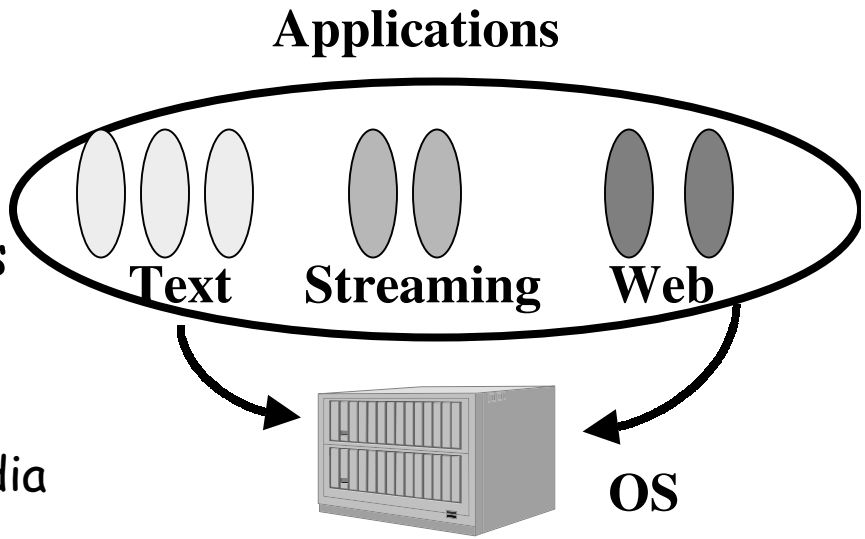
Motivation

- Rapid advances in computing => proliferation of diverse applications
  - Interactive applications
    - E.g. editors, compilers
  - Throughput-intensive applications
    - E.g. http, ftp servers
  - Soft real-time applications
    - E.g. virtual reality, streaming media



- Research Challenge :
  - Design operating system mechanisms to support diverse application mixes

Existing OS: Limitations

- Provide a single class of service
  - Lack service differentiation
  - Lack predictable allocation of resources
- Do not provide application isolation
- Inappropriate accounting of resources
  - Inaccurate accounting of kernel tasks

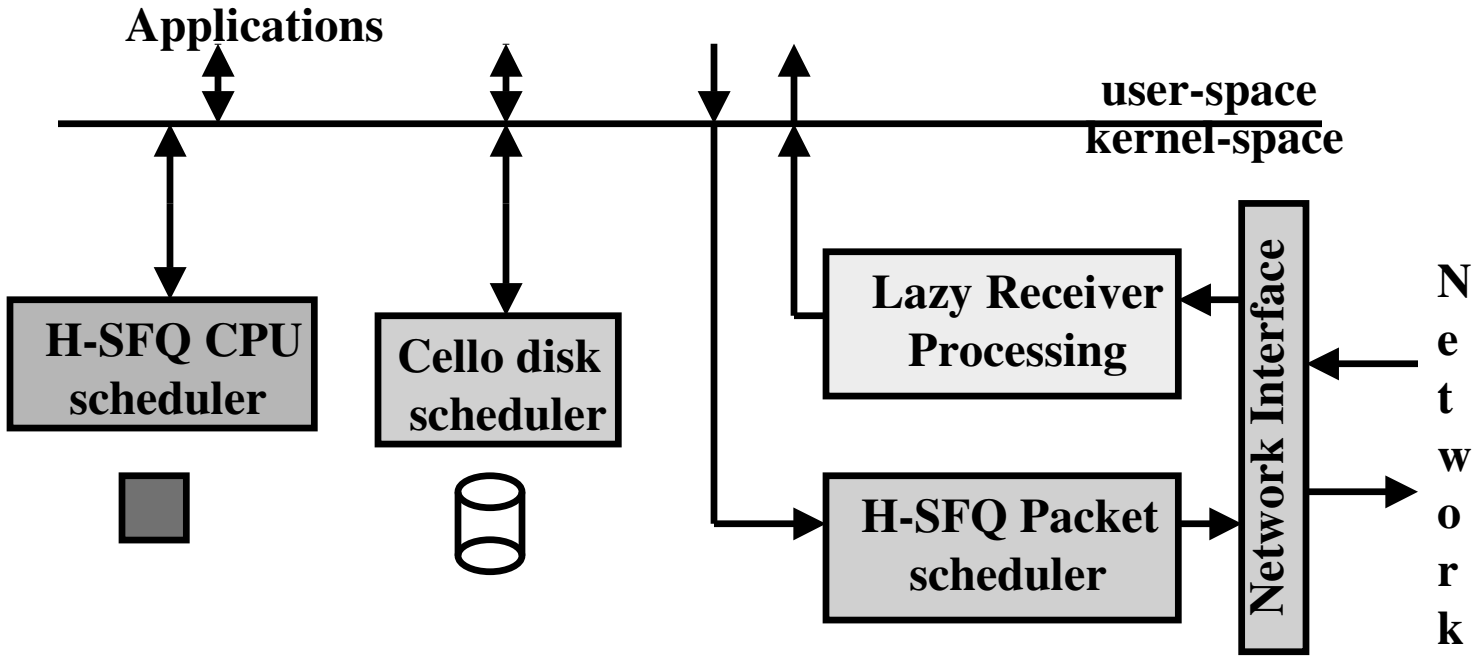
- Approach: QLinux - an enhanced Linux kernel
  - Replaces existing schedulers in Linux
  - Supports diverse application mixes

Design Requirements

- Support for multiple application classes
  - Predictable rate-based allocation
  - Service differentiation
- Application isolation
  - Protect one application class from another
- Appropriate accounting of resource usage
  - Proper accounting of kernel tasks
- Should benefit existing applications

QLinux Architecture

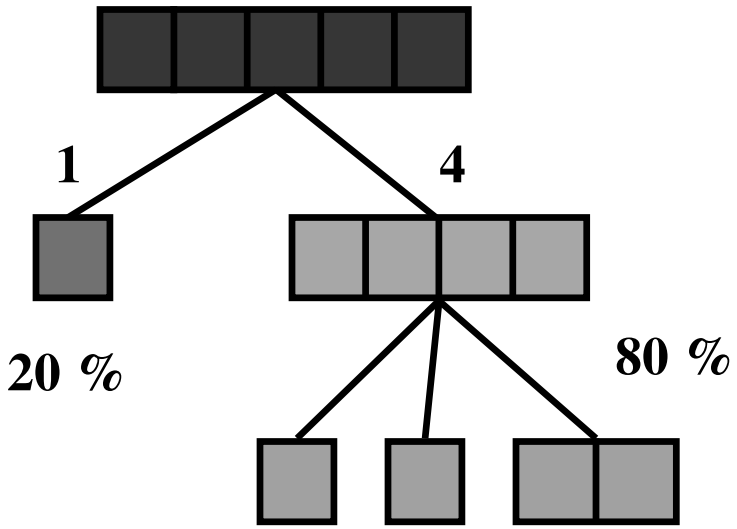
- QoS support using four components

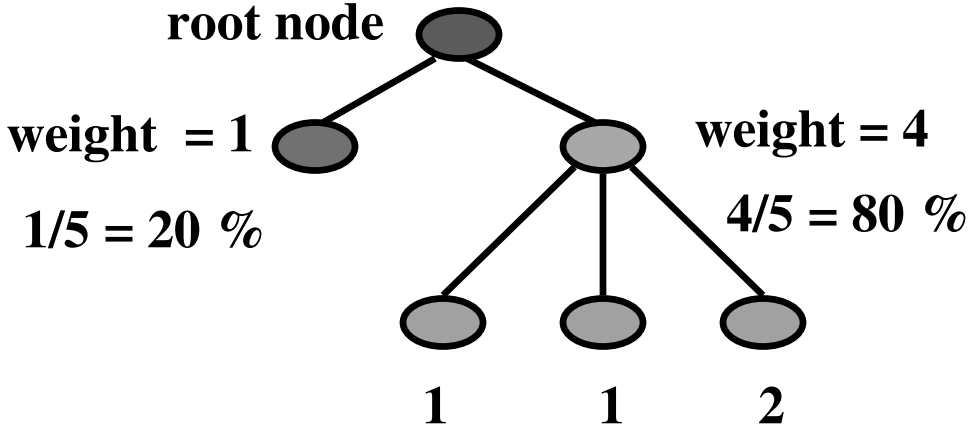


- Key contribution :
  - Integrate resource management mechanisms
  - Study impact on application performance

Hierarchical Weighted Fair Sharing

- Resource requirements specified using weights
- A node gets bandwidth in proportion to it's weight

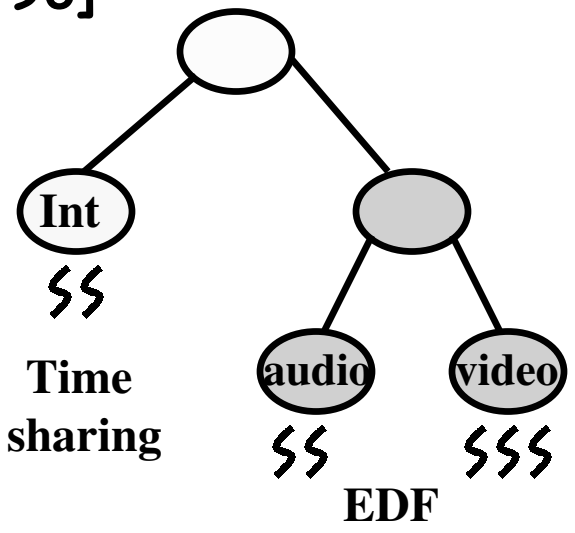




- Applications are attached to leaf nodes

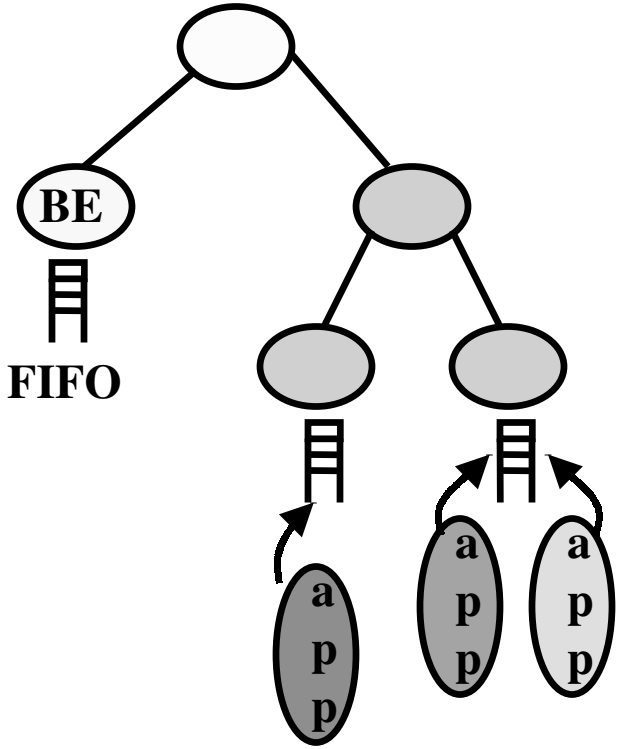
CPU Scheduler

- Implements hierarchical start time fair queuing (HSFQ) **[OSDI'96]**
  - Weighted fair sharing
  - Leaf-nodes use class-specific schedulers
- QLinux provides system calls to
  - Construct and manipulate the scheduling hierarchy
  - Assign weights to nodes



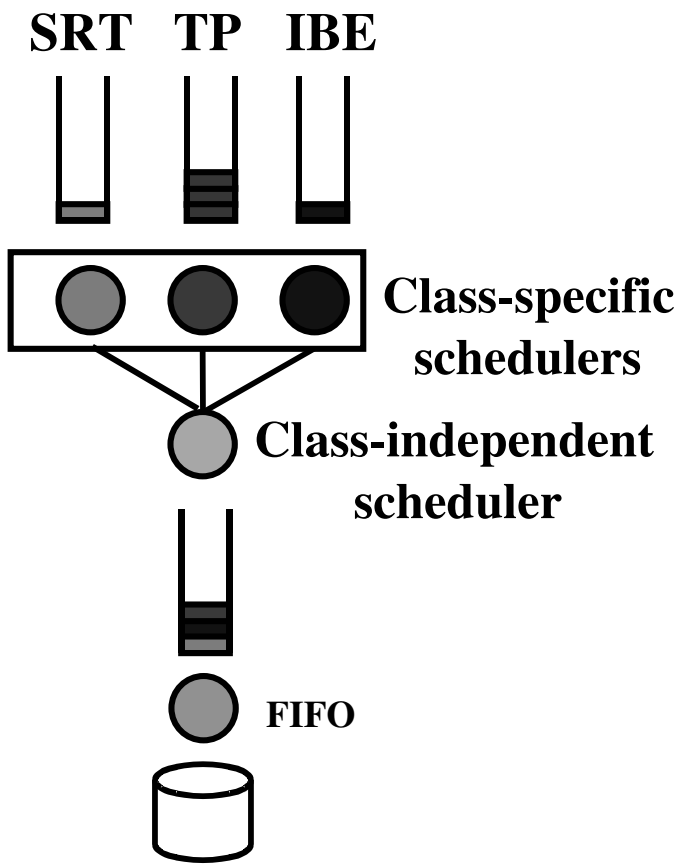
Packet Scheduler

- Implements HSFQ [SIGCOMM'95]
  - Sockets can be attached to queues
  - Sockets can be aggregated and assigned an aggregate rate
  - Uses packet classifiers to map packets to queue
- QLinux provides system calls to
  - Construct and manipulate the scheduling hierarchy
  - Assign weight to queues and attach them to nodes



Disk Scheduler

- Implements Cello [SIGMETRICS'98]
- Two level hierarchy
  - Class-independent scheduler
    - Weighted bandwidth allocation
  - Class-specific scheduler
    - Service differentiation
- Implementation supports three application classes
  - Provides system calls to
    - Manage scheduling hierarchy
    - Associate files to classes



Network Subsystem

- Implements Lazy Receiver Processing (LRP) **[OSDI'96]**
- Existing operating systems
  - Have an interrupt driven network subsystem
  - Results in inappropriate accounting of resources
- LRP implementation
  - Delay protocol processing
    - Achieves proper accounting of resource usage
  - Early demultiplexing
    - Offers better overload behavior

Outline

- ✔ Motivation
- ✔ Design requirements
- ✔ QLinux architecture
- ❒ Experimental evaluation
- ❒ Related work
- ❒ Concluding remarks

Related Work

- Rate-based CPU, Packet and Disk Schedulers
  - Waldspurger95, Chen99, Wijayaratne97
- Eclipse - based on FreeBSD [Blanquer99]
  - Employs hierarchical proportional-share schedulers
  - /reserv file system
- Nemesis [Roscoe95]
  - Multi-service operating system
  - Absolute resource requirement specification