BSc (Hons) Computer Science

University of Portsmouth

Third Year

Distributed Systems and Security

M30225

Semester 2

Hugh Baldwin

hugh.baldwin@myport.ac.uk

Contents

Lecture - Introduction to DS&S

10:00 27/01/25 Amanda Peart

Module Info

Spit into two parts, Distributed Systems covered in the first 8 weeks and lectured by Amanda, Security covered in the last 4 weeks, lectured by Fahad.

- Distributed Systems covered by a 60% exam, Weds 26th March
- Security covered by a 40% exam

Lecture - Distributed Systems

10:00 27/01/25 Amanda Peart

What is a Distributed System?

A distributed system is the interactions between two or more computers, which are connected in some way to form a single computing system. The two main issues with distributed systems are how to connect the devices, and how to make them interact.

Definition

A distributed system is one which is made up of a set of independent computers which appear to the user as a single system. They are typically connected using a coordination system, which may use any of many networking and communications standards.

Networking Requirements

Most systems typically require fast and reliable networking to coordinate between systems. The often also require access to large amounts of data, which may needed to be accessed by any or all of the systems at the same time, so fast and reliable storage is also necessary.

Parts of a Distributed System

There are several levels to a distributed system, from what runs on each machine to the system as a whole. The low-level systems work on a per-machine level, and include processes, threads, concurrency, etc. Middleware acts to synchronise and coordinate the different parts of the system and ensure efficiency. The application itself sits on top of the Middleware and manages high-availability and fault tolerance.

Design Issues

Naming

It’s important that every part of a distributed system has a useful and descriptive name which have a global meaning. This may be supported by a name interpretation or translation system to allow programs to access named resources.

Access

It’s important to limit who and what can access the system, both in terms of security and accessibility. They should support as many systems as possible

Communication

The performance and reliability of the communication system is very important to both the availability and overall performance of the system as a whole.

Software

Data abstraction is very important as it allows many different systems to interact with the same data on a high-level without needing to convert data constantly between different formats. Part of this is well designed and documented APIs that allow access to the information and systems.

Resource Management

Optimisation of resources is very important to make sure that there is capacity and availability where it's needed. This includes load balancing and load shifting.

Consistency

The data must be consistent across the system, and must appear the same for all users.

Lecture - Distributed Shared Memory

09:00 17/02/25 Amanda Peart

Distributed Shared Memory (DSM) simplifies communication between nodes in distributed systems by providing a single memory space that all nodes have equal access to. It simplifies programming, as there is no need to perform explicit message passing between nodes or processes. It theoretically also reduces the overhead inherent in resource sharing, as it does not require the data to be duplicated on every node.

DSM is an abstraction which allows multiple computers which do not share any physical memory to access the same data as if it were a single unified address space. It is usually entirely transparent to the system running on it, as there is typically no need for the program to directly interact with the system implementing the abstraction.

Key Features

- **Transparency**– The user and the system do not need to know that the memory is distributed
- **Scalability**– Nodes should be able to be added and removed without affecting performance or reliability
- **Consistency Control**– It is essential to ensure that the memory is exactly the same on all nodes at the same time, to ensure that nodes are not using different or stale values
- **Synchronisation**– Using locks, semaphores or other synchronisation techniques to prevent conflicts between nodes
- **Performance**– Store data locally to ensure quick access

Centralised vs Distributed DSM

- Centralised
  - Shared memory managed by a single server
  - Data is accessed only through the single server
  - Data is stored in a single place
  - This makes it easier to manage and ensure consistency, but has a single point of failure and a large bottleneck on the performance of the single server
- Distributed
  - Memory is stored across multiple nodes but appears as one
  - Each node manages part of the memory space
  - This reduces the reliance on a single machine both in terms of reliability, and performance, but is significantly harder to manage and ensure consistency between nodes

DSM Models

- Page-Based
  - Transfers entire memory pages

  - Can be efficient for large datasets
  - Is susceptible to page faults causing large overhead
- Object-Based
  - Transfers single object instances
  - Reduced data transfers for each object
  - Harder to enforce consistency
- Variable-Based
  - Transfers single variable values
  - Ideal for sharing small amounts of data infrequently
  - Very bad in terms of scalability
- Library-Based
  - Run-time RPC/IPC
  - Very scalable if implemented correctly
  - Code may fail at runtime, causing unrecoverable errors

Consistency Models

Insert table here