Distributed Systems

Introduction to Distributed Systems

WEEK 1: Fundamentals, Design Goals, Scalability, and Fault Tolerance

Online Lecture Series - 1

Felix Edesa

Addis Ababa Science and Technology University

AASTU

Contents Outline

- From networked systems to distributed systems

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness
  - Dependability

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness
  - Dependability
  - Security

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness
  - Dependability
  - Security
  - Scalability

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness
  - Dependability
  - Security
  - Scalability
- A simple classification of distributed systems

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness
  - Dependability
  - Security
  - Scalability
- A simple classification of distributed systems
  - High-performance distributed computing

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness
  - Dependability
  - Security
  - Scalability
- A simple classification of distributed systems
  - High-performance distributed computing
  - Distributed information systems

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness
  - Dependability
  - Security
  - Scalability
- A simple classification of distributed systems
  - High-performance distributed computing
  - Distributed information systems
  - Pervasive systems

AASTU

Contents Outline

- From networked systems to distributed systems
  - Distributed versus decentralized systems
  - Why making the distinction is relevant
  - Studying distributed systems
- Design goals
  - Resource sharing
  - Distribution transparency
  - Openness
  - Dependability
  - Security
  - Scalability
- A simple classification of distributed systems
  - High-performance distributed computing
  - Distributed information systems
  - Pervasive systems

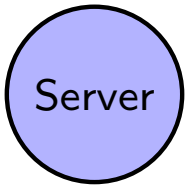
AASTU

Distributed Vs Decentralized Systems

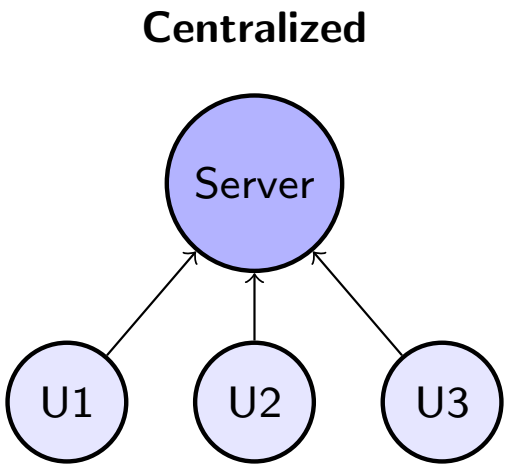
Centralized

Distributed Vs Decentralized Systems

Centralized



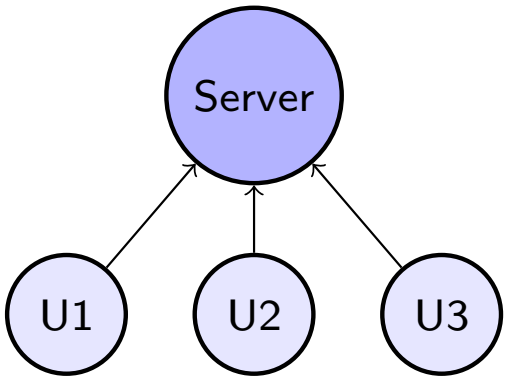
Distributed Vs Decentralized Systems



Distributed Vs Decentralized Systems

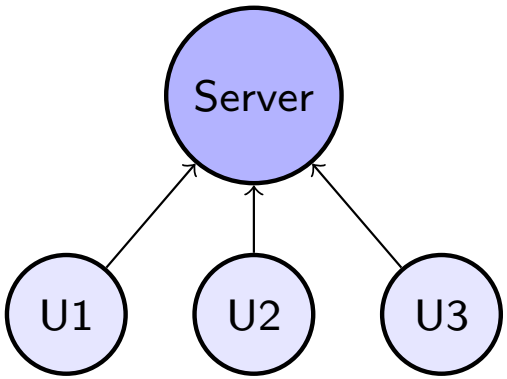
Decentralized

Centralized

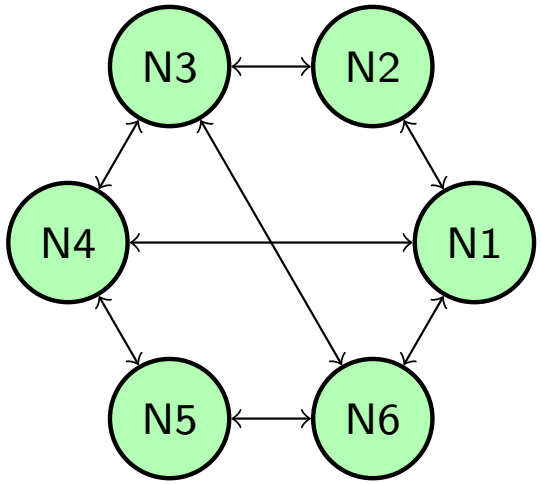


Distributed Vs Decentralized Systems

Centralized

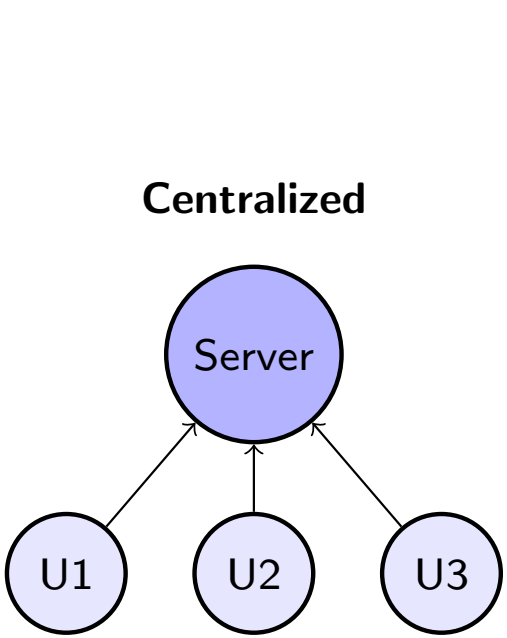


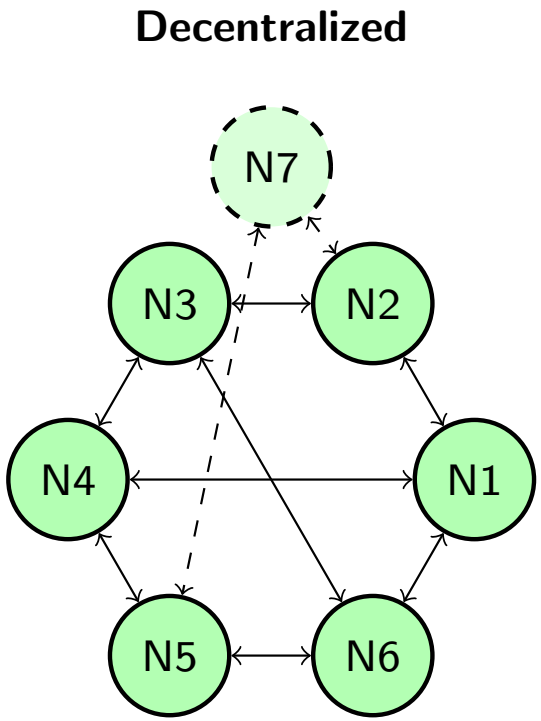
Decentralized



AASTU

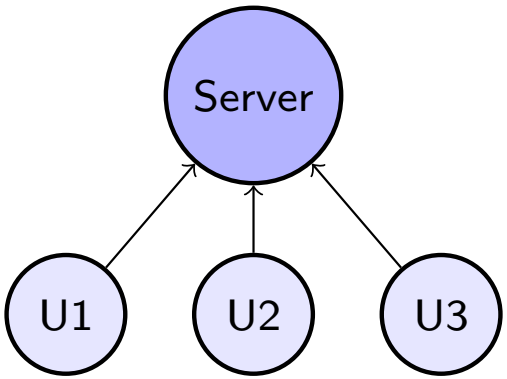
Distributed Vs Decentralized Systems



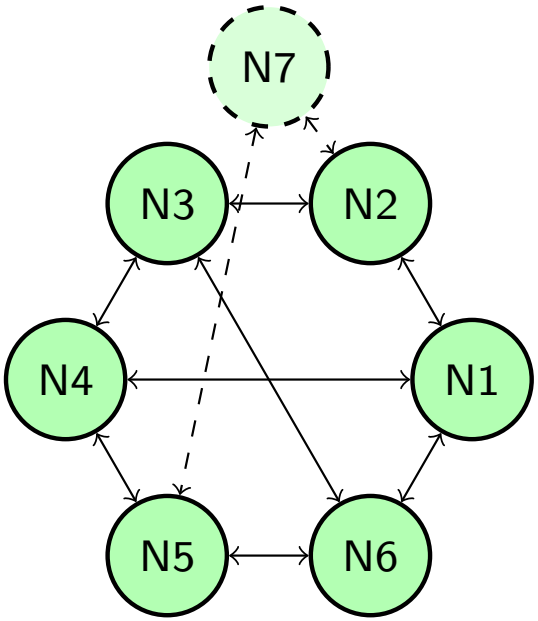


Distributed Vs Decentralized Systems

Centralized

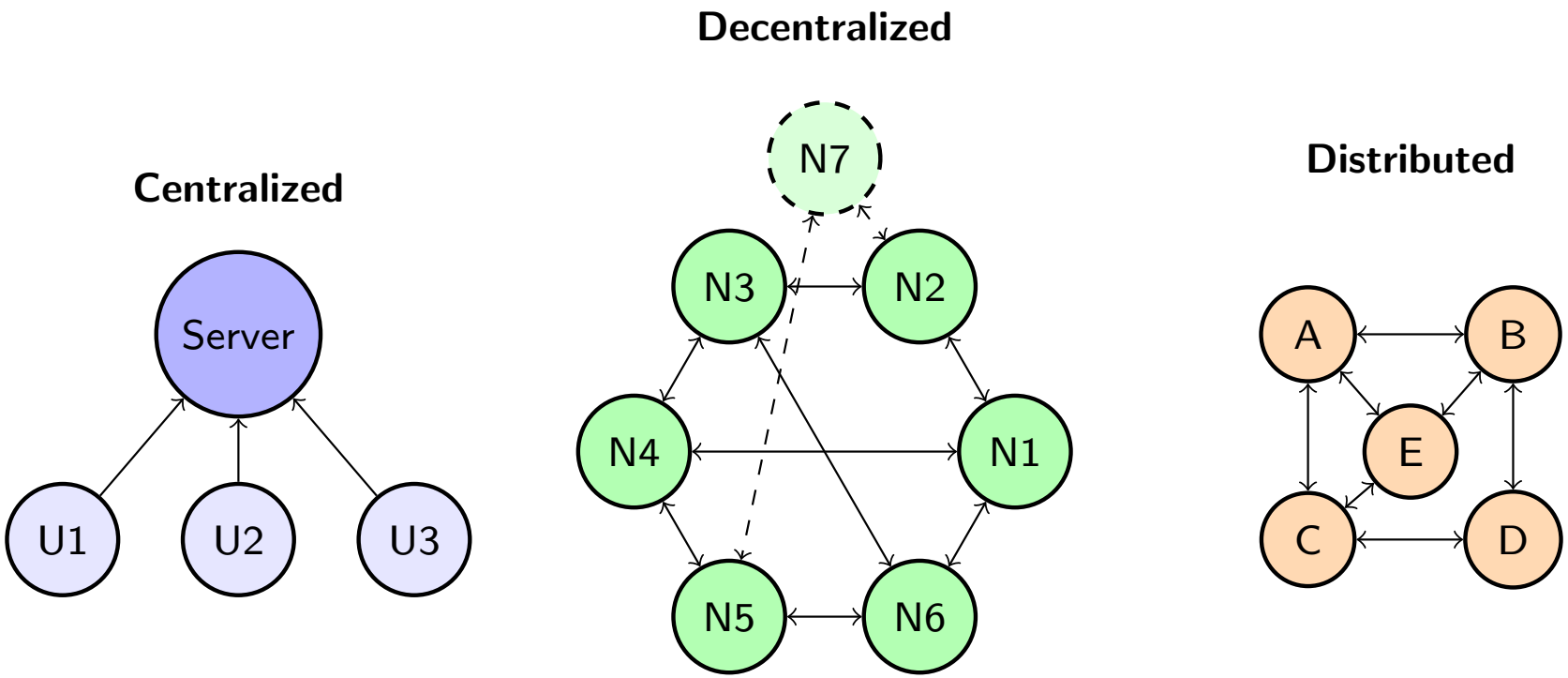


Decentralized



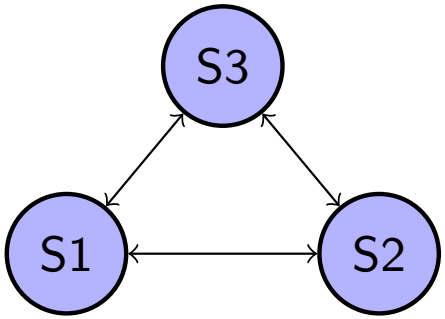
Distributed

Distributed Vs Decentralized Systems



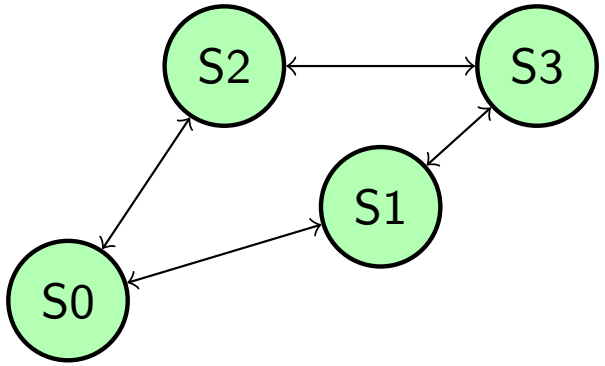
Two Views on Realizing Distributed Systems

Integrative View



Existing systems connected

Expansive View



Original system extended

Two Definitions

Decentralized System

- Processes and resources are necessarily spread across multiple computers
- No single point of control

Distributed System

- Processes and resources are sufficiently spread
- Components coordinate and collaborate
- Emphasizes system-wide integration

AASTU

Some Common Misconceptions

Centralized Solutions and Scalability

- Distinguish between:
  - Logically centralized
  - Physically centralized
- Example: Domain Name System (DNS)
  - Logically centralized
  - Physically massively distributed
  - Decentralized across multiple organizations
- Centralized solutions may face scaling limitations

AASTU

Some Common Misconceptions

Fault Tolerance and Management

- Centralized systems often perceived as having a single point of failure
- Not always true (e.g., DNS root)
- Single points of failure can be:
  - Easier to manage
  - Easier to make robust
- Common misconceptions exist regarding:
  - Scalability
  - Fault tolerance
  - Security

AASTU

Distributed Systems Goals

Overall Design Goals

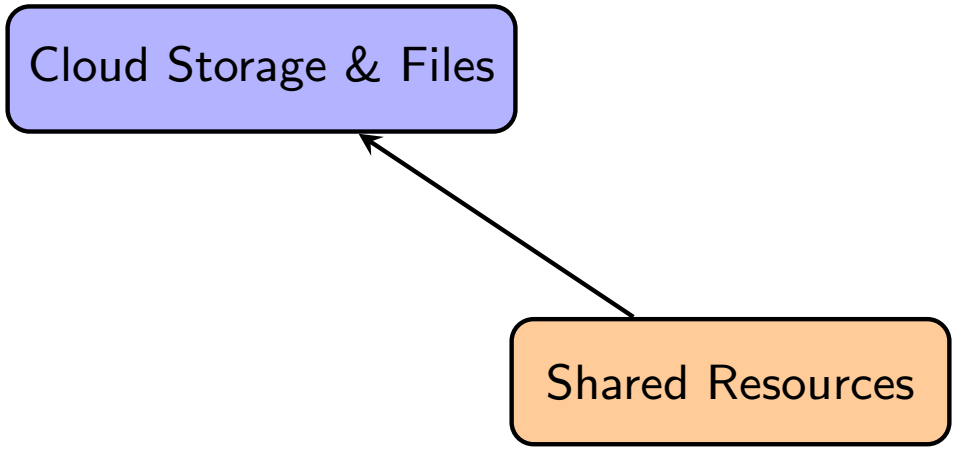
- **Support sharing of resources**
  Efficient use of distributed hardware and software resources.
- **Distribution transparency**
  Hide the complexity of the distributed system from users.
- **Openness**
  Provide standard interfaces for interoperability and extensibility.
- **Scalability**
  System should grow in size, geography, and administration seamlessly.

AASTU

Sharing Resources in Distributed Systems

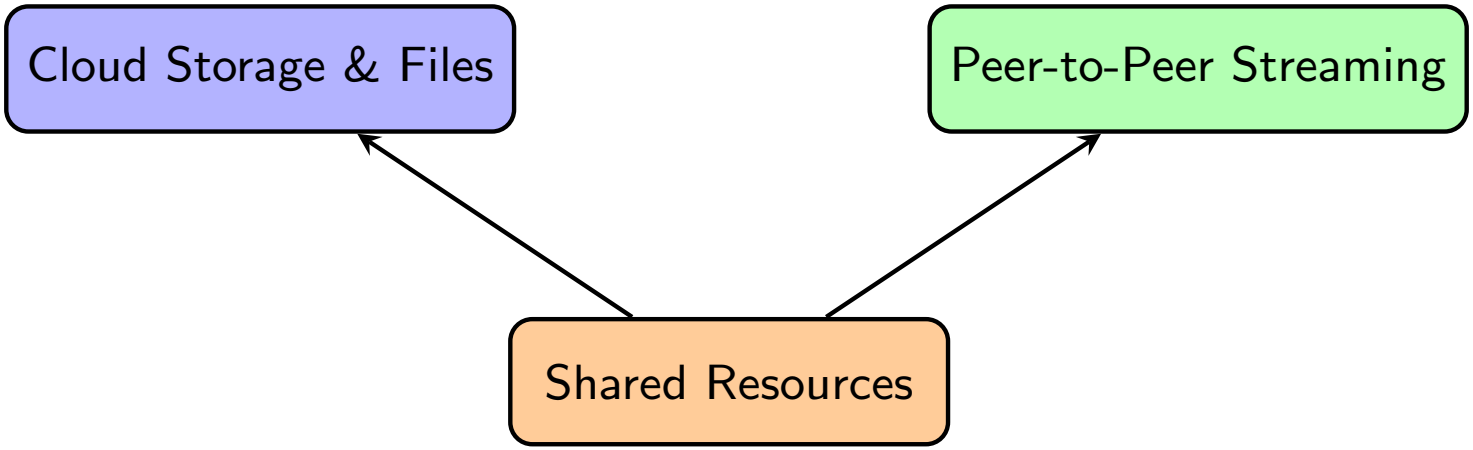
Shared Resources

Sharing Resources in Distributed Systems



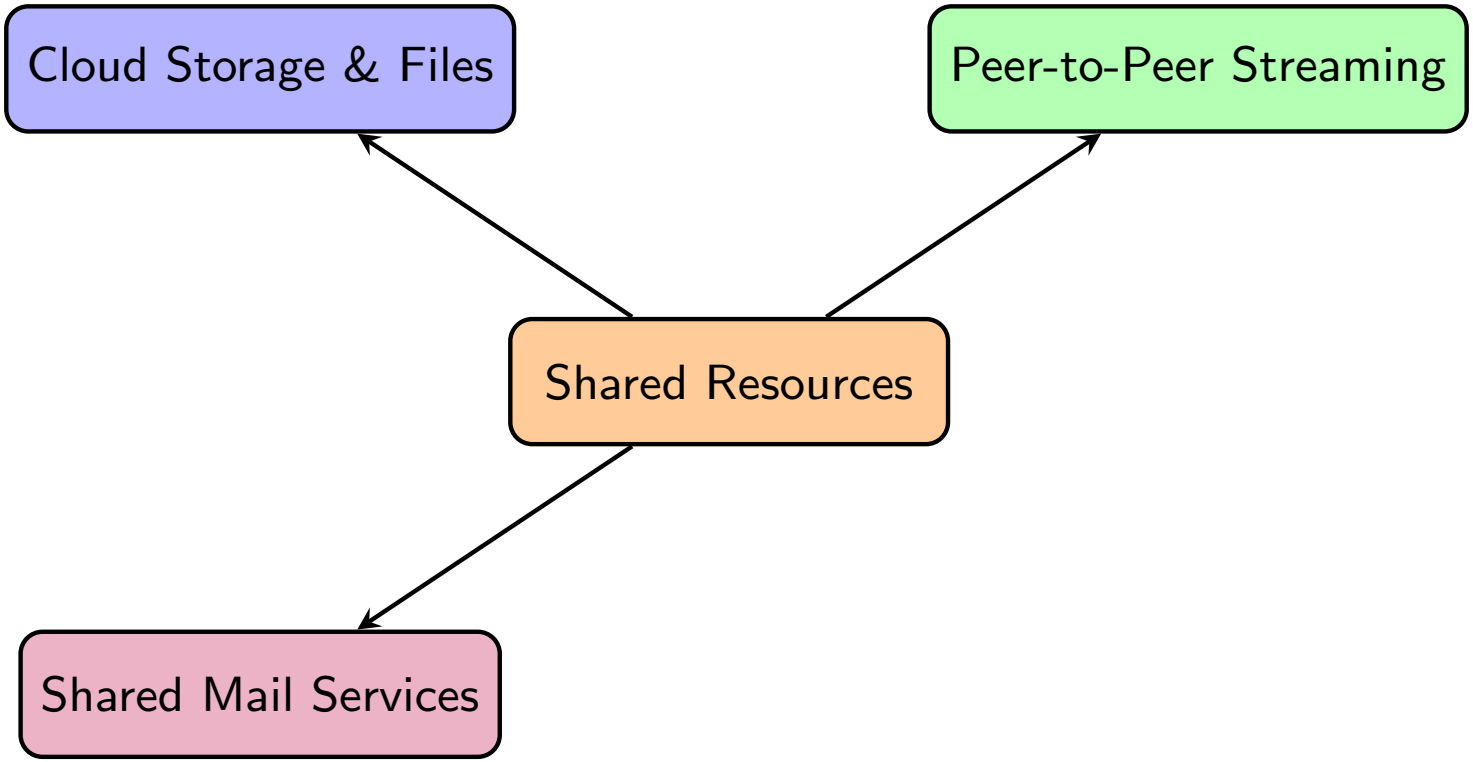
AASTU

Sharing Resources in Distributed Systems



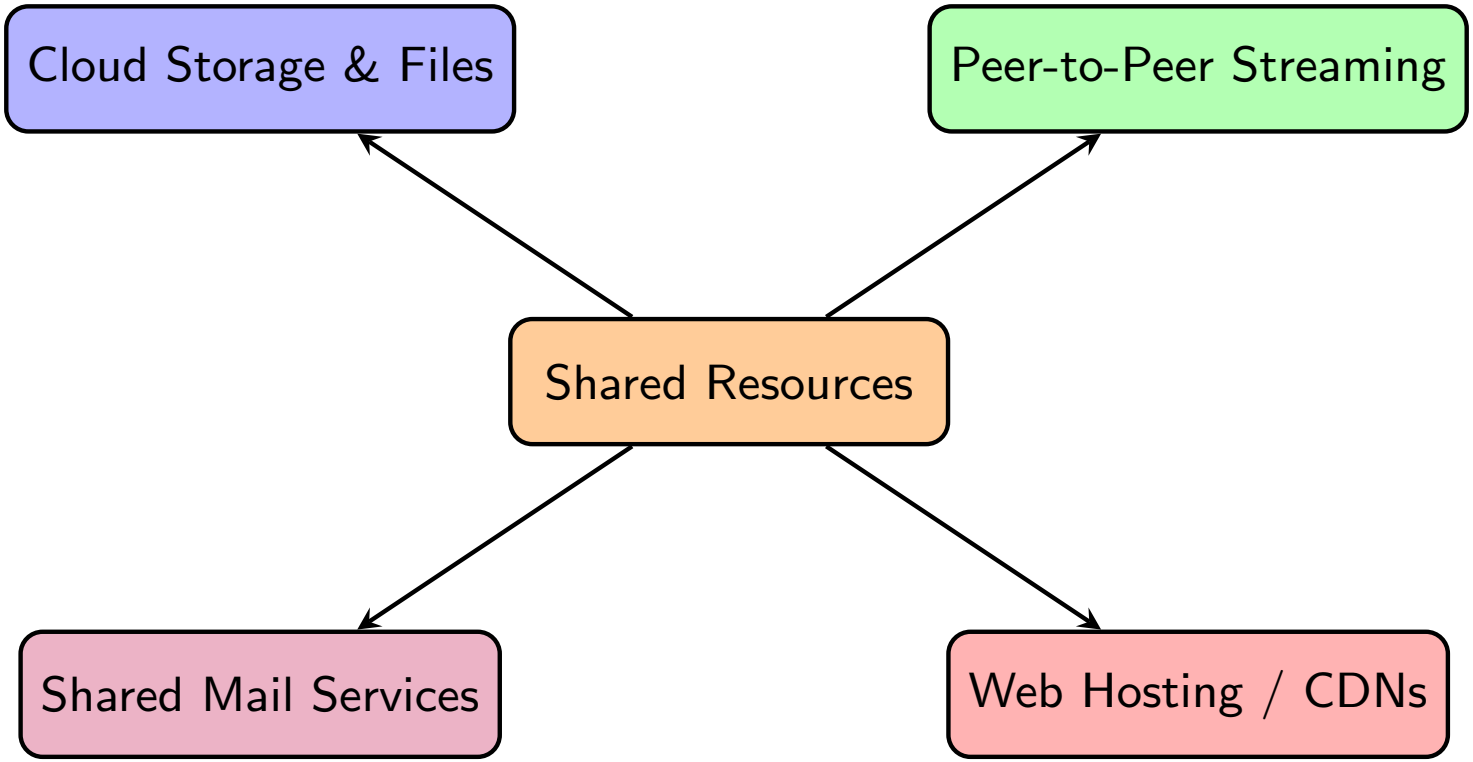
AASTU

Sharing Resources in Distributed Systems



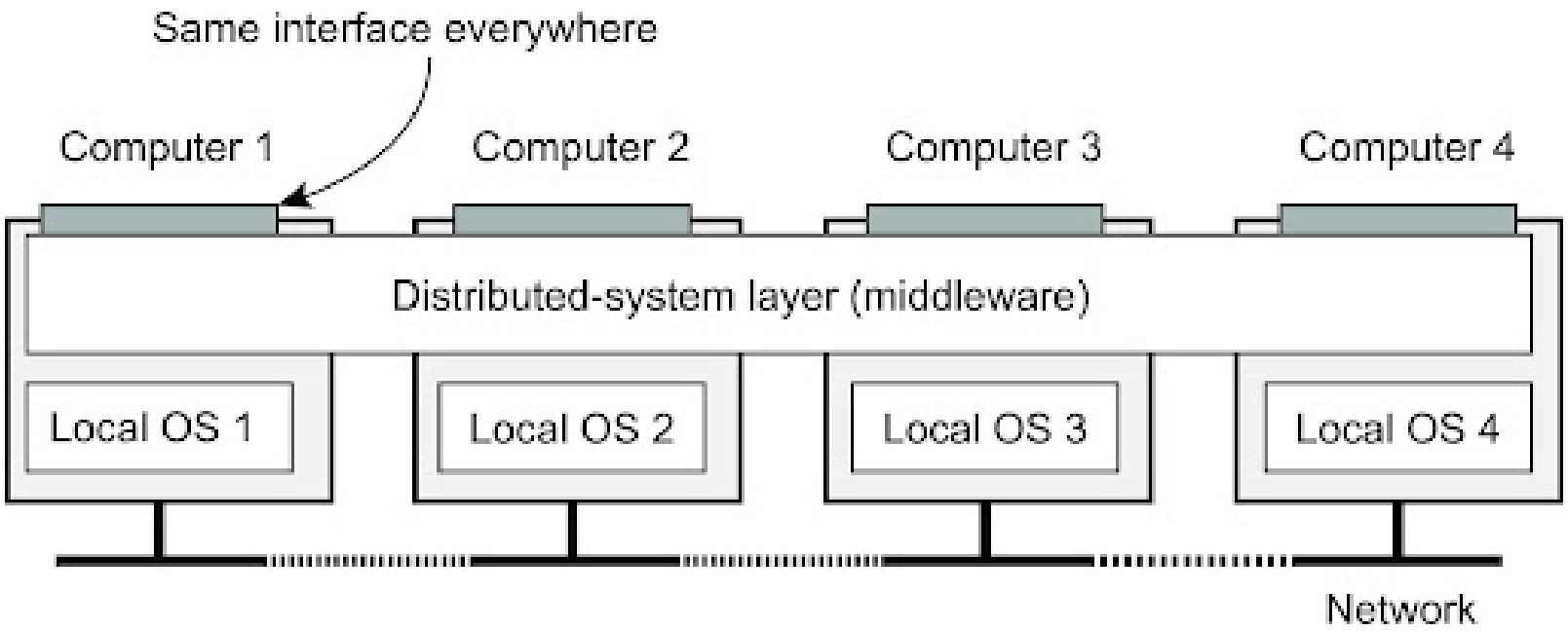
AASTU

Sharing Resources in Distributed Systems



AASTU

Distribution Transparency



M. van Steen and A. S. Tanenbaum, Distributed Systems, 4th ed., Version 02.

Transparency in Distributed Systems (Part 1)

Transparency	Description
Access	Hides differences in data representation and object access.
Location	Hides the physical location of an object.
Relocation	Hides that an object may move while in use.
Migration	Hides that an object may be moved elsewhere.

M. van Steen and A. S. Tanenbaum, *Distributed Systems*, 4th ed., Version 0.2, 2024.

AASTU

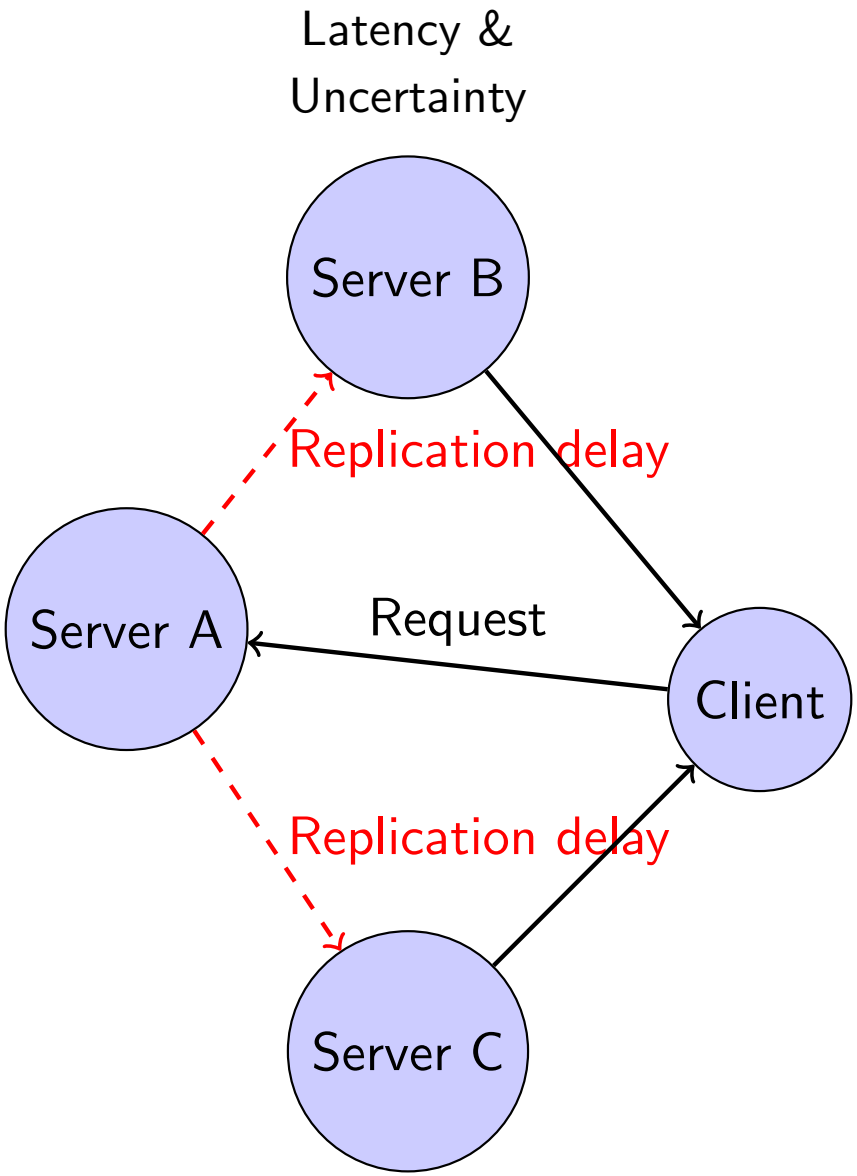
Transparency in Distributed Systems (Part 2)

Transparency	Description
Replication	Hide that an object is replicated.
Concurrency	Hide that an object may be shared by several independent users.
Failure	Hide the failure and recovery of an object.

M. van Steen and A. S. Tanenbaum, *Distributed Systems*, 4th ed., Version 0.2, 2024.

AASTU

Degree of Transparency in Distributed Systems



Openness in Distributed Systems: Definition

What is Openness?

- Components can be easily reused or shared across systems
- Integration with other independently developed components is seamless
- Supports flexibility and collaboration between different sources
- Encourages standard interfaces for smooth communication

AASTU

Openness of Distributed Systems: Key Features

Key Features:

- **Standardized Interfaces:** Ensures consistent communication

AASTU

Openness of Distributed Systems: Key Features

Key Features:

- **Standardized Interfaces:** Ensures consistent communication
- **Interoperability:** Works seamlessly with other systems

AASTU

Openness of Distributed Systems: Key Features

Key Features:

- **Standardized Interfaces:** Ensures consistent communication
- **Interoperability:** Works seamlessly with other systems
- **Portability:** Applications/components move across environments

AASTU

Openness of Distributed Systems: Key Features

Key Features:

- **Standardized Interfaces:** Ensures consistent communication
- **Interoperability:** Works seamlessly with other systems
- **Portability:** Applications/components move across environments
- **Extensibility:** New components or services added easily

AASTU

Openness of Distributed Systems: Key Features

Key Features:

- **Standardized Interfaces:** Ensures consistent communication
- **Interoperability:** Works seamlessly with other systems
- **Portability:** Applications/components move across environments
- **Extensibility:** New components or services added easily
- **Reusability:** Components reused in multiple systems

AASTU

Openness of Distributed Systems: Key Features

Key Features:

- **Standardized Interfaces:** Ensures consistent communication
- **Interoperability:** Works seamlessly with other systems
- **Portability:** Applications/components move across environments
- **Extensibility:** New components or services added easily
- **Reusability:** Components reused in multiple systems
- **Loose Coupling:** Minimal dependencies

AASTU

Openness of Distributed Systems: Key Features

Key Features:

- **Standardized Interfaces:** Ensures consistent communication
- **Interoperability:** Works seamlessly with other systems
- **Portability:** Applications/components move across environments
- **Extensibility:** New components or services added easily
- **Reusability:** Components reused in multiple systems
- **Loose Coupling:** Minimal dependencies

AASTU

Openness of Distributed Systems: Key Features

Key Features:

- **Standardized Interfaces:** Ensures consistent communication
- **Interoperability:** Works seamlessly with other systems
- **Portability:** Applications/components move across environments
- **Extensibility:** New components or services added easily
- **Reusability:** Components reused in multiple systems
- **Loose Coupling:** Minimal dependencies
- **Scalability:** Can grow/shrink easily

Openness in Distributed Systems: Examples Use Cases

Use Cases:

- Microservices (independent, reusable services)

AASTU

Openness in Distributed Systems: Examples Use Cases

Use Cases:

- Microservices (independent, reusable services)
- Open APIs (external integration)

AASTU

Openness in Distributed Systems: Examples Use Cases

Use Cases:

- Microservices (independent, reusable services)
- Open APIs (external integration)
- Cloud platforms (AWS + Azure components)

AASTU

Openness in Distributed Systems: Examples Use Cases

Use Cases:

- Microservices (independent, reusable services)
- Open APIs (external integration)
- Cloud platforms (AWS + Azure components)
- Collaborative tools (Git, Docker, Kubernetes)

AASTU

Openness in Distributed Systems: Examples Use Cases

Use Cases:

- Microservices (independent, reusable services)
- Open APIs (external integration)
- Cloud platforms (AWS + Azure components)
- Collaborative tools (Git, Docker, Kubernetes)
- IoT systems (devices interoperate via standards)

AASTU

Policies versus Mechanisms

Policies

- Consistency of client-cached data
- Allowed operations for downloaded code
- QoS adaptation to bandwidth changes

Mechanisms

- Dynamic caching policies
- Support for varying trust levels
- Adjustable QoS per data stream

AASTU

Dependability in Distributed Systems: Definition

Definition: Dependability is the system's ability to provide reliable services to clients, even when components interact with or rely on other components.

AASTU

Dependability in Distributed Systems: Key Points

Key Concepts: Component Dependencies

- A component provides services to clients but may require services from other components.

AASTU

Dependability in Distributed Systems: Key Points

Key Concepts: Component Dependencies

- A component provides services to clients but may require services from other components.
- Component C **depends on** C* if C's correctness relies on C* behaving correctly.

AASTU

Dependability in Distributed Systems: Key Points

Key Concepts: Component Dependencies

- A component provides services to clients but may require services from other components.
- Component C **depends on** C* if C's correctness relies on C* behaving correctly.
- Dependencies can exist between processes or communication channels.

AASTU

Requirements Related to Dependability

Requirement	Description
Availability	Ready for usage when needed
Reliability	Continuous, consistent service delivery
Safety	Very low probability of critical failures
Maintainability	Easy to repair or restore after failure

Source: Maarten van Steen Andrew S. Tanenbaum, *Distributed Systems*, Fourth Edition, Version 4.02, February 2024

AASTU

Reliability of a Component ()

- **Reliability** $R(t)$**:** Conditional probability that component C functions correctly during $[0, t)$, given it was working at $T = 0$.
- **Traditional Metrics:**
  - **Mean Time To Failure (MTTF):** Average time until a component fails.
  - **Mean Time To Repair (MTTR):** Average time required to repair a component.
  - **Mean Time Between Failures (MTBF):** $MTBF = MTTF + MTTR$
- Important: Higher reliability $\rightarrow$ longer operational periods and lower downtime.

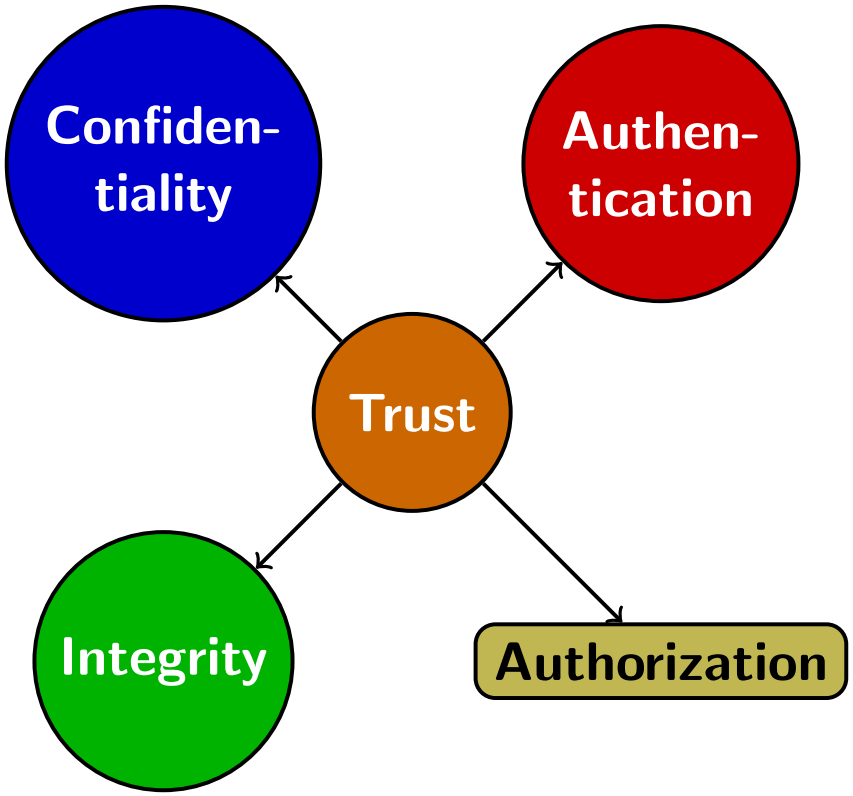
AASTU

Security

- **Observation:** A distributed system that is not secure, is not dependable.
- **What we need:**
  - Confidentiality – information disclosed only to authorized parties *(e.g., encrypted WhatsApp messages).*
  - Integrity – alterations only in authorized ways *(e.g., preventing bank transaction tampering).*
- **Key Concepts:**
  - Authentication – verifying correctness of identity *(e.g., login with username and password).*
  - Authorization – checking access rights *(e.g., file access permissions).*
  - Trust – ensuring expected actions are performed *(e.g., cloud provider reliably replicates data).*

AASTU

Security Diagram

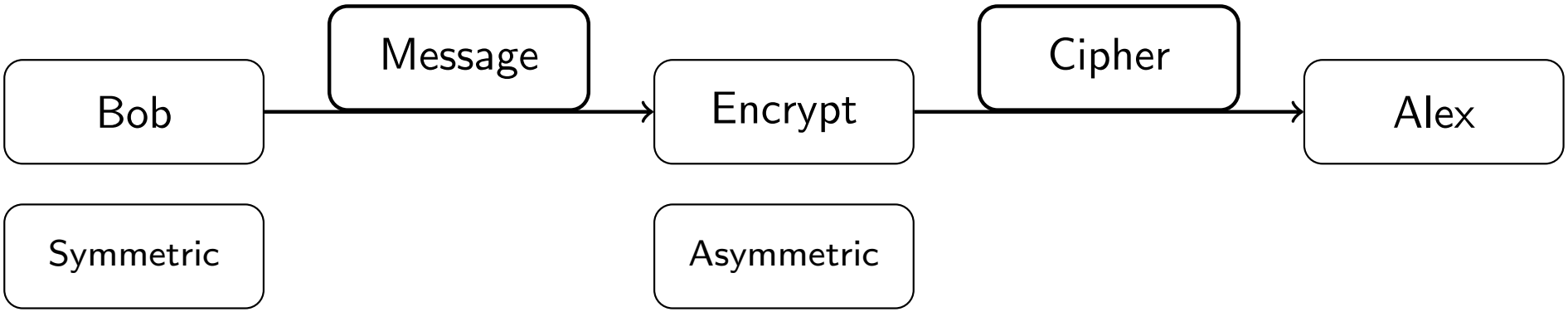


Security Mechanisms in Distributed Systems

- **Symmetric Cryptosystem:**
  - Uses a single secret key for both encryption and decryption.
  - Example: AES for encrypting files in cloud storage.
  - Fast and efficient for large data.
- **Asymmetric Cryptosystem:**
  - Uses a pair of keys: public key (encrypts) and private key (decrypts).
  - Example: RSA used in HTTPS websites, secure emails.
  - Enables secure communication without sharing the private key.

AASTU

Secure Message Exchange



Key idea: Messages must be secure in distributed systems.

Why It Matters in Distributed Systems

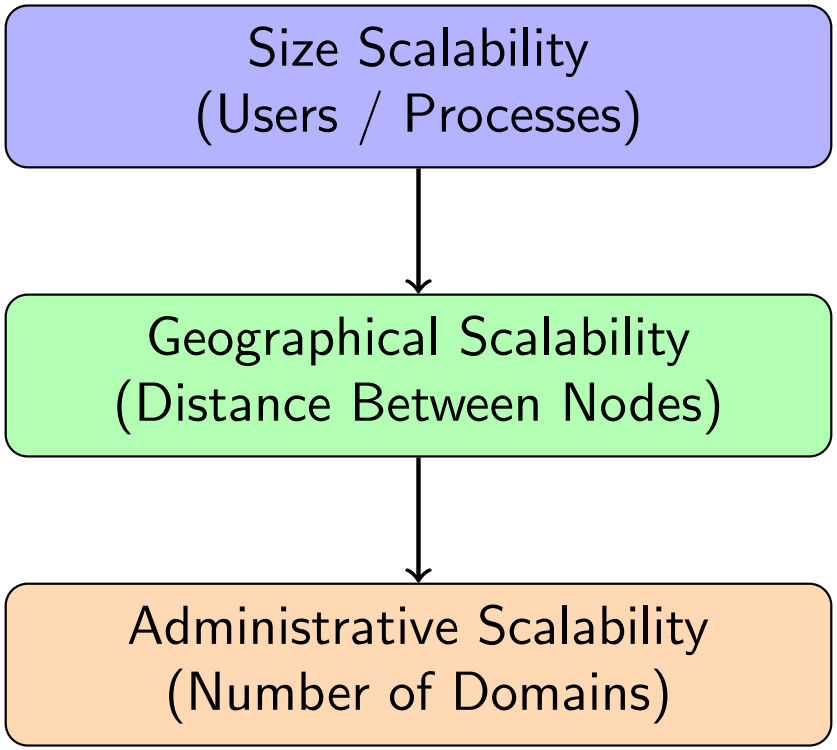
- Ensures that communication between nodes cannot be intercepted or tampered with.
- Maintains confidentiality, integrity, and trust across distributed nodes.
- Examples:
  - Cloud storage (AES encryption)
  - Secure web browsing (HTTPS using RSA/TLS)
  - Messaging apps (WhatsApp, Signal)

AASTU

Components of Scalability in Distributed Systems

Observation

Many distributed systems claim to be scalable. It is important to understand the types of scalability.

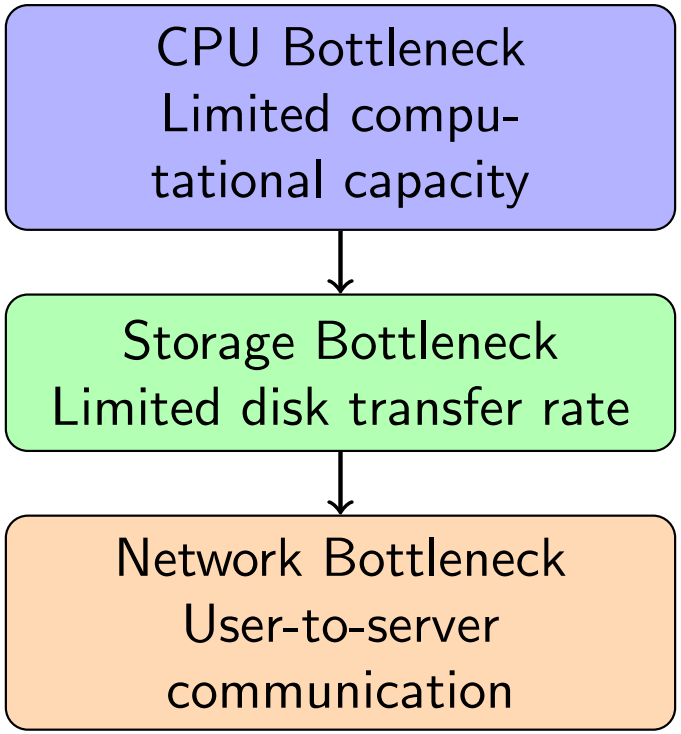


AASTU

Size Scalability in Distributed Systems (Part 1)

What is Size Scalability?

Ability of a system to handle **increasing number of users or processes** efficiently.



AASTU

Size Scalability in Distributed Systems (Part 2)

Observation

Centralized systems often face **size scalability problems** because:

- Increasing users overload the CPU.
- Disk I/O cannot keep up with demand.
- Network bandwidth becomes a bottleneck.

AASTU

Size Scalability in Distributed Systems (Part 3)

Solutions for Size Scalability

- Parallel processing: multiple servers work together.
- Load balancing: distribute user requests evenly.
- Optimize data transfer: caching compression.
- Horizontal scaling: add more machines instead of upgrading one.

AASTU

Size Scalability in Distributed Systems (Part 4)

Practical Insights

- Cloud platforms (AWS, Google Cloud, Azure) implement horizontal scaling.
- Overcome CPU, storage, and network bottlenecks efficiently.
- Example: 10,000 concurrent users handled across 5 servers instead of one.

AASTU

Formal Analysis of Centralized Services (Part 5)

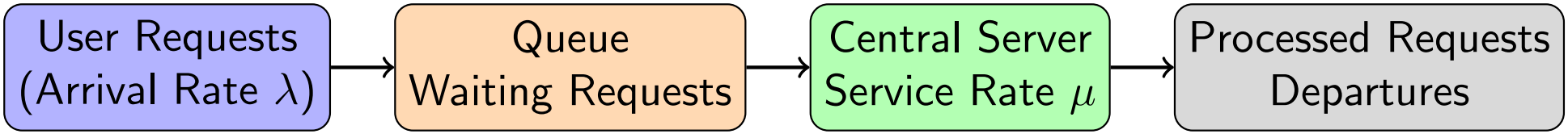
Observation

A centralized service can be modeled as a **simple queuing system**, where:

- Users arrive and wait for service.
- The server processes one request at a time (or limited in parallel).
- Bottlenecks appear when arrival rate exceeds service rate.

AASTU

Formal Analysis of Centralized Services (Part 6)



AASTU

Formal Analysis of Centralized Services (Part 8)

Insight

Key points from formal queuing analysis:

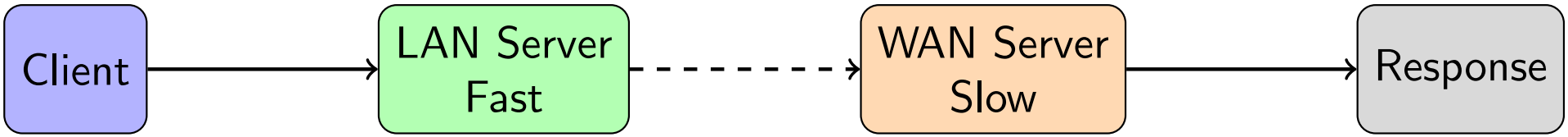
- Predict response time for users.
- Identify potential bottlenecks in the system.
- Determine system limits and scalability constraints.
- Provides quantitative measures for design decisions.

AASTU

Geographical Scalability Challenges

Key Issues

- LAN vs WAN latency: synchronous requests may fail
- WAN links are often unreliable $\rightarrow$ streaming transfers fail
- Multipoint communication is limited $\rightarrow$ need naming/directory services



AASTU

LAN vs WAN: Real Example

LAN

- Low latency (¡1ms)
- Synchronous requests succeed
- Streaming file transfer reliable

WAN

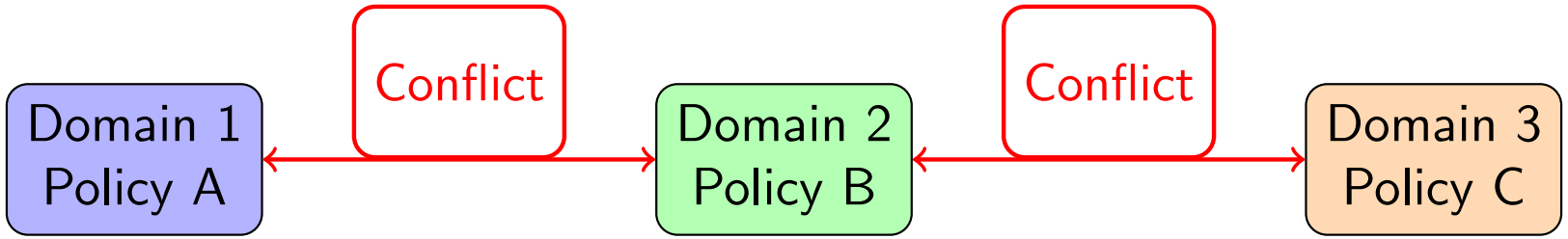
- Higher latency (50-200ms)
- Synchronous requests may fail
- Streaming often fails $\rightarrow$ need replication/caching

AASTU

Administrative Scalability Challenges

Essence

- Conflicting policies on usage, payment, management, and security
- Different administrative domains may not cooperate efficiently



AASTU

Administrative Scalability: Examples

Examples

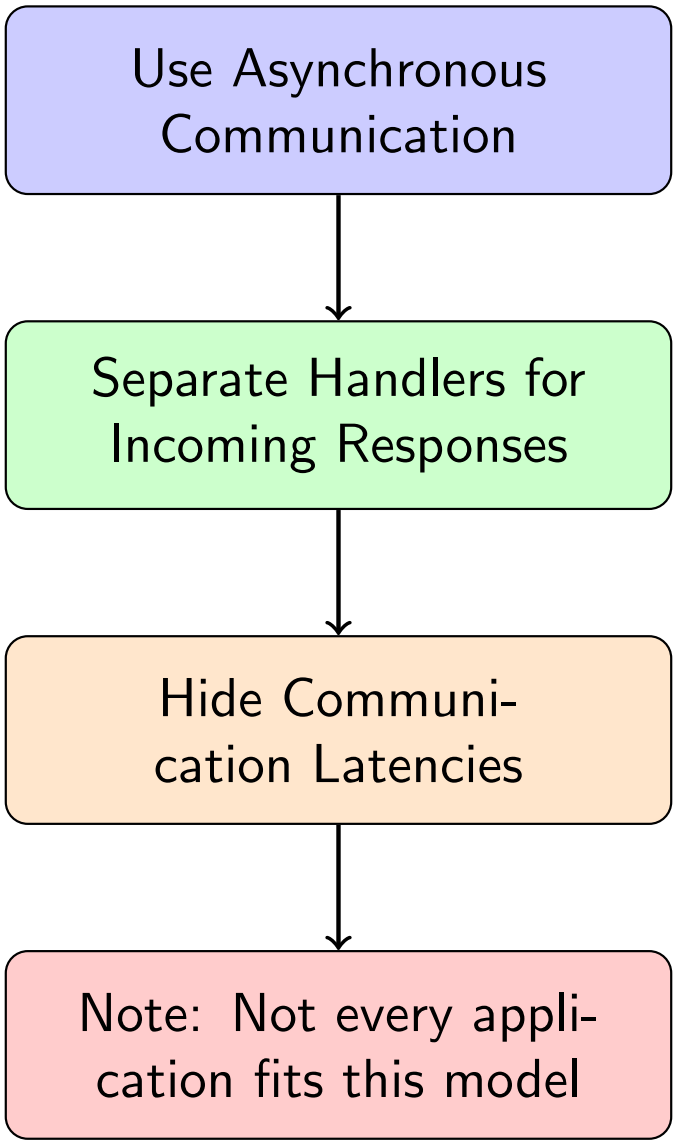
- Computational grids: share expensive resources between different domains
- Shared equipment: e.g., large-scale radio telescopes

Exceptions: Peer-to-Peer Systems

- File-sharing (BitTorrent), P2P telephony (early Skype)
- Peer-assisted audio streaming (Spotify)
- End users collaborate directly, not administrative entities

AASTU

Techniques for Scaling Distributed Systems



AASTU

Techniques for Scaling: Practical Examples

Example 1: Web Servers

- Asynchronous API calls reduce waiting time for requests
- Separate threads/processes handle client responses efficiently

Example 2: Messaging Systems

- Kafka or RabbitMQ use async communication to hide latencies
- Handlers process messages independently, improving throughput

AASTU

Scaling: The Problem with Replication

Replication: Simple but Tricky

- Applying replication is easy in theory.
- Multiple copies (cached or replicated) may become inconsistent.
- Modifying one copy can make it different from the others.
- Maintaining consistency globally requires synchronization on every update.
- Global synchronization limits scalability.

AASTU

Observation: Tolerating Inconsistencies

Insight:

- Some applications can tolerate temporary inconsistencies.
- Reducing global synchronization allows better scalability.
- Applicability depends on the type of application and its tolerance for inconsistency.

Key idea: Scalability can be improved if slight inconsistencies are acceptable.

Conclusion: Introduction to Distributed Systems

- **Distributed systems** unify multiple computers as a single system.
- **Core goals:** Scalability, reliability, transparency.
- **Techniques:** Asynchronous communication, separate handlers.
- **Examples:** Cloud storage, messaging platforms, peer-to-peer networks.
- Enables **efficient resource use** and **system resilience**.
- Fundamentals prepare for **designing scalable, dependable systems**.

AASTU

Questions?

Thank you for your attention!

AASTU

References

Suggested References

1. M. van Steen and A. S. Tanenbaum, *Distributed Systems*, 4th ed., Version 4.02, February 2024.
2. G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed., Addison-Wesley, 2012.
3. N. Lynch, *Distributed Algorithms*, 2nd ed., Morgan Kaufmann, 1996.
4. R. E. Bryant and D. O'Hallaron, *Computer Systems: A Programmer's Perspective*, 4th ed., Pearson, 2024.

AASTU