

# Section 2.5 and 3.1

## Stream Oriented Communication, Threads

John P. Daigle

Department of Computer Science  
Georgia State University

05.16.06

# Outline

- 1 Stream Oriented Communication
  - Support for Continuous Media
  - Streams and Quality of Service
  - Stream Synchronization
  
- 2 Threads
  - Introduction to Threads
  - Threads in Distributed Systems

# Understanding Streams

## Problem Statement

In many situations, it does not matter *when* a particular communication process takes place. But what if we are attempting to serve audio or video, or a combination of both? Time dependent data can be served using **streams**.

# Continuous and Discrete Media

- Continuous Representation Media
  - The temporal relationship between data items is important to the meaning of the data.
  - Examples include video and audio data
- Discrete Representation Media
  - Not time dependent
  - still images, text, executables
- Data Streams handle Continuous Media

# Data Stream

**data stream** A sequence of data units.

## Transmission Modes

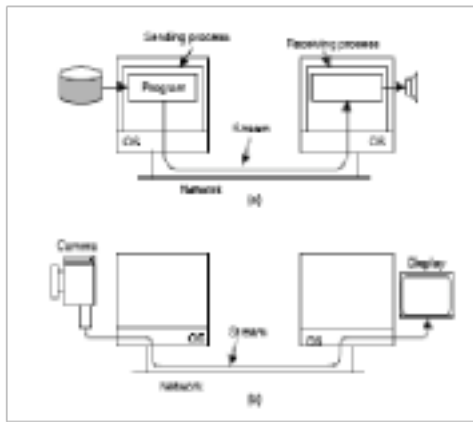
Asynchronous	No timing constraints
Synchronous	Constrained to a maximum delay
Isochronous	timing constrained to a minimum and a maximum

**continuous data stream** a connection oriented communication facility that supports isochronous data transmission

- A **simple stream** consists of a single sequence of data
- A **complex stream** consists of many substreams, which may need to synchronize with each other

# Sources and Sinks

Streams can be set up between different machines, or directly between devices, or both.



# multiparty communication

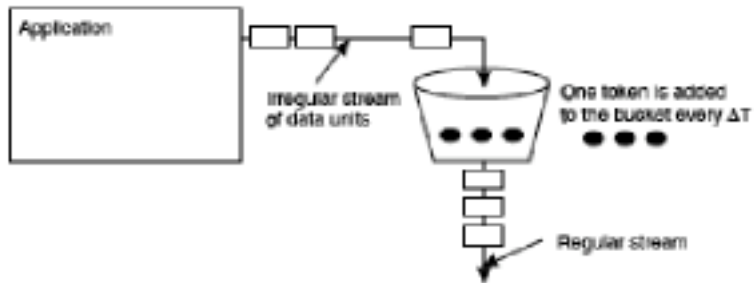
- Different receivers may have different abilities
- The stream should have **filters** to to adjust the the incoming stream.

# Flow Specification

Characteristics of the Input	Service Required
Maximum data unite size	Loss sensitivity
Token bucket rate	Loss interval
Token bucket size	Burst loss sensitivity
Maximum transmission rate	Minimum delay noticed
	Maximum delay variation
	Quality of Guarantee

# Token Bucket Algorithm

Token Bucket Algorithm: A **Flow Specification** for bandwidth requirements, delays, etc.



# Mutual Synchronization

- Discrete Data Stream/Continuous Data Stream
  - Slide Show/Soundtrack
  - The data unit may be several seconds, easy to synchronize
- Continuous/Continuous
  - Movie/Soundtrack
  - The data unit is 40-60 msec, harder to synchronize

# Mechanisms

- Difficult to check streams for synchronization on client side
- MPEG-2 compresses all data to 1 stream at sender side
- This is considered the optimal approach

# processes and threads

## The Operating System

The operating system establishes virtual processors, one for each running program or process. These are protected by the OS, so that they cannot affect each other.

## Threads

Threads are much like processes, but they are not protected at the OS level. This protection is left to the application.

- Switching processes is expensive, because they carry more context
- Processes have their own blocks of memory
- Threads are cheaper, because they carry less context
- protecting threads is more work for developers

# Threads in Nondistributed Systems

## Advantages

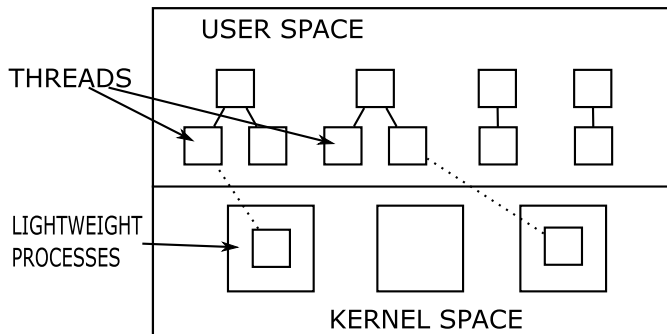
- Allows one process to execute while another waits (for example, for disk access)
- Allows parallelism to be easily exploited
- Useful in large applications with many functions
- Allows compartmentalization of development

## Implementation

- User Level
  - Switching is easy, because the Kernel is not involved
  - A blocking system call will stop the whole process
- Kernel Level
  - Solves the blocking problem
  - But every thread switch requires a system call

# Lightweight Processes

- A compromise between User & Kernel Levels
- Several LWP's per process
- LWP's pick up runnable threads from the scheduler
- If a system call is executed, the OS will switch to another LWP



# Client & Server Threads

- Client
  - Reading HTML while waiting for images, sound
  - Picking up data for one page from several mirrors
- Server
  - dispatcher thread to read requests
  - worker thread handles requests
  - a non-threaded system would be blocked, waiting for the “worker” portion of the task.