



# University of Dublin Trinity College



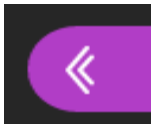
## CS7CS3: Software Architecture - Technical

Prof. Siobhán Clarke

Ext. 2224 – L2.15

[www.scss.tcd.ie/Siobhan.Clarke/](http://www.scss.tcd.ie/Siobhan.Clarke/)

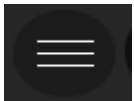
### The Session Will Begin Shortly



1. Click on the pink Collaborate button (bottom right) to open the chat window and enter your message.



2. You can close the Collaborate panel at any time so you can see more of the current presentation.



3. Click on the menu icon at the top left when you want to exit the online lecture



→ A recording will be made available afterwards in case you get disconnected or have technical issues.

# Student Online Teaching Advice Notice

**The materials and content presented within this session are intended solely for use in a context of teaching and learning at Trinity.**

**Any session recorded for subsequent review is made available solely for the purpose of enhancing student learning.**

**Students should not edit or modify the recording in any way, nor disseminate it for use outside of a context of teaching and learning at Trinity.**

**Please be mindful of your physical environment and conscious of what may be captured by the device camera and microphone during videoconferencing calls.**

**Recorded materials will be handled in compliance with Trinity's statutory duties under the Universities Act, 1997 and in accordance with the University's [policies and procedures](#).**

**Further information on data protection and best practice when using videoconferencing software is available at [https://www.tcd.ie/info\\_compliance/data-protection/](https://www.tcd.ie/info_compliance/data-protection/).**

© Trinity College Dublin 2020



**Trinity College Dublin**  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin

# Software Architecture

---

## What is it?

- Defines the basic components and important concepts of a system
- Describes the relationships between components/concepts

## Different Views:

- Functional Architecture
  - *view of software components*
- Technical Architecture
  - *view of where software components reside*
  - *strong focus on **quality attributes** (e.g., latency, security,*

# Technical Architecture

---

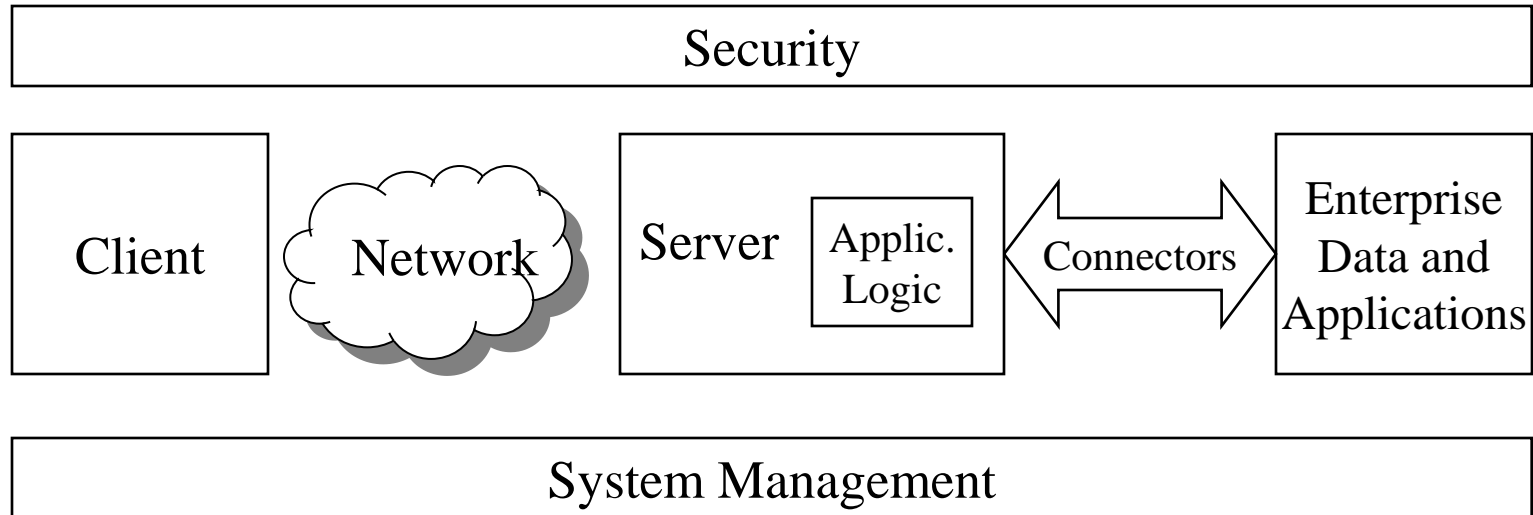
Possibilities for physical placement of software:

- Single machine
- Distributed
  - *Client-server*
  - *Peer-to-peer*
  - *Internet of Things*
  - *Edge network*
  - *Ad hoc*
  - ....

Trade-offs for non-functional quality requirements

# Architecture Building Blocks

---



Source: “An Approach to Designing e-Business Solutions”, SG24-5949-00  
available from <http://www.redbooks.ibm.com>

# Security – Decision Points

---

## Encryption

- Do transactions need to be encrypted?
- Level of encryption? (e.g., 40-bit encryption in US)

## User Identification

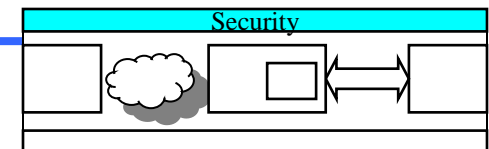
- uid/pw, cookies, certificates, application-level?
- Existing customer database that should be used to identify online visitors?

## Access to data

- Do you need to restrict access to parts of the site?
- What **privacy rules** should be applied to information provided by users

What are the legal requirements and company policies for auditing content, changes and transactions? GDPR?

Does the company already have a secure demilitarised zone into which the Web server could be placed?



# System Management – Decision Points

---

Do you have the infrastructure to install and run you own server?

What are the **response time** targets?

## ***Availability:***

- What hours should the service be available?
- Is it acceptable to have any scheduled downtime for maintenance?
- How important is it that the service be never interrupted, even for unscheduled component failures?
- If interruptions do occur, what should be the target time for resuming service?

How should partial or total service failures be monitored and handled?

Do you need a recovery plan, or will it be covered by existing processes?

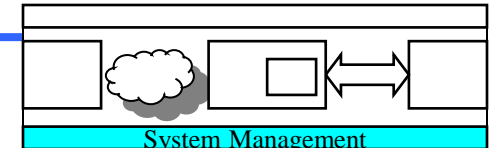
## ***Tracking/Documenting:***

How should the architecture support the process of problem reporting, tracking and fixing?

What statistics do you need to keep about the site, and how will they be analysed?

What instrumentation should be included in the design to measure performance, response times and availability?

Should the architecture include a repository for statistical data?



# Client – Decision Points

---

## ***About the user:***

Who is the customer? (Internet or Intranet) – affects browser choice

What is the level of the user's skill?

What languages should the site support?

What are the user's usage patterns? (search or browse)

Mobile device support?

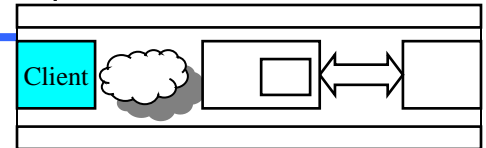
## ***About the application:***

How will the application maintain state?

Is there a need to distribute application code, and if so, how will it be done?

How will the choice of client affect end-to-end response? (HTML, JavaScript, AJAX, JQuery, VBScript?)

Is the browser the only user interface? (e.g., mail?)





# Network – Decision Points

---

Will my solution involve the internet?

What protocols will I use?

*HTTP? HTTPS? FTP? RMI? Messaging? etc*

What about data, object and application placement?

*projected transaction volumes, amount of data, interaction?*

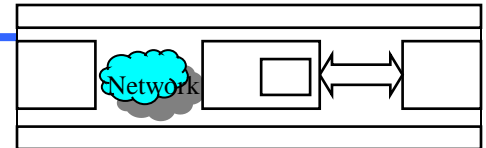
What security functions are required/provided by chosen protocol?

*level of encryption will affect this, and also performance!*

Will existing network function as required?

*expansion needed?*

How does the network affect end-to-end response time?



# Server – Decision Points

---

## ***Functional considerations:***

Are mail or conferencing facilities needed?

Are there workflow requirements?

Are indexing, searching or other site navigation aids required?

## ***Technical considerations:***

Single server or multiple servers? Peer-to-Peer? Edge Network? IoT?

Geographic location for servers?

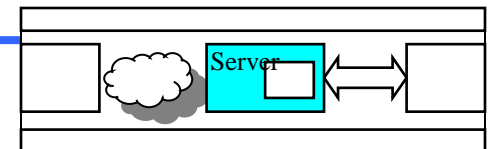
End-user client to server, or server to server required also?

Do client options affect the design of the server?

What security functions are required on the server?

How can impact of server on end-to-end response time be estimated?

*some benchmark numbers may be available but not straightforward!*



# Application Logic – Decision Points

---

Is this a logical two-, three- or n-tier solution?

Will object technology development, traditional programming development or integrated packages be used?

Application control (for end-user print/save) required?

Is normal HTML adequate, or need an enhanced UI?

Is client-side scripting needed? At what level?

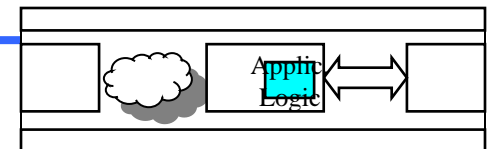
*check all possible browsers?*

Will site use proprietary scripts/tags/plugin-ins?

Will site use client-side executables? What are their connectivity requirements?

How will application be split between client-side and server-side logic? *affects communications for validation etc/performance?*

Additional access security required?



# Connectors – Decision Points

---

What enterprise systems, applications and data does e-business application need to access?

How should data be transferred between different systems?

How current does the information have to be? *Use caches?*

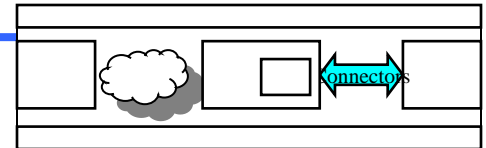
Is synchronous or asynchronous access required? *Off-line OK?*

Is access to different operating systems, network protocols, application environments required? *which connector? CICS? MQSeries? RPC?*

Is a new user interface required? If so, what kind?

Are additional security policies required?

Can scalability and performance requirements be predicted?



# Enterprise Data and Applications – Decision Points

---

Do service hours of enterprise data repository match e-business app targets?

How will access authorisation rules for database map to e-business user identification?

Is e-business application using the corporate data for reference? (i.e., read only)

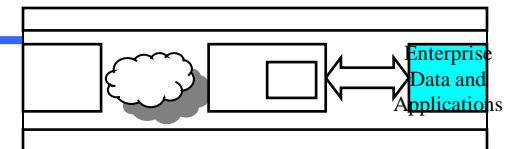
Is the data in a format easily accessed by distributed systems?

If additional code is needed to gain access to data, how will this be developed?

If access is to relational databases, can the SQL be structured to minimise network traffic?

What are the commit and rollback requirements of the application?

Is there a need for caching?

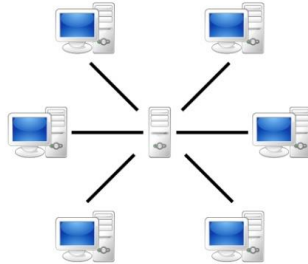


---

Ultimately, your technical  
architecture choices will involve a  
trade-off between all the  
requirements

# Example – Architectural Tradeoffs between P2P and Client-Service Architectures

---

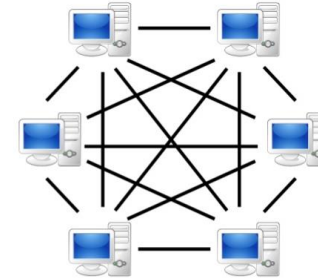


Server-based

- Clients request services from a server
- Very well-known model

*However:*

- Scalability possible issue
- Single point of failure
- Management required
- Potential for un-used resources



P2P-network

- Every node participates as both client and server
- Very scalable (*need have only part of whole system on each node*)

- No central point of failure

*However:*

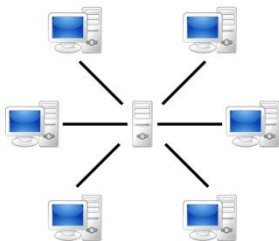
- Decentralised coordination (state consistency, etc)
- All nodes not equal (performance issues)

# Infrastructure Availability Tradeoff: P2P and Client-Service Architectures

---



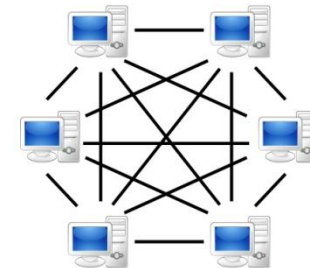
Application



Server-based



- No existing infrastructure
  - e.g., disaster area
- Application is ad hoc



P2P-network



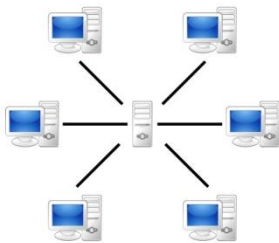
# Scalability Tradeoff: P2P and Client-Service Architectures



Application

*Too hard to manage DHT*

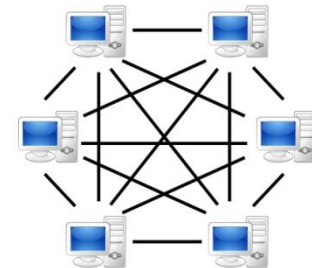
*Dynamic increases in load,  
processing, capacity?*



Server-based



- Dynamic increases in system load?
- More processing, storage capacity required?
- Too hard to manage a distributed hash table (DHT)?



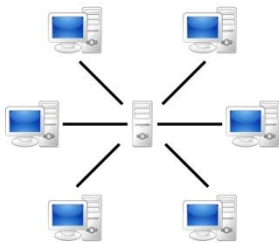
P2P-network

# Robustness/Reliability Tradeoff: P2P and Client-Service Architectures

---



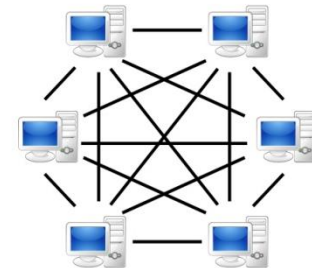
Application



Server-based



- Expect high availability in server-based architectures
- P2P generally not expected to be robust, though could be improved (though at the expense of scalability?)



P2P-network

# Performance Tradeoff: P2P and Client-Service Architectures

---



Application

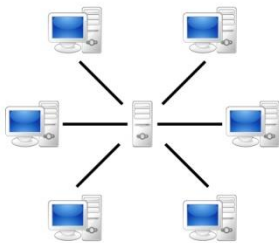
*DB queries?  
No cooperating peers*

?



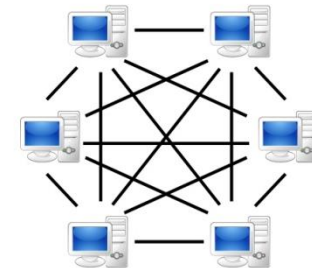
*Regular (similar) computers,  
with parallel tasks*

?



Server-based

- Performance of P2P system depends on many factors like:
  - task type: can it be divided into independent, parallel subtasks?
- Centralised DB vs DHT?
- lack of peer cooperation



P2P-network

# Energy Consumption Tradeoff: P2P and Client-Service Architectures

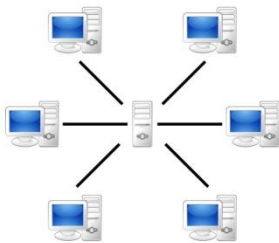
---



Application

*Geographic spread  
High energy task*

?

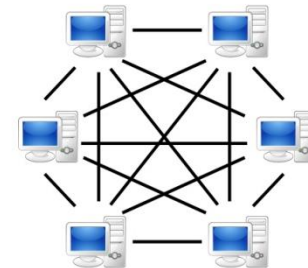


Server-based



*Energy-aware protocols ?  
Nodes already in use?*

?



P2P-network

- Performance of P2P system depends on many factors like task type: whether the nodes are up regardless of task; energy consumption; geographic spread.

# Cost Tradeoff: P2P and Client-Service Architectures

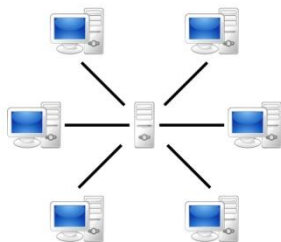
---



Application

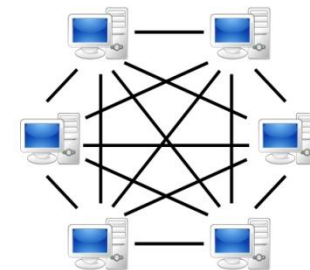


*Cost is a big issue*



Server-based

- Is capital or operative cost an issue?
- is the cost of a highly available cloud solution prohibitive?

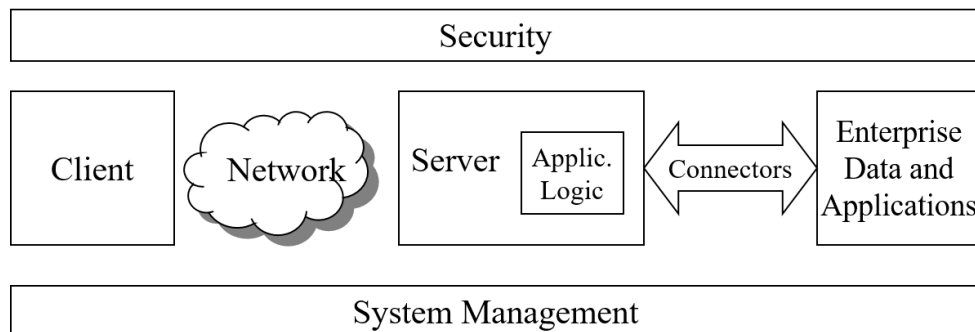


P2P-network

# Again, and in conclusion

---

Ultimately, your technical architecture choices will involve a trade-off between all the requirements



You will need to clearly indicate the technical choices made for each architectural element. This includes technologies you plan to use plus how all non-functional requirements are met.