Computing Parable

- The Cow

- Courtesy: S. Keshav

Message-oriented Persistent Communication

- Message queuing systems
  - Support asynchronous persistent communication
  - Intermediate storage for message while sender/receiver are inactive
  - Example application: email
- Communicate by inserting messages in queues
- Sender is only guaranteed that message will be eventually inserted in recipient’s queue
  - No guarantees on when or if the message will be read
  - “Loosely coupled communication”
**Message-Queuing Model (1)**

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put</td>
<td>Append a message to a specified queue</td>
</tr>
<tr>
<td>Get</td>
<td>Block until the specified queue is nonempty, and remove the first message</td>
</tr>
<tr>
<td>Poll</td>
<td>Check a specified queue for messages, and remove the first. Never block.</td>
</tr>
<tr>
<td>Notify</td>
<td>Install a handler to be called when a message is put into the specified queue.</td>
</tr>
</tbody>
</table>
General Architecture of a Message-Queuing System (2)

• Queue manager and relays
  – Relays use an overlay network
  – Relays know about the network topology and how to route

Message Brokers

• Message broker: application level gateway in MQS
  – Convert incoming messages so that they can be understood by destination (format conversion)
  – Also used for pub-sub systems
IBM’s WebSphere MQ

- Queue managers manage queues
  - Connected through message channels
- Message channel agent (MCA)
  - Checks queue, wraps into TCP packet, send to receiving MCA

Stream Oriented Communication

- Message-oriented communication: request-response
  - When communication occurs and speed do not affect correctness
- Timing is crucial in certain forms of communication
  - Examples: audio and video (“continuous media”)
  - 30 frames/s video => receive and display a frame every 33ms
- Characteristics
  - Isochronous communication
    - Data transfers have a maximum bound on end-end delay and jitter
    - Push mode: no explicit requests for individual data units beyond the first “play” request
Streams and Quality of Service

- Properties for Quality of Service:
- The required bit rate at which data should be transported.
- The maximum delay until a session has been set up
- The maximum end-to-end delay.
- The maximum delay variance, or jitter.
- The maximum round-trip delay.
Quality of Service (QoS)

- Time-dependent and other requirements are specified as quality of service (QoS)
  - Requirements/desired guarantees from the underlying systems
  - Application specifies workload and requests a certain service quality
  - Contract between the application and the system

<table>
<thead>
<tr>
<th>Characteristics of the Input</th>
<th>Service Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>• maximum data unit size (bytes)</td>
<td>• Loss sensitivity (bytes)</td>
</tr>
<tr>
<td>• Token bucket rate (bytes/sec)</td>
<td>• Loss interval (µsec)</td>
</tr>
<tr>
<td>• Token bucket size (bytes)</td>
<td>• Burst loss sensitivity (data units)</td>
</tr>
<tr>
<td>• Maximum transmission rate (bytes/sec)</td>
<td>• Minimum delay noticed (µsec)</td>
</tr>
<tr>
<td></td>
<td>• Maximum delay variation (µsec)</td>
</tr>
<tr>
<td></td>
<td>• Quality of guarantee</td>
</tr>
</tbody>
</table>

Specifying QoS: Token bucket

- The principle of a token bucket algorithm
  - Parameters (rate r, burst b)
  - Rate is the average rate, burst is the maximum number of packets that can arrive simultaneously
Enforcing QoS

- Enforce at end-points (e.g., token bucket)
  - No network support needed
- Mark packets and use router support
  - Differentiated services: expedited & assured forwarding
- Use buffers at receiver to mask jitter
- Packet losses
  - Handle using forward error correction
  - Use interleaving to reduce impact

Enforcing QoS (1)
Enforcing QoS (2)

- Can also use forward error correction (FEC)

HTTP Streaming

- UDP is inherently better suited for streaming
  - Adaptive streaming, specialized streaming protocols
- Yet, almost all streaming occurs over HTTP (and TCP)
  - Universal availability of HTTP, no special protocol needed
- Direct Adaptive Streaming over HTTP (DASH)
  - Intelligence is placed at the client
Stream synchronization

- **Multiple streams:**
  - Audio and video; layered video
- **Need to sync prior to playback**
  - Timestamp each stream and sync up data units prior to playback
- **Sender or receiver?**
- **App does low-level sync**
  - 30 fps: image every 33ms, lip-sync with audio
- **Use middleware and specify playback rates**

**Synchronization Mechanism**

- Procedure that reads two audio data units for each video data unit
- Application tells middleware what to do with incoming streams
- Multimedia control is part of middleware
- Middleware layer
- Incoming stream
- Network
- Receiver's machine
- Application
Multicasting

- Group communication
  - IP multicast versus application-level multicast
  - Construct an overlay multicast tree rooted at the sender
  - Send packet down each link in the tree
- Issues: tree construction, dynamic joins and leaves

Overlay Construction
New Topic: Naming

- Names are used to share resources, uniquely identify entities and refer to locations
- Need to map from name to the entity it refers to
  - E.g., Browser access to www.cnn.com
  - Use name resolution
- Differences in naming in distributed and non-distributed systems
  - Distributed systems: naming systems is itself distributed
- How to name mobile entities?

Example: File Names

- Hierarchical directory structure (DAG)
  - Each file name is a unique path in the DAG
  - Resolution of /home/steen/mbox a traversal of the DAG
- File names are human-friendly
Resolving File Names across Machines

- Remote files are accessed using a node name, path name
- NFS mount protocol: map a remote node onto local DAG
  - Remote files are accessed using local names! (*location independence*)
  - OS maintains a mount table with the mappings

Name Space Distribution

- Naming in large distributed systems
  - System may be global in scope (e.g., Internet, WWW)
- Name space is organized hierarchically
  - Single root node (like naming files)
- Name space is distributed and has three logical layers
  - Global layer: highest level nodes (root and a few children)
    - Represent groups of organizations, rare changes
  - Administrational layer: nodes managed by a single organization
    - Typically one node per department, infrequent changes
  - Managerial layer: actual nodes
    - Frequent changes
  - Zone: part of the name space managed by a separate name server
Name Space Distribution Example

An example partitioning of the DNS name space, including Internet-accessible files, into three layers.

### Name Space Distribution

<table>
<thead>
<tr>
<th>Item</th>
<th>Global</th>
<th>Administrative</th>
<th>Managerial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographical scale of network</td>
<td>Worldwide</td>
<td>Organization</td>
<td>Department</td>
</tr>
<tr>
<td>Total number of nodes</td>
<td>Few</td>
<td>Many</td>
<td>Vast numbers</td>
</tr>
<tr>
<td>Responsiveness to lookups</td>
<td>Seconds</td>
<td>Milliseconds</td>
<td>Immediate</td>
</tr>
<tr>
<td>Update propagation</td>
<td>Lazy</td>
<td>Immediate</td>
<td>Immediate</td>
</tr>
<tr>
<td>Number of replicas</td>
<td>Many</td>
<td>None or few</td>
<td>None</td>
</tr>
<tr>
<td>Is client-side caching applied?</td>
<td>Yes</td>
<td>Yes</td>
<td>Sometimes</td>
</tr>
</tbody>
</table>

- A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, as an administrative layer, and a managerial layer.
- The more stable a layer, the longer are the lookups valid (and can be cached longer)