Unit 2

Layered Protocols

- No shared memory in DS. Thus, communication is through message passing.
- OSI reference model
  - Communication divided into 7 layers.
  - App, Pres, Sess, Trans, Net, DL, Phy
  - Process creates message, passes it to app layer. App layer adds header, sends to Pres layer and so on..
  - At receiving end, message is passed upwards. Each layer strips off its corresponding header/tailer.

Types of Communication

- Persistent vs Transient
  - Persistent: Data is stored in middleware till it is transferred to the receiver.
  - Transient: Sender and receiver must be active at the time of transmission. Data is not stored in the middleware.
- Asynchronous vs Synchronous
  - Async: Sender continues immediately after submitting message for transmission.
  - Sync: Sender is blocked until
    - Middleware notifies it'll take over.
    - Request is transmitted.
    - Response is received.

Remote Procedure Call

- Idea is to allow a process to call a procedure stored on a different machine.
- When a process on machine A calls procedure on machine B…
- Info from caller to callee: parameters
- Info from callee to caller: return value
- Message passing not visible to programmer.
- Remote procedure call should look like a local one. Transparent.
- Steps:
  1. Process calls client stub.
  2. Client stub packs parameters into a message, calls local OS.
  3. Local OS sends message to remote OS.
  4. Remote OS gives message to server stub.
  5. Server stub unpacks message and calls remote procedure.
  6. Remote procedure sends return value to server stub.
  7. Server stub packs return value in a message, calls remote OS.
  8. Remote OS sends message to local OS.
  9. Local OS gives message to client stub.
  10. Client stub unpacks message, sends return value to local process.

Multicast Communication

- Sending data to multiple receivers.
- Major issue has been setting up communication paths.
- Advent of peer-to-peer systems has made that simpler.
- Can also be done in other ways (apart from setting up explicit comm. paths). E.g. gossip based.
- Application-level tree-based multicasting
  - Nodes are organised into an overlay network which is used to spread info.
  - 2 approaches in constructing overlay network.
    - Tree. Unique path between nodes.
    - Mesh. Multiple paths between nodes. More robust.
- Flooding-based multicasting

Network Virtualisation: Overlay Networks

- Growing range of apps co-exist in the Internet.
- Impractical to alter the Internet communication protocols for each app.
- This has led to interest in network virtualisation.
- It deals with the construction of many different app-specific networks over an existing network like the Internet without altering the characteristics of the underlying network.
- Just like computer networks, each virtual network has its own addressing schemes, protocols and routing algorithms, redefined to suit the needs of the app.

Election Algorithms (Bully)

- Consider N processes $\{P_0 \dots P_{N-1}\}$ and $id(P_k) = k$.
- When any process notices that the coordinator is not functioning, it initiates an election.

- The election process is:
  - P_k sends ELECTION messages to all processes with greater ids.
  - If no one responds, P_k becomes the coordinator.
  - If anyone answers, it takes conducts its own election and P_k's job is done.
- The process with the biggest id wins. Biggest guy. Therefore “bully algorithm”.
- If a process that was previously down comes back up, it holds an election.

Election Algorithms (Ring)

- We assume that each process knows who its successor is.
- When any process notices that the coordinator is not functioning, it builds an ELECTION message containing its own process identifier and sends the message to its successor.
- If the successor is down, the sender skips over the successor and goes to the next member along the ring, or the one after that, until a running process is located.
- At each step along the way, the sender adds its own identifier to the list in the message effectively making itself a candidate to be elected as coordinator.
- Eventually, the message gets back to the process that started it all. That process recognizes this event when it receives an incoming message containing its own identifier.
- At that point, the message type is changed to COORDINATOR and circulated once again, this time to inform everyone else who the coordinator is (the list member with the highest identifier) and who the members of the new ring are.

Distributed Event Matching

- At the heart of publish-subscribe systems.
  - Process 1 subscribes to events. (S = subscription)
  - Process 2 publishes notification (N). System needs to check if S matches N.
  - If matched, system sends N to process 1.
- Naive implementation: Single central server. Not scalable.
- To scale up, deterministically divide work among servers.
- sub2node(S) and not2node(N).
- Central server idea can be extended by distributing the matching to many servers.

Gossip Based Coordination

- Interesting application of epidemic protocols is collecting (aggregating)
- Every node P_i has an initial arbitrary value v_i.
- When P_i contacts P_j, they update their values (mean of v_i and v_j).
- Finally, all nodes have the same value.
- Set all v_i to 0 except v_1=1. Start communication from P_1.
- When all nodes have the same value, any node can estimate the size of the system as $1/v_i$.