Switch Assisted Peer to Peer Transfer

Cloudlab and Disk Image Loading

- Cloudlab allows to access nodes of different types and topologies
- It has ~1500 nodes and allocates bare-metal nodes
- Fresh disk images are installed each time a request is instantiated
- Disk images contain OS (Distribution image/With user data)
- Disk image loading is on the critical path to provisioning

Frisbee System

- Multicast based image distribution system
- Single server supports multiple clients
- Multicast allows server to send one copy for all clients running in parallel
- Allows clients to join when transfer is in progress
- Provides fast and scalable disk imaging distribution

Issues

- Server process sends more data than ideal
- Thus more load on Server’s CPU, Disk and Cloudlab Network

SWP2P Idea Introduction

- New transfer mechanism using programmable switches
- Prefers peers instead of server to serve the chunk requests
- Switch keeps track of the image availability among connected clients
- Request packets are redirected at the switch

Thesis Statement

In the Frisbee disk loading system, SWP2P can significantly lower load on the server while maintaining high performance.

Related Work - Frisbee

- Splits image into Chunks and Blocks
- Works on Request-Reply mechanism
- Supports Request suppression and Request merging

M. Hibler, et al. Fast, Scalable Disk Imaging with Frisbee. In Proc. USENIX Annual Technical Conference, San Antonio, TX, June 2003.

Related work - LMS

Light Weight Multicast Services

- Elects one link per switch as the replier link
- Requests are redirected to the replier blindly
- On absence of data, replier forwards the request back to the switch
- Only requests from the replier link are forwarded upstream

SWP2P

- Switch keeps track of availability
- Peers are selected in round-robin fashion at the switch

C. Papadopoulos, et. al. LMS: A Router Assisted Scheme for Reliable Multicast.

Related work –P2P

- P2P transfers doesn't prefer closest peers
- Blocks needs to be transferred across topology, increases trunk link usage
- P2P system that support closest peer selection, suffers from overhead of maintaining overlay network
- SWP2P achieves the same benefits due to switch's position in the topology

M. Castro, et. All. Topology-aware routing in structured peer-to-peer overlay networks

SWP2P working principle

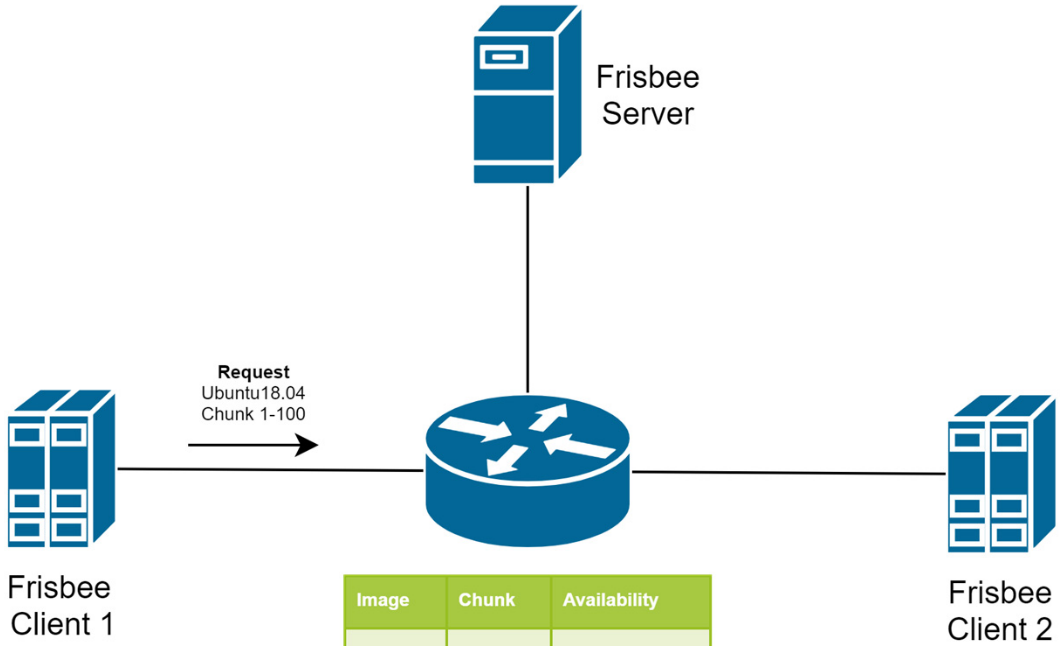
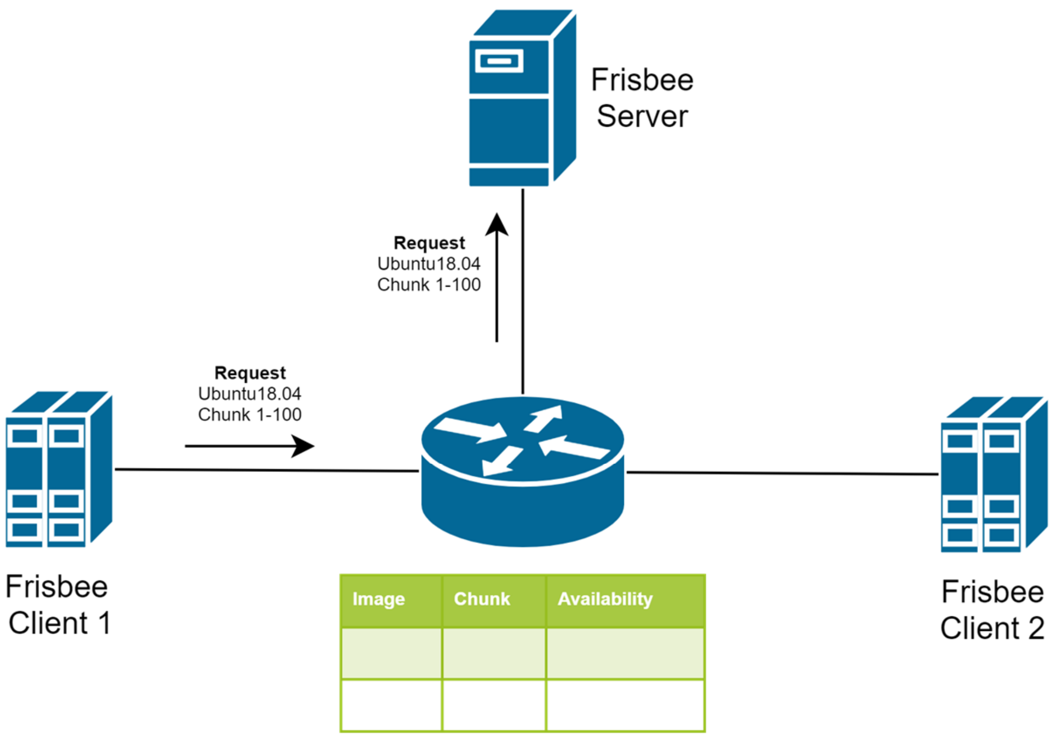
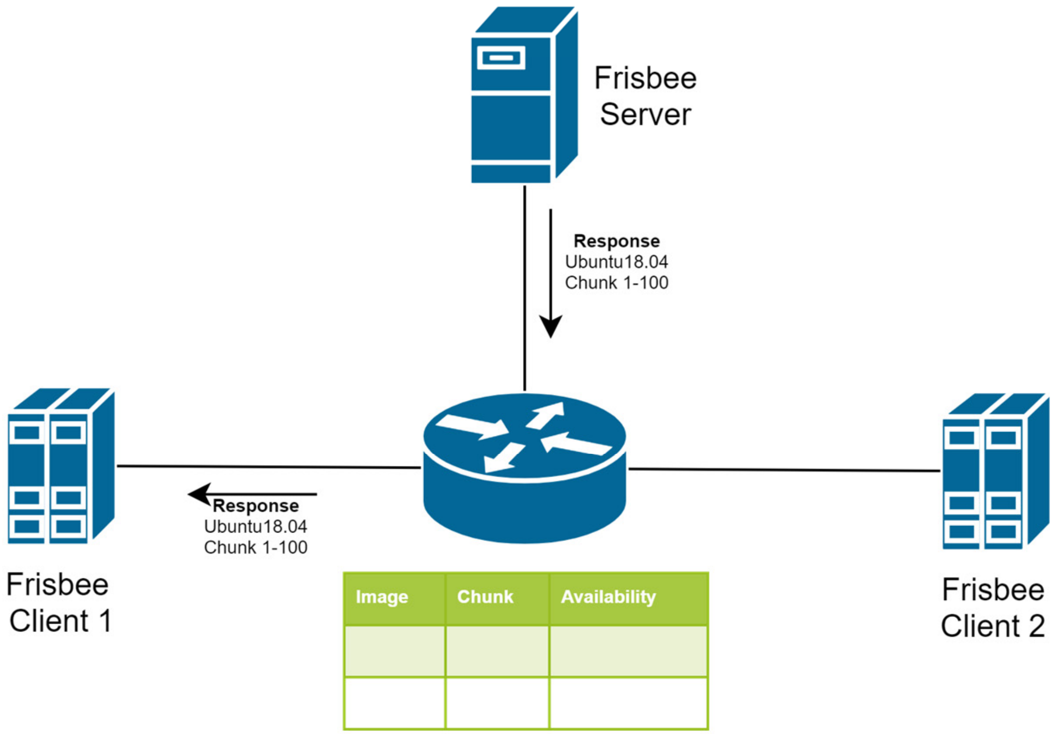


Image	Chunk	Availability

SWP2P working principle



SWP2P working principle



SWP2P working principle

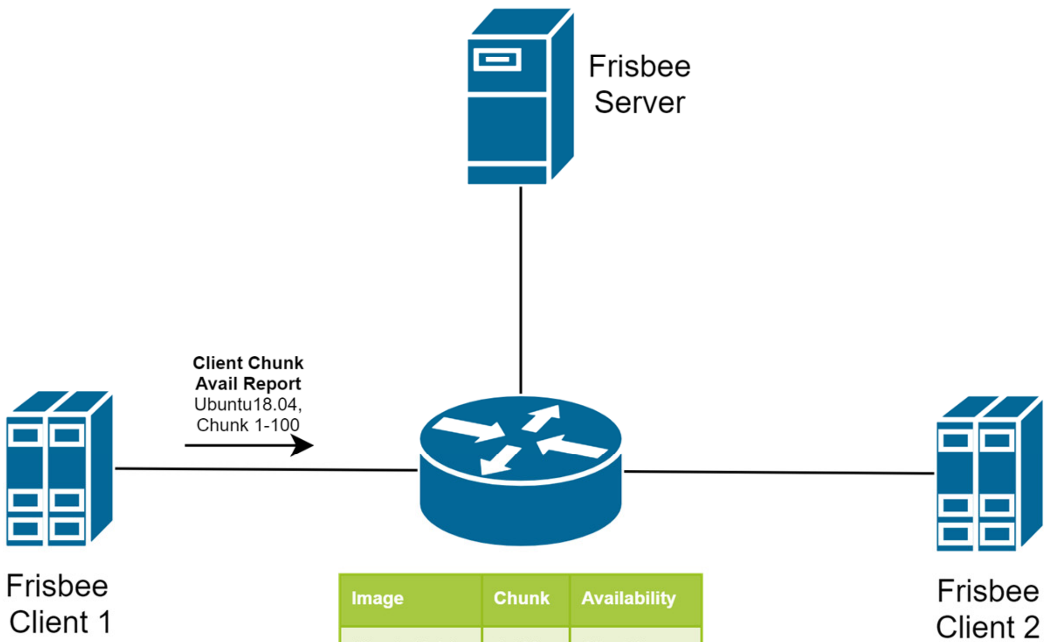


Image	Chunk	Availability
Ubuntu18.04	1-100	Client 1

SWP2P working principle

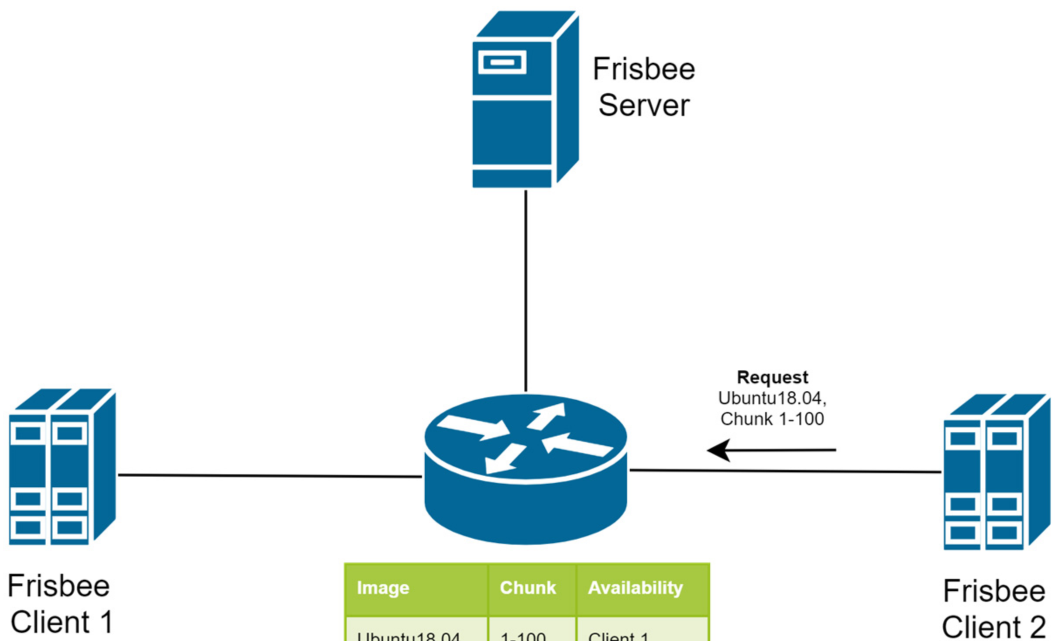


Image	Chunk	Availability
Ubuntu18.04	1-100	Client 1

SWP2P working principle

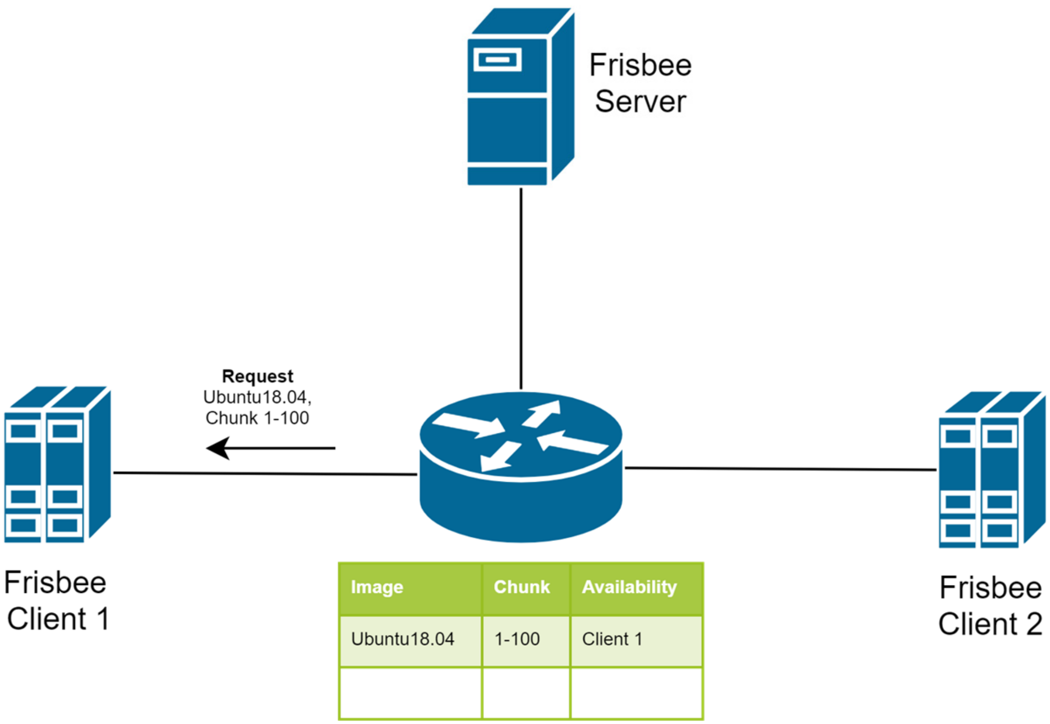
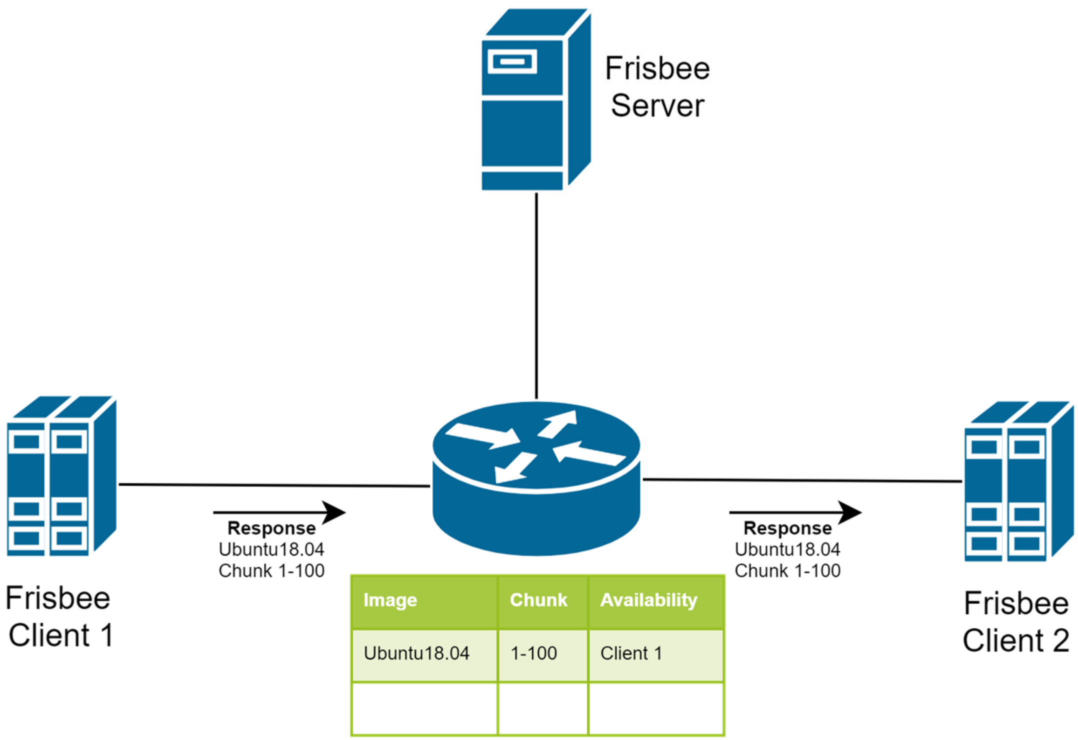


Image	Chunk	Availability
Ubuntu18.04	1-100	Client 1

SWP2P working principle



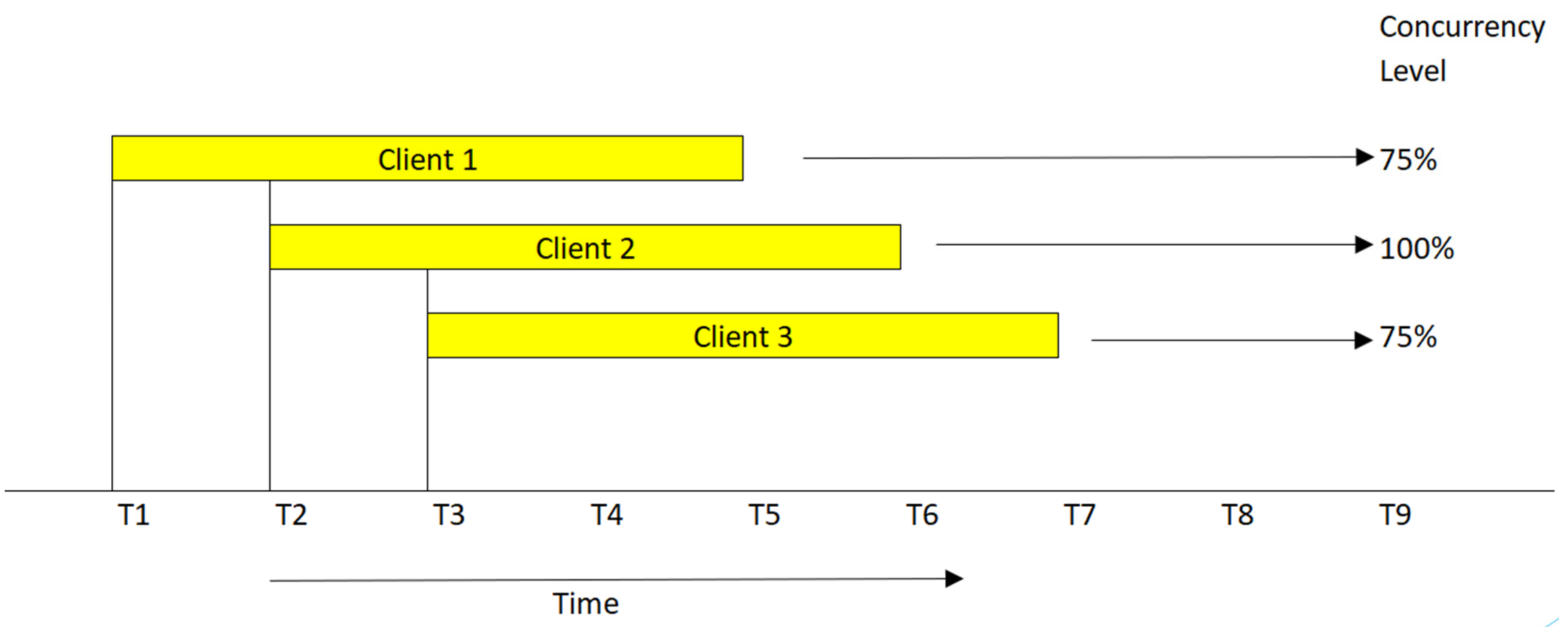
Analysis of existing system
Sample Server Log

```
Server Id                   : 55626
Serving Image               : /usr/testbed/images/UBUNTU16-64-
                              X86/UBUNTU16-64-X86.ndz:20
Total clients served        : 4
File read time              : 0.48
File size in MB             : 709.0
File repeated reads         : 775946240 Bytes
Total blocks on image       : 726016
Total blocks on multicast   : 1483776
Saving using multicast      : 1420288
    Client ID          : 1657444948
    Runtime            : 93.72
    Concurrency        : 100%
    Client ID          : 2058447516
    Runtime            : 93.705
    Concurrency        : 100%
    Client ID          : 1590336582
    Runtime            : 58.478
    Concurrency        : 99.18%
    Client ID          : 374246217
    Runtime            : 36.353
    Concurrency        : 99.02%
```

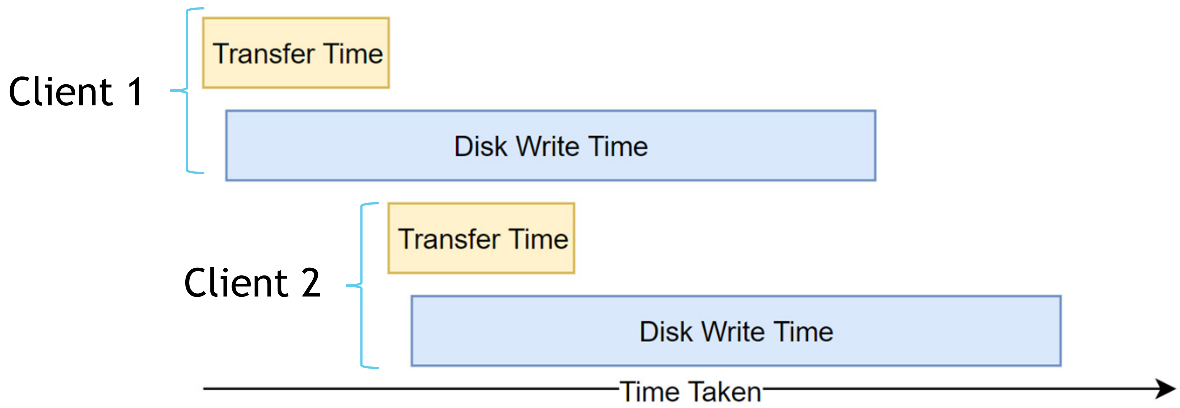
Server has sent twice the required number of blocks

Very high concurrency among clients

Concurrency Introduction

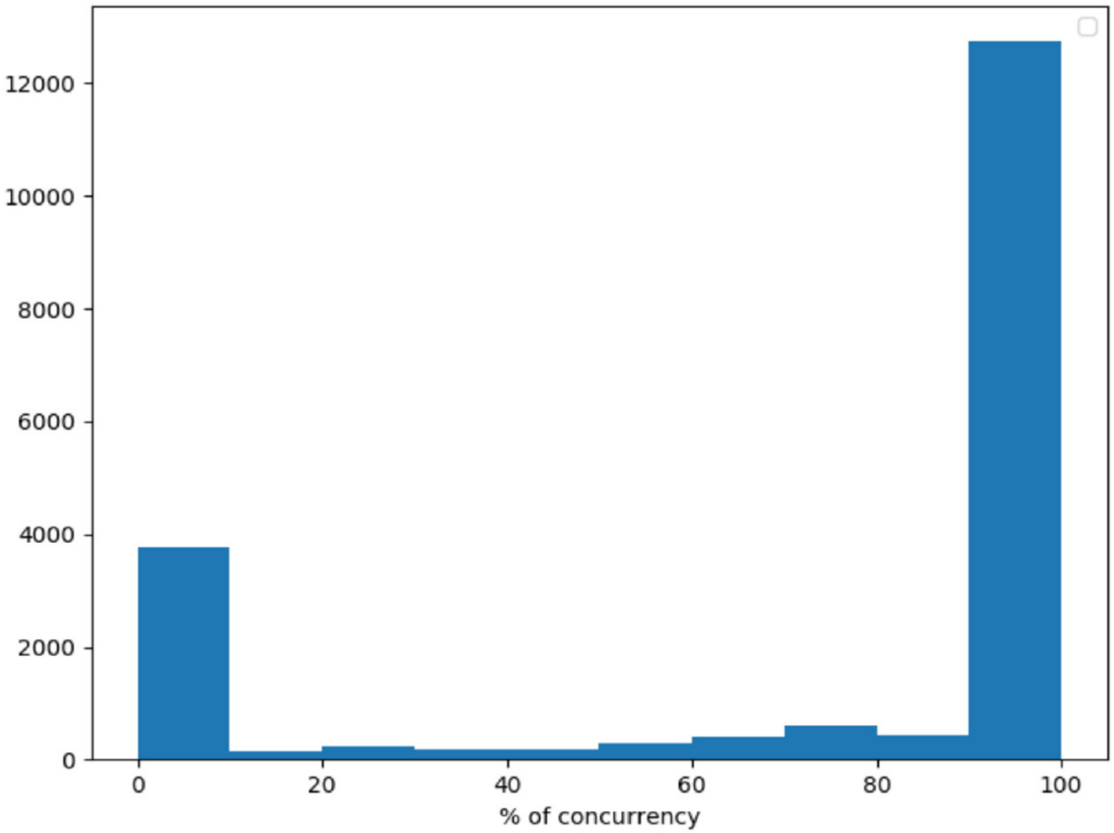


Components in Client Runtime



- Multicast systems considers clients as concurrent only if transfer time overlaps. This is the root cause of the performance gap.

Concurrency among client Data from past logs



Period : Jan 1st to Mar 22nd, 2019

Servers : 4178

Clients : 18998

- Out of the 18998 Clients we analyzed, 12737(67%) Clients had 90 to 100% Concurrency.

Computing transfer efficiency

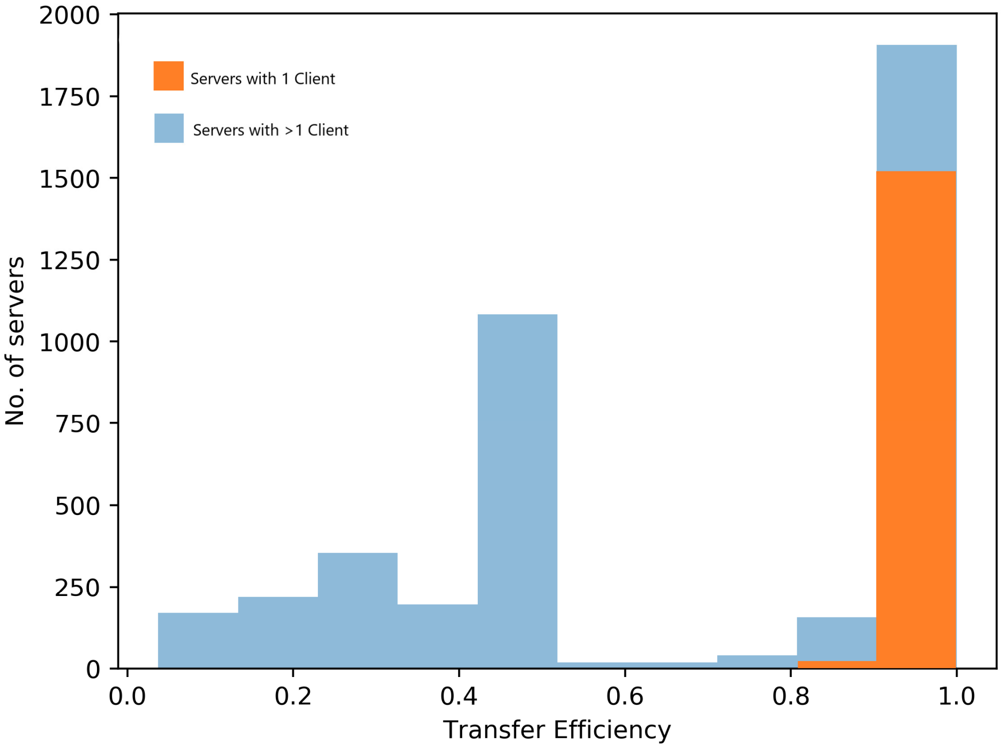
Below computations are for a group of clients that has concurrency > 90%

B = Number of blocks in image

T = Total blocks sent

Multicast Transfer Efficiency = B/T

Multicast transfer efficiency



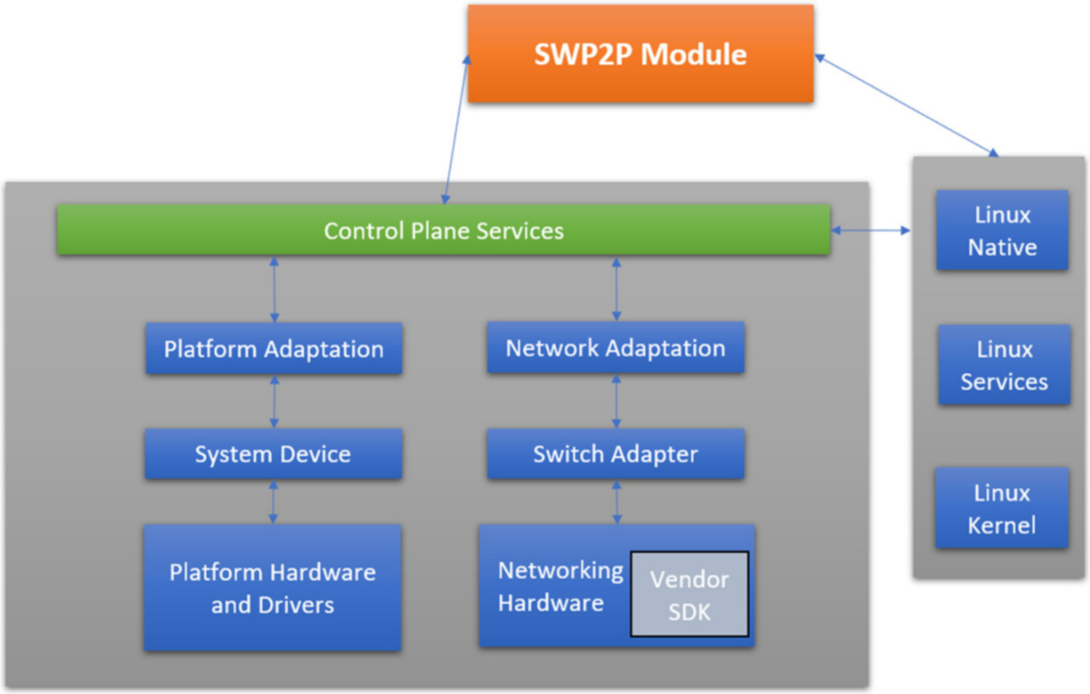
- 79.29% of the servers having max efficiency had just 1 client.
- Average efficiency of servers with >1 client is 0.53
- As 67% of clients had 90-100% concurrency
- Most servers with more than one client has significant concurrency

SWP2P Design

Changes in all three components involved

- **Switch**
- **Frisbee Client**
- Frisbee Server

Switch Environment



Built on OPX (OpenSwitch NOS)

SWP2P python module on the switch

Interacts with CPS to install ACL rules

Lifted packets are delivered via Linux Kernel's Ethernet interface

Switch’s Responsibilities

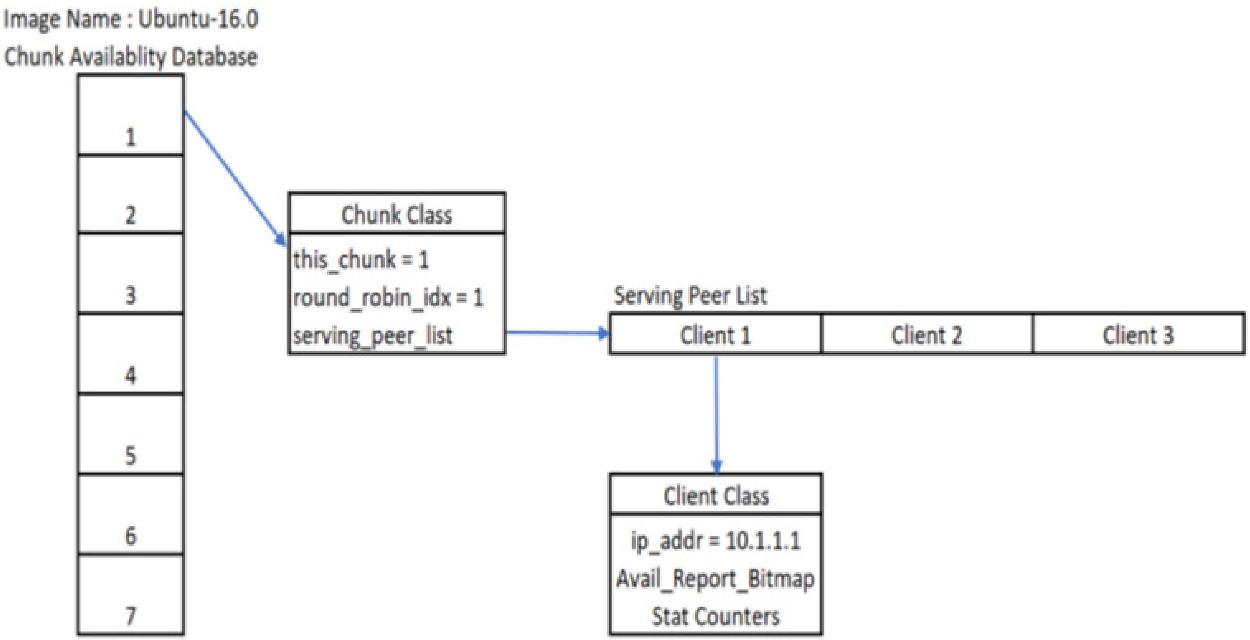
- Receiving Frisbee control packets from the clients
- Build Chunk Availability Database using Report and Leave message
- Parsing and processing request message from clients.
- Redirecting the request to the appropriate peer/server.

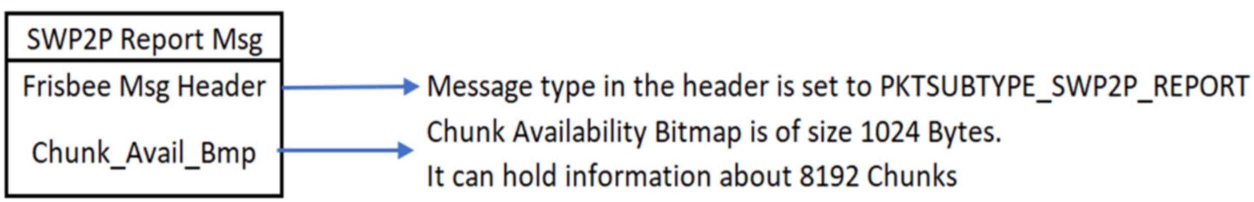
Packet Processing at Switch

Message Type	SWP2P Module Action	Number of messages N = Image Size (MB)
PKTSUBTYPE_JOIN	Forwarded to the server	O(1)
PKTSUBTYPE_REQUEST	Redirected to another client/server based on availability in the database	O(N)
PKTSUBTYPE_BLOCK	This message is not lifted	O(N)*1024
PKTSUBTYPE_SWP2P_REPORT	Used to update the chunk availability database for the client	O(N)
PKTSUBTYPE_LEAVE	Chunk availability database is updated as the client is no longer available to serve chunks	O(1)

The number of messages are for a single client
Total number of clients per switch is bounded to a maximum of 48

Chunk Availability Database





- Chunk Availability Database is maintained using new report message and leave message
- Clients are selected in round robin method to serve the requests

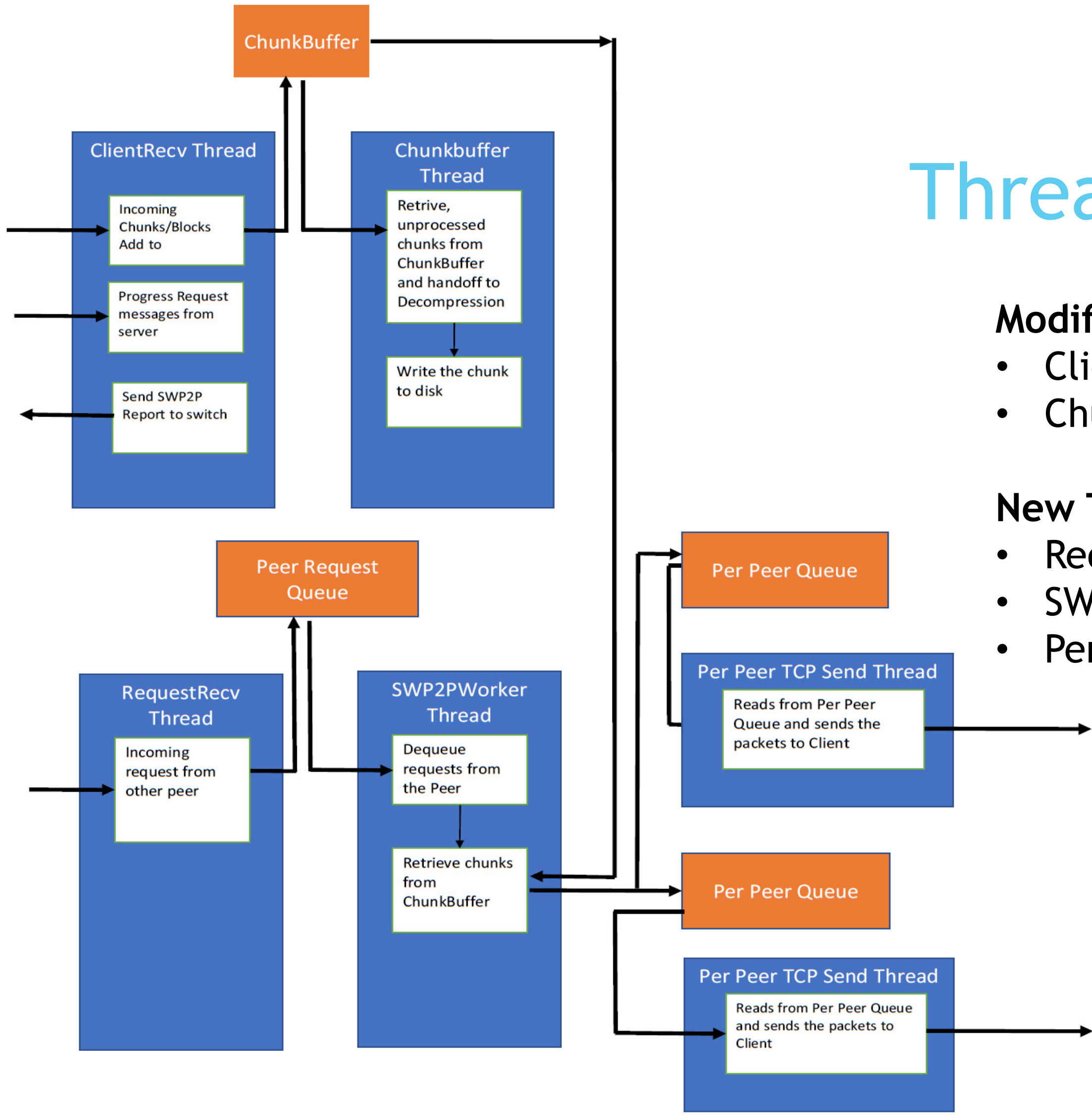
Changes in Client

P2P Essentials

- Transfer changed from Multicast to Unicast
- Receiving redirected request messages from the switch
- Sending requested chunks to the peers
- Updating the switch with report messages

Performance Optimizations

- Moved to deterministic order for requesting chunks
- Request batching
- Transport mechanism changed from UDP to TCP



Threads on Client

Modified Threads

- Client Recv Thread
- ChunkBuffer Thread

New Threads

- Request Recv Thread
- SWP2P Worker Thread
- Per Peer TCP Send Thread

Move from UDP to TCP

- Regardless of our attempts to do congestion control UDP still has loss on receiving side
- Using TCP for the image blocks transfer solves this problem
- We still use UDP from Frisbee control packets as they are redirected at the switch

Request Order and Batching

Goal:

Approximately ensure that first N chunks sent by the server makes a complete copy of the image

Number of Chunks N = 1000

Number of Clients C = 4

Ci = Represents ith Client

Node Number	Ci	Request Order
1	1	1-250, 251-1000
2	2	251-1000, 1-250
3	3	501-1000, 1-500
4	4	751-1000, 1-750

Request Batching Cascade Effect

Range A	1-250
Range B	251-500
Range C	501-750
Range D	751-1000

Time	Client 1	Client 2	Client 3	Client 4
T1 (0 to 2 seconds)	Range A	Range B	Range C	Range D
T2 (2 to 4 seconds)	Range B	Range C	Range D	Range A
T3 (4 to 6 seconds)	Range C	Range D	Range A	Range B
T4 (6 to 8 seconds)	Range D	Range A	Range B	Range C

- Request Batching ensures first 1000 chunks sent by the server constitutes the whole image
- It also assigns approximate ownership for each range to a client.
- Round-robin selection at the switch avoids overloading a client

Design and Implementation Summary

- Modifying existing Frisbee system to support Unicast transfer.
- Installation of ACL rules in Switch for lifting control packets.
- Maintenance of Chunk Availability Database.
- Redirecting request packets at the switch.
- Adding serving capability to clients.
- Efficient multi-threaded architecture on clients for low latency service.
- New sequential request mechanism coupled with request batching.
- Change of transfer protocol of blocks from UDP to TCP.

Evaluation

Key evaluation metrics

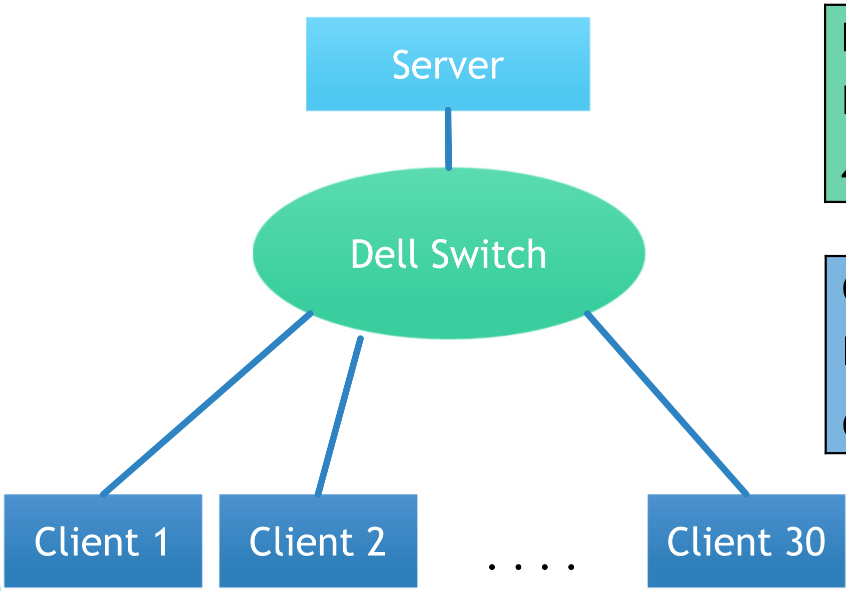
1. Server Load

   Server load is defined as the number of chunks sent out by the server to address the requests from the clients.

2. Network Receive Time in Clients

   Measuring impact on the time taken by clients to receive the image

Experiment Setup



Dell Switch

Intel(R) Atom(TM) CPU @ 1.74GHz

4 GB Memory

Other Nodes

Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz

64 GB Memory

Example of client runtime components

Client Runtime is affected by

1. Compression Ratio
2. Image Size

Network Recv time is always smaller than Decompress and Disk Write time

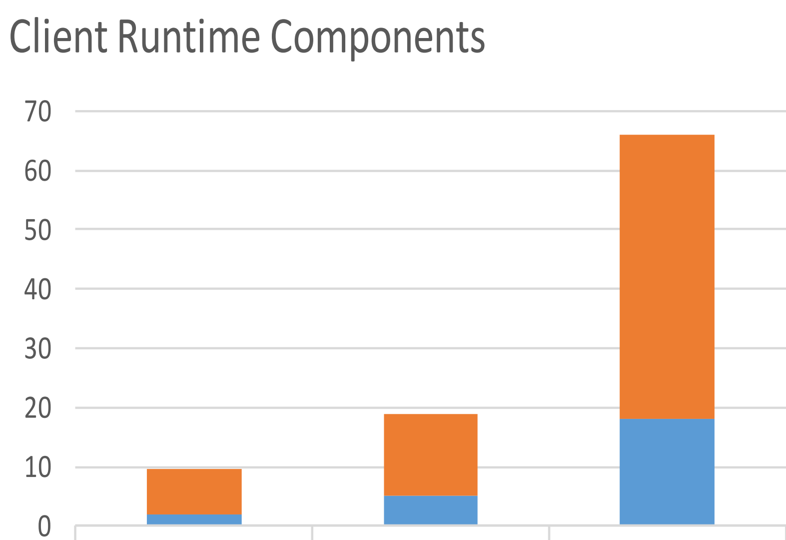
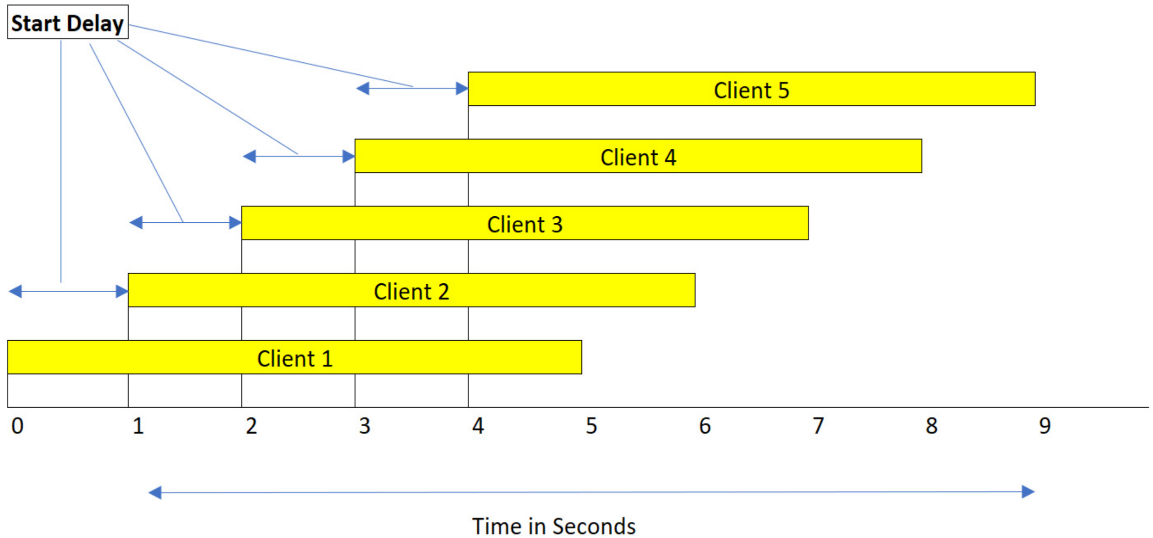


	Image 1 - 730 MB	Image 2 - 1984 MB	Image 3 - 6827 MB
Decompress and Write Time	7.66	13.66	48.06
Network Recv Time	1.996075	5.195049	17.963275

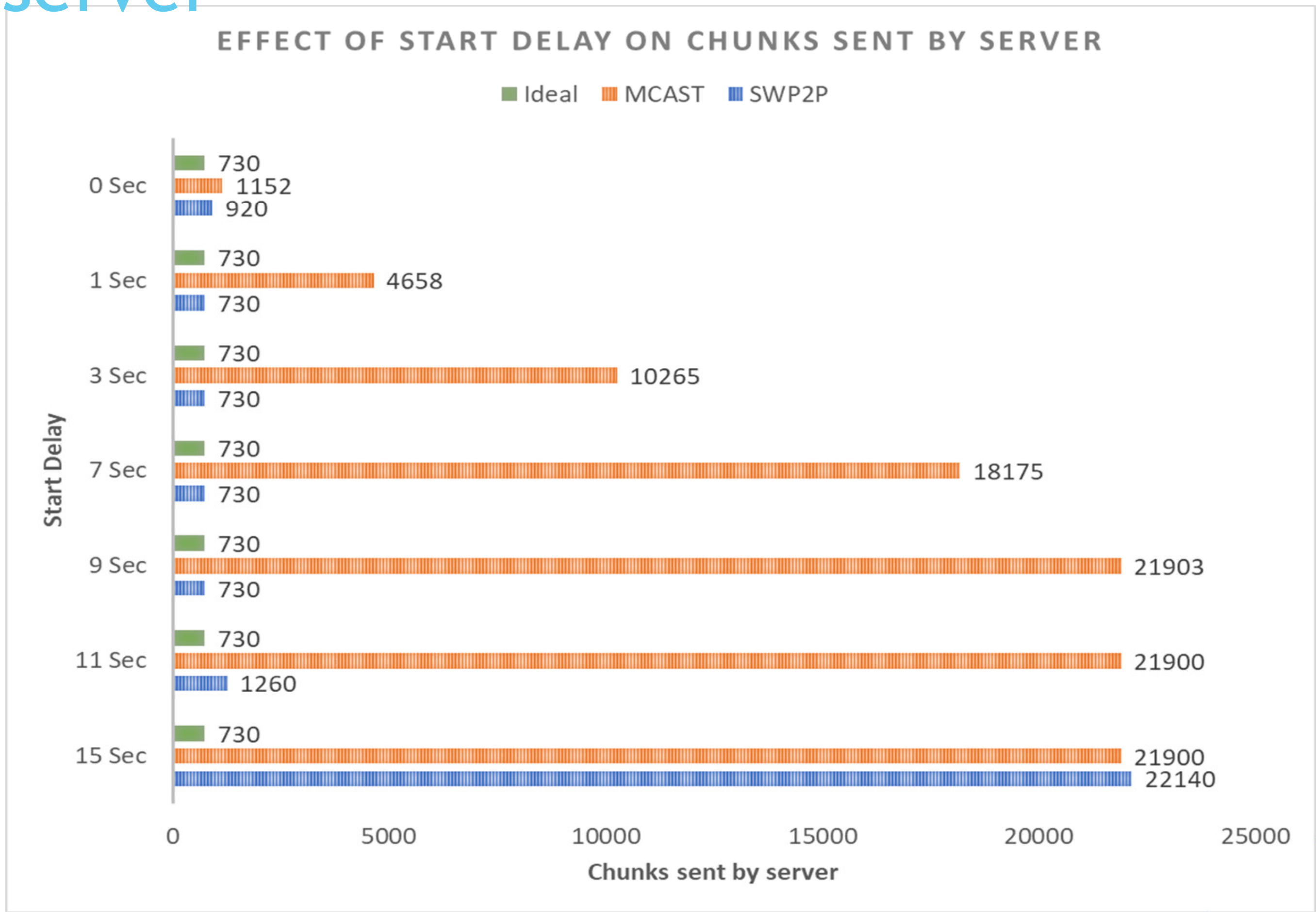
Network Recv Time Decompress and Write Time

Defining start delay



Start delay is used as a parameter to simulate various levels of concurrency

Effect of start delay on chunks sent by server

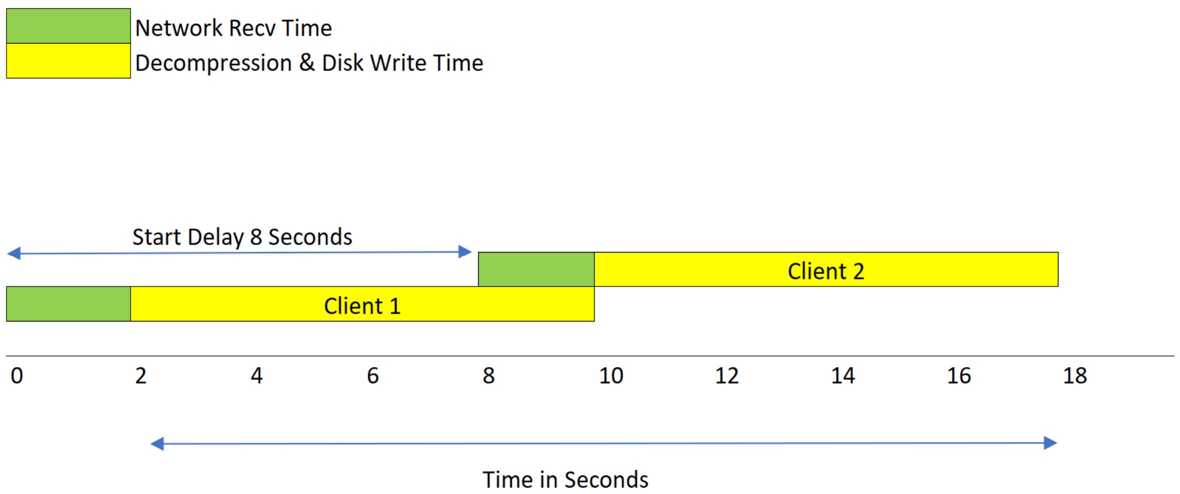


Experiment using image of size 730 MB and 30 clients

Analysis for Start Delay Between 1 to 9 Sec

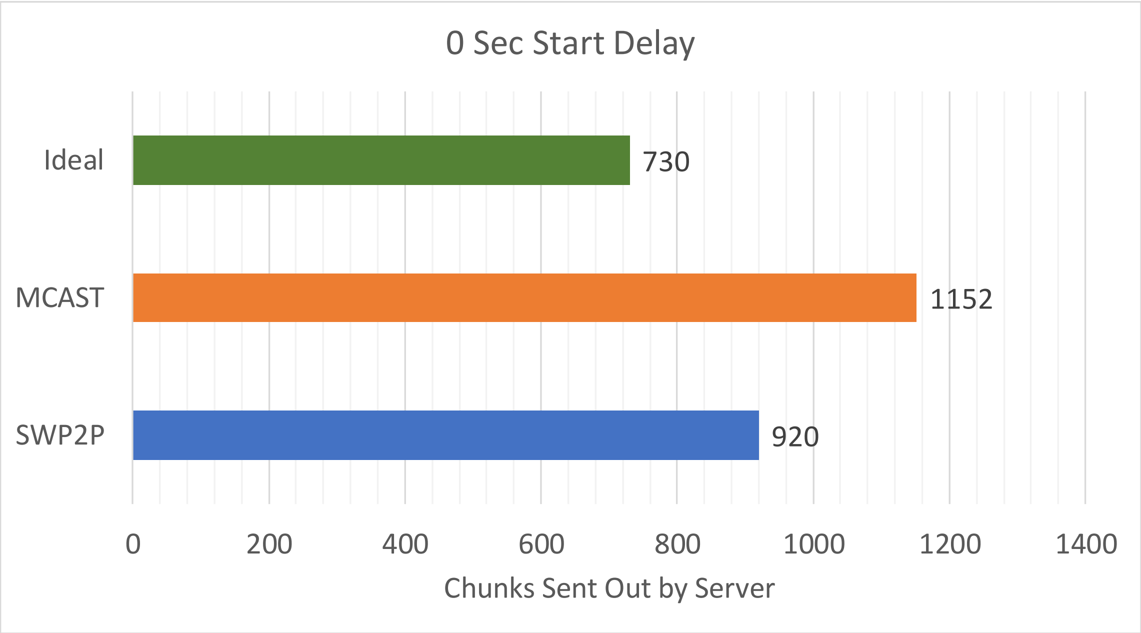
Typical Client Runtime Components for 730 MB Image

Component 1	Network Recv Time	2 to 3 Seconds
Component 2	Decompression and Disk write Time	8 to 9 Seconds



Start delay < 10 still has clients running in parallel

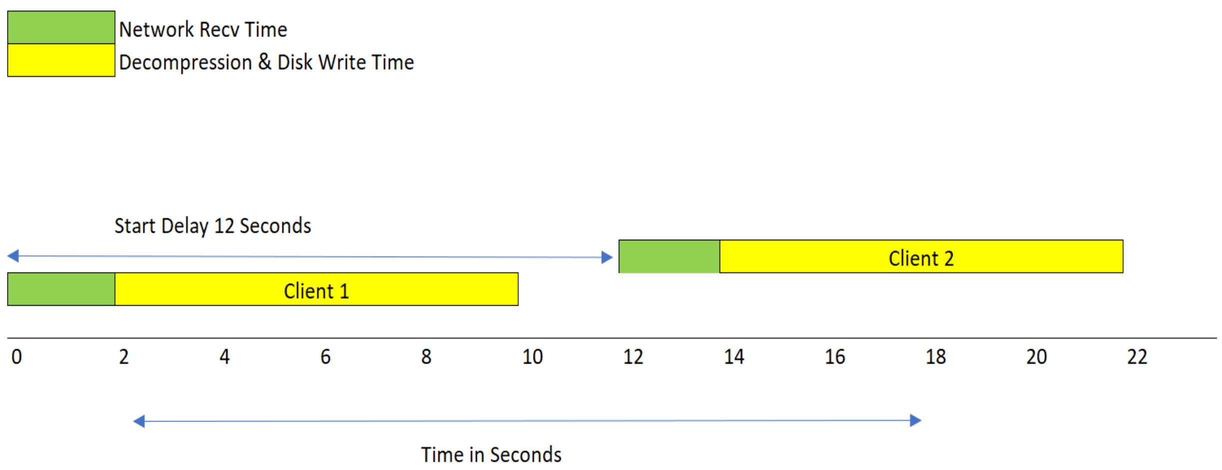
Analysis for 0 Sec Start Delay



SWP2P performs better than MCAST on MCAST ideal start delay.

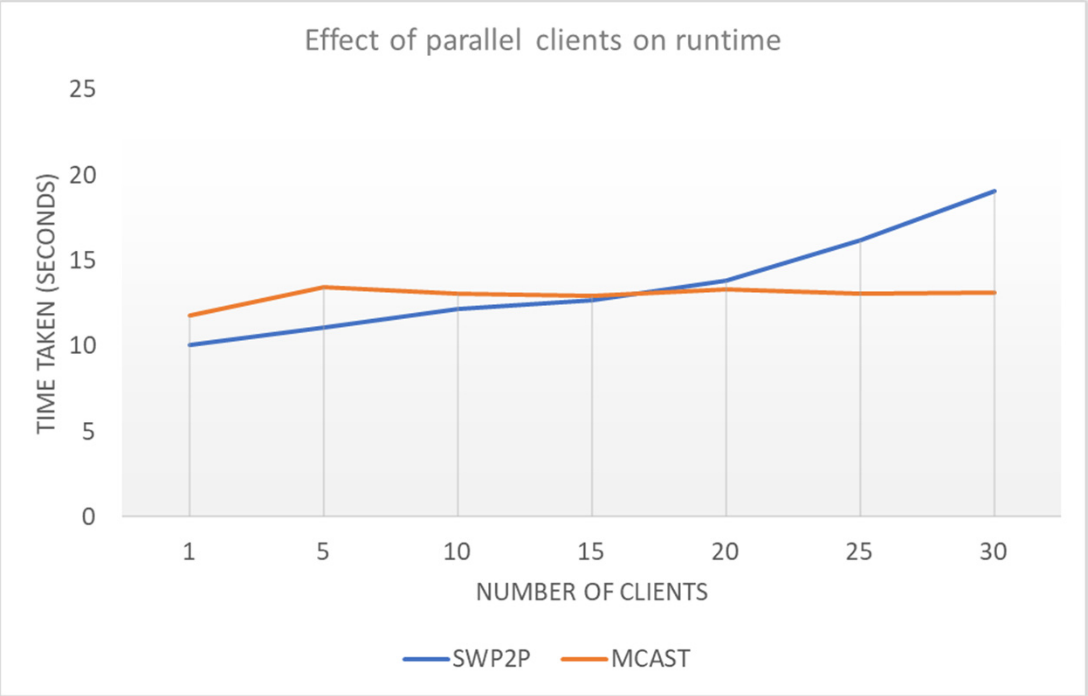
P2P serving design effectively proves its efficiency

Analysis for Start Delay Greater than 12 Sec



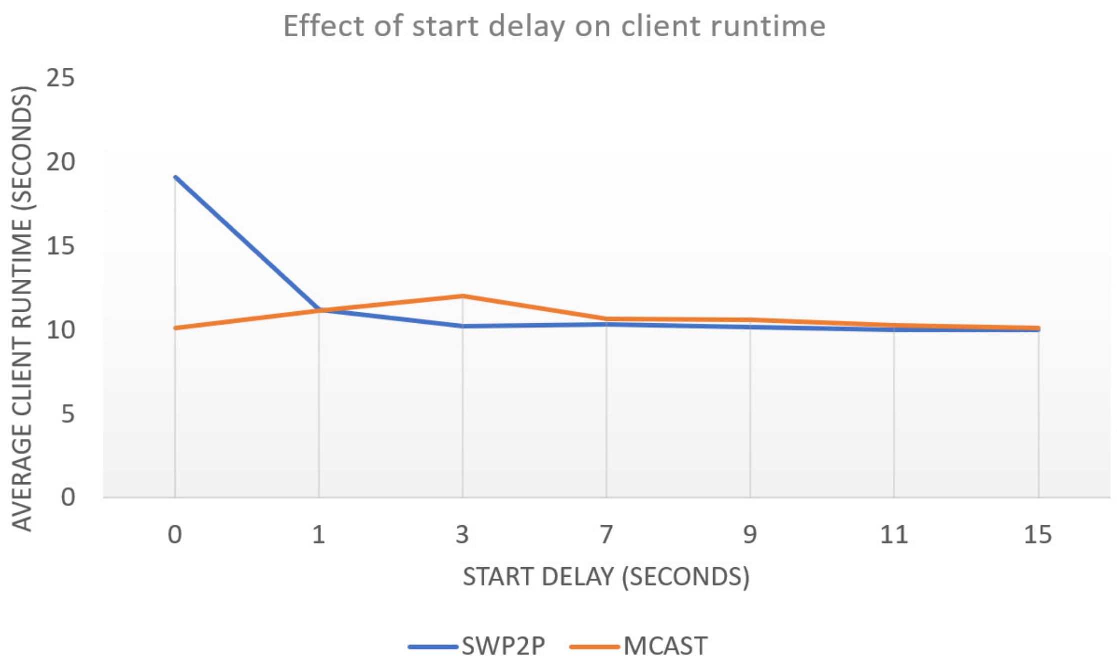
When Start delay > 12, there are no clients to leverage

Increase in client runtime due to switch processing



As more clients start in parallel their runtime goes higher

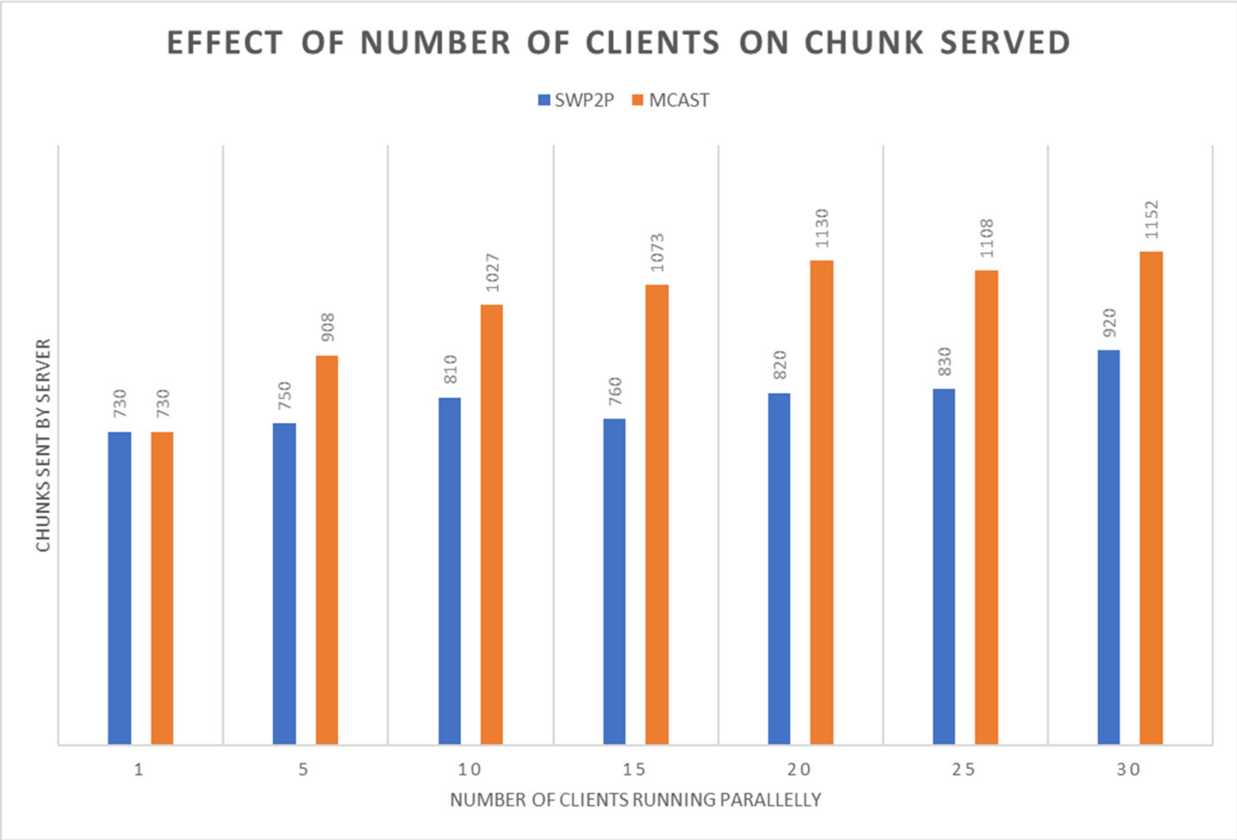
Using start delay to mitigate the switch processing time



30 Clients when started with various start delays and their runtime

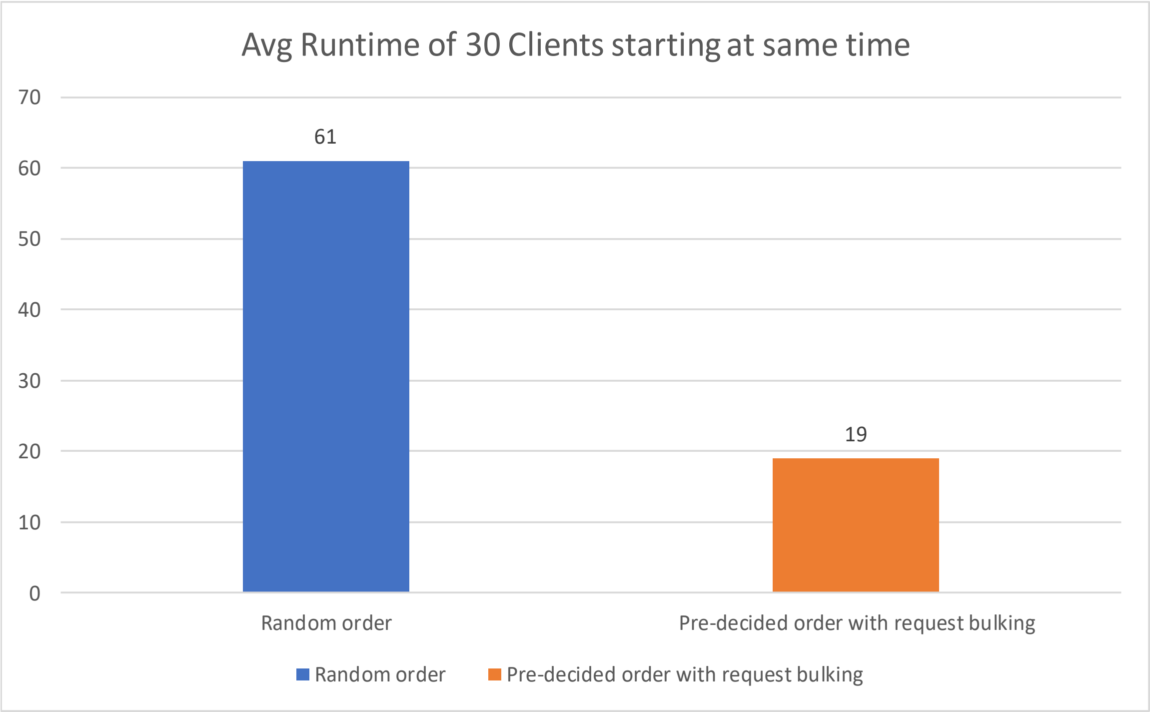
Starting with delay ensures no hiccup at switch, thus flat runtime

Server's perspective SWP2P vs Multicast



When all clients start in parallel, SWP2P consistently has lower load on server

Performance effect of request batching



Request Batching reduces the number of requests sent by clients

This lowers processing load on switch ensuring no queuing at switch

Evaluation Summary

- SWP2P outperforms Multicast in all scenarios we tested
- Switch’s low processing power caused hiccups
- Hiccups were overcome using start delay as a parameters
- Start delays that are less than client runtime is ideal
- Proves the efficiency of leveraging client concurrency

Conclusion

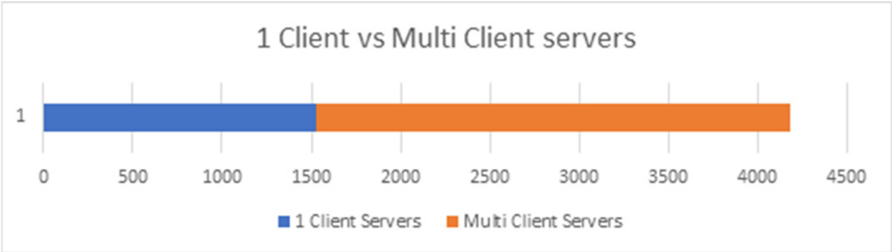
- High load on servers and trunk links were our motivation for SWP2P
- SWP2P performs better than multicast by leveraging client concurrency
- Longer decompression and disk write helps achieve our goal
- Challenges on working with the Switch and solutions were explored
- SWP2P reduces the server load from 21% to 96% depending on start delay.
- System can be extended to a multi-switch topology in future

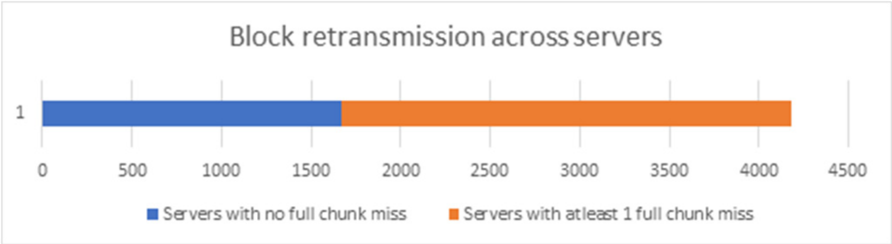
Thesis Statement

In the Frisbee disk loading system, SWP2P can significantly lower load on the server while maintaining high performance.

Q & A

Eliminating packet loss as the reason of retransmit

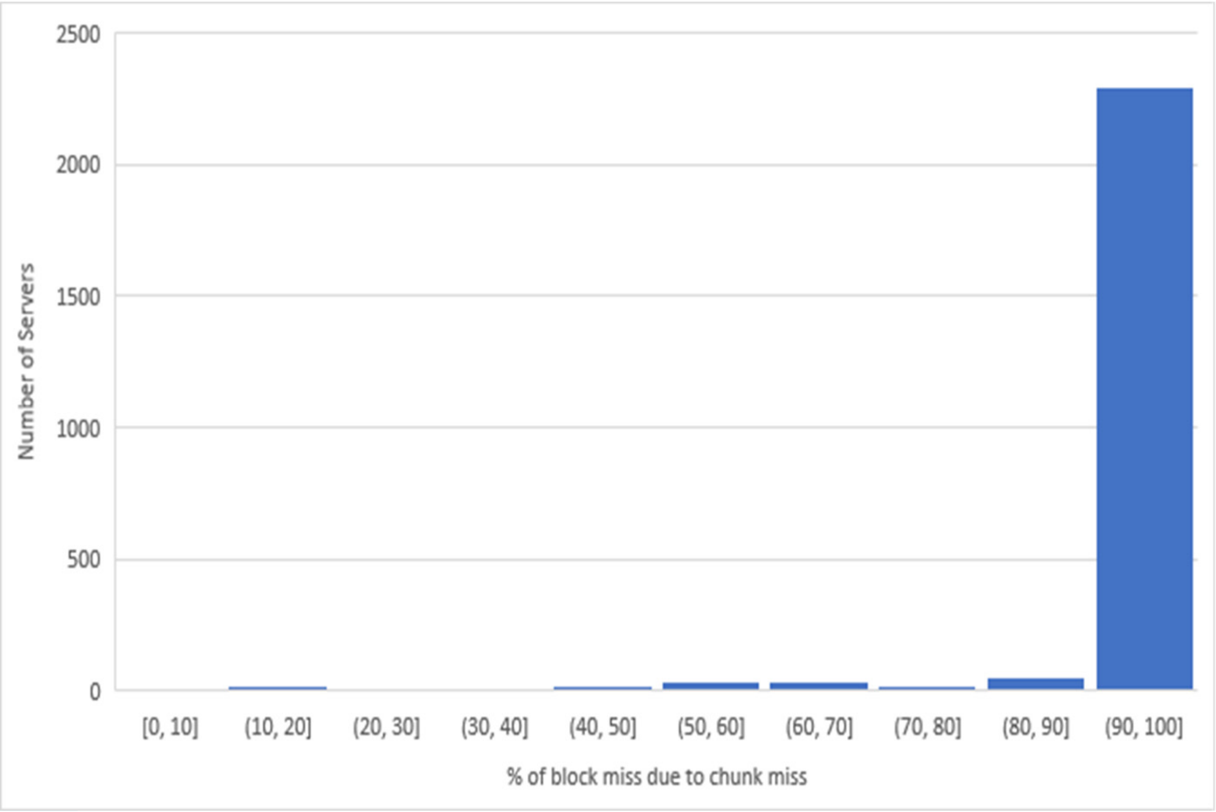




- High percentage of loss comes from total chunk loss.
- We considered network loss, will in general not result in total chunk loss.

Server's clients	**Average % of blocks retransmitted that can be attributed to full chunk miss**
1	0.11
>1	91%

Eliminating packet loss as the reason of retransmit



- Servers in general have clients block miss when they miss the entire chunk.
- Rarely individual blocks are missed by clients