Increasing Reliability and Efficiency via Distributed Data Dissemination

Francine LALOOSES
The MITRE Corporation
Bedford, MA 01730

and

Howard KONG
The MITRE Corporation
Bedford, MA 01730

ABSTRACT

As enterprises become more distributed, disseminating data in a timely manner between nodes of the enterprise becomes increasingly critical to doing business. Peer-to-peer technologies are one promising avenue for disseminating data to distributed nodes, while also limiting performance impact. This paper discusses Project MONSOON, a system which seeks to develop reliable and efficient distributed data dissemination within an enterprise, while also preserving data integrity.

This paper discusses the motivation for developing the Project MONSOON system and the architecture behind it. The evidence presented in this paper suggests that the peer-to-peer approach not only increases performance as more nodes are added, but the overall reliability of the network is increased as additional nodes participate in the dissemination. This conclusion is reinforced by simulations presented in this paper.

Keywords: Data dissemination, peer-to-peer, transfer protocols.

1. INTRODUCTION

Many network communications infrastructures contain various points of congestion, concerns of reliability, and an ever increasing demand for capacity. Bandwidth conditions vary throughout the whole spectrum, with outer edge nodes experiencing various frequencies of disconnect and the poorest throughput. Current data dissemination techniques do not always provide an efficient, reliable method of delivering content over these networks.

There is an essential and urgent need to provide critical information from data sources to consumers in the Net-centric environment. Consider, for example, the mass email you check every day at work containing large attachments, like a movie or audio file, high resolution pictures, etc. On average, an employee sends and receives as many as 200 emails each day. Business email volumes sent annually exceeded one billion gigabytes in 2003 [1] – a huge demand for the infrastructure. In the Net-centric environment, information is continuously increasing, be it in volume, size, frequency, "richness", or usage, and the technologies that support this must keep up.

This research investigated the use of software to increase content delivery performance, reliability, and efficiency. We developed a reusable, component-based system, designed to communicate with and reside within a Net-Centric architecture capable of efficiently managing data flows, bandwidth usage, failover recovery, and distributed data stores.

Our research is looking to answer what the common classes of data dissemination requirements are today, how and where can peer-to-peer technologies be effectively applied to our sponsors' needs, and how does a peer-to-peer system compare to traditional client/server systems? This paper will highlight our conclusion on whether the peer-to-peer (P2P) approach is appropriate for reliable and efficient data dissemination.

2. BACKGROUND

Before we begin, let us review some background information on the data dissemination techniques that we compared, which include pure File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), and the BitTorrent P2P protocol.

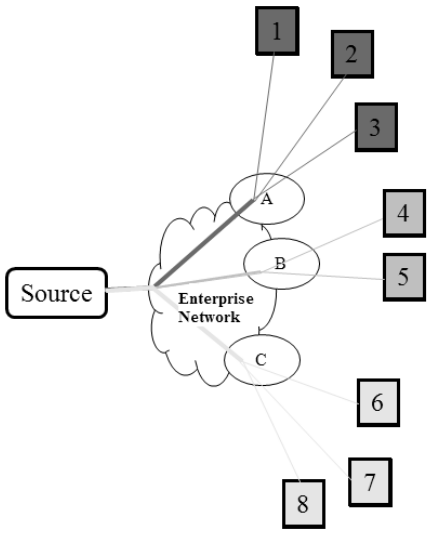


Figure 1: Traditional Pure FTP

Pure FTP is a traditional client/server protocol used to disseminate data from one source, typically called the server, to another computer, called the client, through a network. In Figure 1, the server has to transfer data to three groups of clients, shown in different color patterns. The patterns here refer to the set of data each group receives; for example, the entire dark group (consisting of clients one, two, and three) receives the same data, and similarly for the other two groups. The source does this by first disseminating to Client 1, then to Client 2, etc. (we are simplifying the situation here for illustration purposes, but multithreaded servers exhibit the same behavior since the data is interleaved on the wire.) Note even in this simple example, the bottleneck for this system is at the server, and the network near the server, which must carry the bulk of the workload in the FTP model. The key benefits to this model are its simple architecture and well-known behavior. However, the drawbacks include: single point of failure, since all the data must be transferred one at a time from a single source to each client; centralized processing, control and knowledge; and the observation that increasing demand increases resource requirements.

HTTP is another client/server protocol, and it exhibits similar behavior as described for FTP above, with the exception that while FTP can originate from either the server or client, HTTP usually originates with a client request followed by a server response.

The last protocol we looked at is BitTorrent, which is a P2P file sharing communications protocol. In this protocol, data is shared among peers, each of which can be a source and a consumer for every other peer on the network. [4] A BitTorrent peer can be defined as one instance of a BitTorrent client running on a computer on the Internet that you connect to and transfer data.

P2P data dissemination starts with the source, just like in the client/server case. But BitTorrent divides the data into a set of small, identically-sized units called pieces, and when peers request data, they request not the whole data, but a *random piece*. In Figure 2, Peer 1 requests Piece J from the source, and the source fulfills the request. The system recognizes that there are now 2 copies of Piece J on the network, one at the source, and one at Peer 1. So when Peer 3 requests Piece J, Peer 1 can fulfill that request, and the source can be freed to send other pieces to other peers. Similarly, when Peer 2 requests Piece J, it can get it from either the source, Peer 1, or Peer 3. In this way data is shared, and the load of the work of distributing that data is also shared. The system "load-balances" itself automatically by the fact that the slowest connections tends to less data, while the fastest connections tends to serve more. In Figure 2, there are three hops from the source to Peer 1, and one hop from Peer 1 to Peer 3, and again one hop from Peer 3 to Peer 2. So the top three peers tend to share what they have among themselves locally, and tend to only get original pieces from the source. The "automatic load balance" and the favoring of local fast connections are key to the efficiencies we will see in the BitTorrent peer-to-peer model.

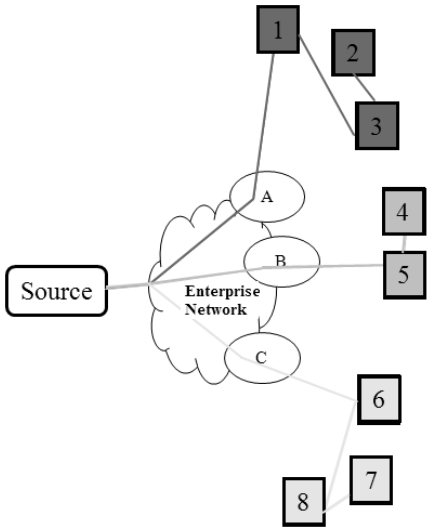


Figure 2: P2P Network Characteristics

The drawbacks of BitTorrent are its complex architecture, protocol, and behavior. The benefits include its ad-hoc adaptability; scalability—since all peers contribute resources; and resilience—as we will see later. This is a highly distributed architecture, and the effects of network problems are decreased, by scale and redundancy.

3. TECHNICAL APPROACH

Our hypothesis is that there exists a reliable and secure approach in distributed data dissemination to send data efficiently to its peers. Our initial approach is focused on addressing BitTorrent as a data dissemination protocol

and demonstrating the performance of several protocols. Our work program is divided into four general tasks:

1. Distill common data dissemination characteristics for consumer requirements and constraints—so that we can answer questions about data dissemination in a general way, one that can benefit multiple sponsors.
2. Build a prototype to meet key requirements.
3. Compare performance characteristics of the peer-to-peer prototype with more traditional client-server technologies—these are hands-on laboratory tests designed to compare relative performance and reliability of the protocols.
4. Build simulation models to test scenarios "in the large"—these models enables us to scale up the node counts that would be prohibitive with physical equipment.

Data Dissemination Characteristics

Our first task was to find the common, basic constraints that govern data dissemination. The main reason we did this was to look at data dissemination in a general way across multiple consumers. Furthermore, by distilling the problem space into smaller, more manageable components, we can gain a better understanding of the factors that affect data dissemination performance.

We identified five broad categories of constraints that govern data dissemination, depicted in Figure 3.

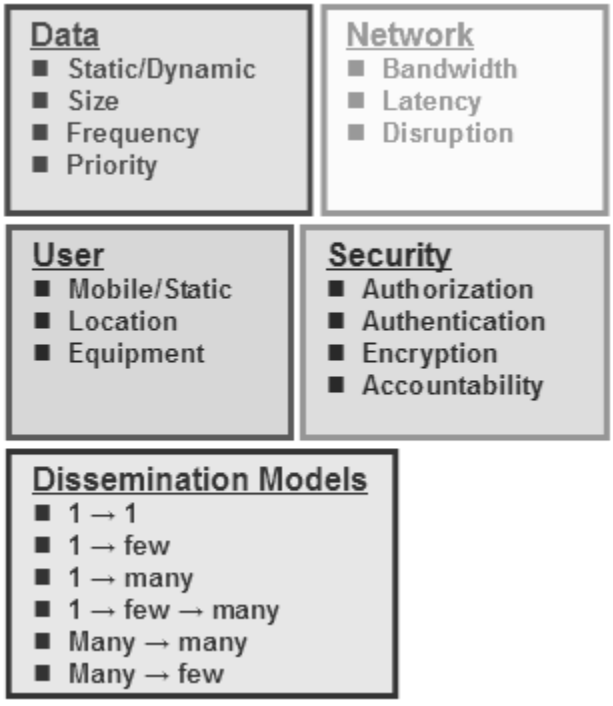


Figure 3: Common categories of constraints that govern data dissemination.

In addition to these five categories, we found it helpful to think of data dissemination in 3 separate problem spaces, as illustrated in Figure 4. The three problem spaces are:

- small data (such as email, metadata, XML)
- large data (such as rich media files, large imagery files)
- streaming data (such as rich streaming media, live video feeds)

What we found most important about these problem spaces is that different network application protocols are designed for each space, and no single protocol is a panacea in all spaces. In this paper, we concentrated mostly in the large data space.

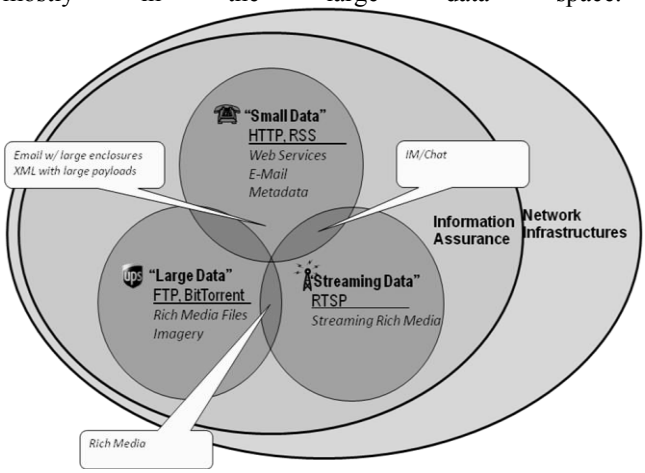


Figure 4: Data dissemination problem space

Prototype and Design

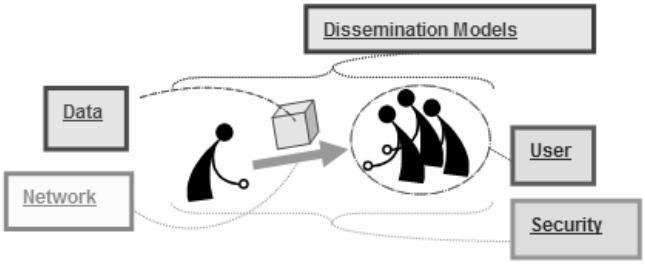


Figure 5: The bigger picture

Our second task was to build a prototype. We again wanted to broaden our scope to accommodate multiple consumers. Realizing that each consumer will have different emphasis for requirements and constraints, we decided to build our prototype using a plug-in architecture—something that can accommodate sharing code among customers as well as allow customization and independent development to meet each consumer's specific needs. As can be seen in Figure 6, we built our prototype on top of the Eclipse RCP, a proven architecture built by IBM and maintained by eclipse.org.

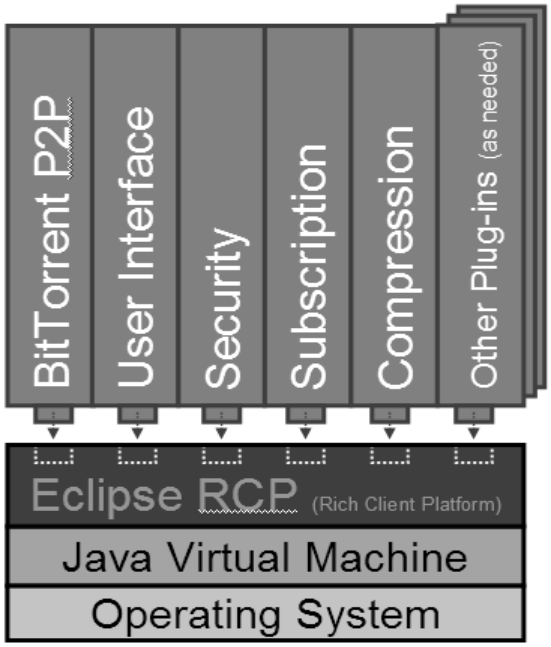


Figure 6: Monsoon high level architecture

Peer-to-Peer versus Client/Server

Our next task was to conduct laboratory experiments comparing the performance of the BitTorrent peer-to-peer system with that of traditional client/server systems, such as FTP and HTTP.

What we found was that while FTP and HTTP are simple protocols that are easy to understand, they present a single point of failure at the server. Furthermore, as the number of clients increase, the server becomes *the* bottleneck, and its capabilities (e.g., CPU, memory, network bandwidth, etc.) must be increased in order to meet the increased demand. The problem is that the singleton server is the node in the network doing almost all of the work—it is the supplier of information to the clients who are the consumers. Furthermore, the data dissemination is centrally managed at the server—none of the client nodes know about other client nodes.

On the other hand, peer-to-peer systems, such as BitTorrent, make every node both a server and a client, and therefore each node becomes both supplier and consumer. Furthermore, nodes are aware of each other, and are free to choose the best ones to exchange data with. With redundant storage of data, and the benefit of each additional node bringing in new supply as well as new demand, bottlenecks disappear by distributing the work throughout the entire system.

Modeling and Simulation

Since we cannot physically scale up to hundreds of machines for lab tests, our last task is to build a software simulation in the AnyLogic simulation tool [8] to model large numbers of nodes. Figure 7 illustrates a small 15-node version of our simulator [*], running FTP and BitTorrent tests. We found that, just as the lab tests suggest, peer-to-peer performance scales very well compared to client-server systems.

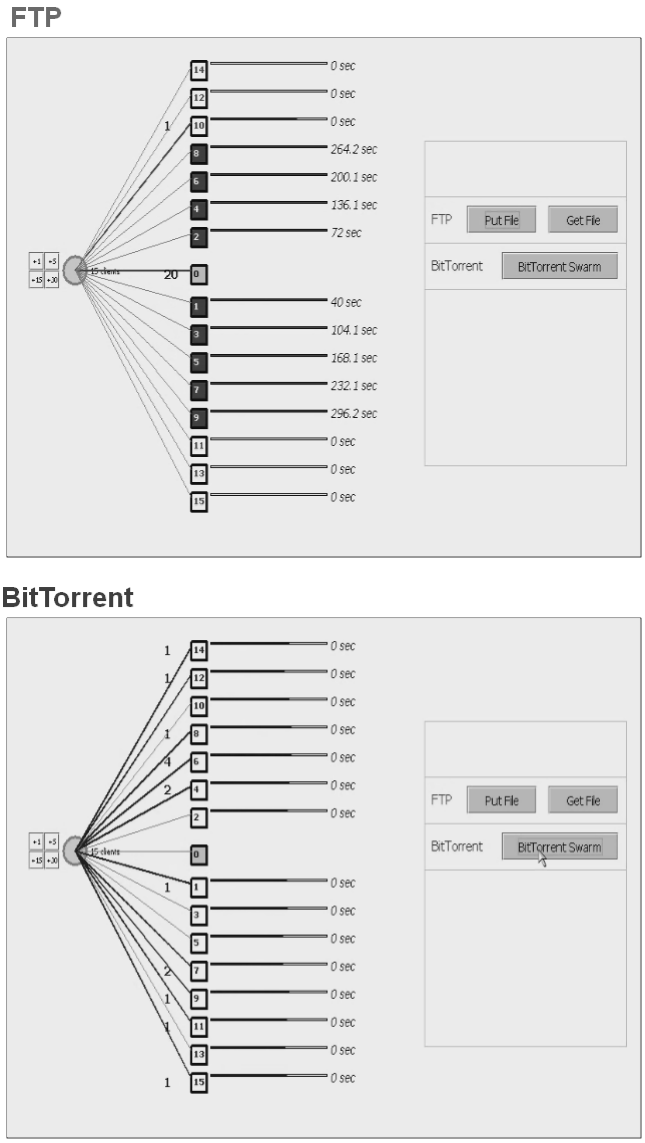


Figure 7: Two models simulated using AnyLogic.

4. FINDINGS

We compared the performance of P2P systems to client/server in Figure 8. Here, a relative small (100MB) file is disseminated to multiple nodes, in a "flat" network with all of the nodes connected together to a single Cisco switch. The x-axis represents the number of nodes, while the vertical axis displays the average total time to complete the dissemination.

As you can see, when there is a single client, HTTP is fastest for this size data, followed closely by FTP, with BitTorrent coming in last. However, with each new additional node, both FTP and HTTP slows down

[*] We've run the simulator with up to 1024 nodes.

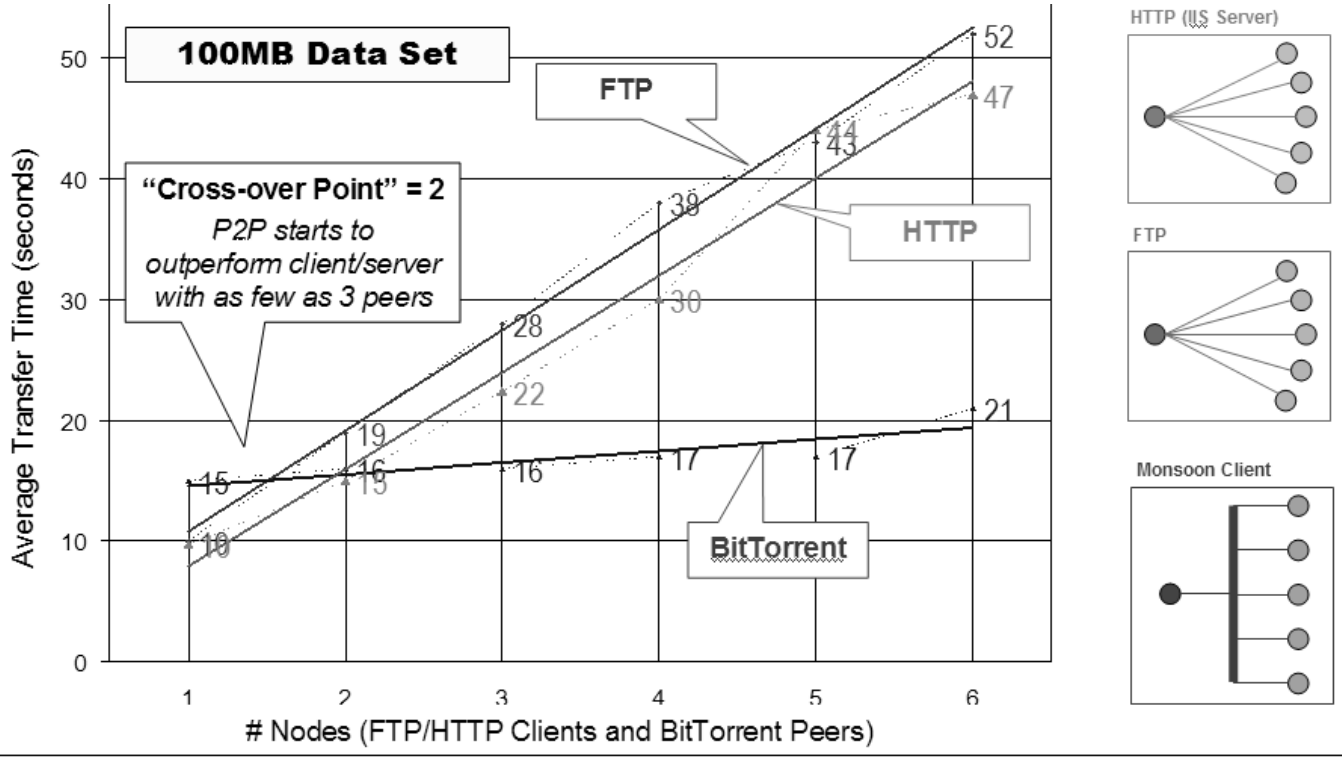
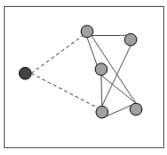


Figure 8: Performance Comparisons of FTP, HTTP, and BitTorrent

proportionately, since the server now must do that much more work. BitTorrent peers however, bring both supply and demand to the dissemination, and the average time does not increase very much at all. In fact, the cross-over point—that is, the point at which peer-to-peer performance matches client-server performance is at two nodes. This means that a peer-to-peer system is at least as fast as a client-server system when information needs to be disseminated to more than 1 other node. Of course, this benefit increases significantly with each new additional node.

The peer-to-peer approach also benefits overall reliability, since each peer is also a server to other peers. And since each peer can choose to exchange data with new peers if communication to old peers is disrupted, local disruptions only have local effects. In fact, we found that the more peers participate in the dissemination, the more reliable the overall system becomes, because the "server" serving a piece of information is redundantly duplicated across the entire network of consumers for that piece of information. The more tests we ran in the lab with larger data files, the better performance results BitTorrent scored.

	# peers	Disrupted at 25%	Disrupted at 50%	Disrupted at 75%
FTP	1	25%	50%	75%
	2	42%	73%	97%
	3	49%	80%	100%
	5	52%	77%	100%



Peer-to-peer decreases total dependence on a single server

Figure 9: Disruption tolerance findings

Figure 9 illustrates a chart where we disrupted the network by physically unplugging the connection to the server when the file transmission completion was 25%, 50%, and 75%, respectively.

As discussed previously, the case of having only one peer is the pure FTP model, highlighted in red. Using the chart, one can conclude that as the number of peers was increased, there is a greater chance of completing your file transmission even though the network was disrupted as a result of storage redundancy.

5. Conclusion and Future Work

The initial design demonstrated a need for some additional research and development to peer-to-peer systems. No single technology can solve all requirements in the data dissemination problem space illustrated in Figure 4. We also found that P2P works well for static large data sets sharing data among at least a few peers. We found it is resilient on unreliable networks.

As we continue to become more Net-centric, there is a need to address data dissemination issues, such as transmission latency, data integrity, and [6] security. When dealing with dissemination of large data sets, there is a large security risk to maintain the data integrity of the

network. Security risks and mitigations will need to be investigated in the future.

6. ACKNOWLEDGMENTS

Thank you to the Monsoon team for their following contributions: Dan Potter for his software development, Amanda Martino for her tedious laboratory tests, and Ron Couture for his security investigation. A final thank you goes to the MITRE Corporation for acknowledging the need for additional research in this area (Approved for Public Release; Distribution Unlimited. Case Number 08-0884. ©2008 - The MITRE Corporation. All rights reserved).

7. REFERENCES

[1] J. Raikes, "An Information Worker's View of Microsoft Office Evolution", 2005. http://office.microsoft.com

[2] F. Harrell, et al, "Survey of Locating & Routing in Peer-to-Peer Systems", University of California, San Diego, December 2001.

[3] R. Hasan, et al, "A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems", In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II - Volume 02, 2005.

[4] B. Cohen, "Incentives Build Robustness in BitTorrent", Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, May 2003.

[5] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving BitTorrent Performance", Carnegie Mellon University and Microsoft Research, Microsoft Technical Report MSR-TR-2005-03, February 2005.

[6] D. Wallach, "A Survey of Peer-to-Peer Security Issues", Rice University.

[7] R. Rodrigues, B. Liskov, and L. Shrira, "The Design of a Robust Peer-to-Peer System", In *10th ACM SIGOPS European Workshop*, (Saint Emilion, France), September 2002.

[8] AnyLogic – Multi-Method Simulation Software, http://www.xjtek.com/.