**Physical Models:**

The baseline physical model for a distributed system, is one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages. Examples of distributed system physical models are below:

* Early distributed systems – A distributed system typically consisting of between 10 and 100 nodes interconnected by local area network, with limited connectivity and supported a small range of services across the Internet.
* Internet-scale distributed systems – A distributed system in which the underlying physical infrastructure consists of an extensible set of nodes interconnected by a network of networks, exploiting the infrastructure offered by the Internet to become truly global.
* Contemporary distributed systems – A distributed system deploying an increasingly varied set of network technologies and offer a wide variety of applications and services, and potentially involve up to hundreds of thousands of nodes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Distributed System** | **Early** | **Internet-Scale** | **Contemporary** |
| **Scale** | Small | Large | Ultra-large |
| **Heterogeneity** | Limited | Significant priority with range of services introduced | Added dimensions introduced including radically different styles of architecture |
| **Openness** | Not a priority | Significant priority with range of services introduced | Major research challenge with existing standards not yet able to embrace complex systems |
| **Quality of Service** | In its infancy | Significant priority with range of services introduced | Major research challenge with existing standards not yet able to embrace complex systems |

I will develop my distributed system on an internet-scale physical model, this is because ‘MSc Properties’ main requirements are to share data accurately in a timely manner across a large geographical scale, and internet-scale distributed systems offer this. LOOK INTO MORE ABOUT WHAT I CAN SAY!!

**Architectural Models:**

Architectural elements

* Communicating entities – What is communicating in a distributed system?
  + Objects – A number of interacting objects within a system representing natural units of decomposition for the given problem domain.
  + Components – Similar to objects, but also provide the assumptions made in terms of other components/interfaces that must be present for a component to fulfil its function.
  + Web Services – Again similar to both objects and components but are intrinsically integrated into the World Wide Web, using web standards to represent and discover services, meaning they are defined by the web based technologies they adopt.
* Communication paradigms – How entities communicate in a distributed system?
  + Interprocess communication – low-level support for communication between processes in distributed systems, including:
    - Message passing primitives
    - Direct access to the API offered by Internet protocols (socket programming)
    - Support for multicast communication.
  + Remote Invocation – Most common communication paradigm covering a range of techniques based on two-way exchange between communicating entities in a system, resulting in remote operations/procedures/methods being invoked. There are a few remote invocation types such as:
    - Request-reply protocols – Message passing service to communicate involves a pairwise exchange of messages from client to server and then from server to client.
    - Remote procedure calls – Procedures on remote computers can be invoked as if they are procedures in the local address space.
    - Remote method invocation – Similar to remote procedure call but using objects, in which an object can invoke a method in a remote object.
  + Indirect Communication
    - Group Communication – Is concerned with delivery of messages to a set of recipients, supporting one-to-many communication.
    - Publish-subscribe systems (Distributed event-based systems) – Is concerned with providing an intermediary service that efficiently ensures information generated by procedures is routed to consumers who desire this information, supporting one-to-many communication.
    - Message Queues – Is concerned with a producer process sending messages to a specified queue and consumer processes receiving messages from that queue or be notified of the arrival of new messages, supporting point-to-point communication.
    - Tuple Spaces –
    - Distributed Shared Memory
* Roles and responsibilities – What roles and responsibilities do they have in the overall architecture? – Architectural styles stemming from the role of individual processes:
  + Client-Server – Processes take on the roles of being clients or servers, in particular, client processes interact with individual server processes in order to access the shared resources that they manage.
  + Peer-to-peer – All processes involved in a task play similar roles, interacting cooperatively as peers without any distinction between client and server processes or the computers on which they run.
* Placement – How are the mapped on to the physical distributed infrastructure? (What is their placement?)
  + Mapping of services to multiple servers – services may be separated as several server processes in separate host computers interacting as necessary to provide a service to client processes.
  + Caching – A cache is a store of recently used data objects that is closer to one client or a particular set of clients than the objects themselves, and when a new object is received from a server it is added to the local cache store, replacing some existing objects if necessary.
  + Mobile Code – Is where code is downloaded from a server and run locally to accomplish a task.
  + Mobile Agents – A mobile agent is a running program that travels from one computer to another in a network carrying out a task on someone’s behalf, such as collecting information, and eventually returning with the results.

Architectural patterns

* Layering – A complex system is partitioned into a number of vertical layers, with a given layer making use of services offered by the layer below.
* Tiered architecture – Organize functionality of a given layer and place this functionality into appropriate servers and on to physical nodes. Can be 2 tier (client (front end) and server (application logic & database)), 3 tier (client (front end), application server, and database server or n-tier.
* Thin clients – Complexity is moved from the front end-user device towards services in the internet (cloud or client-server), enabling clients to access sophisticated networks services, with few assumptions or demands on the client.
* Proxy pattern
* Brokerage pattern
* Reflection (used in RMI)

CORRECT BEHAVIOUR IN DISTRIBUTED PROGRAMS DEPENDS UPON CHECKS, ERROR-CORRECTION MECHANISIMS AND SECURITY MEASURES.