Transparency in Distributed Systems

- Transparency is a key issue within Distributed system:
  - Advantages:
    - Easier for the user:
      - doesn't have to bother with system topography
      - doesn't have to know about changes
      - easier to understand
    - Easier for the programmer:
      - doesn't have to bother with system topography
      - doesn't have to know about changes
      - easier to understand
  - Disadvantages:
    - Optimization can not be done by programmer or user
    - Strange behavior when the underlying system fails.
    - Underlying system can be very complex.

Different types of Transparency (1)

- Network Transparency
  - The programmer/user does not have to know about the network.
    The system looks like a stand-alone computer.
- Name Transparency
  - Names will not change when the system is reconfigured.
- Location Transparency
  - The location of an object is not visible in its name.
- Semantic Consistency
  - The execution of a program looks the same and gives the same result even if it is run on different platforms.
    - Today this might not be the case if you run a Java program on Windows or Linux. E.g. different coding of characters (unicode, utf-8) might lead to different behavior.

Different types of Transparency (2)

- Access Transparency
  - The objects is accessed in the same way even if they are on different type of platforms.
- Execution Transparency
  - Migration of processes will not be visible to the user.
- Replication Transparency
  - Reading and updating of replicated data will appear to the user/programmer as if it is performed on one single local copy.
- Performance Transparency
  - Performance should not be affected by the actual configuration of the system.
  - This is probably the hardest to achieve, but for some real-time systems it might be crucial.
- Configuration Transparency
  - The configuration does not affect the programming or use of the system.