
University of Portsmouth
BSc (Hons) Computer Science
Third Year

Distributed Systems And Security (DSSEC)

M30225

January 2026 - June 2026

20 Credits

Thomas Boxall
thomas.boxall1@myport.ac.uk

Contents

I Distributed Systems	2
1 Lecture - Module Intro & Overview of DS (2026-02-03)	3

Part I

Distributed Systems

Page 1

Lecture - Module Intro & Overview of DS

📅 2026-02-03

⌚ 09:00

👤 Amanda

1.1 Introduction to this Module

This is a module of two parts:

- Distributed Systems taught by Dr. Amanda Peart, first 8 weeks with 1 lecture (1 hour) and 1 seminar (2 hours) per week. Exam to cover this content worth 60% which will be held during March, before the Easter break.
- Security taught by Dr. Fahad Ahmad, which will have a different schedule. The exam to cover this content will be during the summer exam period and worth 40% of the overall module.

For Distributed Systems - there are two books recommended, one by Coulouris which is available in the UoP E-Library and on GitHub; and one by Tanenbaum. Both are relevant in places to supplement lecture material, regardless of their publish date.



There's some more information on the content to be covered in this Module available on the Moodle page.

1.2 What is a Distributed System?

A *Distributed System* is the interaction between two or more computing based devices which are connected in some fashion. Historically, this would have been seen as two Laptops, or PCs, then more recently as mobile phones, and even more recently as Meta Glasses (etc). This is an ever evolving field, however the two primary issues remain the same:

- How do the devices communicate? For example through which networking protocols and methods
- How do the devices interact? For example through distributed software

Definitions

Distributed System “A distributed system is a collection of independent computers that appear to the users of the system as a single computer” - Tanenbaum.

Distributed System “A system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing” - Coulouris

1.3 Why use Distributed Systems?

In short, we want to do things faster. Expectations for technology are increasing - the users demand the service to be instant, the webpage to load immediately, and to never see the loading wheel of

death.

There are three methods which can generally be employed to do things ‘faster’: working harder; working smarter; and getting help.

1.3.1 Working Harder

Working Harder is the idea of putting more effort into the task, in the hopes of completing it faster or better. For example, the night before exams cramming as much as possible. It can also apply to the thought of finding shortcuts to achieve given tasks. For example Michaelangelo carrying his lunch up the scaffold with him so he didn’t have to stop painting, or a desperate student taking their lunch to the library to avoid that 2 hour side quest to Greggs.

In computing - we can see working harder through having higher performing hardware.

However working harder alone often isn’t enough.

1.3.2 Working Smarter

Working Smarter is the idea of finding a way to reduce the amount of work needed to accomplish something. Taking a task and making it more efficient or smarter has a positive side effect, in that you get more out of it with less effort. For example, McDonalds have a streamlined kitchen, or a racing line used by Formula One drivers to minimise the distance to cover.

In computing - we can see working smarter through more efficient algorithms.

Although, even if we manage to work harder and more efficiently, that still may not be enough.

1.3.3 Get Help

Getting Help is the idea of more computers, computational devices, etc, being involved in the process to increase capacity. For example, a group of roadworkers digging a hole in the road is more efficient than one person on their own.

In computing - we can see getting help through distributed or parallel processing.

However, there are negative cases of getting help too. For example, some committees spend longer discussing and thinking than doing, where an individual or smaller committee may make the decision in a quicker and less painful method. In saying this, getting help is good, but you must be aware of the risks associated.

1.4 Where are Distributed Systems Used?

Distributed systems can be seen in lots of places. They’re commonly embedded in technology:

- Network File System (NFS)
- SSH / telnet
- Web - retrieving a page or used in CGI Scripts
- Network Printers
- ATM (Cash Machine)

Alternatively in more specialist technology:

- Distributed Shared Memory (DSM)
- Distributed Objects (COBRA or Java RMI)

- Process Migration (Condor)
- Replication (NIS)
- Network Video & Audio

Most of the above systems / applications communicate in some way with other systems, applications or entities. This means they communicate through a network. There are some requirements for the network. (Flashbacks to Networks L4 module).

1.4.1 Networks Requirements

In a traditional layered protocol stack, we may use the OSI or TCP/IP protocol stacks. These have defined layers and defined functions:

- Data link layer - handles the bits on the wire
- Packet switching
- Addressing and routing
- Reliable data streams
- Remote Procedure Calls (RPC) & other high level protocols (SNMP, HTTP, etc)

However in the land of distributed systems, the layered architecture isn't so present; where it's more common for protocols, APIs and services to be mixed and matched as required. Some layers do exist, however:

- RPC, Time & directory & security, File Server
- Storage, Directory, Replicated FS.

1.5 Structure of Distributed Systems

A distributed system can be considered to have three levels: Fundamentals, Distributed OS, and Application.

The *Fundamental Elements* level is the lowest level. This is concerned with the individual processes, threads, and concurrency controls. It also handles the synchronisation between different devices, and the Remote Procedure Calls (RPC). The lowest level also deals with naming the different components for clarity and ease of access, caching and finally network protocols. Many of the topics listed here were covered during the Operating Systems and Internetworking module at Level 5.

The *Distributed Operating Systems* level sits in the middle. This is concerned with scheduling, transactions, replication, fault tolerance, persistence and high availability. Many of these elements are to do with ensuring the multitude of devices in the distributed system are all using and referencing the same same data.

The *Application* level is the highest level. It handles any distributed shared memory, distributed objects, manages security and protected environments.

1.6 Issues & Considerations in Distributed Systems

Within a distributed system, there can be a need for *costly communications*. Most commonly this is seen in a setting where Symmetric Multiprocessing (SMP) is not available, leading to multiple discrete but connected nodes needed. The bandwidth and latencies of SMP is unparalleled to that of multiple nodes - causing performance degradation.

Unreliable Communications are a significant issue between distributed systems. If the connection

is unreliable or unavailable altogether, messages may be lost or corrupted which means a remote computer may not be usable.

Distributed Systems need to have resiliency built in; a system should be able to continue its operation after the *independent failure* of a node, preferably without the user noticing any performance issues unless the failure is significant.

The communication between nodes can also be problematic, where *insecure communications* could be exposed leading to unauthorised access, which potentially may be malicious.

1.7 Design Issues

When designing a distributed system, or even a system which may need to be distributed in the future, there are a number of issues which should be considered to ensure the effective functioning of the system.

1.7.1 Naming

Names assigned to resources or objects must have global meanings. This refers to any names or identifiers used in the code or deployments. The names should be independent of the locations of the objects. There should be a name interpretation system that can translate names in order to enable different programs to access named resources. Names used should scale and translate efficiently.

1.7.2 Access and Authorisation

Authentication should be a global service across the entire distributed system; users shouldn't need to authenticate for every different component of the same application. The same functions should be usable everywhere and should have reasonable performance.

1.7.3 Communications

The performance and reliability of the communication techniques used for the distributed system are critical to the performance of the system. This means that the data communicated within the system should always be up to date, so definitely not yesterday's data presented as though it's today's.

1.7.4 Software Structure

Open software is achieved through the design and construction of software components with well defined interfaces, such as APIs. Through an open system, more people can improve and access the system, however it does pose security issues through the number of people accessing the system.

Data abstraction within the software is important, as well as designing a framework that can work with existing and new services without duplicating the existing services.

1.7.5 Resource Management

While designing the distributed system, consideration must be given to the deployment. Optimisation of one component of the processing or communication resources within the network can simply move the bottleneck to another component in the system. Load balancing can be employed to move the workload between different systems.

1.7.6 Consistency Maintenance

Within a distributed system there may be multiple nodes performing the same function or needing to access the same data; herein lies issues with consistency. Consistency is concerned with:

- Data - where multiple nodes attempt to access a single data item, will they both get the same item?
- Replication of data - fault tolerance in this
- Cache - performance improvements, FT, availability
- Failure - is there a master / slave architecture setup
- Consistent time
- User Interface - ensuring ease of use and smoothness to users

1.7.7 Security

Authentication ensures users have the right access to resources. Access Control provides services to give different users different levels of access. Auditing keeps logs of all actions taken.

FREELY GIVEN, HIGHLY QUESTIONABLE

These notes are a byproduct of my learning process - mistakes, bad jokes and snark included. If you take an error of mine as gospel and ruin your perfect 1st, that is a *you* problem boo. I am a student, not a prophet, use a textbook if you want certainty.

Yes, they have been typeset in L^AT_EX; I'd rather typeset in stone with a blunt chisel than use Word Online for another minute. If the formatting looks obsessive, it's because it was either this or a breakdown. You can find the source on my GitHub.

They're licensed under CC-BY-4.0. So do what you will with them; just attribute them to me. Plagiarism is a bad look, and I've got just enough snark left for you bestie.

Thomas Boxall is a Computer Science Student studying at the University of Portsmouth, UK. In all honesty, he should probably be revising or doing some work towards his dissertation, not fondling the template of his notes again. You can find Thomas online at thomasboxall.net