

## **ELEN627 Lecture 9**

- Review of last lecture
- Processor scheduling

## **Processor Scheduling**

- How to allocate processor resources?
- Important that a user process doesn't eat up the processor
- Traditional techniques:
  - Round robin
  - Running time weighted round robin
  - Priority driven

### Round robin scheduling

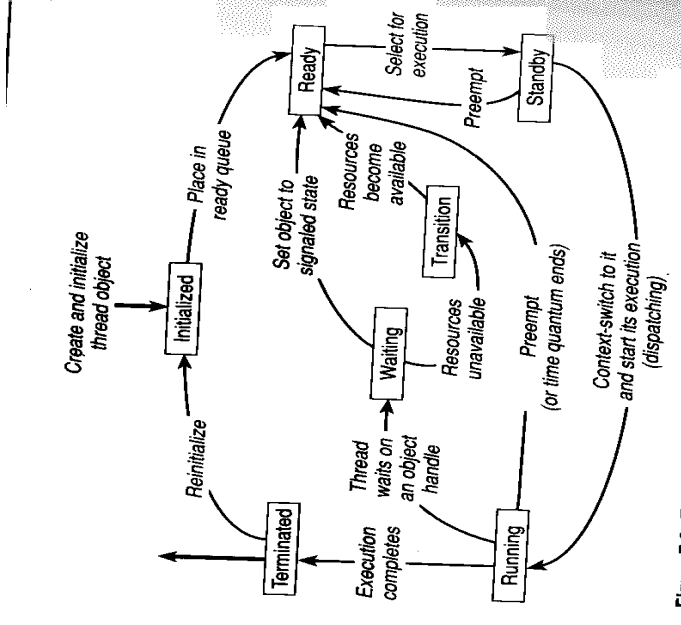
- Allocate a time slice for every process
- Share processor equally among active processes
- Fair division
- Can't allocate more time for one process
- More processes - less share of the processor
- Processor allocation depends on number of processes
- Weighted round robin
- Assign weights to increase the share
- Need to readjust weights with more processes

### Round robin scheduling

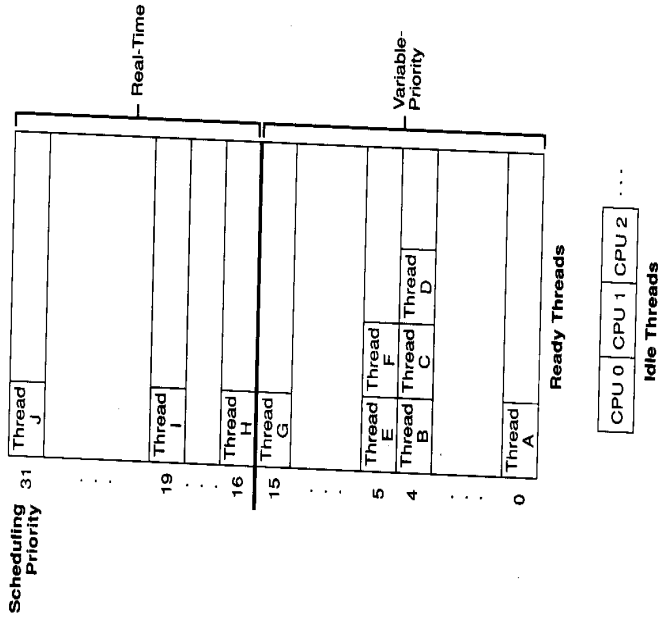
- Give less priority to long running jobs
- Reduce weight depending on running time
- Short jobs get better service
- More users happier

## Priority Driven scheduling

- Assign priorities to each job
- Higher priority jobs get better service
- Can use preemption
  - Preempt a low priority job to run higher priority job
- Processes at same priority can use round robin
- Normally works well in real-time systems
- Difficult to assign priorities
  - Same process may need different priorities at different times
- Tough to predict delivered service
- Priority inversion can occur
  - High priority thread waiting for low priority thread to complete



Windows NT scheduling



Windows NT scheduling

### Multimedia requirements

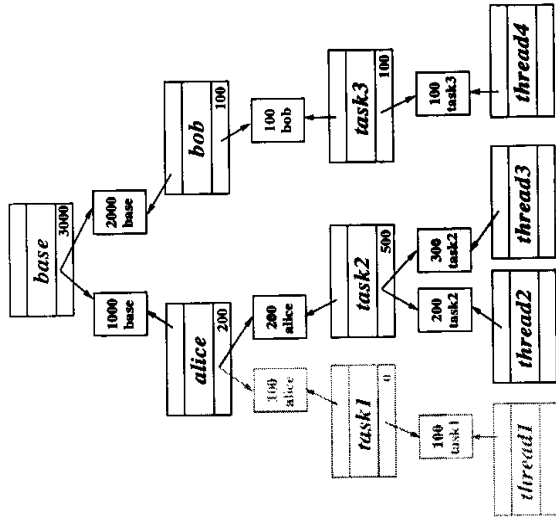
- Allocate certain amount of processing BW
  - Shouldn't change as system gets loaded
- Provide different levels of service
- Different applications may need different schedulers
  - EDF style scheduling for real-time
  - Round robin style for others

## Lottery scheduler

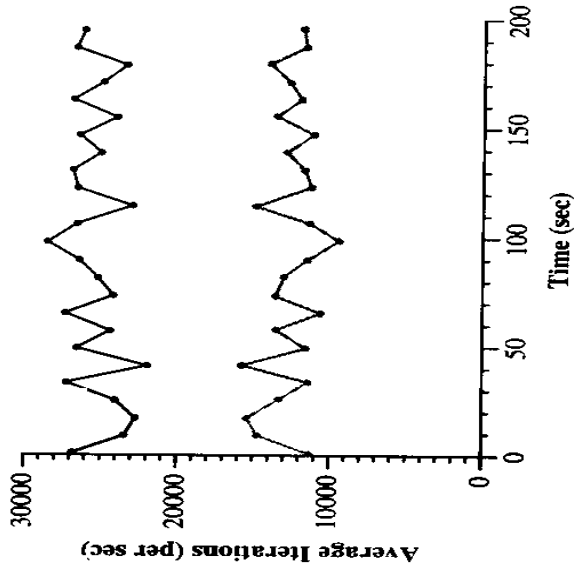
- Allocate tickets to processes
- Every few ms, draw a lottery
- Select the process holding the ticket
- Probabilistically fair
- Can assign processor BW by allocating tickets
  - For more BW, give more tickets
  - Share of bandwidth  $p = t/T$
  - Response time inversely proportional to number of tickets
  - $E[n] = 1/p$

## Issues in lottery scheduling

- Thread is blocked
  - What to do with its lottery tickets
  - Pass them onto the thread holding it up
- Subthreads/subtasks
  - A thread can print its own tickets
  - When allocated processor, hold lottery to pick subtask



Example lottery schedule



Performance of lottery scheduling

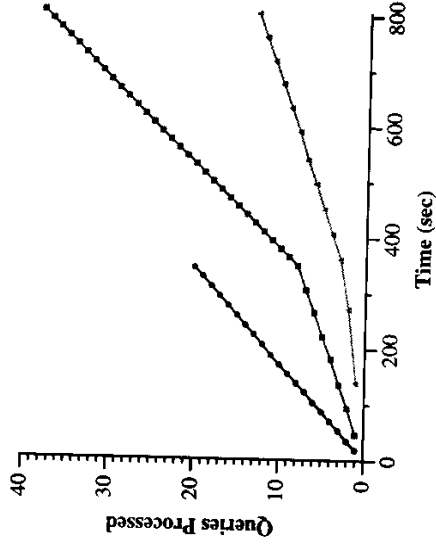


Figure 7: **Query Processing Rates.** Three clients with an 8 : 3 : 1 ticket allocation compete for service from a multithreaded database server. The observed throughput and response time ratios closely match this allocation.

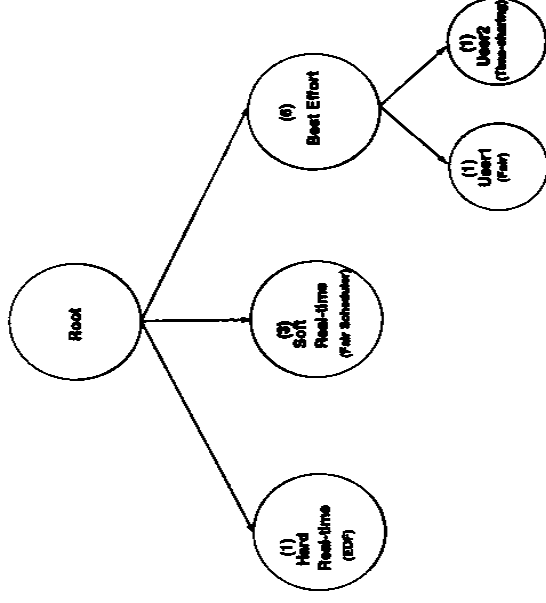
Performance of lottery scheduling

## Hierarchical scheduler

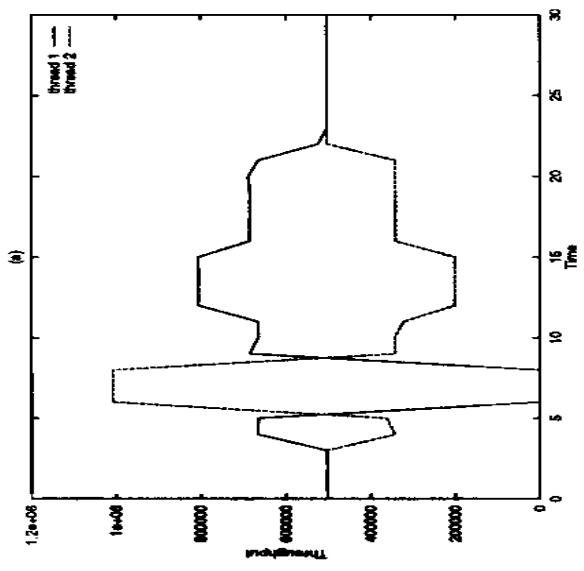
- Allows different scheduler for different tasks
- Separates BW allocation from scheduling
- If thread doesn't use all allocated BW
  - Gets higher priority later when ready to run
- Can't accumulate too much unused BW
  - Other threads will suffer
- Tries to make  $w1(t1,t2)/r1 - w2(t1,t2)/r2 = 0$

## Hierarchical scheduler

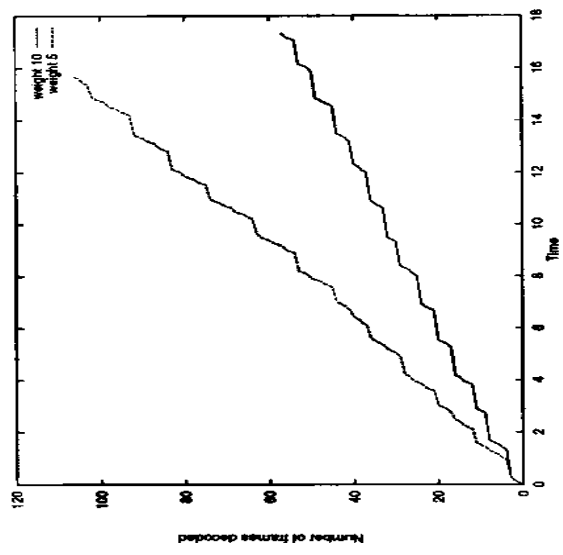
- Assign a start tag to each thread
- Schedule in increasing order of start tags
- Start tag  $S_f = \max\{v(A(q_f^j)), F_f\}$ 
  - $A(q_f^j)$  is the last time the thread used the processor
  - $F_f$  is the finish tag  $= S_f + l_f^j / r_f$
  - Virtual time  $v(t) =$  start tag of thread in service at time  $t$



Hierarchical scheduler model



Hierarchical scheduler performance



Hierarchical scheduler performance

## Strict scheduling

- When processor is not heavily loaded, work well
- Occasionally, load may be higher than can be handled
  - Strict schedulers *ala EDF* break down
  - Work hard to satisfy task requirements
  - When about to complete the task, may switch to another task
  - No notion of incremental utility of processor time
- Need to have best-effort Earliest Deadline First
  - Implement non-strict EDF when resource demand is too high