Paper comments

- Questions so far on the paper:
  
  “Do we need to produce new research results?”
  - No, not at all. Think of the paper as a survey of a research topic. Summarize the state of the art – no need to create new results.

  “What would a research paper look like?”
  - See next slide.
Research paper structure

- Define topic
- Discuss major research results/projects in the area
- Summarize what these papers tell you about the state of the art in the topic, and where it appears to be going
- Quick skim through examples
  - No, I will not post these. They are from last year, and I don’t give away student work.
Onward

- Let’s finish talking about models, and then dive into details on networking.
- Goal: Dig through most of chapter 3 today.
Interaction model

- How a set of processes communicate to achieve some task.
  - Similar to the model of the sequence of algorithmic steps to accomplish something.

- In distributed systems, we call these distributed algorithms. They are composed of:
  - The steps taken by each participant.
  - The messages transmitted between them to coordinate.

- Processes are assumed to have private state that is revealed through messages only.
Interaction model

Two big limiting factors

- Communication performance
  - **Latency**: Amount of time to send a minimal message from point A to point B.
  - **Bandwidth**: Amount of information that can be transmitted per unit time.
  - **Jitter**: Variability in the time to deliver each of a sequence of messages. This is very critical to consider in multimedia apps.

- A single global notion of time is impossible to maintain.
  - Each computer has a single internal clock.
  - Two clocks read simultaneously will yield different values. Real clocks drift from “real time” and each other.
  - Synchronization methods required to correct for drift.
Variants in Interaction Model

- Based on these limiting factors, we can coarsely break interaction models into two classes.

  - **Synchronous systems**
    - Time to execute a step has known lower/upper bounds.
    - Messages received in a known bounded time.
    - Processes with local clocks drift at a known bound for the drift rate from real time.

  - **Asynchronous systems**
    - No assumptions on bounds for three points above.
    - *Real systems tend to be asynchronous.*
      - Bounds difficult to quantify, or upper bound may not be finite.
Real-time ordering of events

Instructor's Guide for Coulouris, Dollimore and Kindberg
Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005
Processes and Communication Channels

We typically assume a simplified channel when we want to reason about communication between processes for modeling purposes.
Failures

- We have a model for failure to precisely define different failure classes.

- As mentioned last time, not all failures are computer fatalities.
  - Many common situations are less dire, yet still are a form of failure.
Failure model

- **Omission failures**
  - Communication channel fails to perform some action it was supposed to.

- **Arbitrary failure**
  - Any type of error can occur. These are pretty bad.
  - Examples: Undetected data corruption, reordering, duplication.

- **Timing failure**
  - Failure to respond in some time interval. Related to synchronous systems with expected bounds.
Masking failures

- Hiding a failure by converting it to a less-bad type.
  - Checksums to change arbitrary failure into an omission failure by determining that a packet is corrupt and must be rejected.
  - Protocols can make some failures totally vanish to the user or application. E.g.: Retries or resends when omissions occur.

- One of the points of middleware layers is to provide this masking.
  - Checksums on messages.
  - Logical time to order messages and prevent dupes and out of order messages.
# Omission and Arbitrary Failures

<table>
<thead>
<tr>
<th>Class of failure</th>
<th>Affects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail-stop</td>
<td>Process</td>
<td>Process halts and remains halted. Other processes may detect this state.</td>
</tr>
<tr>
<td>Crash</td>
<td>Process</td>
<td>Process halts and remains halted. Other processes may not be able to detect this state.</td>
</tr>
<tr>
<td>Omission</td>
<td>Channel</td>
<td>A message inserted in an outgoing message buffer never arrives at the other end’s incoming message buffer.</td>
</tr>
<tr>
<td>Send-omission</td>
<td>Process</td>
<td>A process completes a <code>send</code>, but the message is not put in its outgoing message buffer.</td>
</tr>
<tr>
<td>Receive-omission</td>
<td>Process</td>
<td>A message is put in a process’s incoming message buffer, but that process does not receive it.</td>
</tr>
<tr>
<td>Arbitrary (Byzantine)</td>
<td>Process or channel</td>
<td>Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.</td>
</tr>
</tbody>
</table>
# Timing Failures

<table>
<thead>
<tr>
<th>Class of Failure</th>
<th>Affects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock</td>
<td>Process</td>
<td>Process’s local clock exceeds the bounds on its rate of drift from real time.</td>
</tr>
<tr>
<td>Performance</td>
<td>Process</td>
<td>Process exceeds the bounds on the interval between two steps.</td>
</tr>
<tr>
<td>Performance</td>
<td>Channel</td>
<td>A message’s transmission takes longer than the stated bound.</td>
</tr>
</tbody>
</table>
Reliability of one-to-one communications

- Validity and integrity define reliability characteristics, related both to security and failure.

- **Validity**
  - Any message in an outgoing buffer eventually makes it to a corresponding incoming buffer.

- **Integrity**
  - Message received is identical to the one that was sent without duplication.

- Threats to integrity:
  - Protocols that retransmit but don’t reject multiply transmitted messages.
  - Malicious users who inject, replay, or tamper with messages.