CS 499 – Distributed Systems  
Wednesday, August 26, 2009

Notes:

* go over the boring stuff: introductory stuff today
* Intro to Distributed Systems (DS):
  + What is a DS?
    - “A distributed system is a collection of independent which give the user the impression, that he is working with one computer”
  + Definition by Mullender
    - “A distributed system is various computers doing something together. Thus it is characterized by three things:
      * a number of independent computers, each consisting of CPU…i/o
      * Some of the i/o are used for communications between c computers 🡪 communication network. if the computers cannot communicate with each other, this distributed system is not very interesting
      * The computers share a common state. if state is thought of a range of global invariants, the maintenance of these invariants require the common/coordinated cooperation of the computers.”
  + Sun Def:
    - “the network is the computer”
  + Coulouris Def:
    - “We define a distributed system as one in which hardware or software components located at networked computers communicate and coordinate their acions only by passing messages.”
  + Class Def:
    - No common physical clock
    - No shared memory
      * // concurrency
    - geographic separation
    - Autonomy, heterogeneity
* Motivation for DS:
  + Problems are too big for one to handle
    - >> Load distribution, **Performance boost**
    - On one computer you can get stuck waiting for one process to finish. once that process finishes it allows other processes to continue normally.
    - Performance boost 🡪 coarsely grained programs
  + Bridge distances
    - resource access
    - // Hackers 🡪 “use resources that are not their”
  + Reliability
  + Scalability
    - “upgradability”
* // Inherently DS are not good!
* Parallel systems are close to DS, therefore we need to define the two to see the difference they share:
  + Multiprocessor Systems (UMA – Uniform Memory Access)
    - Direct access to shared memory
    - connected by bus (commonly used)
    - switches (two by two switches) 🡪 Omega network and Butterfly network
      * Multi-switch
  + Multi-computer parallel system (NUMA -
    - no direct access to main memory
      * but that is no fully limiting its access
    - each sections have their own memory
    - commonly don’t have clock
  + Array processors
    - tightly copulated
  + Relation between SW components
    - Image (look for)
* Classifications of Primitives of DS:
  + Synchronous (send/receive)
    - Handshake between sender and receiver!
  + Asynchronous (send)
    - control network after the data to the buffer
  + Blocking (send/receive)
    - control returns after processing of primitives
  + Non-blocking (return right away)
    - control returns to process immediately
    - check way handle
    - wait (handle)
    - // Blocking can be synchronous and asynchronous (only for sending) 🡪 research at home
* Challenges to DS:
  + Transparency “there but not seen”
    - Access Trans., location Trans. >> Network Trans.
      * don’t care where the resources are, you just care of getting it (i.e., mirrors)
    - Migration Trans.
      * move things around without the user knowing anything (no consequence to the user)
    - Replication Trans.
      * duplicating a file without seeing the process
    - Concurrency Trans.
      * keeping data synchronized without user input
    - Failure Trans.
      * when something fails and something takes over and the user is unaware of the issue
    - Performance Trans.
      * the load changes and the performance is still good
    - Mobility Trans. (close to Migration)
      * Close to access trans.
      * but deal with mobile devices; laptop, wireless devices
    - Scaling Trans.
      * if you add more to the system the system does not fail
  + // Visible (there and seen), Virtual (not there but seen)

TODO:

* THIS FRIDAY: Present our conceptual design to the project: Peer to peer
  + network with the assumption that one peer is always there (any peer)
  + peers joining, leaving, connecting to another
  + free to use any architecture
* FRIDAY AFTER: Documentation
* FRIDAY AFTER THAT: Checkpoint