PEER TO PEER VS CLIENT-SERVER : WHICH IS BETTER?

PETER KEOGH
DT228C
ADVANCED SOFTWARE DEVELOPMENT
D94140029

Contents

Part 1. Introduction

The choice of network architecture in the design of a ubiquitous computing system is crucial to the success of the system. The number of devices owned by a single user and has increased the amount of data produced and exchanged by systems. This has brought with it an increase load on networks exchanging this data. As ubicomp systems become more and more common this load is set to increase. Therefore determining how data will be exchanged and stored in a systems architecture is crucial to the success of Ubicomp. Ubiquitous Computing can be defined as computation, communication and sensing being integrated with the physical world. First described in the paper by Mark Weiser "The Computer in the 21st Century", it is a paradigm where computers would "disappear. They (would) weave themselves into the fabric of everyday life until they are indistinguishable from it."[?] The pervasiveness of computers being the defining characteristic of Ubicomp it's applications touch on and integrate a broad range of topics within computer science and electronic engineering. Ubicomp has the potential be utilised in any and every industry or part of life. With such a broad range of devices and potential applications it is impossible to truly say that one network architecture is better than another. Client-Server networking and Peer-to-Peer networking both have their strengths and weaknesses. We cannot look at one and say that it is better that the other. We can however look at them and determine which is better for a particular application. This paper will present information that will hopefully help guide the decision making process when developing an application.

The growth of the internet has seen the emergence of two typical network architectures in many applications. These architectures are the Client-Server model and the Peer-to-Peer model. The popularity of these architectures can be easily seen when we look at their respective shares of network traffic. Although in recent years video over IP has come to dominate internet traffic we can still get a sense of the popularity of these architectures. Fixed web and data, mostly client-server resources currently account for 5,625 Petabytes of data a month. Similarly P2P file transfer consumes 5,254 Petabyes of data worldwide per month.[?]

The Client-Server model is the driving force behind much innovation today, most obviously in Cloud Computing. Gmail, Facebook and Dropbox are all examples of a client-server architecture. It is an appealing architecture for online businesses as they can store and access customer information. The advent of faster and more reliable internet access in many parts of the world and the growth in the number of connected devices users have access to has been a driving factor. People want to be able to access their resources on all of their devices. Client server architectures make this possible. In years gone by users would have to back all their work up on a CD or floppy disk. They would forget to do so at their peril. Now one can work on a project locally and easily save it to a remote server to back it up. Using cloud services can allow the user to spend less money on their devices. "Thin" clients like Google's Chrome book have come to fill a new gap in the market. The processing and storage requirements of a laptop can now be transferred to the cloud.

P2P first emerged in the popular conscious in the form of Napster. P2P allows clients to connect to each other in a decentralised way. P2P has caused quite a stir in the past and continues to do so. It's most common use is file sharing, often illegally. The development of this technology has had a major impact on the entertainment industry. As a result they have lobbied continuously for it's regulation. In the case of file sharing peer to peer provides a greater availability of popular material. The more popular a resource is the more likely a peer is to be sharing it. As a result it can be more robust for a client server model, in which, once the server goes down the client is unable to avail of it's resources. On the other hand, if a resource is unpopular it is unlikely that many peers will be seeding that resource at any point in time. In this case a client server architecture might be preferable.

Both P2P and Client-Server are not restricted to being independent of one another. In fact, companies such as Spotify and Skype use a hybrid approach.

Part 2. The Architectures

1. Client-Server

In the Client-Server Architecture one always-on host known as a server services requests from other hosts known as clients. The clients exchange data with the server and the server provides the clients with services. The classic example of a client server architecture is in web applications. A client, usually a web browser sends a request to the server using the http protocol. The server then responds by sending a html page for the browser to render. In a client-server application clients do not directly communicate with each other. The can however communicate with each other via the server. For example, in the Facebook chat application a client can send a message to the server using AJAX which the server forwards to another client. To the user it appears the same as if the two clients were communicating directly. The server has a fixed IP address which is known by any client that wishes to connect to it. Increasing numbers of clients typically put more strain on a server. As one host needs to accommodate many requests it requires a robust infrastructure. A single host usually isn't enough to accommodate

multiple requests. Large applications usually make use of a cluster of hosts known as a
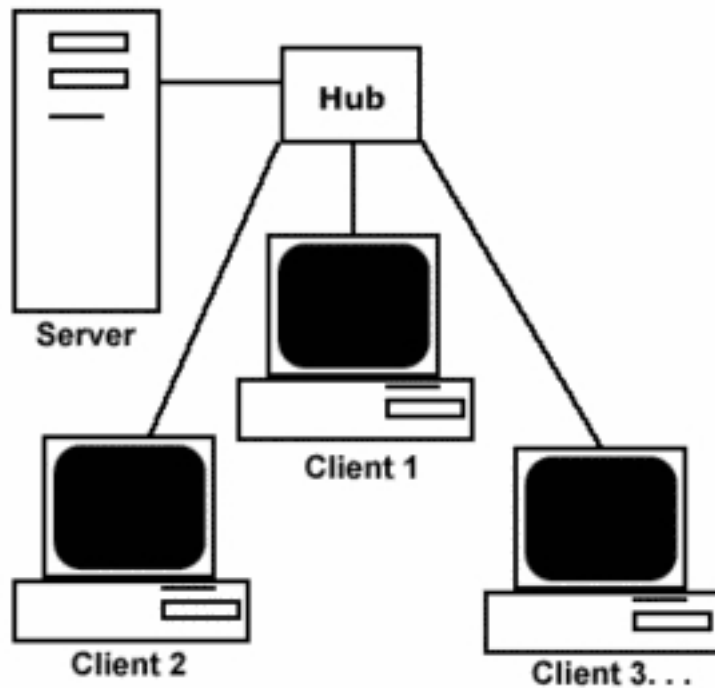
server farm.   [?]

2. Peer-to-Peer

In the Peer-to-Peer (P2P) Architecture nodes communicate directly and intermittently. The hosts in a P2P architecture are known as peers and there is little or no reliance on always on servers. P2P provides many opportunities the development of applications that require collaboration, decentralisation or high network throughput. This has lead to it's use in many popular applications such as Bittorrent and Bitcoin.
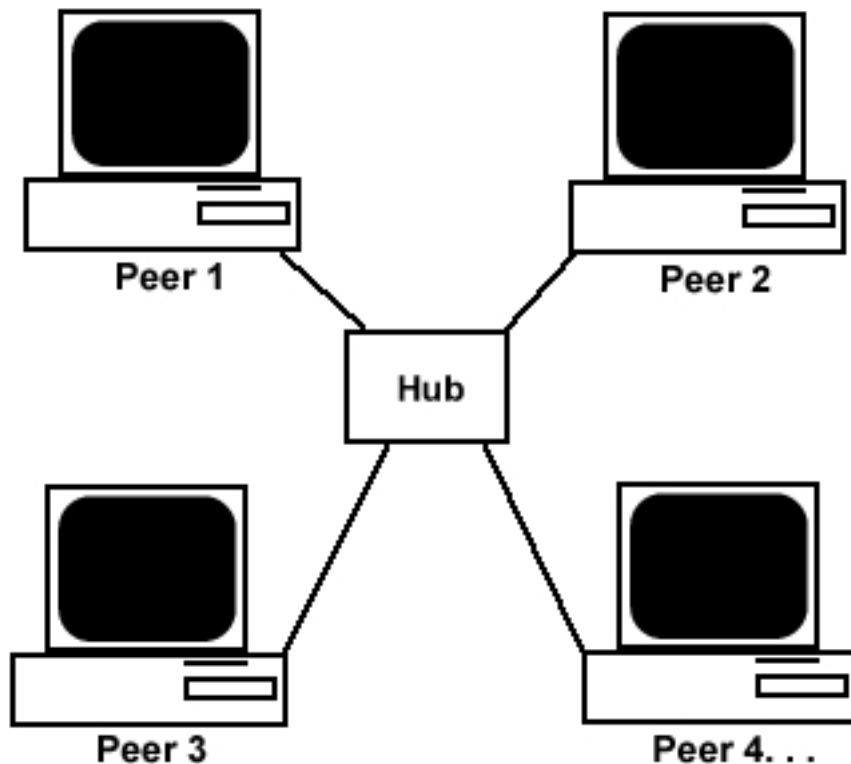
[?] A P2P architecture is 'self-scalable'. While the addition of an extra peer increases the workload of the system by increasing the number of requests the peer also increases the ability to handle that workload by providing the service to other peers. Securing P2P applications can be difficult due to their open and distributed nature. P2P can be broken down further into two subcategories of architecture; centralised, decentralised and hybrid P2P.

2.1. **Centralised P2P.** An example of this would be Bittorrent. In order to communicate with peers in the network a client must obtain information about the peers from a centralised server known as a tracker. Information about a tracker is obtained from a metadata file. A user needs to visit centralised server such as The Pirate Bay or Kick Ass Torrents in order to obtain this file.

2.2. **Pure P2P.** In a decentralised or pure P2P Architecture each peer acts as an indexed server. It also acts like a router by relaying queries between peers. An example of this would be Bitcoin or Gnutella 0.4.

2.3. **Hybrid P2P.** In hybrid peer to peer hosts with a good internet connection act as routers for other peers in the network who's connection isn't as strong. An example of this is Gnutella 0.6.

Peer-to-peer systems are most effective at storing large amounts of immutable data. They perform less well at storing mutable data.

Part 3. **Example Protocols**

3. Protocols

A protocol is an agreed method of communicating between applications. The protocols outlined below provide details of how current technologies using the architectures outlined above communicate. While these particular protocols may not be directly applicable to Ubicomp directly they provide a useful outline of how the architectures need to work in reality. Designers of Ubicomp systems are likely to encounter similar problems that system designers have encountered in the past. These protocols outline methods for overcoming these problems. An outline of these protocols will be useful in reasoning about the architectures as applied to Ubicomp systems later.

3.1. **HTTP Protocol.** The client server architecture is the typical architecture of web applications. In web applications the protocol used for transferring information between the client and server is the HTTP protocol. HTTP (HyperText Transfer Protocol) follows a request response cycle. The client makes requests to the server and the server responds to the request. The server doesn't send any information to the client if not prompted by a request. This has implications for applications using the protocol. If nodes wish to collaborate with each other or obtain realtime information about other nodes in the system they must "poll" the server by repeatedly sending requests to the sever. This is very inefficient as it results in significant network overhead and results in the server processing requests even when the state of the system remains unchanged. A solution for this problem has recently emerge the the standard of web sockets which allows servers and web browsers to set up a dedicated end to end bidirectional connection.

3.2. **Gnutella 0.4.** [**?**]
Gnutella was the first large scale decentralised P2P network of it's kind. It "builds at the application layer a virtual routing mechanism."[**?**] Early versions of Gnutella were truly decentralised P2P networks. No centralised servers were needed to facilitate the transfer of files from node to node. One of the implications of not using centralised servers in a p2p application is the challenge of keeping track of network nodes and what resources are available in the network. In order to obtain this information "network flooding" was used. Peers all maintain at least one TCP connection to other nodes in the system. The network flooding algorithm "floods" the network with requests; each node sends messages on to all of it's neighbours except the one it received the message from.

3.3. **Gnutella 0.6.** The flooding model used in Gnutella 0.4 had scalability problems. Searches were costly and usually terminated prematurely. Gnutella moved from a "pure" P2P paradigm to a hybrid approach. This new approach makes use of ultra peers and leaf nodes to create a hierarchically structured network. An ultra peer is a node that has a faster connection and elects to route more traffic for the network. The ultra peers also act as proxies for the leaf nodes. Searches become more efficient since searches only need to flood the ultra nodes instead of the whole network.

3.4. **Bittorrent Protocol.** [**?**] [**?**]

The bit torrent protocol is a P2P file sharing protocol that runs over http and tcp. The owner of a file hosts a "bencoded" metadata file (with the extension .torrent) on a server. A user downloads the metadata file and opens it in a bit torrent client. This client uses a url contained in the metadata file to connect to a "tracker". A tracker is a HTTP service that maintains information about the torrent, including a list of peers. The tracker may also provide stats for the torrent at an "scrape" url.

Once a client has obtained information from the tracker it can communicate with peers using TCP. Peers exchange the blocks defined in the metainfo file. Peers all have connections with each other that they maintain state info about. Peers exchange information based on these states.   These states are choked and interested. A peer may choke a client to stop it requesting data. A peer is interested if the client has something that the peer wants. If a peer is interested than it may unchoke the client and they can begin exchanging data.

Part 4. **Comparison of Performance**

Systems that are built with the Client-Server architecture and P2P systems are both examples of distributed systems. There are several challenges associated with distributed systems that provide interesting headings to analyse these architectures under. Performance, the amount of work a system can do. Scalability, the ability of the system to handle greater loads and to get bigger. Security can be defined as the ability keep malicious parties from doing things to or with our system we don't want them to do.

3.5. **Availability, Fault Tolerance and Scalability.**

Availability in client-server systems is dependent on the server architecture and network connection. If a firewall is preventing a user from accessing a server than to that user the server is unavailable.

A single server may be enough to handle requests for a service with few customers. However, more popular services need more complicated server architectures.

A service that works internationally can reduce the network latency of requests by having the same service running on multiple servers in locations that are closer to their customers. This also reduces the load on a single server.

Data can be managed through replication and partitioning. Data can be partitioned so that data can be used closer to where it is accessed. By replicating data the system becomes more fault tolerant.

All of the techniques for improving availability, fault tolerance and scalability in client-server architectures come at a cost. There's the financial cost of paying for more storage and processing power. There's also cost associated with developer time. Developing and deploying distributed server architectures can be time consuming and expensive. While replicating data improves the fault tolerance of the system they system now needs to make sure that the data is consistent throughout the system. There is also an additional cost associated with system administration.[**?**]

Availability in p2p systems is a function of the number of system nodes and the popularity of that resource. If a node wishes to request a resource that is popular in a network with a large number of nodes then it should have no problem getting it. It is

likely that there will be nodes geographically nearby that are hosting the information and are currently available for processing.

For example, if a Bittorrent user wanted to download Miley Cyrus's "Bangerz" album they would find 1143 peers hosting that file in a torrent file on The Pirate Bay. That resource would be highly available in the P2P system.[**?**]

If on the other hand a user wanted to read "Everyware, The Dawning Age of Ubiquitous Computing" they would find just 3 peers hosting the file. This file would be far less available than Cyrus's album. A user attempting to download this album would only be able to download the resource when at least one peer was active.[**?**]

If a user wanted to obtain a copy of the academic paper "Supporting Activity in Desktop and Ubiquitous Computing". It is unlikely that there would be a peer actively hosting this file. A Pirate Bay search revealed no results for this resource at this time. This illustrates just one problem of availability in P2P networks. Niche resources like academic journals are better hosted on a server. Real world institutions can't function based on what's popular.

One important thing to note is that when a user downloads "Everyware, The Dawning Age of Ubiquitous Computing" they increase the availability of the resource. If they choose to host the files then there would be 4 peers for the next user to download from. The availability of a resource increases in proportion to the nodes in the system with that resource.

P2P is also extremely fault tolerant if the network is big enough. If a cup of coffee spills over one of the nodes in the Miley Cyrus network the downloading peers are unlikely to notice a performance difference. P2P is at it's most fault tolerant when it is completely decentralised and has no reliance on centralised trackers.

P2P networks are highly scalable as a result of all of the above since an increase in the number of clients consuming network resources increases the ability of the network to handle it's load. P2P networks can be described as "self-scalable".

3.6. **Security.** One obvious security flaw in P2P is that machines can be used to form a bot-net without the owner knowing. This bot-net can then be used to perform a Distributed Denial of Service Attack on a server.

Part 5. **Notable Systems**

3.7. **IFTTT.** If This Then That (IFTTT) is a web application that allows users to link web services together in "recipes". It allows people with no programming experience to construct conditional statements using powerful web APIs. IFTTT allows a user to coordinate actions across many popular web applications such as Twitter, Dropbox, Evernote, Linkedin etc. The user can also set actions based on conditions of their device. In the simplest sense a "recipe" can be defined as a conditional statement and a corresponding action. If a condition is true in one system then take an action defined in another. Users can share these recipes with other users. Users can defined actions based on the location of their android phone. This allows them to define the kind of context aware actions that would be typical in Ubicomp. At the time of writing the eighth most popular recipe on the site is "Mute my Android device when I get to the office turn on vibrate". Perhaps even more impressive is the recipe that takes advantage

of Philips Hue to "Turn on your lights when you're near home". This application nicely illustrates the type of ubiquitous computing applications that the client server model is suited for. Having one centralised computer (in this case a server) coordinating actions across multiple devices is a design pattern likely to arise in the development of Ubicomp systems. It is particularly suited to coordinating actions in one device many users situations.

3.8. **Bitcoin.** Bitcoin is a digital currency that utilizes the P2P architecture to remove the need for centralised institutions in order to conduct financial transactions online. Commerce has traditionally depended upon an institution ... Nodes in the network "mine" bitcoins by computing cryptographic hashes in order to validate and record the transactions of other bitcoin users. It may take some time before bitcoin and other cryptocurrencies are widely adopted. Currently it's value it's value is highly volatile.

3.9. **BitTorrent Bundle and iTunes.** iTunes came to fill the online music distribution void left after Napster was shut down back in 2001. Attempts are being made to leverage the technology behind bittorrent to put more power into the hands of artists. "Tomorrow's Modern Boxes" by Thom Yorke was released on the 26th of September 2014. It was the first paygated BitTorrent bundle. (Yorkes band "Radiohead" gave their album"In Rainbows" away essentially for free back in 2007. U2 did a similar thing recently via iTunes that was met with mixed reviews.)

3.10. **BitTorrent Synch and Dropbox.** Alternative to cloud synchronisation.

4. Cloud Computing

Thanks to the spread of context aware smart devices users are always aware.

4.1. **Ubiqutous Computing Applications.** Ubiqutous computing can be defined as... When designing ubicomp applications we have to choose an application architecture. The client-server, p2p and hybrid approaches server different use cases.
Context aware applications produce large amounts of data.
Network Throughput requirements Storage Requirements Data Access - context aware systems need to be able to access data. Local storage vs server.
Data Analysis Requirements
How do sensors communicate?
Data mining is far more difficult in P2P than client server. By keeping data on a central server it's far easier to analyse that data. This is the thinking behind data warehousing; Store all of the data at a central location and analyse it there. However this may not be feasable given the amount of data we are collecting. This has given rise to the research area of Distributed Data Mining.

Part 6. **The architectures in the context of ubicomp**

It's easy to compare P2P and the client-server model under traditional standards. While it is tempting to believe these models to be analogous to these architectures in Ubiquitous computing, this is not necessarily the case. The reason for this is because the technologies associated are in different stages of evolution.

The choice of architecture impacts the development of technology. A prime example of this is in the development of the World Wide Web. Web browsers are no longer simple programs that display HTML. Fast Javascript engines have been developed like Chrome's V8. Graphics processing has been made possible in the browser by WebGL. The HTML5 specification is being implemented by vendors and as a result browsers have more access to system hardware than ever before. One of the motivations for this has been to reduce the load on servers. Why should a server waste precious processing power assembling HTML views that can be created on the fly by a browser.

Contrasting this with sensor nets. Sensors, particularly on the nano scale have little processing power. While one day they may increase their processing. Storage of other sensors information. Knowledge of another sensors output. Distribute decision making computation amongst smart sensors.

In a ubiquitous computing context

5. Peer to Peer

5.1. **Tiny Torrents.**

Part 7. **Social and Political Issues**

The client server model has brought with it challenges for law makers. If an Irish customer's data is stored in a server in the USA, what laws govern that data?

6. Censorship

The client server model used in traditional web applications is far easier for governments to censor than peers to peer networks. Governments can force internet service providers to block IP addresses, redirect Domain Name requests to bad IP addresses or force a hosting company to remove a service from it's servers. This is the case in countries like China where the government's "Great Firewall" prevents Chinese netizens from accessing sites like Facebook, Google and Twitter. These services have come to be replaced by Chinese equivalents like Baidu, Renren and Sina Weibo. These sites are subject to heavy regulation from the Chinese government. In Ireland too there is a certain amount of internet censorship. Eircom has been ordered to block sites associated with illegal file sharing such as The Pirate Bay. The centralised nature of the client server model makes it easy for governments to censor applications built on it. One of the reasons for this is because it is easy to shutdown a system with a single point of failure. One of the methods of bypassing such censorship is by leveraging the p2p model. In January 2014 The Pirate Bay announced that it was developing "a browser-like client to circumvent censorship, including domain blocking, domain confiscation, IP-blocking." The software works like both a web browser and a BitTorrent client. Web application files that would usually be stored on a centralised server are instead shared among peers. The application has it's own DNS system in which it uses a fake DNS that links to a unique and verified public key.[**?**]

7. Privacy, Spying and Hacking

However, this flexibility comes at a price. Privacy is certainly an issue. While most people consider these services free they do come at a price. It's an internet cliche now that "If You're Not Paying for It; You're the Product". Privacy is a hot topic these days. In light of the revelations of Wikileaks and people like Edward Snowden the public have started asking if the convenience of these services is worth trading their privacy for.

Client server technologies offer us the opportunity to automatically back up data to a cloud account. Services like Apple's iCloud allow users to synch their iPhone data automatically. Many users simply set this up and don't think about it again until something goes wrong with their device. Users may be unaware of or not give much thought to what they are synching. Having so much personal data automatically synched to a cloud service provides a gold mine for hackers. Recently many hackers have release personal photos of celebrities such as Jennifer Lawrence that were synched to iCloud.

8. Copyright

P2P networks such as bit torrent have been used for illegal file sharing since Napster. The entertainment industry benefitted for years as a result of technology and it finally turned on them. SOPA PIPA. The size of P2P networks and the amount of file sharing that happens within them makes it extremely difficult and expensive to prosecute.

9. Free Riding

[?]
One of the issues in P2P networks is the social dilemma of "free riding". In a file sharing system the availability of a resource is dependent on peers sharing that resource. However, if a peers choose not to share the resource they can improve their network bandwidth and download more resources for themselves. A consequence of this is that as fewer and fewer peers are sharing it becomes easier to prosecute file sharers.

9.1. **Net Neutrality.**

Part 8. **Conclusions**

References

[1] Cisco. Cisco visual networking index: Forecast and methodology, 20132018. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html`. Accessed: 15-10-2014.

[2] TechSoup. Client server. `http://www.ictknowledgebase.org.uk/uploads/RTEmagicC_clientserver.gif.gif`. Accessed: 15-10-2014.

[3] TechSoup. Peer to peer networks. `http://www.ictknowledgebase.org.uk/peertopeernetworks`. Accessed: 15-10-2014.

[4] Matei Ripeanu. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 99 – 100, August 2001.

[5] Bram Cohen. The bittorrent protocol specification. `http://www.bittorrent.org/beps/bep_0003.html`. Accessed: 15-10-2014.

[6] Various. The bittorrent protocol specification. `https://wiki.theory.org/BitTorrentSpecification`. Accessed: 15-10-2014.

[7] The Pirate Bay. Miley cyrus, bangerz. `http://thepiratebay.se/torrent/8996596/Miley_Cyrus_-_Bangerz_(Deluxe_Version)__2013-Album_`. Accessed: 15-10-2014.

[8] The Pirate Bay. Everyware, the dawning age of ubiquitous computing. `http://thepiratebay.se/torrent/4506989/Everyware__The_Dawning_Age_of_Ubiquitous_Computing`. Accessed: 15-10-2014.

[9] Daniel Hughes, Geoff Coulson, and James Walkerdine. Free riding on gnutella revisited: The bell tolls? *IEEE DISTRIBUTED SYSTEMS ONLINE*, June 2005.