Communication Problems

- Flow, loss, congestion, policing
- Messages get lost due to several factors, including collisions, lack of buffer space, lack of computing power, etc
- To allow the flow of data at a good pace, the senders of data should be sensitive to the amount of data that is getting through (throughput of the network). From each sender’s perspective, the maximum bandwidth the network provides should be the upper limit on the amount of data to be transmitted. Does this work for real-time?
- However, several senders at once may cause network congestion
- The protocol being used must provide some means of detecting lost messages (or packets) and of correcting present and future
- Two types: hop-by-hop and end-to-end flow control and loss detection and correction
- Trade-offs?

Flow Control

- Some methods are windows (sliding and jumping) and buckets
- Windows count the number of packets in each window and refuse any packets that exceed the number allowed in the window
- Buckets have a (usually) constant rate of accepting packets. Any packet in excess to that rate is not accepted (dropped)
Flow Control

- Flow control can be done with and without feedback from the net
- If there is no feedback, the sender has to control the number of packets it will send according to some pre-established threshold
- This scheme is often applied to *implicit resource reservation* used in many multimedia and real-time applications (due to the tight timing requirements, for example, video and audio, their synchronization, telephone delays or echoes, etc)
- On the other hand, feedback can come on a hop-by-hop basis, or on an end-to-end basis.
- Feedback can take several forms, for example the time for a packet to arrive to the destination (final or intermediate) plus the time for an ack to be received by the sender (*round trip delay*)
- Another example is the packets that arrive at a destination, or their ordering, for example a destination got packet $n$, but not yet packet $n-k$

Media Access Control

In LANs, usually the medium is shared among allPes
Therefore, there must be a way of detecting when it is safe to transmit, and after that, if the transmission was successful

- *explicit reservation* of the bandwidth and time to send
- *implicit reservations* for each PE in LAN (eg, token rings)

- *hope and pray*: put the packet in the network and wait for ack
- *look, hope, and pray*: attempt to send only if no other traffic on net

Aside from collisions, the PE should be able to collect the bits from the network, put them into packets, put these into messages and give them to the user or forward them to the next hop (switch)

Usually protocols implemented in software are the bottleneck in terms of performance (data transfer rates)

Many new applications are taking the protocols to hardware with an improvement of a factor of 4-20 times their software counterparts
Traffic Police

- The Media Access Control (MAC) layer fits between the physical and datalink layers of the ISO OSI model
- It accepts packets from either connectionless or connection-oriented channels and examines the media to verify when to send the packet
- Combining the flow control exercised in the network layer and the access control makes the network sager with respect to reliability and efficiency
- However, all nodes connected to the net should obey the rules of traffic control, and the switched should behave as traffic police
- These policing mechanisms are essential to RT nets, where each packet or message has associated with it a deadline before which it must be delivered
- The criticalness of the application defines whether the traffic is non-RT (datagram), soft-RT (mm), hard RT (remote surgery)

Stream-Oriented Communication

- All communication discussed so far is *time independent*
- Support for continuous media (CM) is *time dependent*
  - Audio, video, sensor data applications
  - Different types of devices (camera, audio card, etc)
  - Maximum end-to-end delays (synchronous CM) and perhaps maximum jitter (isochronous CM)
- There are sources of data and sinks of data
- Known source and destination/sink
  - Are there multiple sinks? perhaps multiple sources?
  - What is the optimal way to reach all destinations?
- Each source may have multiple flows, or different sources will have multiple flows to the same sink.
Data Streams

a) A stream between two processes across a network.

b) A stream directly between two devices.

Data Streams

• An example of multicasting a stream to several receivers.
Streams, flows and QoS

- The Quality of Service (QoS) of an application is the “look and feel” of how the stream will flow
- The QoS depends on two components:
  - Specification of the flow/stream
  - Implementation to satisfy the flow/stream specifications.
- A flow specification encompasses
  - The rates at which the media is transmitted
  - How bursty the flow/stream is
  - How much error can there be (given a cost function, hopefully!)
- The implementation depends on many, many factors
  - Common among all implementations is an enforcement of the specs

Specifying QoS

- QoS specification parameters for a Token Bucket model
- Token bucket: the flow/stream is serviced when there are “credits”
- Other schemes are also good, but different characteristics
  - Leaky buckets
  - Explicit real-time scheduler

<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 (\mu sec)</td>
</tr>
<tr>
<td>Token bucket size (bytes)</td>
<td>Burst loss sensitivity (data units)</td>
</tr>
<tr>
<td>Maximum transmission rate</td>
<td>Minimum delay noticed (\mu sec)</td>
</tr>
<tr>
<td>(bytes/sec)</td>
<td>Maximum delay variation (\mu sec)</td>
</tr>
<tr>
<td></td>
<td>Quality of guarantee</td>
</tr>
</tbody>
</table>
The principle of a token bucket

- Resource reservation in a distributed system
- OS must support mechanism
- Policy can be implemented at the user level

Setting Up a Stream

- Reservations
  - Statistical (good on average) vs guaranteed (good for important stuff)
  - No standards, much research, no prevalent results
Vnet: Pitt’s approach

- V-net is a versatile network architecture [IEEE Transactions on Computers, Aug 2000]
- Specification of both minimum data rate (mandatory traffic $\), above minimum (optional traffic $$\), and bursty traffic
- Use real-time scheduling techniques to be able to schedule streams:
  - EDF is earliest deadline first: if CPU utilization is less than 100\% of the time, accept new stream
  - RMS is rate monotonic scheduling: if CPU utilization is less than $n(2^{10n}-1)$, accept new stream
  - CPU utilization is sum of all process utilizations, which is $C_i/P_i$, where $C_i$ is the execution time of the process and $P_i$ is the period of the process
- Probabilistic acceptances are also possible, and price-based approaches should also be explored

Added / Alternative protocols

- Best Effort (some call is ‘hope and pray’ method):
  - Do as much as you can, and live with consequences
- Fair Queueing or Processor sharing:
  - Define the “share” of each process
  - System to provide only that much service
  - How is it accomplished?
- RSVP: ReSerVation Protocol
  - Soft-state protocol (refresh messages) allows for routes to be pinned.
  - Need to refresh routes at specific periodicity
  - Overhead is high if large groups or if many short-lived communications
- Multicast Trees
  - Shared Trees (ST): one tree for all sources and destinations (minimum spanning tree, stainer tree problem)
  - Source-Specific Trees (SST): one tree per source.
  - Advantages and disadvantages?
Synchronization Mechanisms (1)

- Synchronization of data and of flows is a big problem, mainly due to the unpredictability of the computing systems we have today
  - Interrupts may disturb media continuity
  - Priority inversion may occur (high priority task blocked by low priority task)
  - Different flows should be synchronized
- Application can do synchronization
- Hardware can be used to help specially when the sync requirements are strict
  - Separate streams should be played within 30 microsecs

Synchronization Mechanisms (2)

- Synchronization supported by high-level interfaces.
- Middleware or specialized applications (or layers) multiplex all substreams into a single stream, and demultiplex at the receiver.
- Synchronization is handled at multiplexing/demultiplexing point
- Example: MPEG