Advanced Packet Scheduling

Jon Turner
Computer Science & Engineering
Washington University

www.arl.wustl.edu/~jst
Queueing with Delay Guarantees

- Best packet scheduling algorithms avoid excessive burstiness and some provide delay guarantees
  - flow with rate $r$ guaranteed service roughly equivalent to what it would get with a dedicated link of rate $r$
  - packets delayed no more than they would be by ideal link plus time to send one packet on real physical link
  - guarantee is independent of behavior of other flows

- Ideal behavior defined by General Processor Sharing (GPS) model
  - view packet flows as fluid and divide link bandwidth among active flows according to their weights
  - schedulers seek to ensure that last bit of each packet is sent not much later than it would be under GPS
Self-Clocked Fair Queueing

- SCFQ is simple scheduler that uses virtual time ($vt$) to enable work-conservation.
  
  define $vft(Q) =$ virtual finish time of first packet in $Q$
  
  on arrival of packet $p$ for queue $Q$ at time $t$
  
  if $Q$ is empty, let $vft(Q) = \max\{vt, vft(Q)\} + length(p)/wt(Q)$
  
  add $p$ to $Q$
  
  on completion of a packet $p$ from queue $Q$ at time $t$
  
  if $Q$ is not empty
  
  $vft(Q) = vft(Q) + length(p_{new})/wt(Q);$
  
  let $R$ be active queue with smallest $vft$ value
  
  send first packet from $R$ and let $vt = vft(R)$

- Eliminates excessive burstiness of WDRR
  
  at cost of $O(\log n)$ time per packet
Comparison of WDRR vs. SCFQ

- 3 CBR flows (.02, .08, .2), 3 bursty flows (.15, .25, .3 avg.), sampling interval 5,000.
Generalized Processor Sharing

- View packet flows as fluid, divide link bandwidth among active flows according to their weights.
- Weighted Fair Queueing (WFQ) selects packets in order they would complete in GPS schedule.
  - Example: A, F, C, B, G, D with finish times 3, 6, 9, 15, 17, 22.
  - Delay seen by packet $p$ entering queue $Q$ is at most $L_{max} + \frac{\text{backlog}(Q)}{\text{wt}(Q)}$ where $\text{backlog}(Q)$ is the amount of data in $Q$ after $p$ is added to $Q.$
Virtual Time and WFQ

- **Virtual time** used to facilitate simulation of GPS
  - if $W$ is total weight of queues that are active in GPS, then each active queue loses $\frac{wt(Q)}{W}$ bits per second
  - if we define 1 virtual second=$W$ seconds, then can say that active queue loses $wt(Q)$ bits per virtual second
  - must update virtual time as real time passes

- To update virtual time
  - let $vt$=old value of virtual time, computed at time $t$,
  - let $W$=weight of active queues at $t$ and now=current time
  - let $vdt(Q)$ be virtual drain time for queue $Q$
    - while smallest $vdt(Q) \leq vt + (now-t)/W$
      - $t=t+(vdt(Q)-vt)W$; $vt=vdt(Q)$; $W=W-wt(Q)$
      - $vt=vt+(now-t)/W$; $t=now$
Implementing WFQ

- Let $vft(Q) =$ virtual finish time of first packet in $Q$
  let $vdt(Q) =$ virtual drain time of $Q$
- On arrival of packet $p$ for queue $Q$
  update $t$, $vt$ and $W$
  if $vdt(Q) > vt$ then
    $vdt(Q) = vdt(Q) + \text{length}(p) / \text{wt}(Q)$
  else
    $vdt(Q) = vft(Q) = vt + \text{length}(p) / \text{wt}(Q)$; $W = W + \text{wt}(Q)$
  add $p$ to $Q$
- To select outgoing packet
  update $t$, $vt$ and $W$
  select $Q$ with smallest $vft(Q)$ and send first packet
- Worst-case time per packet grows linearly with $n$
Limitations of WFQ

- WFQ guarantees each queue performs no worse than in GPS
  » but some do much better
- Leads to greater burstiness than in GPS
- Example:
  » fast flow with weight 0.5
  » 10 slow flows, each with weight .05
  » at t=0, 10 packets arrive for fast flow plus 1 per slow flow
  » WFQ sends 9 from fast flow before any from slow flows
  » prefer to interleave fast with slow
- Worst-case Fair, Weighted Fair Queueing (WFQ+), corrects this
The WF$^2$Q$^+$ Packet Scheduler

- WF$^2$Q$^+$ keeps number of bits sent per flow by time $t$ closer to number sent by GPS scheduler
  - idea: a queue should be eligible to send a packet only when its first packet has started sending in GPS
  - WF$^2$Q$^+$ does not simulate GPS but behaves similarly

When a packet $p$ arrives for $Q$
update $vt$ by adding elapsed real-time
if $Q$ is empty,
  $vst(Q) = \max\{vt, vft(Q)\}$
  $vft(Q) = vst(Q) + length(p) / wt(Q)$
add $p$ to $Q$
- On completion of a packet \( p \) from queue \( Q \)
  update \( vt \) by adding elapsed real-time
  if \( Q \) not empty
    \[ vst(Q) = vt(Q); vt(Q) = vst(Q) + \text{length}(p_{new})/\text{wt}(Q) \]
    if \( vst(Q) > vt \), make \( Q \) ineligible
  make all queues with \( vst \leq vt \) eligible
  if no eligible queues
    let \( vt = \text{smallest } vst \) for ineligible queues and make
    all queues with \( vst = vt \) eligible
  select eligible queue with smallest \( vft \) and send its first packet
Implementing $WF^2Q+$

- $WF^2Q+$ can be implemented to run in $O(\log n)$ time/packet, using data structure that blends search trees and heaps

- Must maintain "smallest VFT field" when balancing
- Approximate version uses two heaps
Comparison of WF2Q+ vs. SCFQ

- 3 CBR flows (.02, .08, .2), 3 bursty flows (.15, .25, .3 avg.), sampling interval 5,000.

SCFQ - trace6, N=100K, B=1M, sampling period 5K

WF2Q+ - trace6, N=100K, B=1M, sampling period 5K