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

PETER KEOGH

Contents

1. Introduction

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. A powerful for the popularity of these architectures is their respective shares of network traffic. HTTP, the protocol of the client server accounts for approximately 46% of all traffic. P2P comes in at a close second consuming 37% of total traffic. The Client-Server model is the driving force behind much innovation today, most obviously in Cloud Computing. Gmail, Facebook and Dropbox are all classic examples of a client-server architecture. It's appeal lies in it's flexibility and robustness. The advent of faster and more reliable internet access in many parts of the world and the growth in the number of users has been a driving factor. People want to be able to access their resources on all of their devices. This ”new” model offers robustness. 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 architecture has also influence the development of client technology. 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. 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. 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? Censorship is and freedom of expression are also at play here. The internet provides the public with the power to communicate and organise effectively. The ability to spread information effectively empowers people to overthrow oppressive regimes. However, centralised servers can be easily blocked by tactics such as DNS poisoning. An authoritarian governments control over servers allows it to effectively control it's population's minds. The Peer-to-peer architecture on the other hand ... Napster, Bittorrent, 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. 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. 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.

2. The Architectures

2.1. **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 eachother. The can however communicate with eachother 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 accomodate many requests it requires a robust infrastructure. A single host usually isn't enough to accomodate multiple requests. Large applications usuallly make use of a cluster of hosts known as a server farm.

2.2. **Peer-to-Peer.** In the Peer-to-Peer (P2P) Architecture hosts communicate directly and intermittently. The hosts in a P2P architecure are known as peers and there is little or no reliance on an always on servers. P2P provides many opportunities 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 request 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.

2.3. **Hybrid Peer-to-Peer.**