**CMPSCI 754 Multimedia Systems** **Spring 2001**

# Lecture 18: Apr 12

*Lecturer: Prashant Shenoy* *Scribe: Zihui Ge*

## 18.1 OS Structure

### 18.1.1 Overview

Goal:

- Design operating system mechanisms to support diverse application mixes.
- Improve application efficiency by reduce the overhead at OS/Application interface.

Design Philosophy:

- redesign the OS; example: Nemesis
- incremental growth; example: QLinux
  Backward compatible: benefit existing applications.

Problem of current OS structure:

- unsuitable for predicable rate-based resource allocation.
- lack of service differentiation.
- no isolation between applications.
- interrupt handling is charged inaccurately.

### 18.1.2 Case study: QLinux

QLinux is an enhanced Linux kernel, which replaces existing schedulers in Linux.

#### 18.1.2.1 CPU scheduler

Implements Hierarchical Starttime Fair Queuing:

- Resource requirements is specified using weights.
- A node gets CPU sharing in proportion to it’s weight.
- Applications are attached to leaf nodes.

- By default, an application is attached to Time Sharing (TS) node.

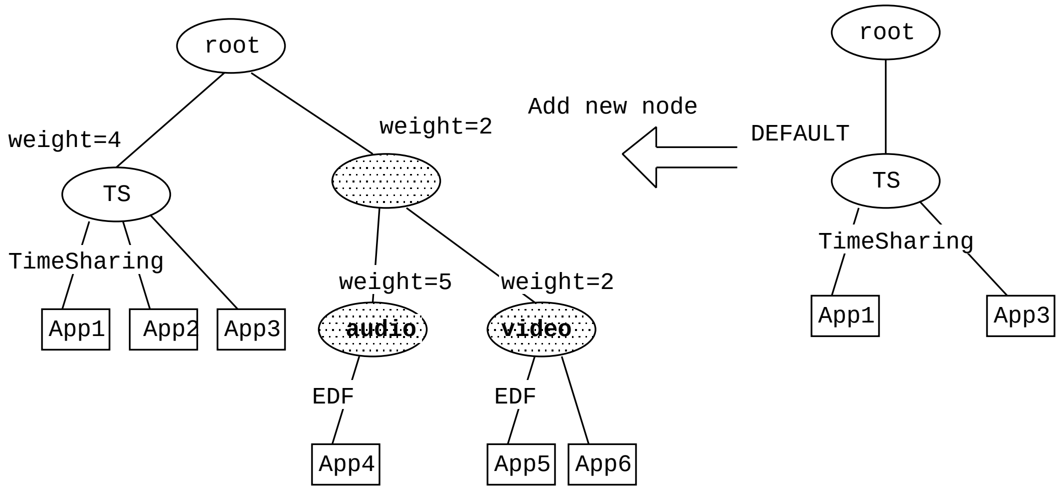


Figure 18.1: H-SFQ CPU scheduler

#### 18.1.2.2 Packet Scheduler

Implements HSFQ:

- Each leaf node maintains a queue.
- Sockets can be attached to queues.
- Sockets can be aggregated(in the same queue) and assigned an aggregate rate.
- New node is created with weigth and queue scheduling policy specified.
- By default, a socket is attached to Best Effort (BE) node which uses FIFO queue management.

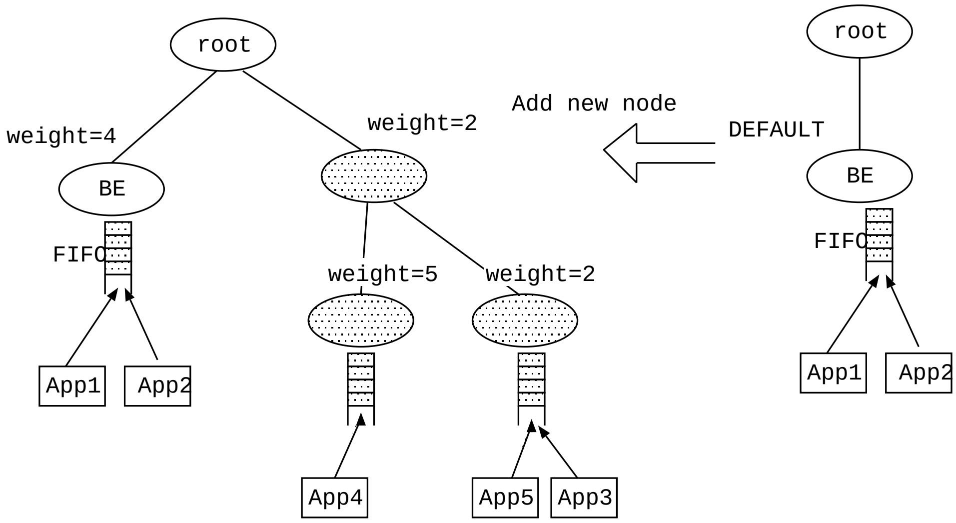

Figure 18.2: H-SFQ Packet scheduler

#### 18.1.2.3 Disk scheduler

Implements Cello:

- Two level hierachy: Class-independent scheduler and class-specific scheduler.
- Associate files to classes.

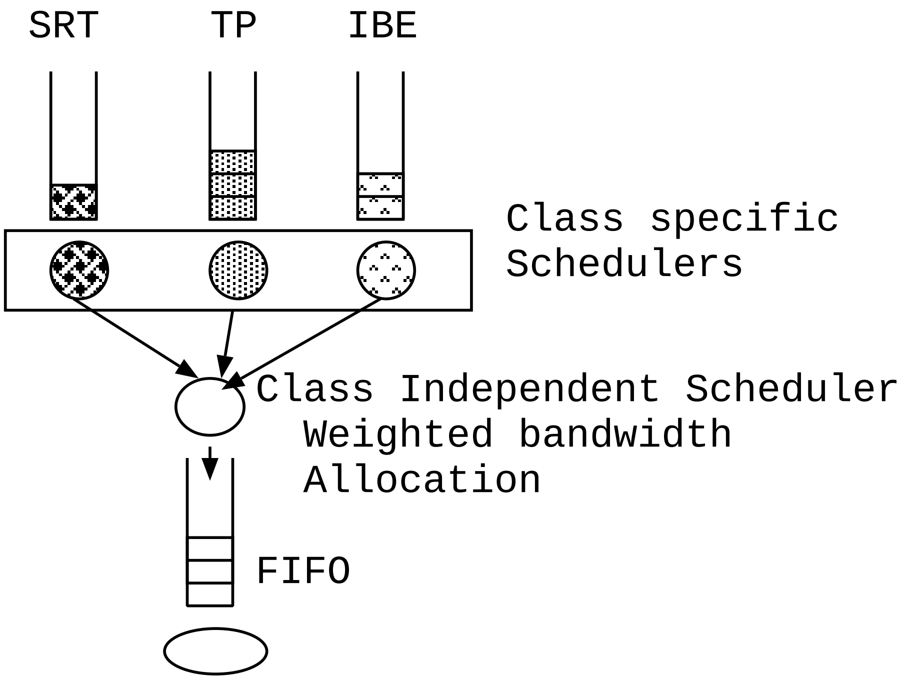


Figure 18.3: Cello Disk scheduler

#### 18.1.2.4 Network subsystem

Implements Lazy Receiver Processing:

- Replacement of old protocol stacks with LRP structure;
- Delay protocol processing;
- Early demultiplexing;

### 18.1.3 Case study: Exokernel

Exokernel Principles:

- Provide applications with control over resources.
- Separate protection from management.

Implementation issues:

- Exokernel only provides resource protection.
  - For CPU: prevents one process hold CPU forever.
  - For Memory: prevents process accessing other user's data.
- Expose allocation, revocation, physical names, system information and data structures to applications.
- Use library operating systems (LibOS) for traditional applications.
- LibOSes are unprivileged; anyone can implement, improve or extend an LibOS.