File Distribution Protocol

# Analysis: Multicast vs Peer to Peer

* Multicast requires only transmitting data once for many to receive.
* Multicast has extremely large limits.
* Multicast should have the best file transfer time on a perfect communication channel
* Multicast does not transit over public internet
* On a local subnet multicast should perform the best.
* Some form of peer to peer should scale relatively well. May take sometime to ramp up though.
* On a local subnet peer to peer may not make the best use of bandwidth due to Ethernet multicast etc.
* Peer to peer will work over the internet
* Peer to peer may be able to cope much better with the central host going down if other peers have the file.
* Peer management in P2P might be complex. Old style Bit Torrent used trackers, a centralised server. New Bit Torrent uses DHT’s and other types of local peer discovery. Very complicated though.
* Peer to peer can handle varying peer speeds. Multicast will transmit at one fixed speed. Slow receiver/consumer.
* Chunked / unchunked P2P. Chunked will reduce the lag time between a peer downloading data and then beginning to upload the same data to another node.

Peer to peer will be able to scale better for retransmitting chunks (e.g. if packets are lost or a host goes down for a short period of time). With multicast either the host could loop through data continuously, or receiving peers would ask the host to re-send chunks, but this may require the host to be connected to every receiving peer which isn’t so scalable. Also, when resending every peer will receive the chunk again, rather than only the peer which needs it.

## Performance Predictions

To analyse the two methods further I decided to calculate the expected performance of Multicast and Peer-to-Peer. The times used in these calculations are based on a 1GB file being transferred over a 100Mbit/s connection. Every node is assumed to be able to upload and download 100Mbit/s at the same time.

Figure 1.0 – Graph predicting file transfer time for different architectures on a lossless network. Note: Lines for Multicast and P2P (Chunked) largely overlap.

Figure 1.0 visualises a number of important facts:

1. All architectures take the same amount of time if number of nodes (n) =1.
2. Serial transfers do not scale well as n increases.
3. P2P (unchunked) demonstrates a log(n) relationship.
4. Multicast’s performance is not related to the number of nodes.
5. The performance of P2P (chunked) is modelled to be nearly identical to Multicast’s.

The last point, chunked P2P performance nearly equalling multicast performance was surprising. However I it stems from the assumption that all nodes will be able to transmit and receive at 100Mbit/s, regardless of other traffic on the network. Unless an extremely high performance switch/router is managing the subnet this is an unlikely scenario.

Though I have no mathematical predictions, I suspect in reality chunked P2P would provide a distinct improvement on unchunked P2P, but would not perform as well as Multicast on an average subnet due to the higher total network resources required.

## Tests performed

Before I select which architecture to use in my file distribution system I decided some tests were needed to verify my predictions and assumptions.

### Important considerations

* Ethernet vs WiFi for congestion. All hosts should be connected via Ethernet.
* Ethernet multicast may cause a considerable benefit.

### Test 1: Client-Server Serial transfer

The most basic file distribution system used as a benchmark for the other two methods. A single host machine will serve the files to connecting clients.

I will measure using iperf (UDP mode) to attempt to avoid packet corruptions or TCP ramp up artefacts impacting results.

### Test 2: Multicast transfer

Using iperf again, this test will measure the time it takes for all clients listening to the group to receive all of the data.

### Test 3: Basic P2P simulation

A bash script will set up iperf connections between a chain of hosts as expected in the chunked P2P model. This should confirm or deny my prediction that Chunked P2P will not perform quite as well as shown in Figure 1.0.

### Test 4: BitTorrent Peer to Peer

Using the most complex software of the 4 tests, this experiment should test whether the performance of a pre-existing P2P file distribution solution matches up with my expectations (Somewhere between Unchunked and chunked P2P in Figure 1.0). I will be using Herd (Garrett & Gadea, 2014) as a P2P system for this test. Herd uses BitTorrent underneath to perform the file transfers.

|  |  |  |
| --- | --- | --- |
| Description | Expected outcome | Actual outcome |
| Test 1: Serial Transfer | Total time proportional to number of clients |  |
| Test 2: Multicast | Total time does not change with number of clients |  |
| Test 3: Attempt iperf p2p simulation? | Total time follows a logarithmic-like curve against number of clients. |  |
| Test 4: Herd/BitTorrent | Total time follows a logarithmic-like curve against number of clients. |  |

# Design: **Multicast OR Peer to Peer System**

# Implementation

**Would be nice to have a program which could test multicast, serial and P2P all with chunked + un-chunked selective reject?**

**Reliability**

# Extras

Compression? File content / file extension testing.

Security – Prevent malicious peers / malicious network. Trust host (or “.torrent file”).

* Compression

# Issues Encountered

# Evaluation

## Functionality

## Code Quality

# Example Output

### Example output CSV file from -stats

### Example Program Run through (With simulation)

# Burst Errors

### Back Channel

### Other Methods

# Appendix

## Graph Figures – Data and Methods

### Figure 1.0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nodes (**n)** | Serial | Multicast | P2P (Unchunked) | P2P (Chunked) |
| 1 | 85.89934592 | 85.89934592 | 85.89934592 | 85.89934592 |
| 5 | 429.4967296 | 85.89934592 | 196.1368399 | 85.983232 |
| 10 | 858.9934592 | 85.89934592 | 251.5964572 | 86.0880896 |
| 15 | 1288.490189 | 85.89934592 | 285.0337001 | 86.1929472 |
| 50 | 4294.967296 | 85.89934592 | 386.4787957 | 86.9269504 |

|  |  |  |  |
| --- | --- | --- | --- |
| Chunk Size(**C**) | Total File Size (**D**) | Bandwidth (**B**) |  |
| 262144 | 1073741824 | 12500000 |  |

|  |  |  |
| --- | --- | --- |
| Architecture | Total transfer time formula | Explanation |
| Serial |  | Bit-containing packets served serially. Total bits to serve |
| Multicast |  | Same packets served to all clients simultaneously. |
| P2P (unchunked) |  | Upload begins only after download is complete. Each subsequent client downloads with 1 more seeder than the last. |
| P2P (chunked) |  | The optimal architecture for speed is a chain of peers, links long.  The total time is the time until the last peer has downloaded the last chunk.  This will be the time taken to transfer the last chunk to the first link + the length of the chain.  number of chunks |

# References

Garrett, R., & Gadea, L. (2014, September 25). Herd: A single-command bittorrent distribution system, based on Twitter's Murder. Github. Retrieved from https://github.com/russss/Herd