WatDFS

A Project for Understanding Distributed Systems

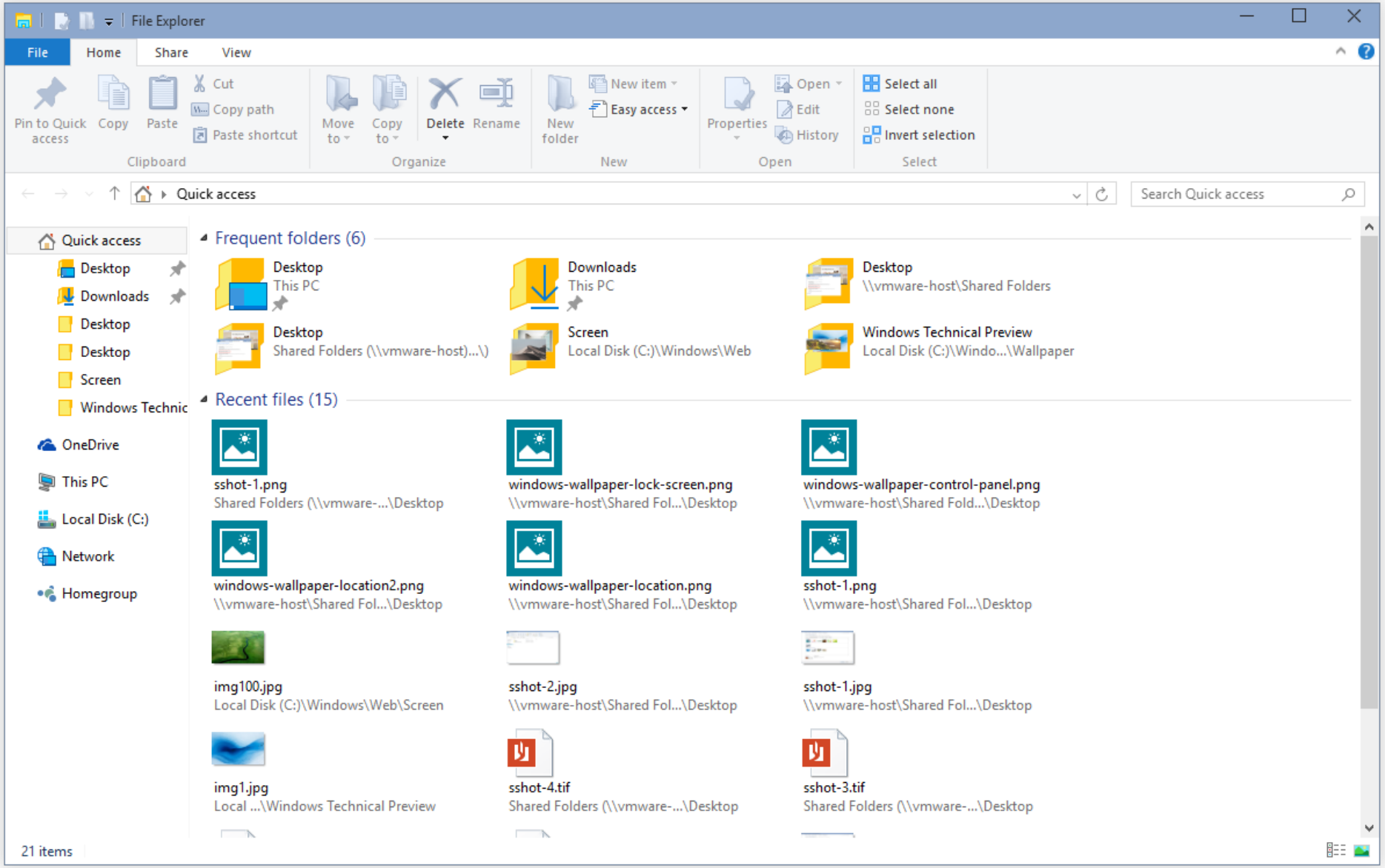
Michael Abebe

Brad Glasbergen

Khuzaima Daudjee

tiny.cc/watdfs

SIGCSE 2019



UNIVERSITY OF WATERLOO

UNIVERSITY OF WATERLOO

UNIVERSITY OF WATERLOO


Sun
microsystems


hadoop


ceph

Sun microsystems

Build Your Own Distributed File System

ceph

UNIVERSITY OF WATERLOO

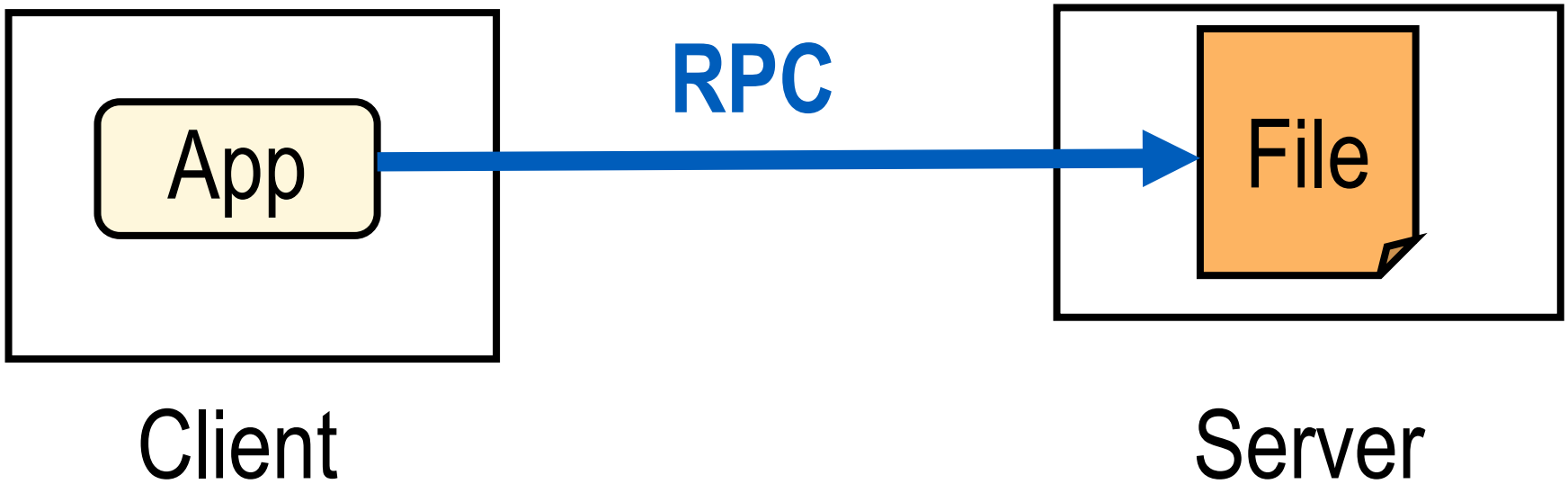
Project Goals

Cover wide range of course material

Interaction with **common** systems & applications

Provide **high-quality** and **timely feedback**

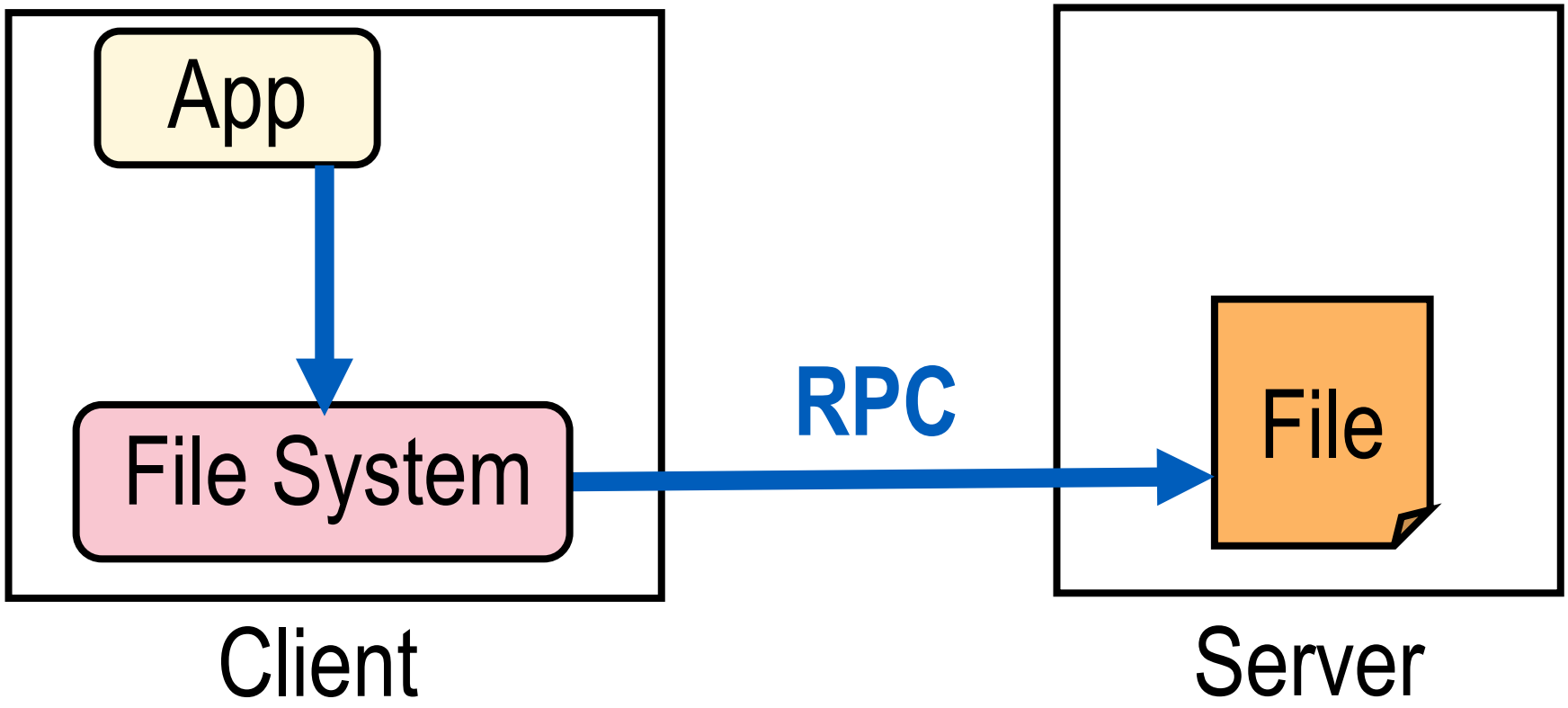
Distributed File Systems



Distributed File Systems

Lacks access transparency

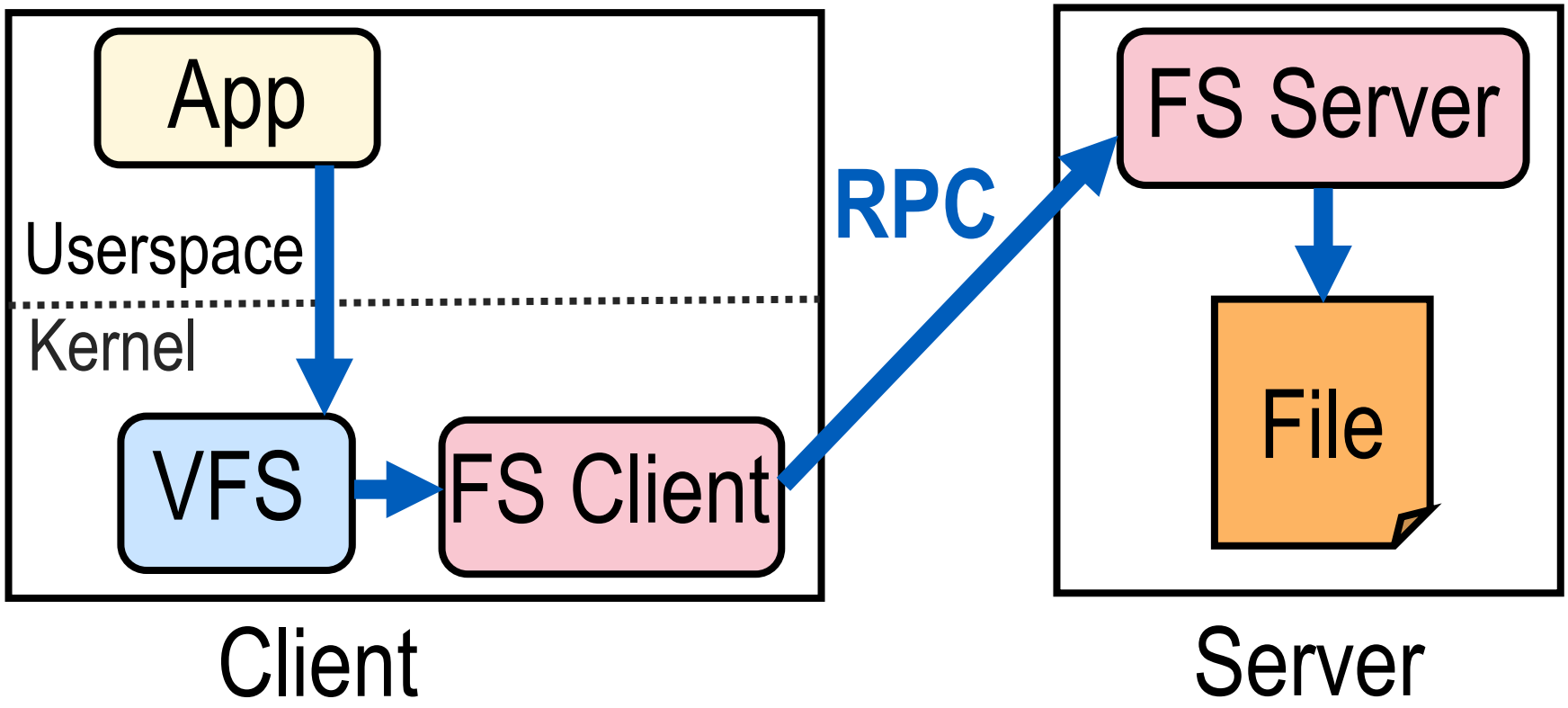
Distributed File Systems



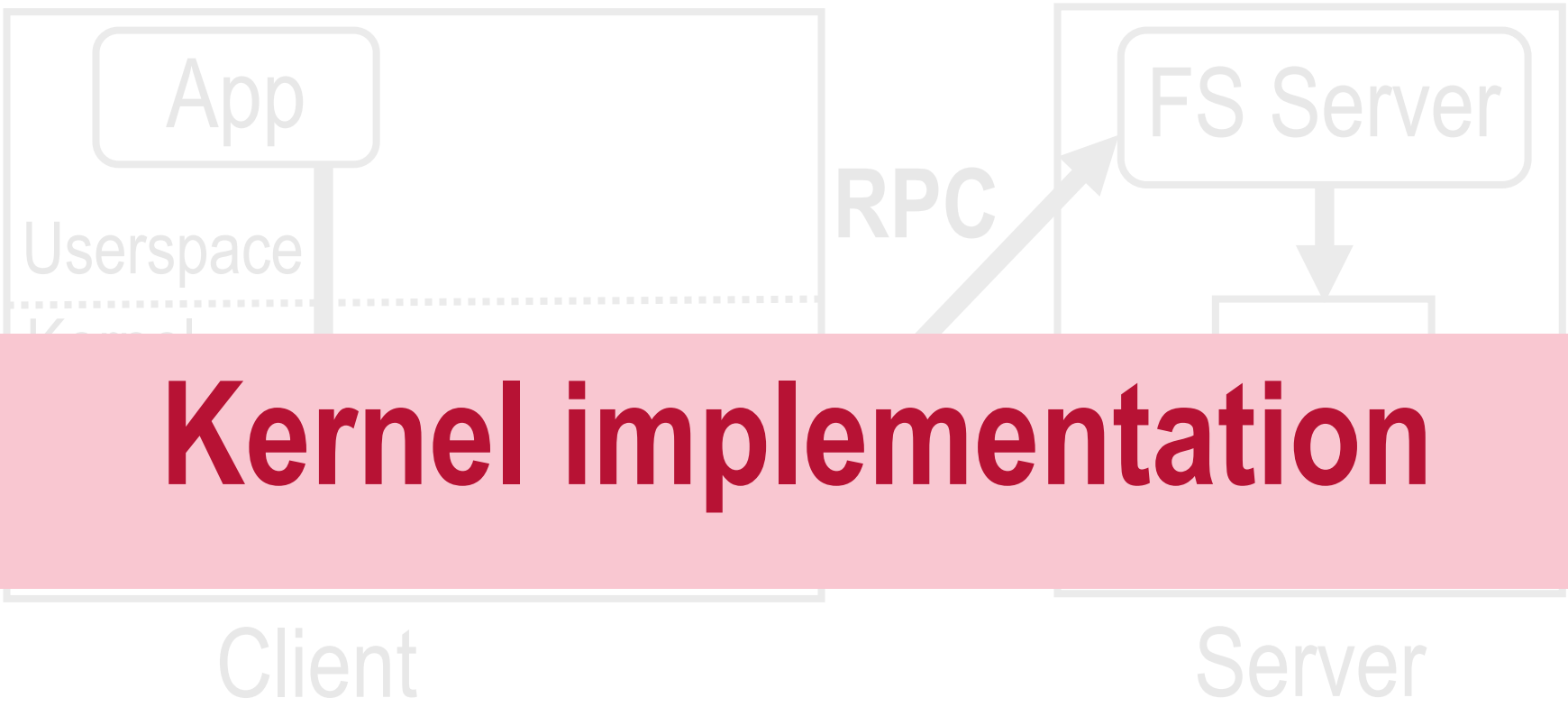
Distributed File Systems

Provides access transparency

Distributed File Systems

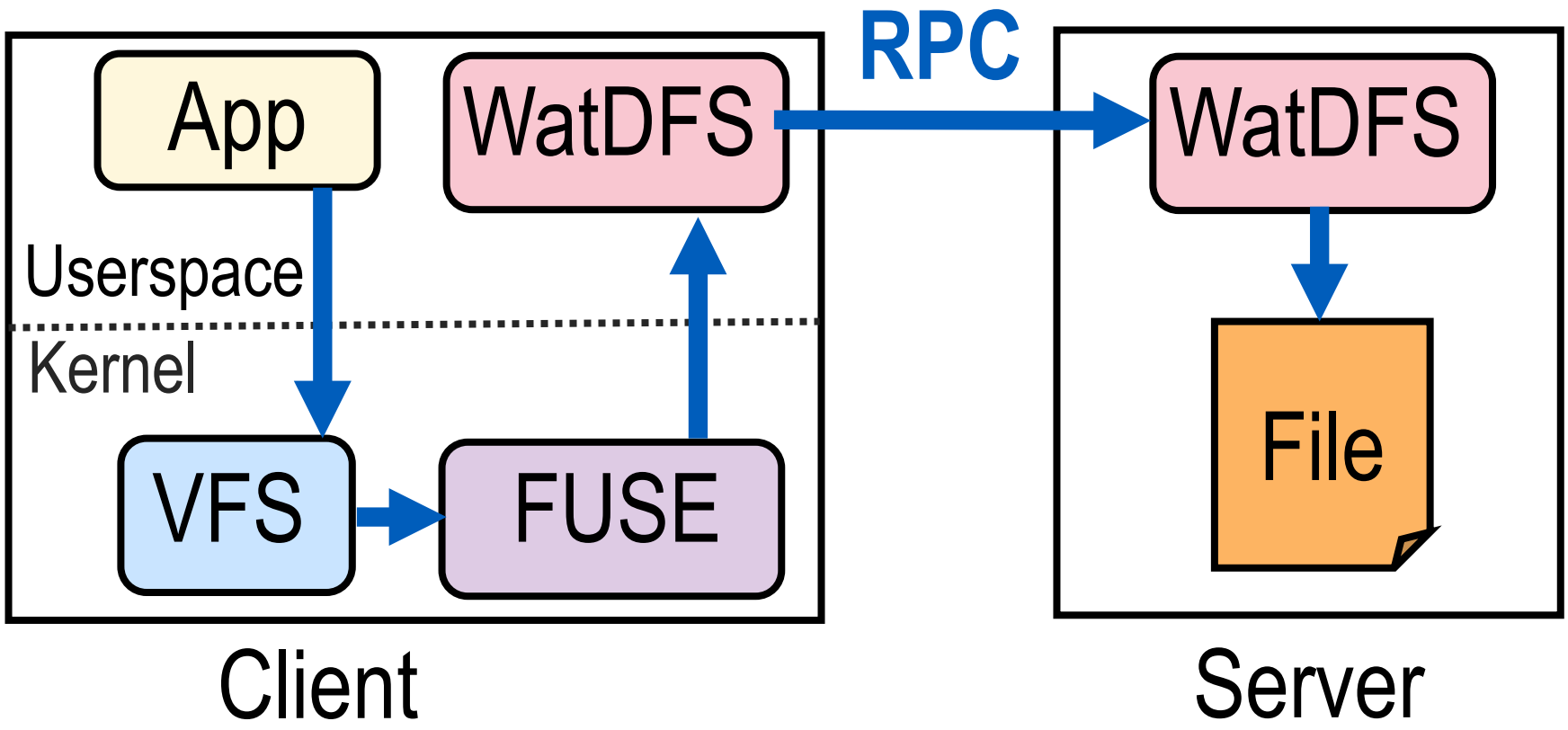


Distributed File Systems

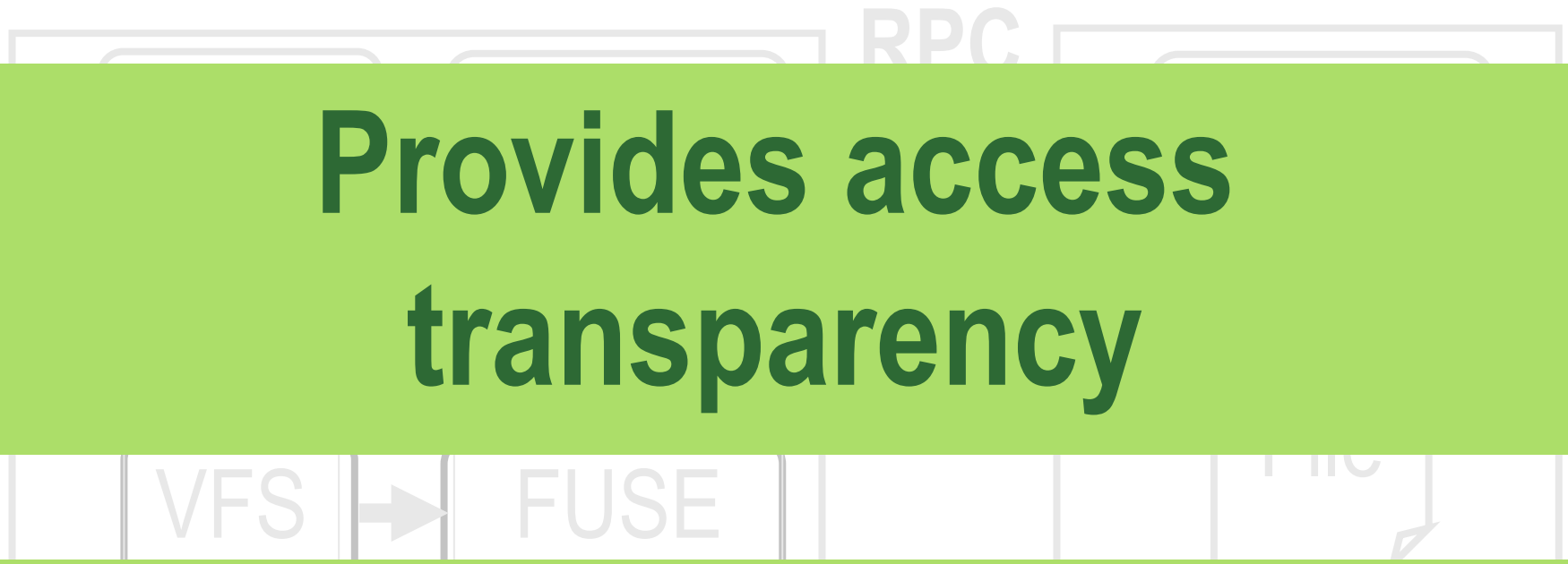


Kernel implementation

Distributed File Systems



Distributed File Systems



Provides access transparency

Userspace implementation

WatDFS Project

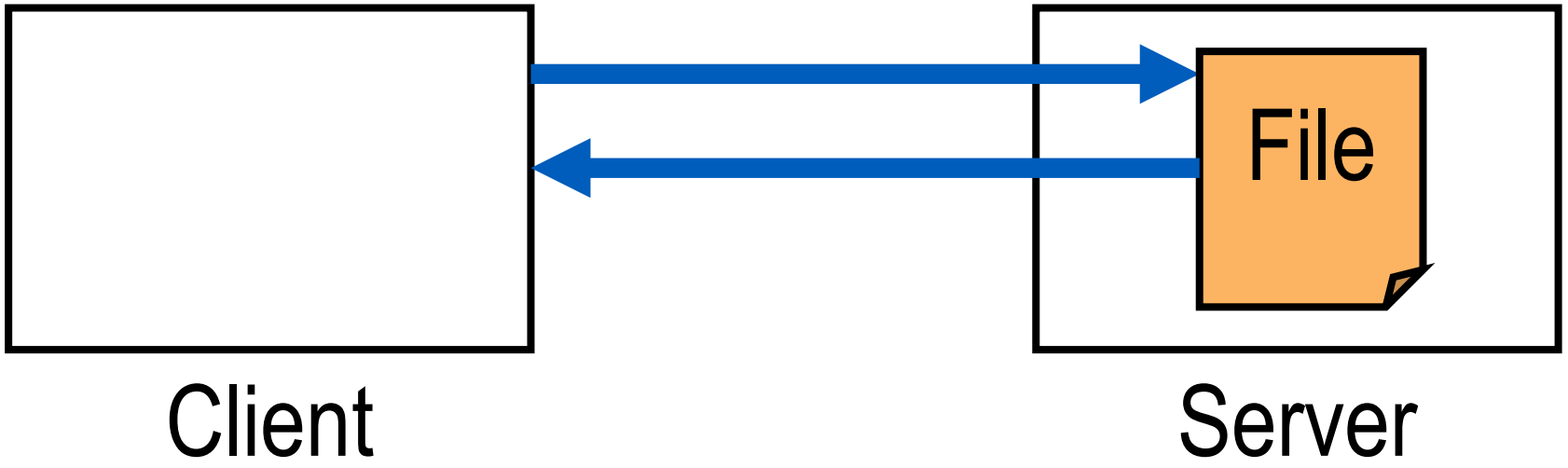
Implement WatDFS client and server:

Support: file creation, open, close, read, write, truncate, and metadata operations

Using two distributed file systems models

Remote-Access Model

Forward operations to server



File stays **at server**

Remote-Access Model

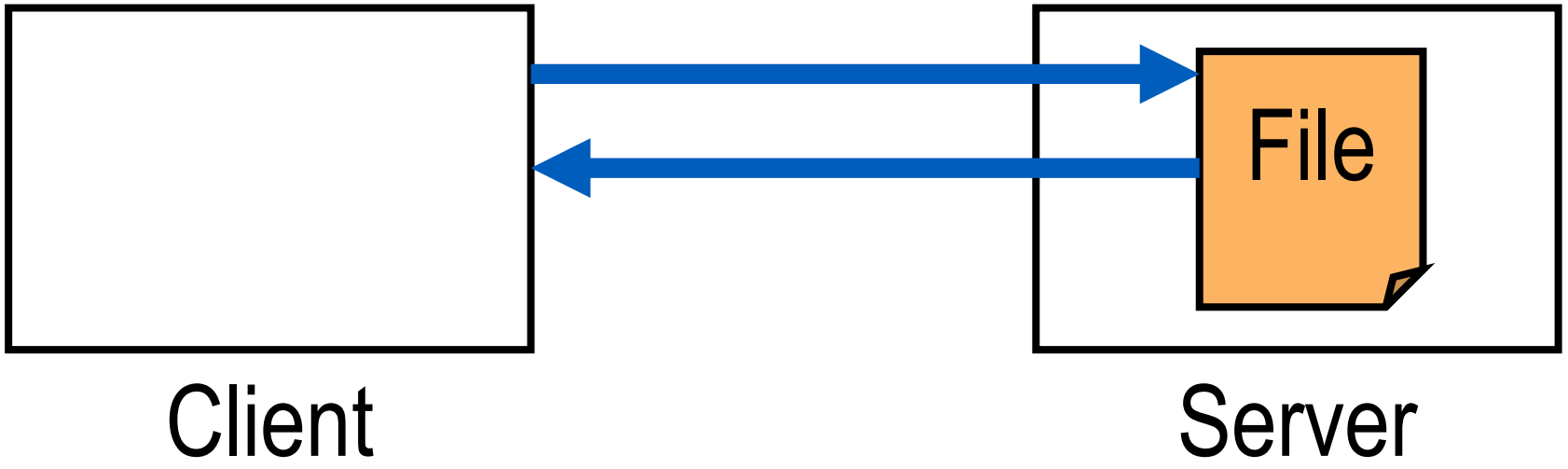
Forward operations to server

Increases Latency

File stays **at server**

UNIVERSITY OF WATERLOO

Remote-Access Model Learning Goals

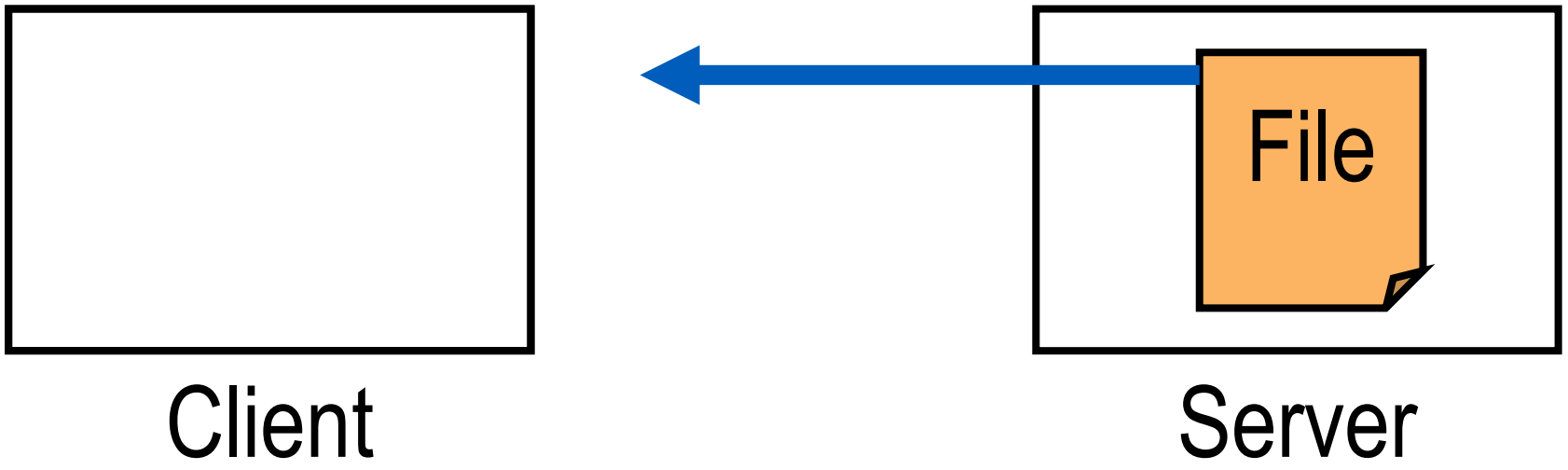


Introduce RPCs and file I/O

Familiarize tools (`libfuse`, `gdb`, `strace`)

Upload-Download Model

Download file from server

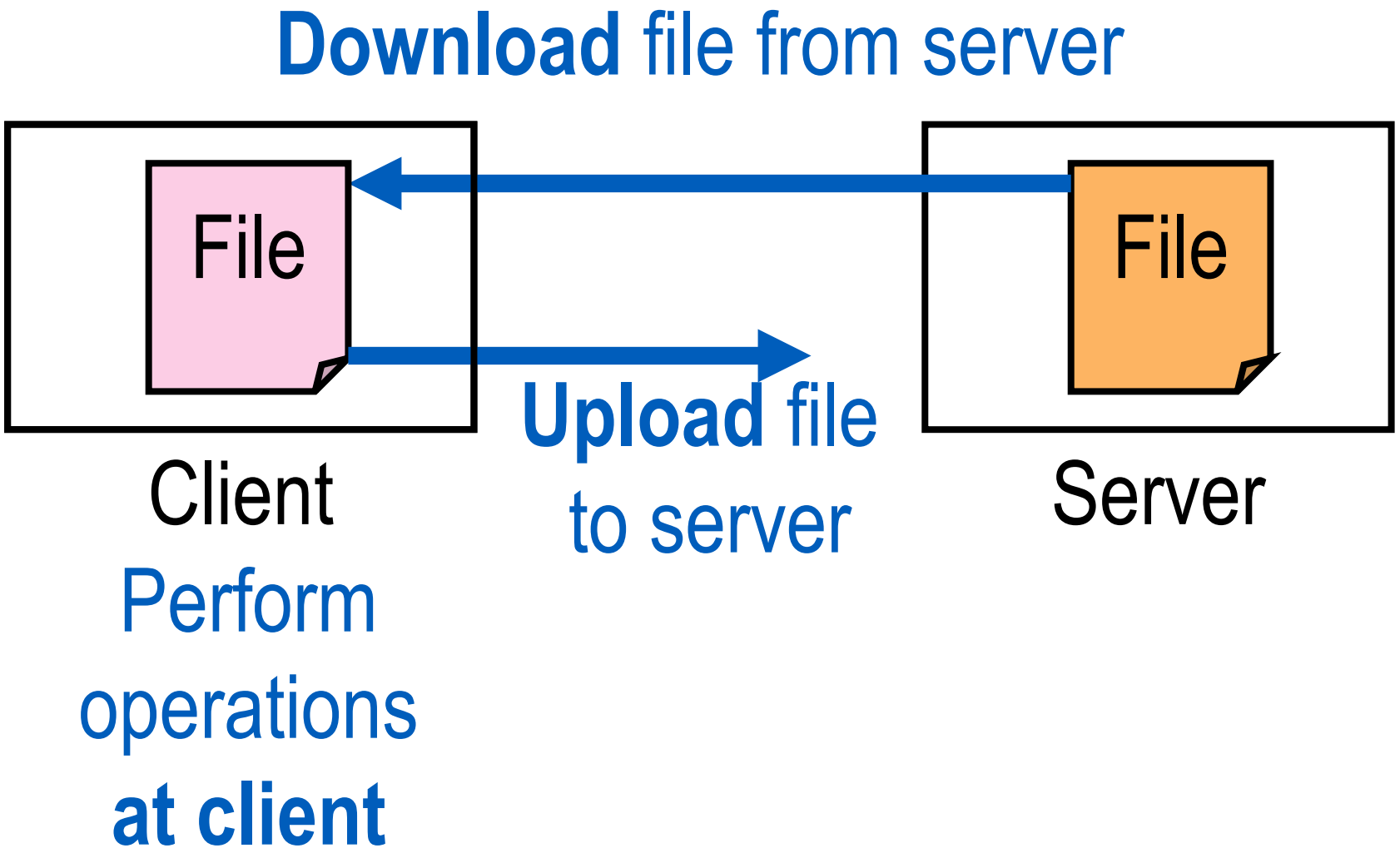


Upload-Download Model

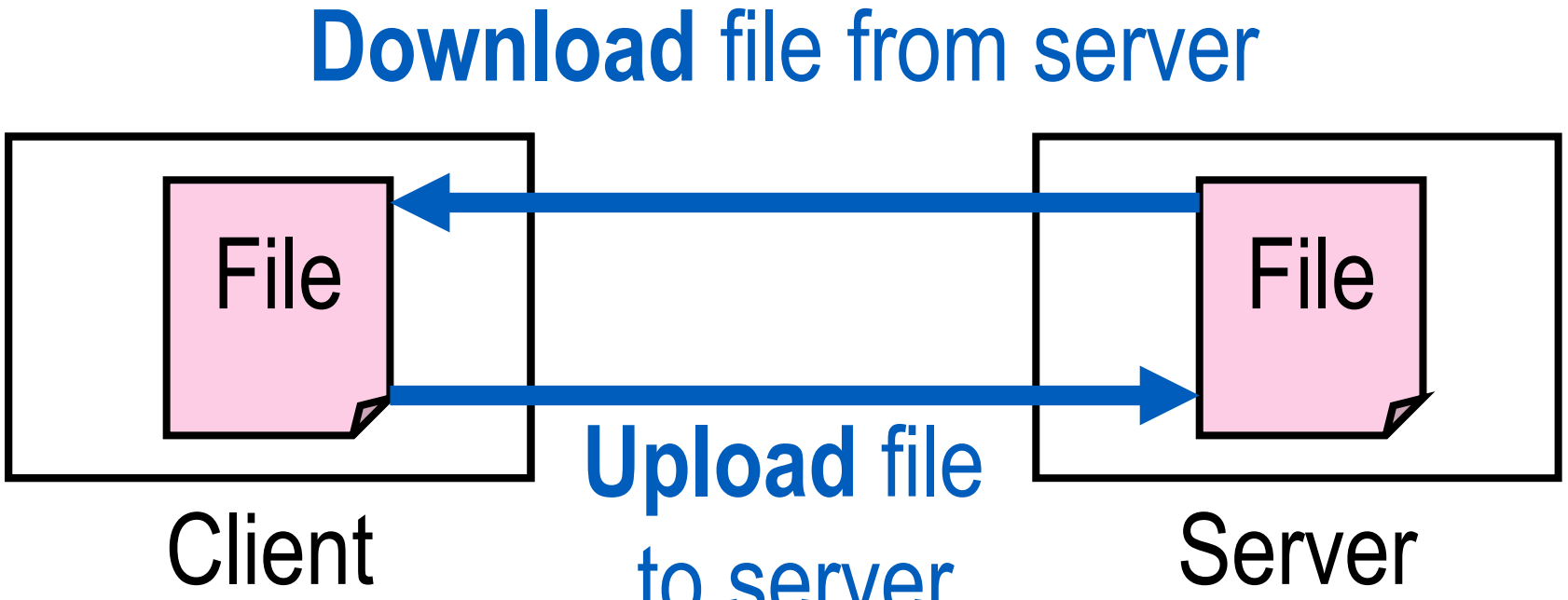
Download file from server

Perform operations **at client**

Upload-Download Model



Upload-Download Model



Perform operations **at client**

Upload-Download Model

Download file from server

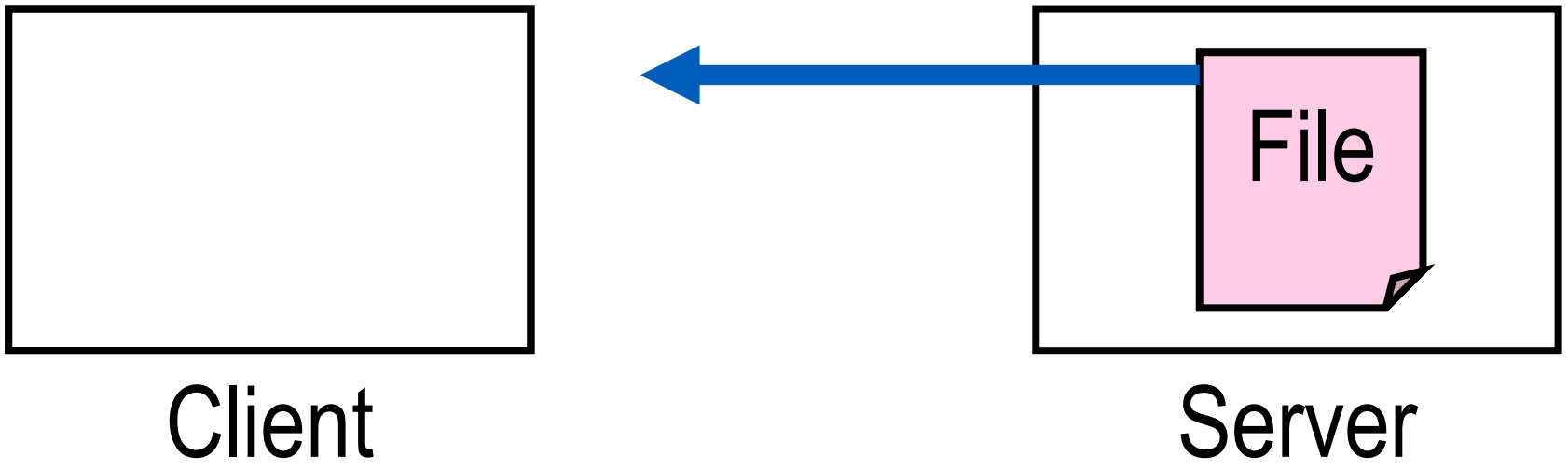
Reduces Latency

Perform operations **at client**

UNIVERSITY OF WATERLOO

Upload-Download Freshness

Download file from server



Upload-Download Freshness

Download file from server

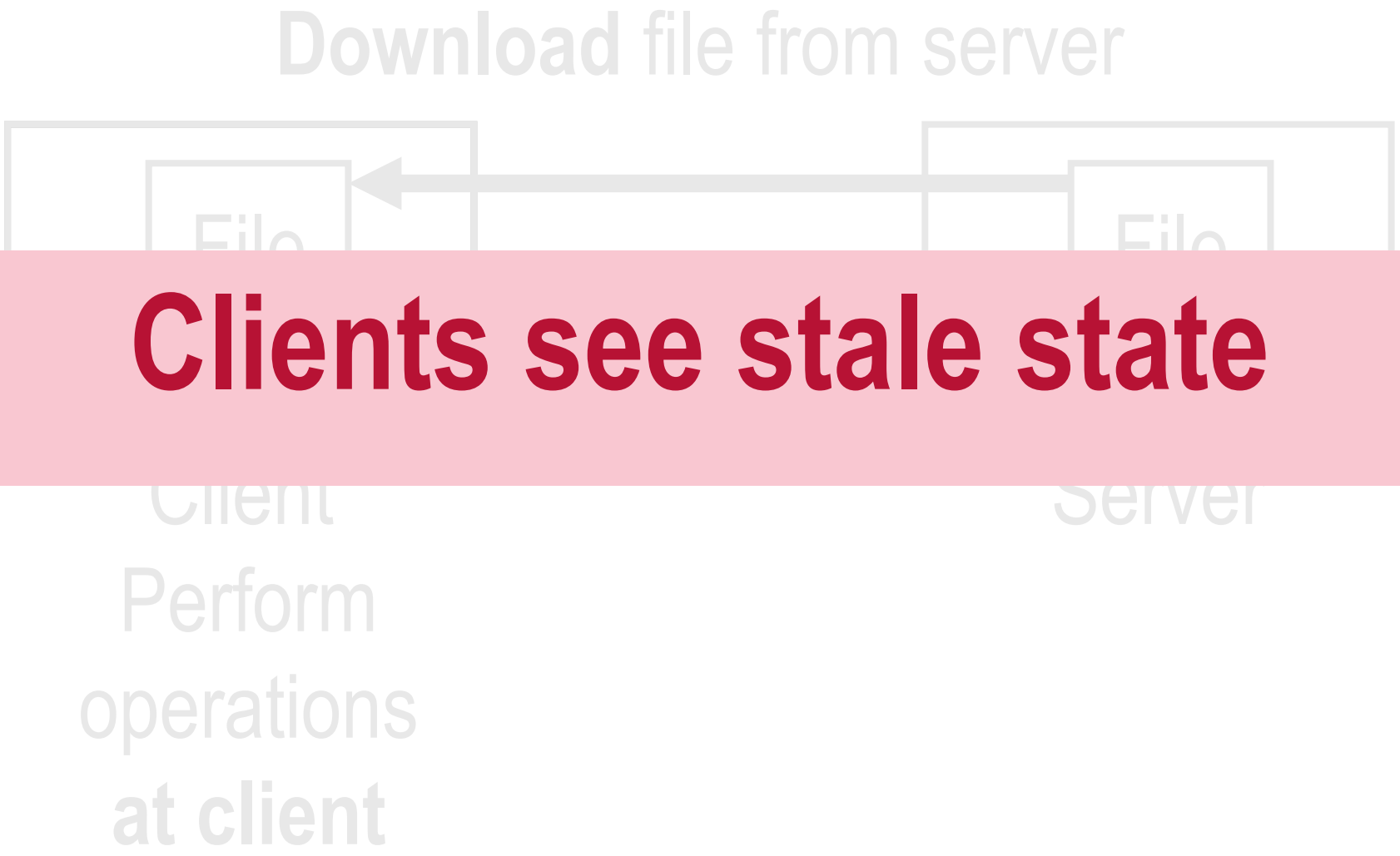
Perform operations **at client**

Upload-Download Freshness

Download file from server

Perform operations **at client**

Upload-Download Freshness



Clients see stale state

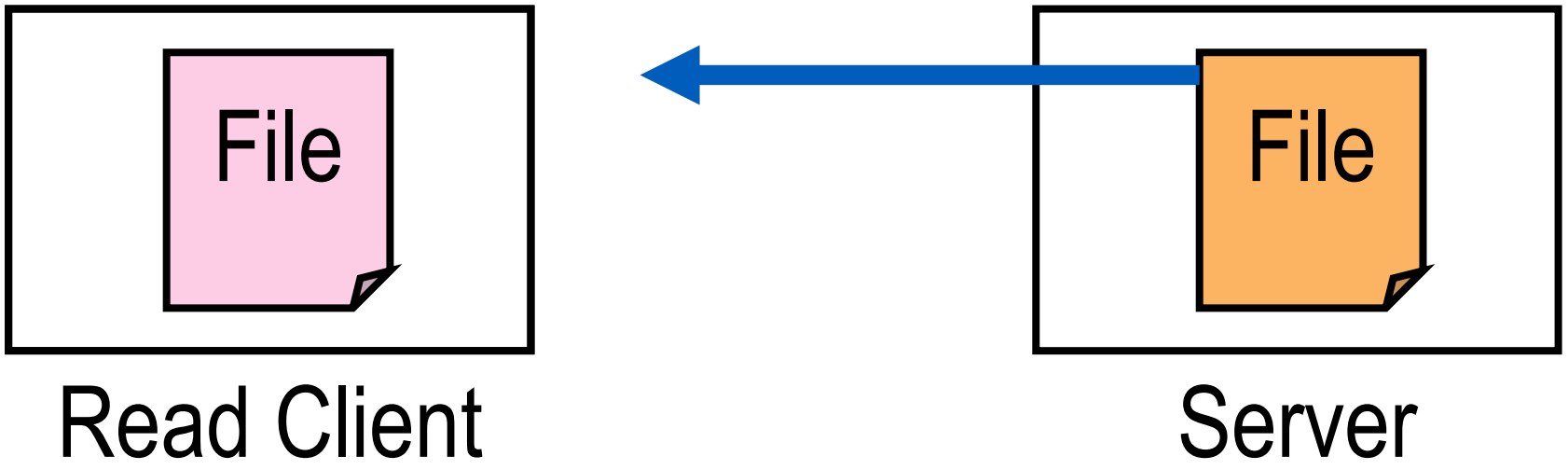
Upload-Download Freshness

How to ensure **freshness**?

Periodically upload and download

Upload-Download Freshness

Download file from server



Upload-Download Freshness

Download file from server

Upload-Download Freshness

Download file from server

Read Client

Server

Perform operations **at client**

Upload-Download Freshness

Download file from server

Perform operations **at client**

Upload-Download Freshness

Download file from server

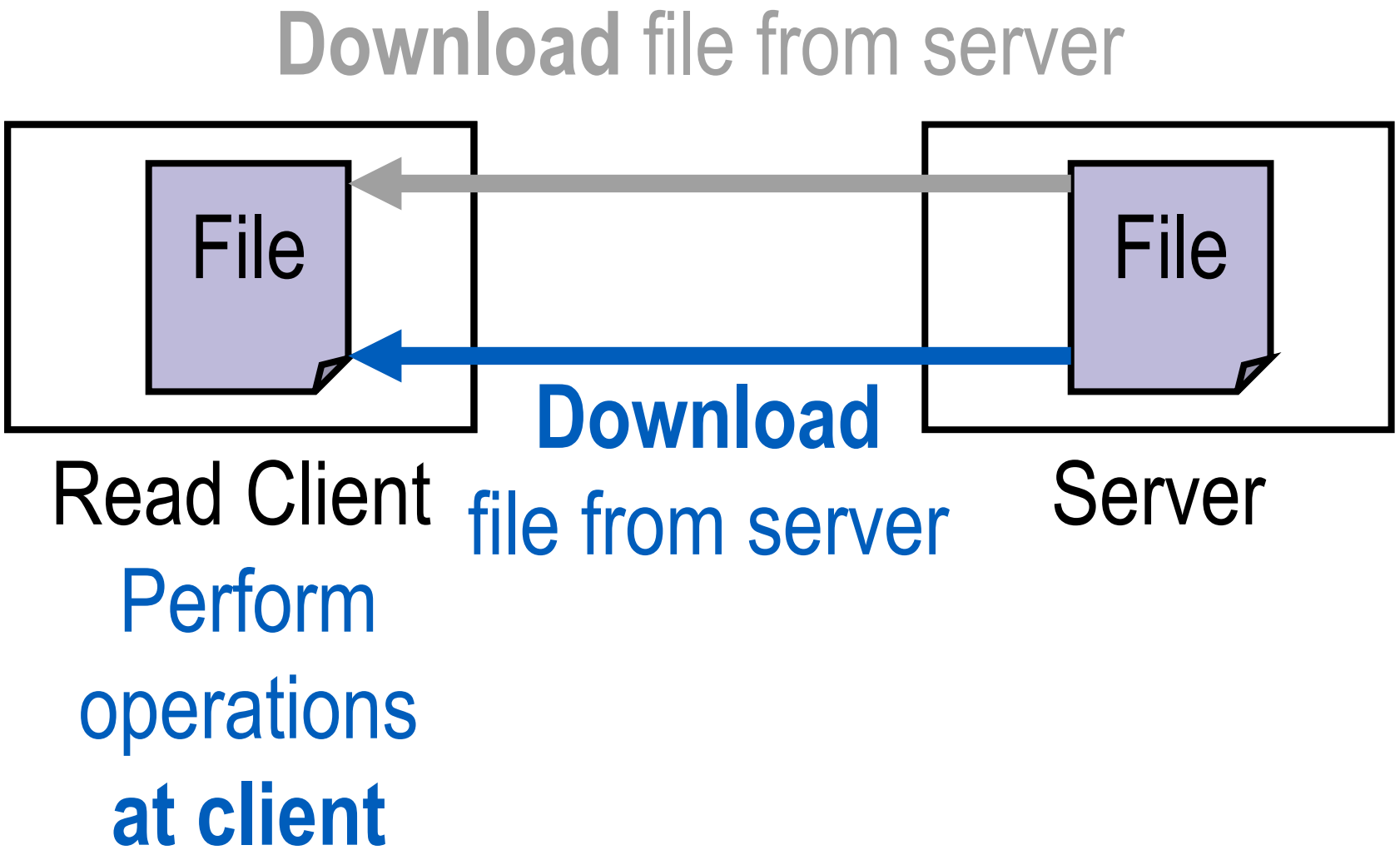
Download
file from server

Read Client

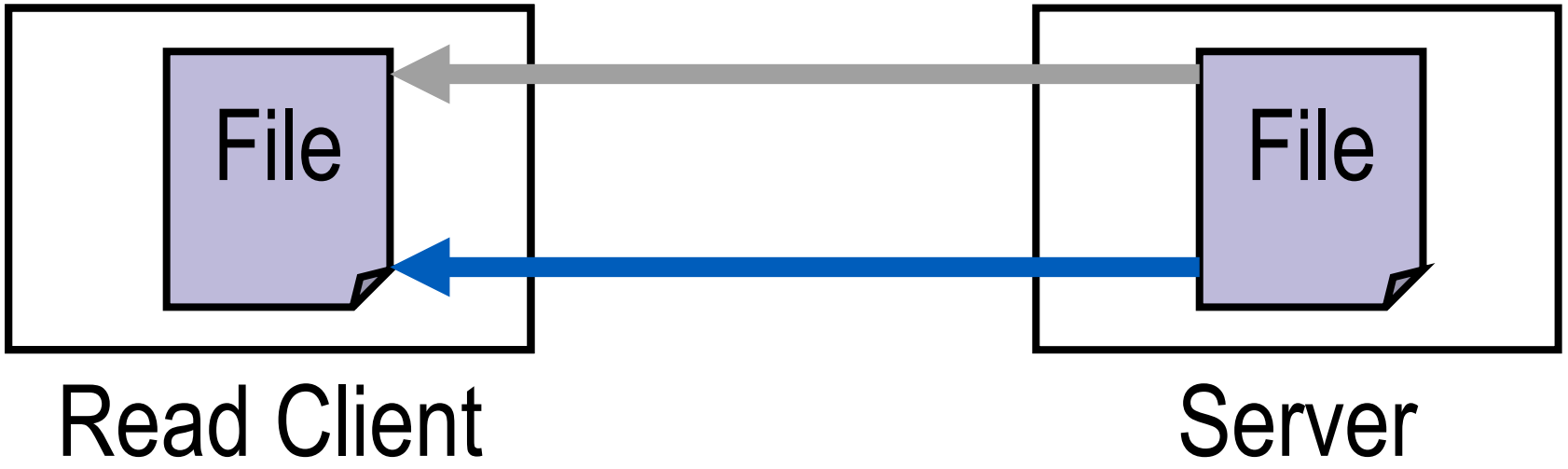
Server

Perform operations **at client**

Upload-Download Freshness



Upload-Download Freshness



Periodically upload and download using **timestamp-based cache consistency**

Upload-Download Freshness

Clients see fresher state

Reduces Latency

Details at tiny.cc/watdfs

Upload-Download Model Learning Goals

Manage distributed state with cache consistency

Use locks for atomicity and mutual exclusion

Experiences with WatDFS

Provide students with:

Detailed specification and Q&A forum

Public and release **tests** (Marmoset)

Starter code: ~300 lines of code

Experiences with WatDFS

Students implement

~760 lines of code for remote-access model

~1425 lines of code for upload-download model

Design document about upload-download model

Experiences with WatDFS

95% **passed all** remote-access model tests

80% **passed majority** of upload-download model tests

Most **common errors and questions** about timestamp-based cache consistency

WatDFS Project Summary

Implement two distributed file systems models

Covers wide range of course material

FUSE allows **usage of existing applications**

Tests provide **high-quality & timely feedback**

Details at tiny.cc/watdfs