Master-Thesis Presentation by Özcan Karaca

Testbed-Development for lectureStudio

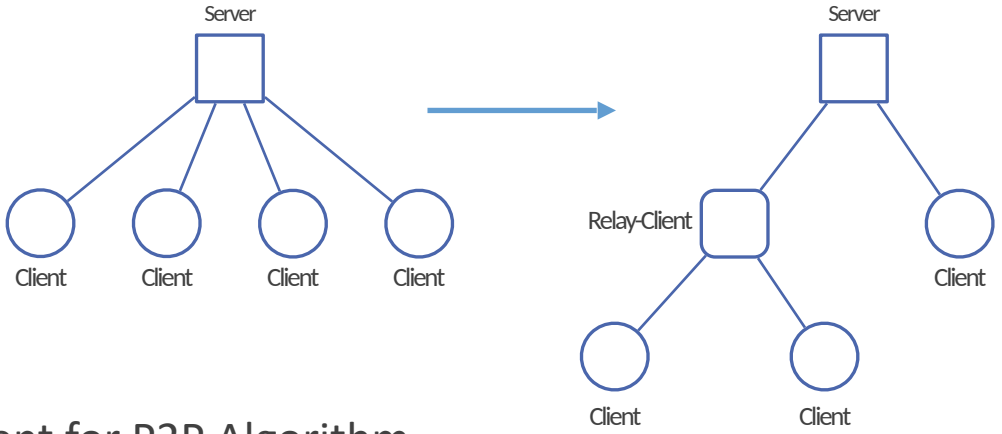
TESTBED-ENTWICKLUNG FÜR LECTURESTUDIO

Fachgebiet Echtzeitsysteme
Fachbereich Elektrotechnik und Informationstechnik
Technische Universität Darmstadt

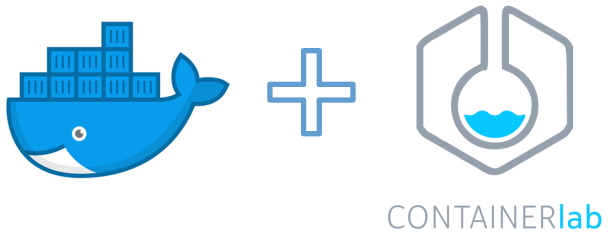
TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation

- P2P Algorithm for lectureStudio
  - Direct Distribution Among Clients
  - Reducing the Central Server's Load
  - Optimizing Bandwidth



- Developing A Container-Based Testbed Environment for P2P Algorithm
  - Simulation of Real Network Data
  - Configuration and Validation of the Network Characteristics
  - Communication and Data Transfer between Nodes



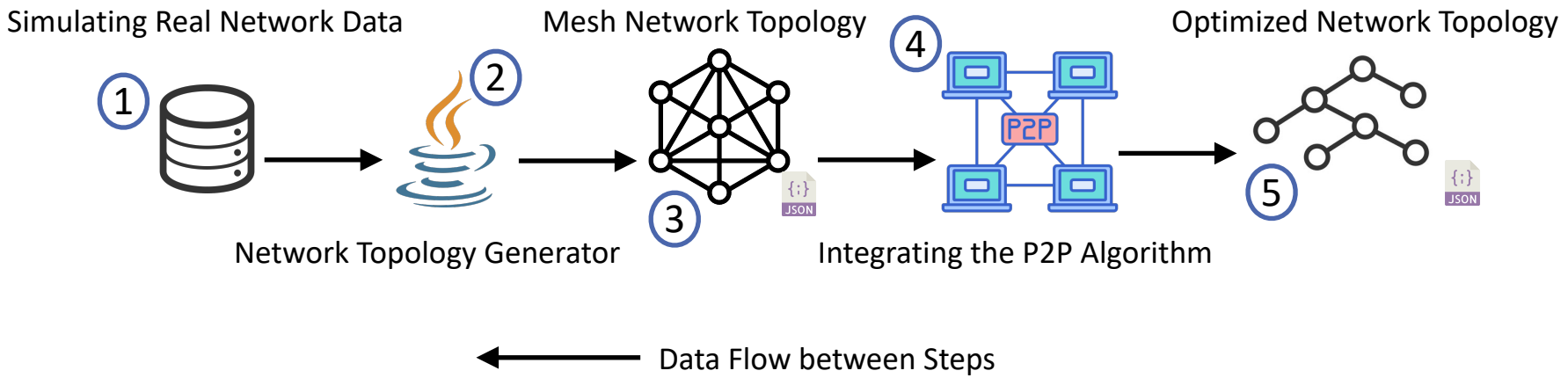
- Performance Evaluation of the P2P Algorithm
  - Analysis of Resource Consumption
  - Analysis of Total Duration

Task Description

- Finding Real Network Data
  - Maximum Upload Speed, Maximum Download Speed, Latency, and Packet Loss
- Simulating Real Network Data Using Normal Distribution
- Generating Network Topology
- Integrating the P2P Algorithm
- Configuring the Components of the P2P Algorithm in the Testbed
  - Creating a Containerlab File and Configuring Network Management, Nodes and Links
- Implementing the Network Characteristics of the Connections
  - Bandwidth, Latency, and Packet Loss
- Validating the Network Characteristics of the Connections
- Managing Communication and Data Transfer
- Tracking , Validating and Analyzing the Data Transfer Process
- Analyzing of Resource Efficiency of the P2P Algorithm Components (lectureStudio server and peers)
- Evaluating of the Testbed and the P2P Algorithm Performance

Initial Steps (Not Repeated) of the Testbed

- Simulating Real Network Data Using Normal Distribution
- Generating Mesh Network Topology
  - Integrating the P2P Algorithm
  - Implementing Traditional Server-Client Based Method
- Calculating Optimized Network Topology with the P2P Algorithm



Simulating Real Network Data

- Analyzing Real Network Dataset
  - Max Download Speed, Max Upload Speed
  - Latency
  - Packet Loss
- Reading Real Network Data from CSV File
- Generating Network Data Using Normal Distribution
  - Having Mean and Standard Deviation from UK and Germany Based Data

Name of Configuration	Mean (μ)	Standard Deviation (σ)
First Configuration	From UK-Based Data	From UK-Based Data
Second Configuration	From Ger-Based Data	From UK-Based Data

Network Topology

- Generating Network Topology with the Testbed
  - Listing Nodes with Network Characteristics
  - Detailing Connections between LectureStudio Server and All Peers
- Optimizing Network Topology with the P2P Algorithm
  - Identifying Super Peers
  - Optimizing Connections between LectureStudio Server and Peers

Network Topology Generated by the Testbed


```
{
  "filename": "test.pdf",
  "filesize": 5000,
  "peers": [
    {
      "name": "lectureStudioServer",
      "maxDownload": 29150,
      "maxUpload": 9209
    },
    {
      "name": "1",
      "maxDownload": 1080,
      "maxUpload": 373
    }
  ]
}
```

```
"connections": [
    {
      "sourceName": "lectureStudioServer",
      "targetName": "1",
      "bandwidth": 1080,
      "latency": 57,
      "loss": 0.0035
    },
    {
      "sourceName": "1",
      "targetName": "lectureStudioServer",
      "bandwidth": 373,
      "latency": 57.15,
      "loss": 0.0035
    }
  ]
}
```


JSON

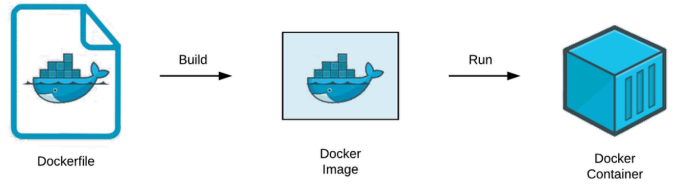
Network Topology Optimized by the P2P Algorithm

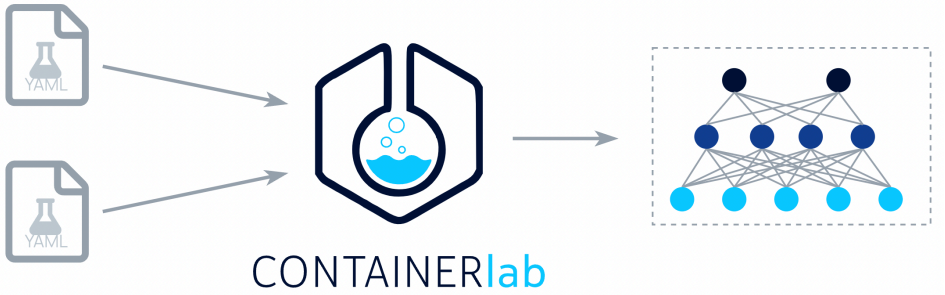

```
{
  "superpeers": [
    {
      "name": "1"
    }
  ],
  "peer2peer": [
    {
      "sourceName": "lectureStudioServer",
      "targetName": "1"
    },
    {
      "sourceName": "1",
      "targetName": "2"
    }
  ]
}
```


JSON

Container—Based Testbed Environment

- Docker and Containerlab: Efficient, Isolated Simulation Environment
- Creation and Management of User-Defined Network Topologies
- Advantages of Using Containerlab for the Testbed
  - Properties of Containerlab (name, image, kind, env, binds, etc.)
  - Speed, Ease of Use, Repeatability
  - Creation of Complex Network Topologies



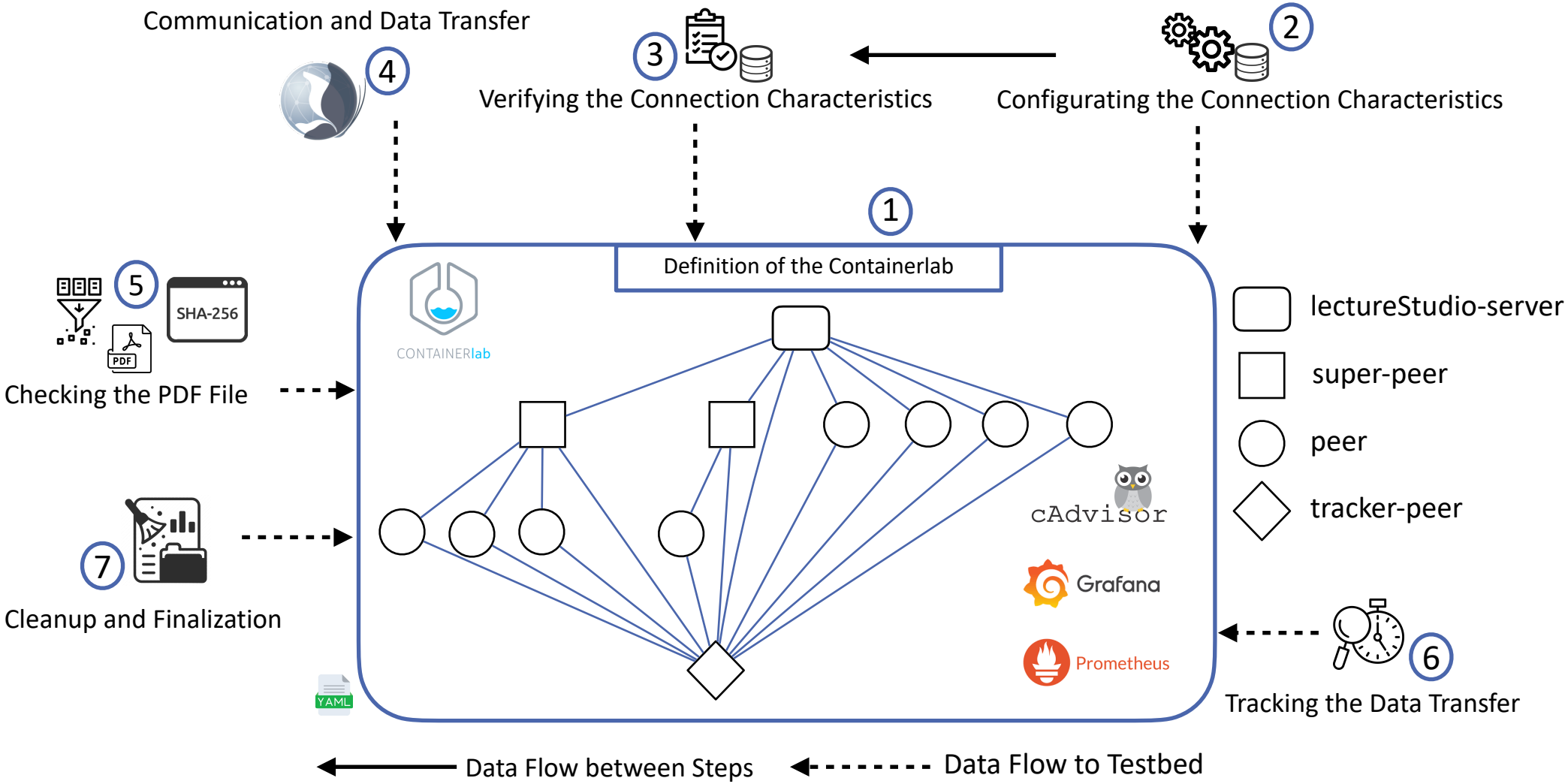


[2]

Configuration of Containerlab File (YAML)

```yaml
name: testbed

topology:
  nodes:
    peer1:
      kind: linux
      Image: image-testbed
    peer2:
      kind: linux
      image: image-testbed

  links:
    - endpoints: [peer1:eth1, peer2:eth1]
```

Execution Steps (Repeated) of the Testbed

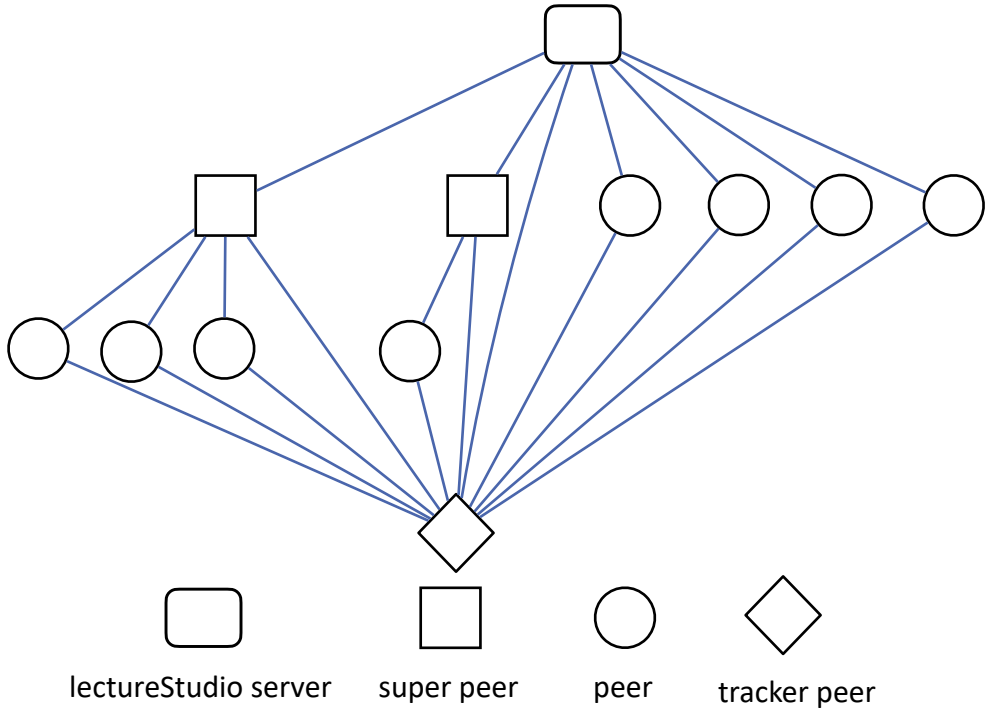


Configuring the Components of the P2P Algorithm

Configuration of Containerlab File (Network Management, Nodes and Links)

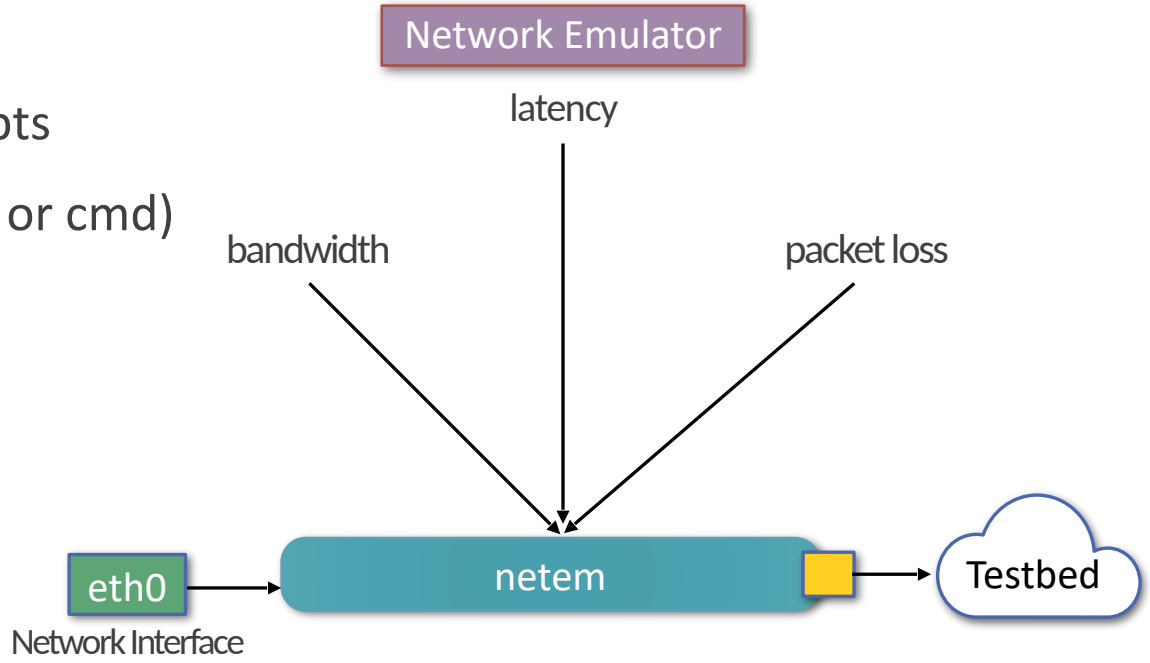
```yaml
name: testbed

mgmt:
  network: fixedips
  ipv4-subnet: 172.100.100.0/24

topology:
   nodes: ...

   links:
     - endpoints: [lectureStudioserver:eth1, 1:eth1]
     - endpoints: [lectureStudioserver:eth2, 2:eth1]
     - endpoints: [lectureStudioserver:eth3, 3:eth1]
     - endpoints: [lectureStudioserver:eth4, 4:eth1]
     - endpoints: [lectureStudioserver:eth5, 5:eth1]
     - endpoints: [lectureStudioserver:eth6, 6:eth1]
     - endpoints: [5:eth2, 7:eth1]
     - endpoints: [6:eth2, 8:eth1]
     - endpoints: [6:eth3, 9:eth1]
     - endpoints: [6:eth4, 10:eth1]
```

Configuration of Nodes



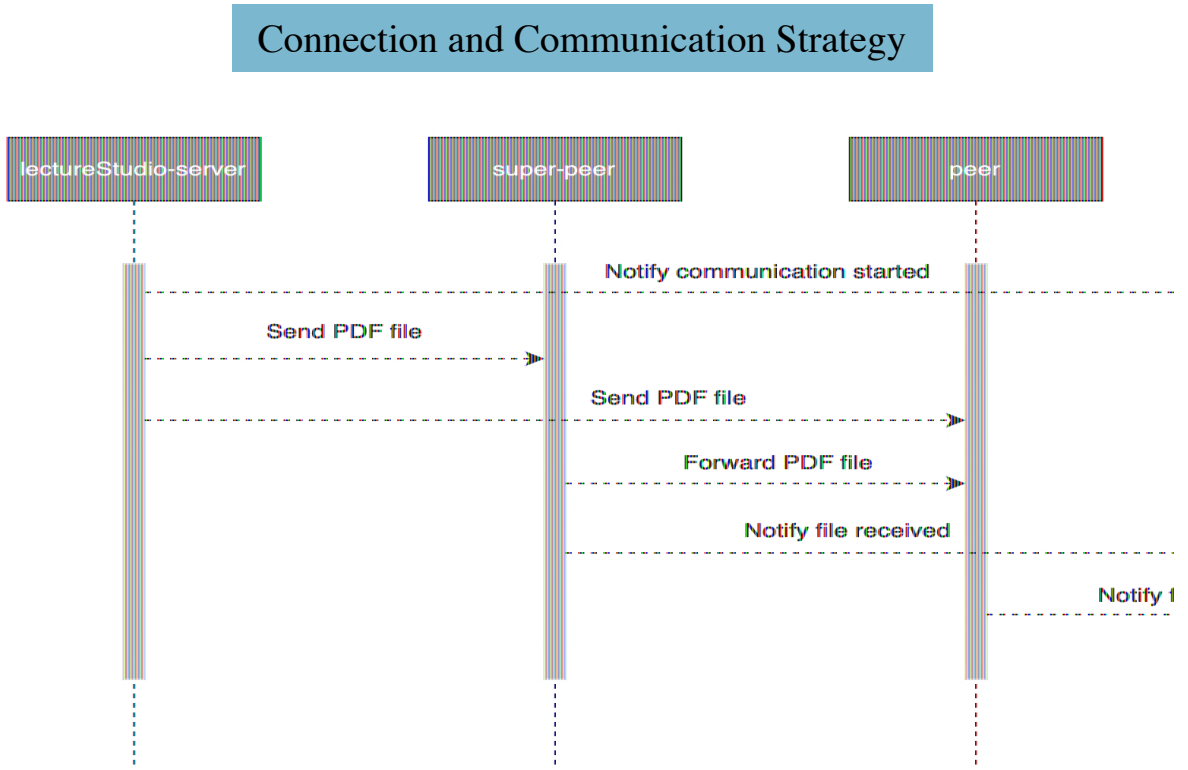
Configuring and Verifying the Network Characteristics

- Configurating the Connection Characteristics Using Traffic Control Commands
  - Bandwidth Limitation
  - Latency Addition
  - Packet Loss Simulation
- Configuring these Characteristics with Scripts
  - Properties of Containerlab (exec, binds or cmd)
- Verifying the Connection Characteristics
  - Using Tools like ping, iperf3



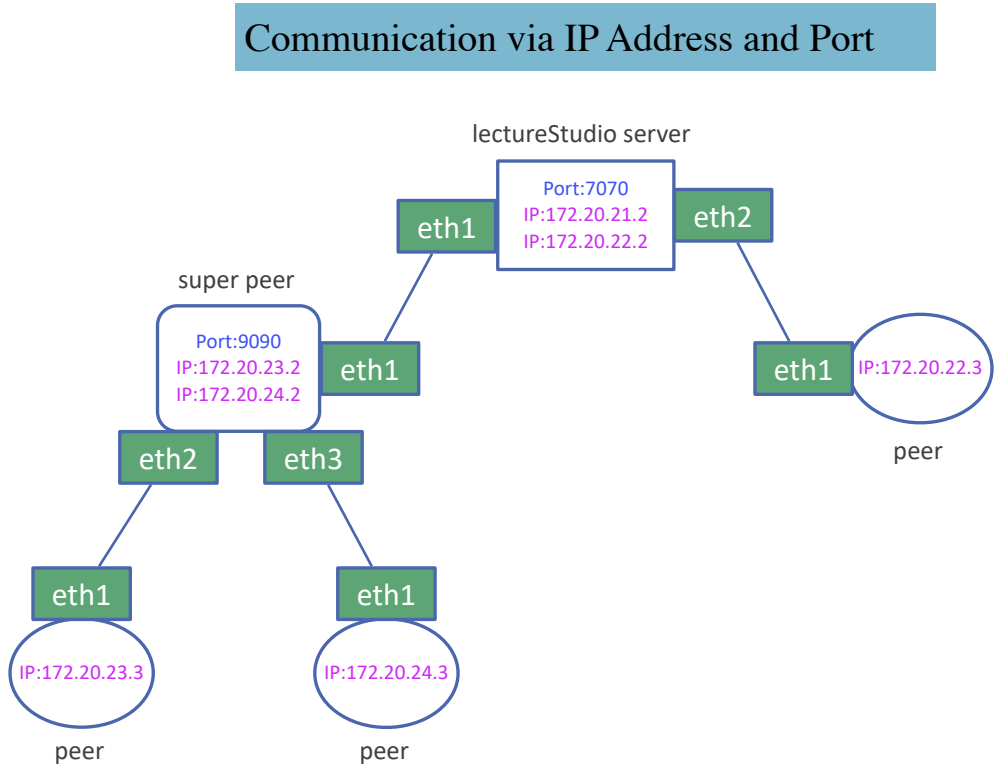
Communication and Data Transfer Processes

- Initial Notification by lectureStudio Server
  - Notification to Tracker Peer
  - Start of Data Transfer Process
- Role of Super Peers
  - Reception and Forwarding of PDF File
  - Transition from Receiver to Sender
- Confirmation Messages
  - From Peers and Super Peers



Connection Strategy Among Peers

- Add Node Info in Testbed Setup
  - Port Number
  - IP Address
- Use Netty Framework for Server
  - Handshaking
  - Authenticating Peers
  - Establishing Connections
  - Preparing the Network for Data Transfer
- Transfer in Segments
- Monitor Process and Performance



Validating and Tracking of the Data Transfer

- Integrity Checks of PDFs with Hash Value Calculation
- Comparison of Hash Values in all Docker Containers
- Methods to Inspect Data in Containers
  - Direct Container Access or Docker Command Feature for Data Movement and Control
- Role of the Tracker Peer in Monitoring Data Transfer
- Confirmation Message System
  - Initial Confirmation from the LectureStudio Server
  - Peer or Super Peer Confirmation Upon Data Receipt
- Calculation of Data Transfer Duration
  - Counting Received Confirmations
  - Total Duration Calculation from First to Last Acknowledgment

Evaluation Research Questions

Testbed

RQ1.1 How Accurately Does the Testbed Measure the Configured Bandwidth, Latency and Packet Loss?

RQ1.2 How Well Does the Testbed Scale with More Nodes and Complex Topologies Affects The Host in Terms of Resource Utilization?

P2P Algorithm

RQ2.1 How Do CPU and Memory Usage Change of the Participants in Tests with and without the P2P Algorithm?

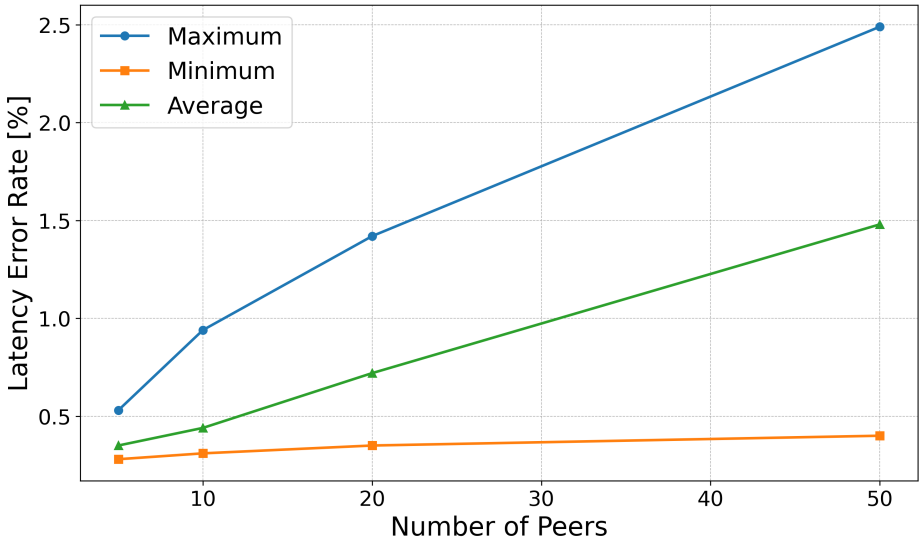
RQ2.2 How Does the P2P Algorithm React to Changing Network Characteristics (Bandwidth, Latency, Packet Loss)?

RQ2.3 Overall, is the P2P Algorithm Efficient for Data Transfer? How Does the Total Duration Obtained with the P2P Algorithm Respond to the Changing Number of Peers and Data Size?

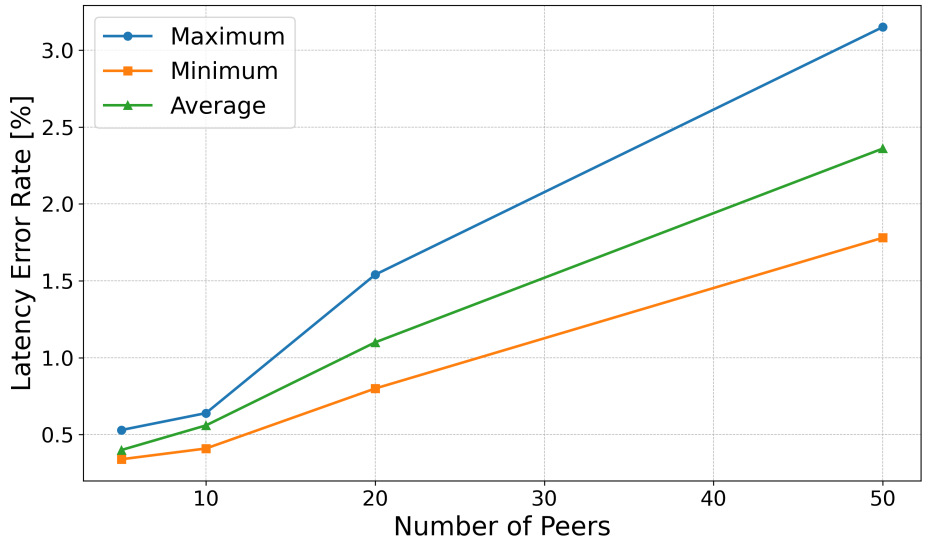
Accuracy of the Testbed, RQ1.1 (1)

- Discrepancy between Desired and Measured Values
- More Nodes Decrease Bandwidth Allocation Per Connection
  - A Reduction in Bandwidth Corresponds to a Rise in Latency.
- High CPU and memory usage impacting network performance and Latency

Error Rate for Latency with P2P Algorithm



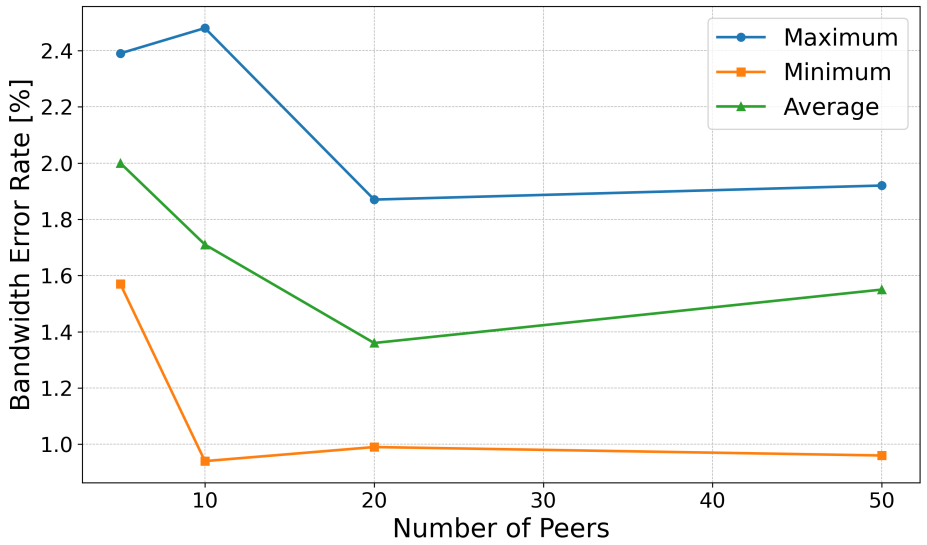
Error Rate for Latency without P2P Algorithm



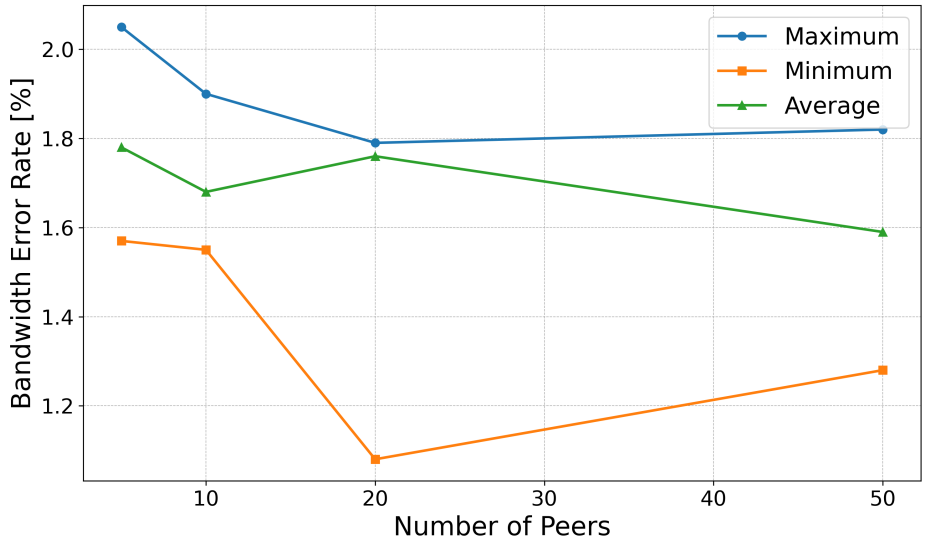
Accuracy of the Testbed, RQ1.1 (2)

- Accuracy of Measuring Tools
  - Iperf Accuracy Results Variation from Actual Performance
  - Variables Like Network Conditions, Configuration, System overhead

Error Rate for Bandwidth with P2P Algorithm



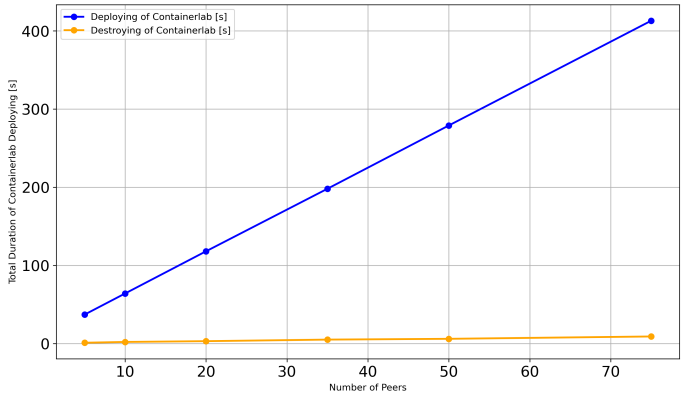
Error Rate for Bandwidth without P2P Algorithm



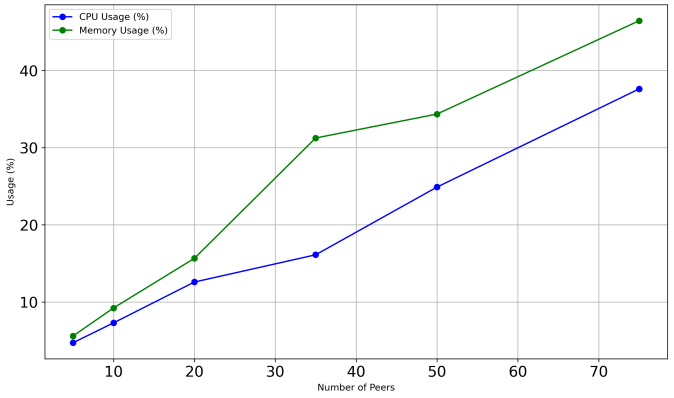
Testbed Scaling and Resource Utilization, RQ1.2

- Scaling Peers with Containerlab for Simplified Topology Deployment
- Deployment Time Increase of 5.37s per Additional Node
- Destroying Process Time Low, Approximately 9 Seconds for 75 Nodes

Duration of Deploying and Destroying of Containerlab



CPU and Memory Usage of the Host

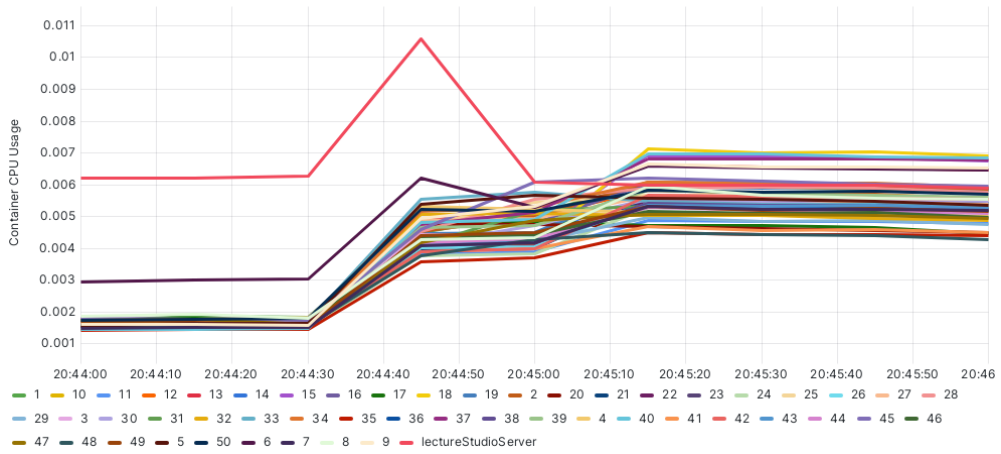


- CPU and Memory Usage Rise with More Peers Indicating Increased Host Load
- CPU Utilization Increase Linearly Showing Scalable Performance
- Memory Usage Growth Suggests Potential Bottleneck with More Peers

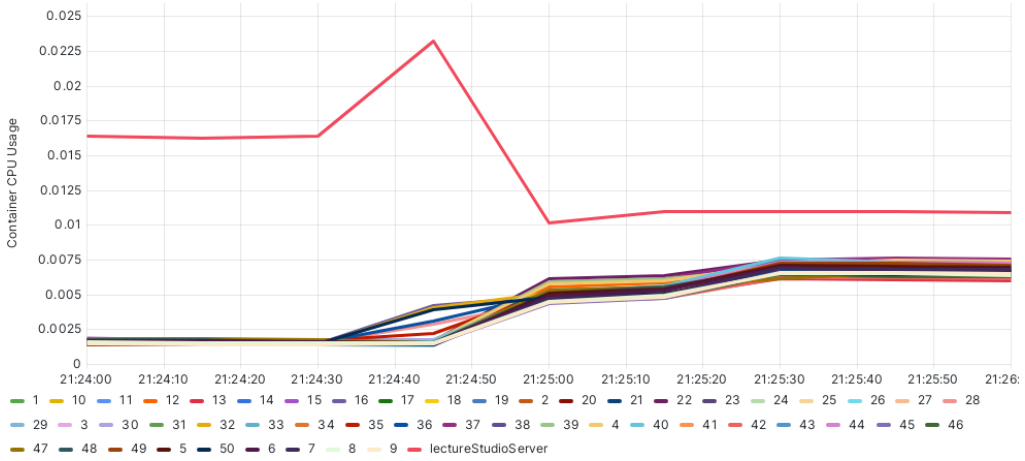
CPU and Memory Usage Analysis of Nodes, RQ2.1 (1)

- Analysis of Resource Consumption of P2P Algorithm Components Including CPU, Memory for LectureStudio Server and Peers
- High Initial CPU Usage on LectureStudio Server at Data Transfer Start
- Peak CPU Usage Reduction of 52% by P2P Algorithm
- Load Distribution Leading to Decreased CPU Usage

CPU Usage with P2P Algorithm



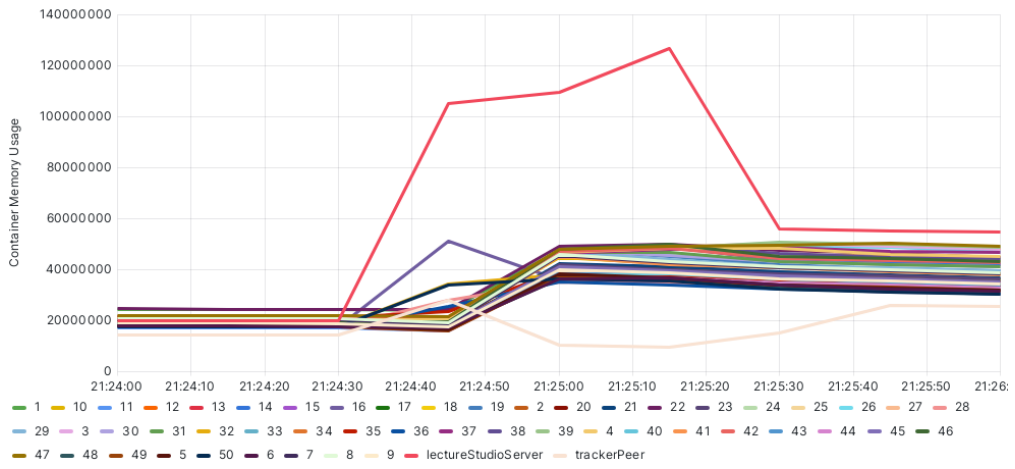
CPU Usage without P2P Algorithm



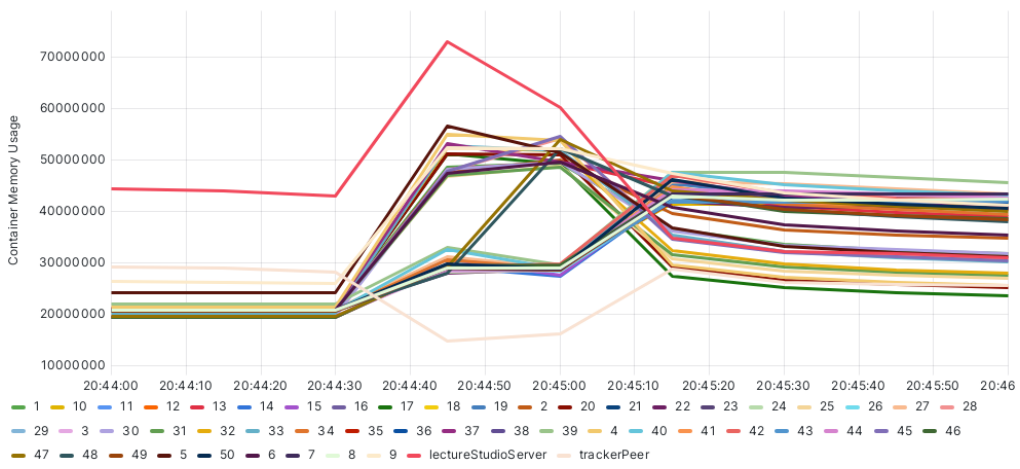
CPU and Memory Usage Analysis of Nodes, RQ2.1 (2)

- Memory Usage Reduction of 41% by P2P Algorithm
- Significant Resource Savings Due to P2P Utilization
- Task Distribution to Super Peers Lowers Server's Memory Requirement

Memory Usage with P2P Algorithm



Memory Usage without P2P Algorithm



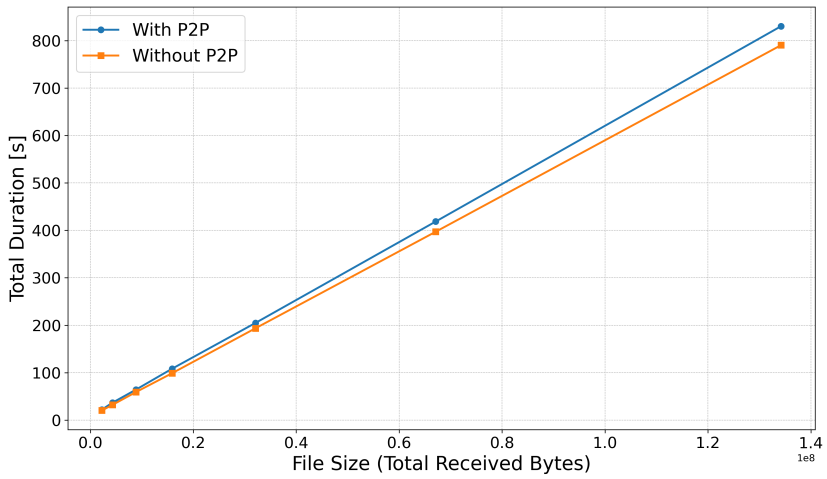
Performance Evaluation of P2P Algorithm, RQ2.2 and RQ2.3 (1)

- Test Duration Measured from First to Last Acknowledgment Message
  - Over 1000 Tests Conducted
  - Varying Data Size or Number of Peers
- First Configuration Simulates Real Network Data for Performance Analysis
  - Minimal Performance Difference between P2P and Server-Client Models Observed
- Second Configuration Employs Varied Mean Values for Data Simulation
  - P2P Algorithm Improved Efficiency with Increased Peers and Data Size
- Small File Sizes Not Significantly Benefited by P2P Optimization
- Larger File Transfers Show P2P Algorithm Efficiency
- Different Average Speeds for Upload and Download in Configurations Affect Performance
- Higher Average Speeds in Second Configuration Enhance Network Efficiency, Reduce Bottlenecks
- P2P Algorithm Effectively Identifies and Utilizes Super Peers with Higher Capacity

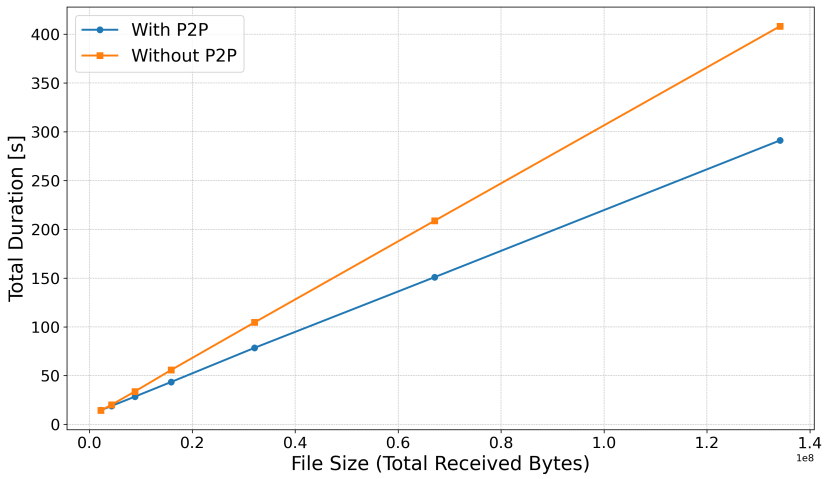
Performance Evaluation of P2P Algorithm, RQ2.2 and RQ2.3 (2)

- First Configuration with lectureStudio Server and 20 Peers
  - P2P Algorithm Performance Little Worse to Traditional Approach
- Second Configuration with lectureStudio Server and 20 Peers
  - Transferring 2MB Data Size, P2P Algorithm Performance Nearly Identical to Traditional Approach
  - Transferring 128MB Data Size, P2P Algorithm Performance 27% Faster than Traditional Approach

Time of Whole Data Transfer with the First Configuration



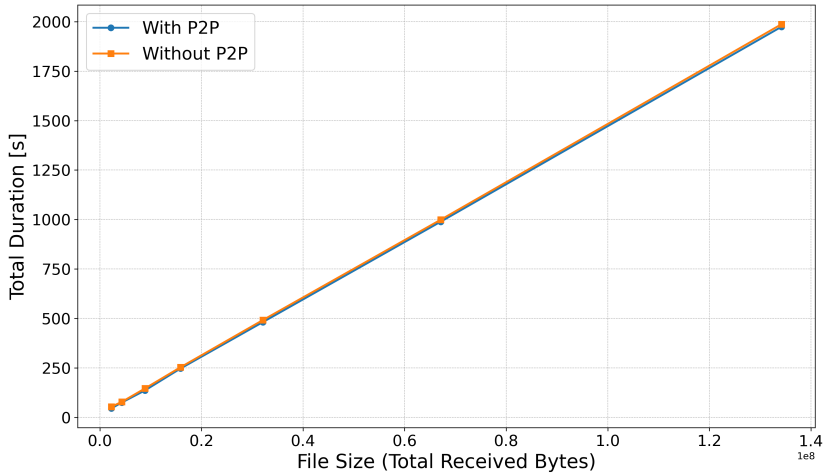
Time of Whole Data Transfer with the Second Configuration



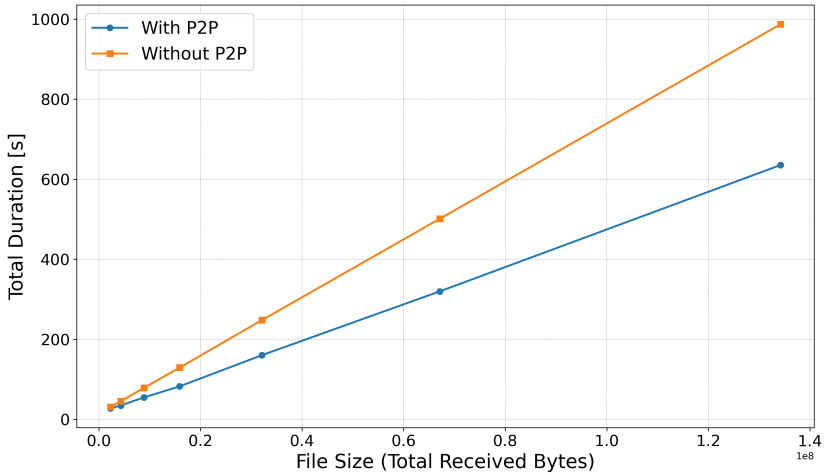
Performance Evaluation of P2P Algorithm, RQ2.2 and RQ2.3 (3)

- First Configuration with lectureStudio Server and 50 Peers
  - Transferring 2MB Data Size, P2P Algorithm Performance Nearly Identical to Traditional Approach
  - Transferring 128MB Data Size, P2P Algorithm Performance 5% Faster than Traditional Approach
- Second Configuration with lectureStudio Server and 50 Peers
  - Transferring 2MB Data Size, P2P Algorithm Performance 12% Faster than Traditional Approach
  - Transferring 128MB Data Size, P2P Algorithm Performance 35% Faster than Traditional Approach

Time of Whole Data Transfer with the First Configuration



Time of Whole Data Transfer with the Second Configuration



Challenges

- Finding real network dataset
- Configuration of the Network Characteristics for Connections
  - Bandwidth Limitation
  - Latency Addition
  - Packet Loss Simulation
- Synchronisation Problem of Total Time
- Allocation of Bandwidth for the Connections between LectureStudio Server and Peers
- Measurement of the Network Characteristics for Connections
- Data Transmission between LectureStudio Server and Peers
- Monitoring by Grafana, Prometheus and cAdvisor

Conclusion and Future Work

- Goal: Develop a Testbed for P2P Data Distribution Algorithm
- Utilization of Docker and Containerlab for An Efficient, Isolated Testing Environment.
- Simulation of Real Network Environments and Complex Network Topologies
- High Replication Accuracy of Bandwidth and Latency
- Effective Scalability with Increasing Nodes
- High resource demand without P2P Algorithm; significant reduction with P2P Algorithm
- Effective Data Distribution and Reduced Server Load through P2P Algorithm
- Limited P2P Benefits for Small Files and Efficiency Improvements in Different Configurations
- Robustness of P2P Algorithm with Increased Peers and Data Size
- Enhancement of Packet Loss Simulation for Accurate Network Behavior
- Automation Between Testbed and P2P Algorithm Optimization
- Development of A Graphical Testbed interface for Easier Configuration and Real-Time Analysis

Thank you for your attention!

ANY QUESTIONS?

References

[1] https://www.data.gov.uk/dataset/dfe843da-06ca-4680-9ba0-fbb27319e402/uk-fixed-line-broadband-performance

[2] https://containerlab.dev/manual/topo-def-file/