2 Distributed System Architectures (week 2)

To motivate classifications of distributed architectures we consider the following types of distributed systems:

- Computing systems, such as cluster or grid computing. Compute clusters consist of powerful machines connected by a high-speed network. Normally those clusters are in one location. A compute grid also consists of powerful machines but they are connected by a virtual network and must not reside in the same location. They can be heterogeneous and geographically distributed.
- Distributed information systems, can be transaction processing systems, such as distributed data bases or enterprise systems. Those systems are normally big, not huge and have properties known from databases (ACID).
- Distributed pervasive systems are small, wireless, connected in adhoc fashion, they have no administration, certainly no central administration. They are used for home automation, health systems, environmental sensor networks, and others.

Why do we need distributed systems? We have them for properties such as:

- performance (we use a distributed system, because we expect better performance than from a monolithic system)
- distribution inherent (distributed nodes want to collaborate, share information, share data, etc.)
- reliability (distributed system includes redundancy, data duplicates, etc.)
- incremental growth (scalability)
- sharing of resources (expensive resources may have better utilisation, pay per usage concept rather than ownership of resources)
- communication (we want to communicate/collaborate/share information across large geographical distances)

2.1 System models

2.1.1 Physical models

Physical models are small reproductions of the real system, such as toy examples of the system. They will not be considered here.

2.1.2 Architectural models/styles

Architectural models can be structured in different ways. We will discuss some types of classifications.

Our first classification divides into

- layers
- tiers

- brokerage of web services

A layered architecture is similar to a protocol stack in that messages pass between neighbouring layers. Normally messages containing requests will be passed from higher to lower layers, while messages containing responses will move from lower to higher layers. Different functionality can be implemented on the layers. The user will normally see the highest layer.

While layers are seen as vertical arrangement, tiers can be regarded as horizontal splitting. Commonly, two- or three-tiered systems are built. The tiers are mostly presentation logic, application logic and data logic. Or presentation server, application server and database server.

Brokerage of web services connects a loosely coupled client and server through a broker. Services register with the broker and clients request the suitable server from the broker and then directly issue their requests to the server.

Figures will be added later.

Another classification arranges in a different way:

- Layered architectures, as defined before
- Object-based architectures. Interaction between objects are key elements in the system. Examples are remote procedure calls, those systems can be client-server systems.
- Data-centered architecture. Data is the key element. Communication over data, shared memory, distributed databases. Web systems are mostly data-centric.
- Event-based architectures. Processes communicate through events. Publish/subscribe systems. Subscriber registers with broker, publisher announces events at broker. Those systems implement loose coupling between publisher and subscriber, since they need not know (of) each other. They are decoupled in space. Scalability is sometimes better than in client-server architecture, since parallel operations are possible, message cashing, etc.

In some systems combinations of the types of architectures exist, such as event-based and data-based can be combined using a shared data space.

We can also divide into

- Interaction model (synchronous, asynchronous, messages,..)
- Failure model (node failure, link failure, message loss, byzantine,..)
- Security model

2.1.3 Fundamental models

Those models will be discussed in much detail in a later section.

2.2 System architectures

We mainly distinguish centralised versus decentralised systems. Those are the most prominent two concepts. We prefer decentralised systems as they solve some problems in centralised systems, such as single points of failure. This comes at a cost and there always is a tradeoff to solve in terms of overhead versus improved properties, such as reliability, scalability, etc.

2.2.1 Centralised architectures

Centralised architectures can be seen as equivalent to client-server systems that work on a request-response basis.

Implementations are mostly using messages, i.e. connectionless protocols.

Transmission failures must be treated with resend (transmission must be **idempotent** - repeatable without harm.)

Mostly reliable transmission using TCP/IP.

Server can be client to another server, application layering (tiers!)

Implies a lot of waiting.

Does not scale well.

2.2.2 Decentralised architectures

Can use **vertical distribution** (multi-tiered systems), different logic on different machines. Please note that we use vertical here for a different meaning than before, where it indicated something like protocol layers. Here we talk about dedicated functionality on different machines.

We can also use **horizontal distribution** (replicated client or server operating on different data). This is often geographical distribution, but need not be.

Cmmonly in use is an **overlay network** that hides the physical structure by adding a logical structure.

We distinguish **structured P2P architectures** and **unstructured P2P architectures**

Structured P2P architectures

Most popular technique uses distributed hash tables (DHT)

Randomly creates 128 bit or 160-bit key k using algorithms such as SHA1 for data and nodes. Two or more identical numbers are very unlikely. Most prominent system is **Chord**.