Middleware
CS 3311

Dr. Dan Grigoras

UNIVERSITY COLLEGE, CORK
Coláiste na hOllscoile Corcaigh

The general context

- Internet is the network of networks with dynamic edge configuration – many devices connect/disconnect frequently.
- More and more connected devices are mobile.
- Internet of Things extends to all communicating tiny devices.
- Most of the computing applications are distributed – e.g., browser/web server, social media, multi-player games.

Distributed systems

- A distributed application is composed of several components that run as processes on different computers – communicating processes.
  - Note: we can also have a distributed application running on one host (e.g., mobiles – smartphones, tablets).

- Processes belonging to the same application communicate, invoke each other's execution, share resources and, sometimes, synchronize their execution – cooperating processes.

- Networked computers run distributed applications.

Computing models

- Historically, for the first distributed applications, the code of the application was deployed on fixed-address networked computers – all communication was programmed a priori.

- Client/server: clients send requests to servers (well-known network and port addresses) that eventually return results.

- Peer to peer (P2P): each peer can be client and/or server.

Main features and requirements of distributed systems

- Heterogeneous: hardware, operating systems, protocols ➔
  1st Requirement: hide heterogeneity!

- Network as the communication infrastructure has an impact on the performances and robustness of the application ➔
  2nd Requirement: make the communication environment reliable and predictable!

- Need to link new software to legacy software in a distributed application (especially in industrial setting) ➔
  3rd Requirement: provide inter-operability!

Challenges

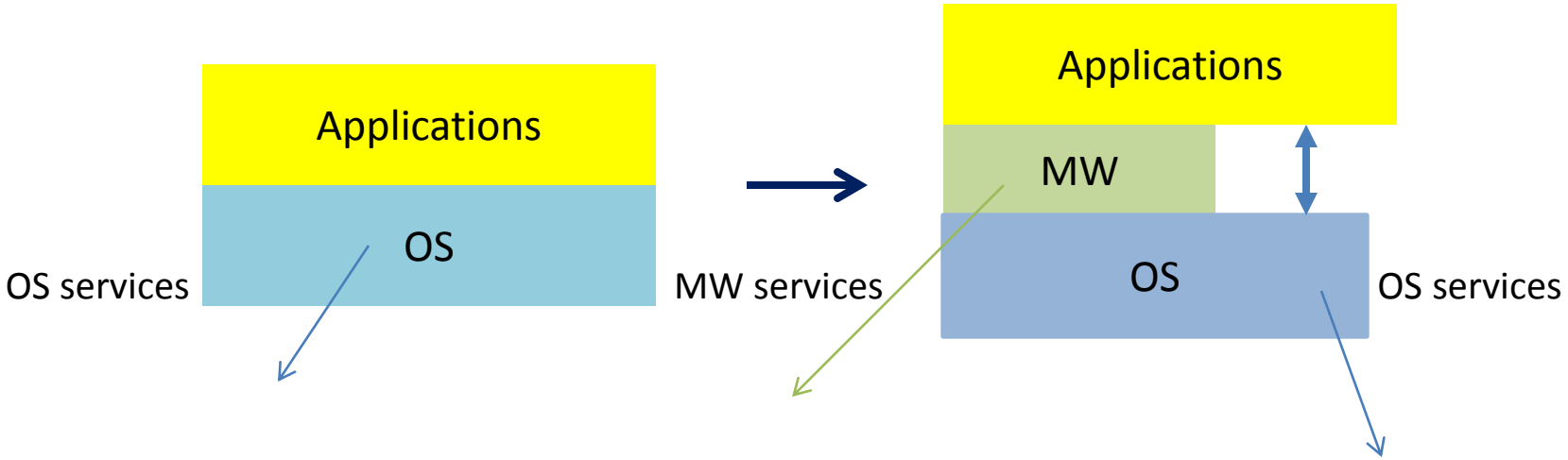
- Hide heterogeneity and complexity
- Manage dynamic environments efficiently: new services need to be learned of, some services may move to different locations, etc
- Predictable performance
- Manage mobility and ad-hoc networks transparently.
- Question: how to overcome these challenges?

The answer is middleware

- Middleware: a common set of services that can be used by any distributed application in order to provide:
  - Transparency
  - Inter-operability
  - Access to services
  - Mobility management
  - Scalability
  - Security
  - Reduced response time
  - Reliability

Location of middleware

- MW is placed between the applications and the operating system.
- It extends the set of services provided by the OS.
- Applications access middleware services by their specific APIs.



Examples of middleware services

- Name service
- Remote execution
- Discovery of services
- Event notification
- Messaging
- Security and privacy

- Join/leave networks and mobility management
- Caching and content adaptation

An example

- *Service*: live traffic webcam; its video stream is fed to the traffic control centre.
- This service can be public. Its different instances (webcams in different locations) need to be discovered by other potential users.

- The *directory service* binds names of services to *{<attribute, value>}* pairs.
- Attributes can be used to lookup service names.
- For a traffic webcam, the attributes are: geographic location, IP address, type (colour/black & white), night vision, resolution etc.

2nd Example

- Mobile users are browsing the Internet while on the move.
- The mobile device running the browser disconnects/(re)connects to Access Points (Wi-Fi) or Base Stations (Cellular networks), getting a new IP address each time.
- The mobile user is not aware of that as the middleware hides all networking operations (IP allocation, caching, DNS, generally, providing the quality of service - QoS).

MW implementations

- Remote Method Execution (RMI)
- Distributed object systems: DCOM, CORBA
- .NET
- Enterprise middleware: TPM, MOM (JMS)
- Cluster, grid and cloud management systems: Globus, Amazon EC2, Microsoft Azure

Sylabus

- Individual services
  - Name service
  - Discovery service
  - Event notification
  - Remote execution
  - Mobility management
  - Content adaptation
- Use cases
  - Home gateway initiative
  - IoT
- Middleware systems
  - Jini, RMI, CORBA, JMS

Course goals and methodology

- *Goals:*

  – To understand the role played by middleware and how it is used by distributed applications.

  – To learn different middleware services such as the name service (DNS), remote method invocation (RMI), event and notification service, activation.

  – To learn the principles of middleware systems such as message-oriented or object-oriented.

- *Methodology:*

  – Attend the lectures.

  – Fulfil tasks of practical work.

  – Use recommended references to learn more about certain topics.

Practical work – 30 marks

- Students are encouraged to make decisions regarding the course topics s/he is more interested to learn during the semester in the form of coursework. Each of you can choose one of the paths presented below. You can either

1. Do all the labs (G21, Tuesday 4-5, or 5-6),

or

2. Choose a middleware service of interest and study it: from literature review up to programming and testing .

Course philosophy: *collaborative learning process*
http://cs4.ucc.ie, key:CS3311_16

Grading:

Continuous assessment: 30%

Final Exam: 70%

Lecture:

50 min + 5 min review + 5 min questions

Contact:

Office: G69, WGB

Email:

grigoras@cs.ucc.ie, Subject: CS3311

Office hours by request