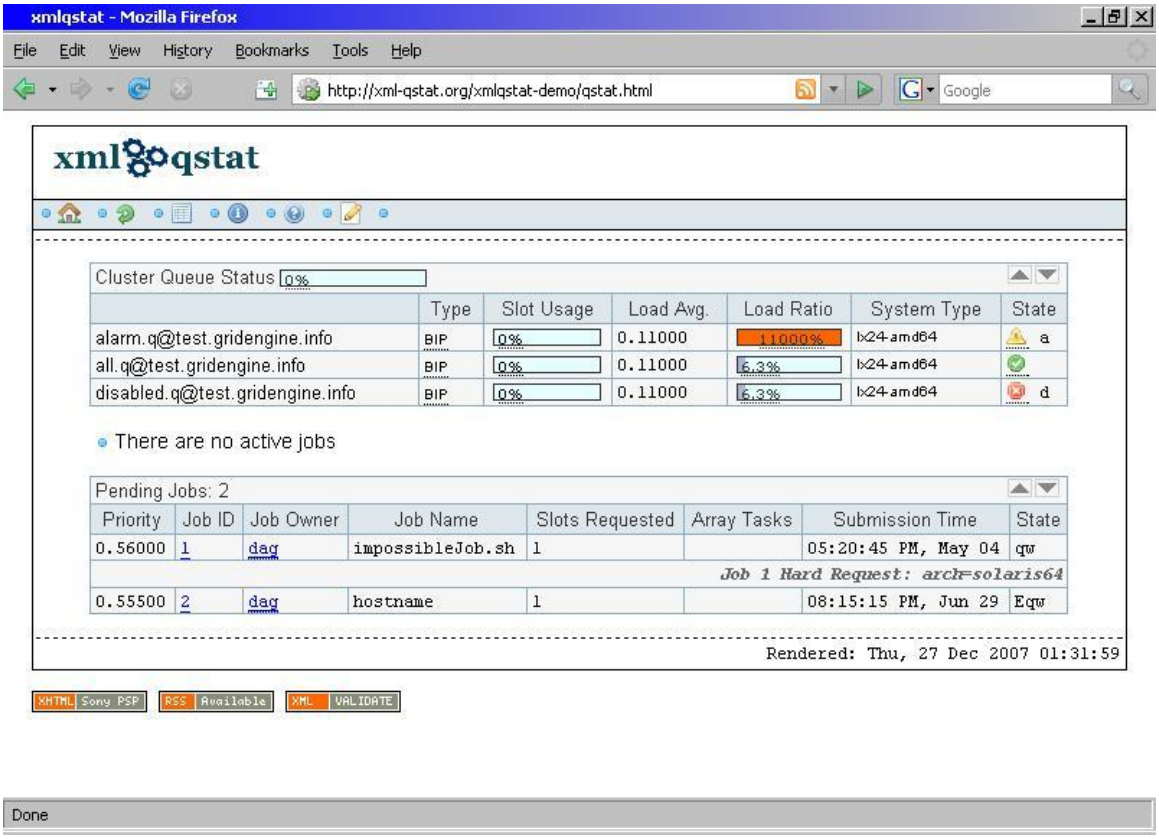
Oracle Grid Engine, previously known as *Sun Grid Engine* (**SGE**), previously known as **CODINE** (COmputing in DIstributed Networked Environments) or **GRD** (Global Resource Director), is an open source batch-queuing system, developed and supported by Sun Microsystems. Sun once also sold a commercial product based on SGE, known as **N1 Grid Engine** (**N1GE**).

Grid Engine is open source and free to use from the project website under the Sun Industry Standards Source License. There is a commercial version available from the Oracle site. It appears that all further versions, starting from 6.2u6, will be commercial (with a 90-day free trial). Licenses cost 500 USD per processor).

SGE is typically used on a computer farm or high-performance computing (HPC) cluster and is responsible for accepting, scheduling, dispatching, and managing the remote and distributed execution of large numbers of standalone, parallel or interactive user jobs. It also manages and schedules the allocation of distributed resources such as processors, memory, disk space, and software licenses.

SGE is the foundation of the Sun Grid utility computing system, made available over the Internet in the United States in 2006, later becoming available in many other countries.

Features



A screenshot of the xml-qstat web interface.

Features new in version 6.2

- Advance reservation
- Array job interdependencies
- Rule-based *Resource Quota* control
- Enhanced remote execution (without using external rshd/rlogind/sshd processes)
- Multi-clustering
- Daemons managed by the Service Management Facility on Solaris
- Pseudo TTY (pty) support for interactive jobs
- Job Submission Verifier (client-side and server-side job verification)
- GUI Installer and SGE Inspect
- Topology-aware scheduling and thread binding
- Hadoop integration, Amazon EC2 integration for cloud computing

Other features of SGE include:

- Multiple advanced scheduling algorithms allow powerful policy-based resource allocation
- Cluster queues

- Job and scheduler fault tolerance - Grid Engine continues to operate as long as there is one or more hosts available
- Job checkpointing
- Job arrays and job tasks
- DRMAA (Job API)
- Resource reservation
- XML status reporting (*qstat* and *qhost*), and the *xml-qstat* web interface
- Parallel jobs (MPI, PVM, OpenMP), and scalable parallel job startup with qrsh
- Usage accounting
- Accounting and Reporting COnsole (ARCO)
- parallel make: distmake, dmake (Sun Studio), and SGE's own qmake
- FLEXlm integration and multi-cluster software license management with *LicenseJuggler*

Platforms

SGE runs on multiple platforms, including:

- AIX
- BSD - FreeBSD, NetBSD, OpenBSD
- HP-UX
- IRIX
- Linux
- Mac OS X
- Solaris
- SUPER-UX
- Tru64
- Windows via SFU (Interix) or SUA (Microsoft Windows Services for UNIX) (as execution hosts only)
- Z/OS (in progress)

Cluster architecture

A typical Grid Engine cluster consists of a master host, and one or more execution hosts. Moreover, multiple *shadow masters* can be configured as hot spares, which take over the role of the master when the original master host crashes.

Support and training

Sun provides support contracts for the commercial version of Grid Engine on most UNIX platforms and Windows. Professional services, consulting, training, and support are also provided by Sun Partners. Sun partners with Georgetown University to deliver Grid Engine administration classes. *The Bioteam* runs short SGE training workshops that are 1 or 2 days long.

Users can obtain community support on the Grid Engine mailing lists.

Grid Engine Workshops were held in 2002, 2003, 2007, and 2009 in Regensburg, Germany.

Prominent users

Notable deployments of SGE include:

- Sun Grid
- the TSUBAME supercomputer at the Tokyo Institute of Technology, which was number 7 on June 2006 TOP500 list.
- Ranger at the Texas Advanced Computing Center (TACC). Ranger has 62,976 processor cores in 3,936 nodes and a peak performance of 504TFlops. Ranger was the 4th most powerful TOP500 supercomputer in 2008.
- San Diego Supercomputer Center (SDSC)
- Geophysical Fluid Dynamics Laboratory (NOAA GFDL)

History

In 2000, Sun acquired Gridware, Inc. a privately owned commercial vendor of advanced computing resource management software with offices in San Jose, Calif., and Regensburg, Germany. Later that year, Sun offered a free version of Gridware for Solaris and Linux, and renamed the product Sun Grid Engine.

In 2001, Sun made the source code available, and adopted the open source development model. Ports for Mac OS X and *BSD were contributed by the non-Sun open source developers.

In 2010, after the purchase of Sun by Oracle, the Grid Engine 6.2 update 6 source code was not included with the binaries, and changes were not put back to the project's source repository. In response to this, the Grid Engine community started the *Open Grid Scheduler* project to continue to develop and maintain a free implementation of Grid Engine.

On January 18, 2011, it was announced that Univa had recruited several principal engineers from the former Sun Grid Engine team and that Univa would be developing their own forked version of Grid Engine. The newly announced Univa Grid Engine will include commercial support and would compete with the official version of Oracle Grid Engine.

Other Grid Engine based products

- Sun Constellation System
- Sun Visualization System
- Sun Compute Cluster
- ClusterVisionOS Distribution
- Rocks Cluster Distribution

- EGEE
- Univa's UniCluster Express
- BioTeam's iNquiry
- Nimbus - uses Grid Engine as a virtual machine scheduler in a cloud computing environment

Add-on software

A number of SGE add-ons are available:

- Solaris Cluster integration
- *Service Domain Management* module in order to meet service level objectives
- Transfer-queue Over Globus (TOG). Globus added support for Grid Engine in Globus Toolkit 5.0.0
- JOb Scheduling Hierarchically (JOSH)

OurGrid

OurGrid is an opensource grid middleware based on a peer-to-peer architecture.

OurGrid is mainly develop at the Federal University of Campina Grande (Brazil), which also run an OurGrid instance named "OurGrid" too, in production since December 2004. Anyone can freely and easily join it to gain access to large amount of computational power and run parallel applications. This computational power is provided by the idle resources of all participants, and is shared in a way that makes those who contribute more get more when they need. Currently, the platform can be used to run any application whose tasks (ie, parts that run on a single machine) do not communicate among themselves during execution, like most simulations, data mining and searching.

OurGrid Vision

The recent advances in computing and networking are changing the way we do scientific research, a trend that has been dubbed eScience. Thanks to the power of computer-based communication, research is now a much more collaborative endeavor. Moreover, computers play an ever-increasing role in the process of scientific discovery. Data analysis without computers sounds antediluvian. Simulation has joined theory and experimentation as the third scientific methodology. As a result, many research labs now demand non-trivial computing capabilities.

As a solution to this demand, Grid Computing has appeared with the enticing promise of turning computing into utility. The vision is plug in the grid and solve your problem. However, turning the grid vision into reality is no trivial matter. In particular, grid

solutions available today require highly specialized computer skills to install, set-up and operate, as well as an off-line negotiation to decide how the resources are going to be shared among the nodes that compose the grid. Therefore, current grid solutions only make sense for large labs. They do have the resources (both human and computer) to undertake the effort of putting a grid into production, as well as staff to negotiate with the other sites that compose the grid.

However, the vast majority of the research labs are small (a dozen people or so), and thus cannot afford deploying grid technology. Yet, these labs increasingly demand large amounts of computational power. OurGrid was designed to fill this gap, catering for small and medium-sized labs around the world that have unserved computational demands.

OurGrid is an open, free-to-join, cooperative grid in which labs donate their idle computational resources in exchange for accessing other labs' idle resources when needed. It uses a peer-to-peer technology that makes it in each lab's best interest to collaborate with the system by donating its idle resources. OurGrid leverages from the fact that people do not use their computers all the time. Even when actively using computers as research tools, researchers alternate between job execution (when they demand computational power) and result analysis (when their computational resources go mostly idle).

For OurGrid to be useful, it must be fast, simple, scalable, and secure. These were the four major goals that guided the design of OurGrid. Putting it in more detail:

- OurGrid must be fast (i.e. the turnaround time of a job must be much better than what is possible using only local resources) otherwise there will be no point in using it.

- Simplicity is also a fundamental requirement for OurGrid. After all, labs want to spend the minimum possible effort on the computer technology that will solve their problems. They want to focus on whatever research they do. Computers are just tools for them.

- OurGrid must scale well; otherwise it will not tap the huge amount of computational power that goes idle in the labs around the world. Note that scalability is not just a technical issue. It also has administrative ramifications. In particular, it is not acceptable to have to go through a human negotiation to define who can access what, when and how (something that is needed to set up current grids). Therefore, OurGrid must be a free-to-join open grid.

- OurGrid must be secure, because its peer-to-peer automatic granting of access will allow unknown foreign code to access one's machine. Nevertheless one's machine must remain safe.

Achieving these goals was a very challenging task. In order to simplify somewhat the problem, at least for now, we reduce OurGrid's scope to supporting Bag-of-Tasks (BoT) applications. BoT applications are those parallel applications whose tasks are independent. Despite their simplicity, BoT applications are used in a variety of scenarios, including data mining, massive searches (such as key breaking), parameter sweeps, simulations, fractal calculations, computational biology, and computer imaging. Assuming applications to be BoT simplifies our requirements in a few important ways. In particular, we can deliver fast execution of applications without demanding any QoS guarantees. It also makes it easier to provide a secure environment, since network access is not necessary during the execution of a foreign task.

Inside OurGrid

OurGrid 4.0 Main Components

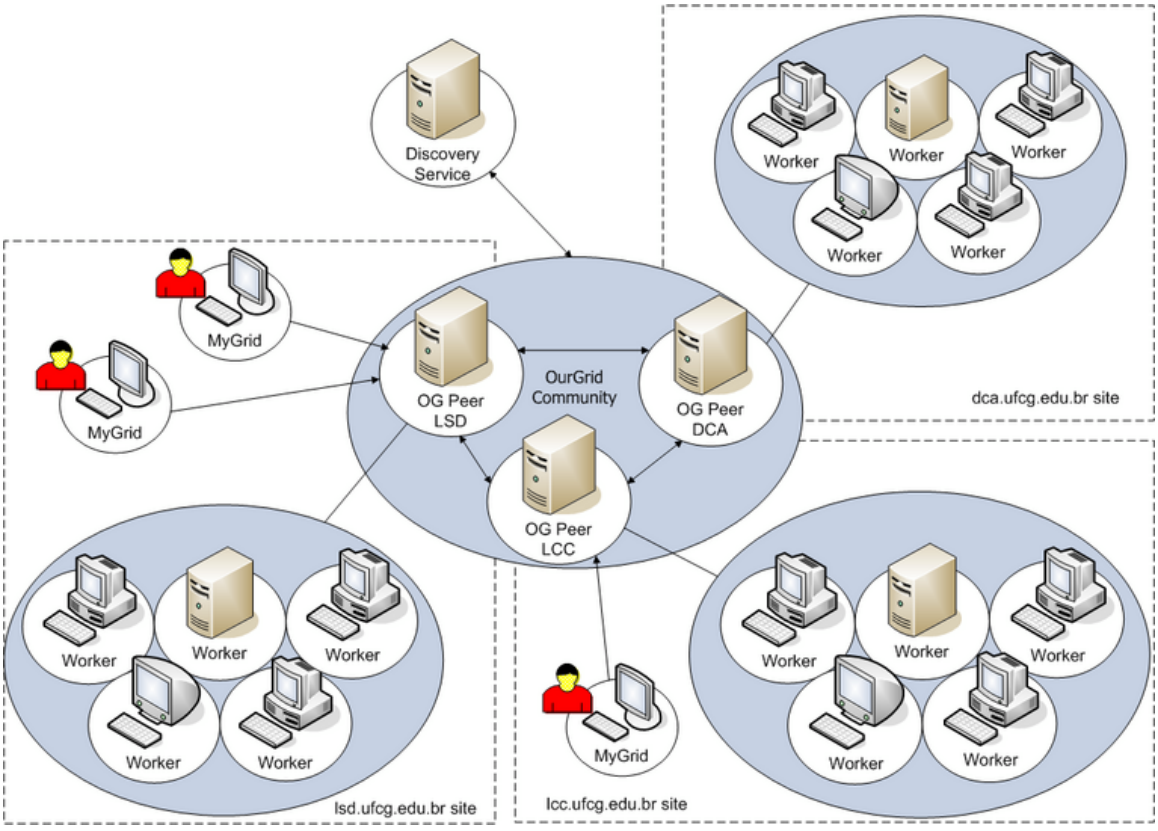


Figure 1.1 OurGrid Main Components

- OurGrid Broker

The OurGrid Broker (originally called MyGrid) is the scheduling component of the OurGrid solution. A machine running the Broker is called the home machine, which is the central point of a grid. During the processing of jobs, it acts as the grid coordinator, scheduling the execution of tasks and doing all the necessary data transfer to and from

grid machines. Due to its central role, grid configuration and management, as well as job specification, is done on the home machine. For these reasons, you will likely use your desktop as the home machine for your grid. This approach decentralizes access to the grid allowing multiple users, each using their own installation of the Broker, to do concurrent processing.

The Broker is OurGrid's user frontend. It provides all the necessary support to describe, execute, and monitor jobs. Job processing is done by machines running OurGrid Workers. During the execution of a job, the Broker gets Workers on-demand from its associated Peer. It is the Broker's role to schedule the tasks to run on the Workers and to deploy and retrieve all data to/from Workers before and after the execution of tasks.

- Peers

An OurGrid Peer runs on a machine called the peer machine. The main role of a Peer is to organize and provide worker machines that belong to the same administrative domain. From the user's perspective, a Peer is a Worker provider, i.e., a network service that dynamically provides Workers for task execution. From an administrative point of view, a Peer determines how and which machines can be used as workers.

In Figure 1.1, there is one Peer for each administrative domain. This architecture allows different site administrators to enforce their own policies regarding the use of their Workers.

- Workers

The OurGrid Worker component runs on each machine that will be available for task execution. The Worker provides necessary access functionality to the home machine. It also provides some basic support for instrumentation and fault handling. Furthermore, combined with the OurGrid Peer, it allows for the use of machines in private networks.

In practice, any computer connected to the Internet can be used as a worker machine, even if it lies in a different administrative domain or behind a firewall. In Figure 1.1, administrative domains, possibly using their own intranets, are illustrated as rectangles containing Workers.

Network of Favours

To encourage resource contribution to the network, OurGrid uses a resource allocation mechanism called Network of Favours. The Network of Favours is an autonomous reputation scheme that rewards Peers that contribute more. This way, there is an incentive for each Peer to contribute as much as possible to the system.

The OurGrid Community is a peer-to-peer resource sharing system focused on providing resources to BoT applications. The central mission of OurGrid Community is that the sharing be done using the network of favours model. In this model, each Peer offers

access to its idle resources to the community. In return, when there is work that exceeds local capacity, a Peer expects to gain access to the idle resources of other participants. The system aims to allow users of BoT applications to easily obtain access and use the community's computational resources, dynamically forming an on-demand, large-scale, grid.

However, the OurGrid solution provides more. Peers may belong to a world wide peer-to-peer community that share resources in a network of favours. This turns your Broker into a world wide out-of-the-box enabled grid. Once your Broker is connected to the community, tasks will be farmed out to machines that belong to different administrative domains all over the world without any further configuration.

Each Peer in the community is an entity that owns a number of resources and occasionally needs more computing power than these resources can provide. Whenever a Peer needs more power, it requests resources to the community. Whenever it has idle resources, it allocates them to one of the requesters. As there are no guarantees about the quality of service obtained from the idle resources donated to the community, not all applications are suitable for OurGrid.

Chapter 21

MTA SZTAKI Laboratory of Parallel and Distributed Systems & National Grid Service

MTA SZTAKI Laboratory of Parallel and Distributed Systems

The **Laboratory of Parallel and Distributed Systems** (LPDS) focuses its research on cluster and grid technologies, applying its research results in products such as the P-GRADE Grid Portal, gUSE and the SZTAKI Desktop Grid. The Laboratory is an active participant in European Grid projects including Enabling Grids for E-sciencE (EGEE) and Enabling Desktop Grids for e-Science (EDGeS).

LPDS and MTA SZTAKI: main building

Products

- The P-GRADE Grid Portal is a Grid portal solution capable of granting access to multiple Grids. It allows users to manage the whole life-cycle of executing a parallel application, enabling the creation, execution and monitoring of workflows through high-level Web interfaces.

- gUSE provides a scalable set of high-level Grid services by which interoperation between Grids and user communities can be achieved. Incorporating a more flexible workflow concept and enabling its distribution on clusters and different Grid sites, gUSE is aimed at extending the objectives and features of the P-GRADE Portal.

- The goal of SZTAKI Desktop Grid is providing an enterprise solution to exploit PCs and clusters located at different sites of a company or institute, solving large scale distributed programs via an easy-to-use application programming interface. It is extended to include clusters as single powerful PCs and to hierarchically propagate work from one desktop grid to the other.

Research

The main research areas of the LPDS are parallel distributed and concurrent programming, graphical programming environments, supercomputing, cluster computing and Grid computing.

Major projects

The Laboratory participated in the CoreGRID Network of Excellence and has been working as a project member in all phases of the grid infrastructure project Enabling Grids for E-sciencE (EGEE), serving as a regional training centre from 2004, as the assistant leader in training from 2007 and as the coordinator of application porting centers from 2008. The LPDS is the coordinator of the EU FP7 EDGeS (Enabling Desktop Grids for e-Science) project with the aim of bridging service grid and desktop grid infrastructures and supporting their user communities from both academic and industry environments. The Laboratory also participates in other large international projects such as SEE-GRID-SCI, S-Cube, ETICS-2 and CancerGRID.

WEB2GRID

The WEB2GRID project aims to facilitate the commercial and non-profit exploitation of the EDGeS EU FP7 Desktop Grid project and the HAGRID project focusing on the Desktop Grid and WEB2 technologies. While WEB2 technologies assist to assure the resources in the desktop grid community on voluntary or settlement bases, the developed platform extended with desktop grid systems offers back-end infrastructure for the operational requirements of WEB2 portals and systems. By combining WEB2 with Desktop Grid technologies, WEB2 can extend its capabilities from the community

contents towards to shared services with the help of grid technologies. The project aims to develop the tools, interfaces and methodologies through which the above mentioned targeted services can be established both in closed (local desktop grid), and in open (global desktop grid) environments.

A comprehensive list of other projects can be found on the LPDS homepage.

Services

GASuC

The Grid Application Support Centre is an EGEE-supported project established in 2008. In close collaboration with application owners, GASuC provides assistance in gridification (i.e. porting legacy applications onto Grid infrastructures). The GASuC team identifies the suitable approaches and tools for the porting process, sets up realistic porting scenarios and organizes workshops and personalized training events for application owners.

Portals for Grids/VOs

The LPDS sets up Grid portal installations serving user communities of international multi-institutional grids and grid based virtual organizations. The list of actual P-GRADE Portal installations can be found on the P-Grade Portal homepage. The list of gUSE Portal installations can be found on the gUSE homepage.

OMNeT++

The LPDS carried out the gridification of OMNeT++, a public-source, component-based, modular, discrete event simulation environment. OMNeT++ is frequently used in a wide area of simulation applications due to its strong GUI support and embeddable simulation kernel. The P-GRADE Portal environment was successfully integrated with the OMNeT++ simulation framework to enable large-scale grid resources to the simulation user community, providing significant performance increase for OMNeT++-based simulations.

GSSVA (Security Assessment Tool)

Grid Site Software Vulnerability Analyzer is a monitoring tool which collects important information about the status of the grid machines, analyzes the information gathered and compares the results using an external information repository to find the security problems of machines. The tool can be set up and maintained easily, as administrators do not need to install the client side or configure the firewalls, and the information is visualized through a graphical user interface.

LPDS Training Room

Training

Since 2004 the LPDS has been operating the Central-European Regional Training Centre of EGEE and plays active role in providing grid trainings in Europe and worldwide. With the national and international trainings the Laboratory provides knowledge transfer and targets new users from industry as well as from science. The LPDS has organized and hosted grid summer schools in 2005, 2006, and 2007. In year 2008 the LPDS organized the world's largest grid training event, the ISSGC08 Grid Summer School, and also hosted several grid related training events.

Personnel

The Head of the LPDS is Prof. Dr. Péter Kacsuk. The Deputy Head of the LPDS is Dr. Robert Lovas. 1 DSc, 5 PhDs, and over 20 full or part-time members work in the laboratory.

National Grid Service

The **National Grid Service (NGS)** aims to help UK academics and researchers carry out their research by providing easy to use access to computational, data and other resources. It is funded by two governmental bodies, Engineering and Physical Sciences Research Council (EPSRC) and the Joint Information Systems Committee (JISC).

The NGS provides compute and data resources that are accessed through a standard common set of services, the Minimum Software Stack. NGS services are based on the Globus Toolkit for job submission and storage resource broker (SRB) for data management. NGS resources host a large number of scientific software packages, such as SIESTA and GAUSSIAN. As well as providing access to compute and data resources, the NGS also offers training (through the National e-Science Centre in Edinburgh) and Grid support to all UK academics and researchers in grid computing.

The NGS has entered its fourth year of production and its third phase with funding of £3M from EPSRC and JISC. This follows an upgrade of the resources at the core sites in September 2007.

With over 500 users and twenty five sites, the NGS is rapidly expanding, with a mission to provide coherent electronic access for UK researchers to all computational and data based resources and facilities required to carry out their research, independent of resource or researcher location.

The NGS provides a link for UK researches into the EGEE international e-infrastructure .

Background

The NGS grew out of requirements within the UK to develop a production quality Grid Computing service for use by academic researchers. Prior to the establishment of the NGS the UK e-Science program funded the development of the Grid Support Centre and later the Grid Operations Support Centre (GOSC). These were both focussed on providing support for end users in the use of and development of grid computing.

Following initial successes by the UK Engineering Task Force in developing the so called 'Level 2 Grid', which was an ad-hoc collection of individual institutes committing a variety of compute resources for use within a Grid Computing infrastructure, the NGS was funded to develop this into a full service. Initially called the ETFp Grid, or Engineering Task Force Production Grid, this soon became known as the National Grid Service (NGS). During phase 1, from October 2004 to September 2006, the NGS worked closely with the separately funded Grid Operations Support Centre (GOSC). The success of this closer collaboration led to a natural evolution of the Grid Operations Support Centre and the NGS becoming a single entity. By the start of phase 2 of the NGS, which commenced in October 2006, the activities of the NGS and the GOSC were harmonised

under the single project name of the NGS. The NGS is currently in phase 3 which will run until the end of September 2011.

Services

The NGS provides the following free-to-use services to UK academic researchers:

- The UK e-Science Certification Authority, which provides digital certificates to identify UK Grid users.
- A web portal, the applications repository, which provides a graphical method for submitting jobs to the NGS.
- A resource broker, which can determine the best site or cluster for a submitted job based on its requirements and the current workloads.
- The GSI-SSHTerm, a Java-based terminal used to give command-line access NGS resources using GSI-enabled SSH.
- Oracle databases.
- The NGS P-GRADE Portal, based on P-GRADE Portal technology, which enables the creation, execution and monitoring of workflows – composed of sequential and parallel jobs – on NGS and EGEE resources.

Sites

All NGS sites provide common interfaces for users to access resources, but they are differentiated on the access they provide to NGS users.

Partner Sites

The NGS consists of the following partner sites:

- Cardiff University
- University of Glasgow Computing Service
- University of Glasgow ScotGrid
- Lancaster University
- University of Manchester
- University of Oxford
- Queen's University Belfast
- Rutherford Appleton Laboratory (STFC/RAL)
- University of Westminster
- White Rose Grid (University of Leeds)

Affiliate Sites

Affiliate sites also provide resources to the NGS but with more conditions than partner sites. Sometimes they will not provide any resources to the general NGS pool, but simply provide access to their own users through the common NGS interfaces. Affiliate sites currently consist of:

- University of Birmingham (GridPP)
- University of Bristol
- Brunel University (GridPP)
- Durham University (GridPP)
- University of Edinburgh (ECDF)
- Imperial College London (GridPP)
- Keele University
- University of Liverpool(GridPP)
- University of Manchester (GridPP)
- University of Oxford (GridPP)
- Queen Mary, University of London (GridPP)
- University of Reading
- Royal Holloway, University of London (GridPP)
- University of Sheffield
- University of Southampton
- STFC SCARF
- STFC Ceramics and Minerals Consortium (Mott) VO
- University of York (White Rose Grid at York)

More affiliate sites are currently under discussion.

Research using the NGS

All academics in the UK are eligible to apply for a free account on the NGS. The science and research achieved using the NGS covers a diverse range of subjects, from the permeation of drugs through a membrane to the welfare of ethnic minority groups in the United Kingdom. Other applications include cases for modelling the coastal oceans, modelling of HIV mutations, research into cameras for imaging patients during cancer treatments and simulating galaxy formation.

As an example use of NGS, "The motivation, methodology and implementation of an e-Social Science pilot demonstrator project entitled: Grid Enabled Micro-econometric Data Analysis (GEMEDA). This used the NGS to investigate a policy relevant social science issue: the welfare of ethnic minority groups in the United Kingdom. The underlying problem is that of a statistical analysis that uses quantitative data from more than one source. The application of grid technology to this problem allows one to integrate elements of the required empirical modelling process: data extraction, data transfer, statistical computation and results presentation, in a manner that is transparent to a casual user".

Chapter 22

Grid File System and DRMAA

Grid File System

A **Grid File System** is a computer file system whose goal is improved reliability and availability by taking advantage of many smaller file storage areas.

Components

Current file systems contain up to three components: -File Table (FAT table, MFT, etc) -File Data -MetaData (user permissions, etc)

A Grid File System would have similar needs: -File Table (or search index) -File Data -MetaData

Comparisons

Because current File Systems are designed to appear as a single disk for a single computer to manage (entirely), many new challenges arise in a grid scenario whereby any single disk within the grid should be capable of handling requests for any data contained in the grid.

Features

Most file storage utilizes layers of redundancy to achieve a high level of data protection (inability to lose data). Current means of redundancy include replication and parity checks. Such redundancy can be implemented via a RAID array (whereby multiple physical disks appear to a local computer as a single disk, which may include data replication, and/or disk partitioning). Similarly, a Grid File System would consist of some level of redundancy (either at the logical file level, or at the block level, possibly including some sort of parity check) across the various disks present in the "Grid".

Framework

First and foremost, a File Table mechanism is necessary. Additionally, the file table must include a mechanism for locating the (target/destination) file within the grid. Secondly, a mechanism for working with File Data must exist. This mechanism is responsible for making File Data available to requests.

Implementation

With the recent advent of Torrent technology, a parallel can be drawn to a Grid File System, in that a torrent tracker (and search engine) would be the "File Table", and the torrent applications (transmitting the files) would be the "File Data" component. An RSS-Feed like mechanism could be utilized by File Table nodes to indicate when new files are added to the table, to instigate replication and other similar components.

A File system which incorporates Torrent technology (distributed replication, distributed data request/fulfillment) would likely be a good start for such a technology.

If both such systems (file table, and file data) were capable of being addressed as a single entity (ie: using virtual nodes in a cluster), then growth into such a system could be easily controlled simply by deciding which uses the grid member would be responsible (File Table and file lookups, and/or File Data).

Availability

Assuming there exists some method of managing data replication (assigning quotas, etc) autonomously within the grid, data could be configured for high availability, regardless of loss or outage.

Challenges

The largest problem currently revolves around distributing data updates. Torrents support minimal hierarchy (currently implemented either as metaData in the torrent tracker, or strictly as UI and basic categorization). Updating multiple nodes concurrently (assuming atomic transactions are required) presents latency during updates and additions, usually to the point of not being feasible. Additionally, a grid (network based) file system breaks traditional TCP/IP paradigms in that a File System (generally low level, ring 0 type of operations) require complicated TCP/IP implementations, introducing layers of abstraction and complication to the process of creating such a grid file system.

Examples

Current examples of high available data include: Network Load Balancing / CARP - splitting incoming requests to multiple computers, usually configured identically or as one whole Shared Storage Clustering / SANs - a single disk (one or more physical disks

acting as a single logical disk) is presented to multiple computers which split incoming requests. This is usually used when more computing power is required than disk access. Data Replication / Mirroring - multiple computers may attempt to synchronize data (usually point-in-time or snapshot based). Used more often for either Reporting (based on last snapshot) or backup purposes. Data Partitioning - splitting data among multiple computers. In databases, data is often partitioned based on tables (certain tables exist on certain computers, or a table is split among multiple computers at certain "break points")... general files tend to be partitioned either by category (category based folders), or location (geographically separated).

Grid computing would bring the benefits from many such solutions, if it were widely adopted.

DRMAA

DRMAA or **Distributed Resource Management Application API** is a high-level Open Grid Forum API specification for the submission and control of jobs to one or more Distributed Resource Management Systems (DRMS) within a Grid architecture. The scope of the API covers all the high level functionality required for Grid applications to submit, control, and monitor jobs to local Grid DRM systems.

In 2007, DRMAA was one of the first two (the other one was GridRPC) specifications that reached the *full recommendation* status in the Open Grid Forum.

Development Model

The development of this API was done through the Global Grid Forum, in the model of IETF standard development, and it was originally co-authored by:

- Roger Brobst from Cadence Design Systems
- Waiman Chan from IBM
- Fritz Ferstl from Sun Microsystems
- Jeff Gardiner from John P. Robarts Research Institute
- Andreas Haas from Sun Microsystems (Co-Chair)
- Bill Nitzberg from Altair Engineering
- Hrabri Rajic from Intel (Maintainer & Co-Chair)
- John Tollefsrud from Sun Microsystems Founding (Chair)

This specification was first proposed at Global Grid Forum 3 (GGF3) in Frascati, Italy, but gained most of its momentum at Global Grid Forum 4 in Toronto, Ontario. The development of the specification was first proposed with the objective to facilitate direct interfacing of applications to existing DRM systems by application's builders, portal

builders, and Independent Software Vendors (ISVs). Because the API was co-authored by participants from a wide-selection of companies and included participants from industries and education, its development resulted in an open standard that received a relatively good reception from a wide audience quickly.

Significance

Without DRMAA, no standard model existed to submit jobs to component regions of a Grid, assuming each region was running local DRMSs. The first version of DRMAA API has been implemented in Sun's Grid Engine and also in the University of Wisconsin–Madison's program Condor. Furthermore C, Java, and IDL binding documents have been made available.

Implementations

- Sun Grid Engine
- Condor
- Torque/PBS
- GridWay
- Xgrid
- EGEE (LCG2 / gLite)
- Platform LSF
- UNICORE
- Kerrighed Cluster Framework
- IBM Tivoli Workload Scheduler LoadLeveler
- SLURM

Language Bindings

- C/C++
- Java/JavaScript
- Perl
- Python
- Ruby

Other language bindings can be generated easily from SWIG, which was first used by the Perl binding.

DRMAA applications

A number of software solutions use DRMAA to interface with different resource management systems:

- tigr-workflow
- eXludus RepliCator
- GridwiseTech Grid Engine-Globus Toolkit adapter