DIS - Introduktion

Trends in Distributed Systems
Side 8

- Mobile and ubiquitous (=allestedsnærværende) computing
  - **Spontaneous Interoperation**
  - **Mobile computing**
  - **Ubiqutous Computing**
  - **Service Discovery**
- **Distributed Multimedia Systems**
  - Webcasting
  - Streams, Skype
- **Distributed computing as a utility (=hjælpeværktøj)**
  - Physical resources
  - Software services
- **Cloud computing**
  - Cluster computing
  - Grid computing

Challenges
Side 16

- **Heterogeneity**
  - Middleware
  - Heterogeneity and mobile code
- **Openness**
  - The openness of a computer system is the characteristic that determines whether the system can be extended and reimplemented in various ways. Dokumentation, API til offentlighed osv.
- **Security**
  - **Confidentiality**: Protection against disclosure to unauthorized individuals
  - **Integrity**: Protection against alteration or corruption
  - **Availability**: Protection against interference with the means to access the resources
  - Security of mobile code
- **Scalability**
  - Controlling the cost of physical resources
  - Controlling the performance loss
  - Preventing software resources running out
  - Avoiding performance bottlenecks
- **Failure Handling**
  - **Techniques for dealing with failures**
    - Detecting failures
    - Masking failures
    - Tolerating failures
    - Recovery from failures
    - Redundancy
  - Availability of a system
- **Concurrency**
- **Transparency**
  - Transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole rather than as a collection of independent components. The implications of transparency are a major influence on the design of the system software
  - **ANSA Definitions**
    - Access transparency enables local and remote resources to be accessed using identical operations
    - Location transparency enables resources to be accessed without knowledge of their physical or network location
    - Concurrency transparency enables several processes to operate concurrently using shared resources without interference between them
    - Replication transparency enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers
    - Failure transparency enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components
    - Mobility transparency allows the movement of resources and clients within a system without affecting the operation of users or programs
    - Performance transparency allows the system to be reconfigured to improve performance as loads vary
    - Scaling transparency allows the system and applications to expand in scale without change to the system structure or the application algorithms
- **Quality of Service (QoS)**
  - Timeliness guarantees

De 7 forhold

- **Samtidighed / Concurrency**
- **Kausalitet**
  A bliver til B. Problem her – samtidig bliver et problem – hvilken rækkefølge sker det her i. Hvis jeg nu har mere end to servere. Hvad sker der, hvis der er en der køber en billet fra den ene server?
- **Konsensus**
  Når vi nu ikke ved om billetten er købt eller solgt. Hvordan finder vi så ud af det? Vi tager en ekstra server på. Nu er der tre servere. Hvem siger det rigtige? Man kan ikke gennemfører en halv transaktion – man skal sige at den skal være solgt på alle sammen, før man går videre.
  Et udtryk for at vi fx kan sige et demokrati – demokrati er en konsensus algoritme – på en eller anden måde søger at blive enige – hvordan servere bliver enige – fx majoritetsalgmoritmer
- **Konflikt**
- **Konvergens**
  Over tid bliver tingene det samme. Sandhederne flyder sammen til én. Det næste var at man som klient ikke skal være afhængig af hvilken server man er på. Man laver en ny server ved navn NTP server som fortæller hvad klokken er. Når man klikker på musen skal man tænke at der er andre der kan gøre det hurtigere.
  Faktum at værdier ligger mange forskellige steder, og samler sig et sted
- **Sikkerhed**
- **Partitionering**
  Det faktum at dele af et distribueret system kan gå ned. Når nået går ned hvordan håndterer vi hvilken del af systemet der går ned? Noget andet tager over? Det er for at sikrer at vi ikke har et single point of failure.
- **Skalerbart**
  Hvordan sørger vi for at det her kommer til at virke med flere brugere og bliver ved med at virke ved flere brugere. Hvor mange servere. Hvordan skal det bygges op. Hvordan sørger vi for at der er én sandhed. Hvordan sørger vi for at tiden går rigtigt.

Definition

We define a distributed system as one in which hardware or software components **communicate** and coordinate their actions only by **passing messages**.
Når to eller flere computere agere som én.

- **CAP-Teormet**
  Consistency, Availability og Partitioning - og aldrig alle tre på én gang. Google drive – vi har fokuseret på tid. Dropbox har det at man ikke kan skrive samtidighed.

Problemer ved global tid/ur

Der findes ikke et globalt ur i et distribueret system grundet at enhederne i systemet bl.a. befinder sig geografisk forskellige steder, der skaber problemer med eksempelvis **forsinkelser/latency**. Hvis vi prøver at forestille os et salg af nogle billetter online, hvornår er købet så foregået? er det når musen klikkes? Er det når det bliver registreret på serveren? Er det når du selv kan se på din skærm, at det er registreret? Og hvad sker der, når der fx er én der køber billetten på umiddelbart samme tid? Dette er alle udfordringer, det giver os.

Der er altid en forsinkelse – vi kan ikke sige hvad klokken er, men vi kan godt måle hvor lang tid tingene tager.