Scheduling on Overlapping Bonded Channels

Fair queuing strategies for single channels

Assumptions:

- Each flow has its own queue from which packets are scheduled.
- Each queue has a limited capacity.
- Packets arriving when the queue is at capacity are dropped.
- An external regulator or controller allows only those packets eligible for immediate scheduling to enter scheduling queues.

Weighted token bucket

This algorithm is one of the class known as *Approximate Generalized Processor Sharing (GPS).* Unlike Weighted Fair Queuing and Worst Case Fair Weighted Fair Queuing, it doesn't require a virtual clock or round counter and the associated iterated deletion at each arrival or completion.

Credit accumulation

- The channel has a service rate *C* in arbitrary service-units/second (e.g. bits / second).
- Each flow *i* has an associated weight *Wi*
- A flow i having at least one packet in queue (or in service) accumulates credits at a rate

  C x W*i* / sum(W*j* for all *j* for which flow *j* has at least one packet in queue or in service)
- Therefore whenever an arrival or service completion occurs
- T = now - last_update_time

```
for each flow i
        flow[i].credits += T x C x Wi / sum(Wj for all j for which flow j has at least one
                packet in the queue or in service )
```

Credit reduction

- When a packet from flow *i* is scheduled or completes service

```
        flow[i].credits = 0
```

It might seem desirable to subtract credits on packet completion instead of simply zeroing them out. Some strategies that have been tried do this reasonably well, if and only if competing workloads are equally weighted and stochastically similar. In the presence of heterogenous workloads or weights each such strategy that was tried produced diverging credit balances as time progressed thus eventually producing priority scheduling. Resetting the credits to 0 guarantees non-divergent behavior.

When to increment and reset *credits (single channel system)*

There are two reasonable times at which credits should be incremented:

- When the system population is > 0 (includes last packet in service)
- When the queue length is > 0 (excludes last packet in service)

and two quasi reasonable times at which to perform the reset.

- When a packet is scheduled and starts service
- When a packet completes service

The choice has impact when on only when an arrival occurs to an empty queue during service of a packet on the same flow.

```
Service                 Service
start                   complete
  |-----------------------|
  a                       b
                |--------
                c
             Packet
             arrival
```

- If credits are reset at the end of service, then a packet arriving at *c* will always have 0 credit when it becomes eligible for scheduling at time *b.*[1]
- If the credits are reset at the start of service, and population is used to control incrementing, then the arriving packet will have credits accumulated over *b* - *a* time units at time b.
- If queue length is used, then the packet will have *b* - *c* credits.

The third choice clearly seems technically fairest, but might or might not produce the most desirable behavior in a complex mix of heterogenous workloads.

For example, when workloads with short packets are competing with workloads with long packets, the first strategy is more penal to the long packet workloads. It can be shown via simulations that strategy 1 measureably improves the performance of the short-packet workload. Since such traffic is likely to be CBR voice or an ACK stream, the fairest strategy may not be the best.

1 For a single channel system, a packet arriving at time *c* can't possibly be scheduled before time *b,* but in a bonded channel system using this approach, a packet that arrives at time *c* could use credits accumumalated by its predecessor to allow it to be scheduled on another channel *before time b.*

The scheduling algorithm

- for each flow i with flow[i].qlen > 0
```
        diff = flow[i].credits - cost of packet at head of queue
        if (diff > maxdiff)
            maxdiff = diff;
            maxflow = i
```
- The head of queue packet on flow maxflow is scheduled

- An alternative approach would be to schedule the flow with maximum credits

- The current approach favors flows with short (low cost) packets and ameliorates head of line blocking (as does Worst Case Fair Weigthed Fair Queing)

Open questions:

Perform an analysis similar to those done for WFQ and WCFWFQ of the performance of Weighted Token Bucket relative to true GPS using the various increment and reset strategies.

What is the effect of using credits (instead of *diff)* in selecting the process to be scheduled *?*

Extension to channel bonding groups

- Scheduling is still done on a per channel basis.
- A map (as used in the mapper) identifies those channels a flow may use.
- Each flow contains three arrays which hold, for each eligible channel,
  - the flow's weight on that channel, this value should determine the fraction of channel capacity that the flow obtains under all flow overload.
  - fraction of the flow's load as apportioned to the channel by the mapper (these values must sum to 1).
  - the flow's present service credit on that channel.
- Array elements for ineligble channels are not used.

```
typedef struct flow_type
{
   int    id;             /* flow id                */
   int    pktid;          /* next packet id         */
   int    qlen;           /* Doesn't count active tx*/
   int    maxq;           /* maximum qlen           */
   int    pop;            /* Includes packet(s)     */
                          /* currently being tx'ed  */

   double weight[MAX_CHANS];     /* WFQ / WCB weight       */
   double frac[MAX_CHANS];       /* Fraction of load/chan  */
   double serv_cred[MAX_CHANS];  /* service credits        */

   double last_finish[MAX_CHANS];
   :
}
```

Weighted token bucket - extension to bonded channels

Credit accumulation

- Each channel has a service rate *Cj* in arbitrary service-units/second (e.g. bits / second).
- Each flow *i* has an associated weight *Wi*
- A flow *i* having at least one packet in queue (or in service, depending on choice of stragegy) accumulates credits on channel *j* at a rate

  flow[i].frac[j] * Cj x W*i,j* / sum(Wk,j for all *k* for which flow *k* is eligible to use channel *j* and has at least one packet in queue (or in service))
- Therefore whenever an arrival or service completion occurs
- T = now - last_update_time

  for each flow i and channel j

  flow[i].credits[j] += T x flow[i].frac[j] * Cj x W*i,j* / sum(Wk,j for all *k* for which flow *k* is eligible to use channel j and has at least one packet in queue or in service)

Credit reduction

There are at least two options possible here:

- When a packet from flow *i* is scheduled or completes service on channel j
  ```
  flow[i].credits[j] = 0
  ```
- In this strategy a flow accumulates credits on all channels to which it has acces but is charged only by the channel that actually carried the packet.
- This approach obviously tends to distribute a flow's load,
- The effect of distributing the cost is not known.

Or alternatively the credit on all channels can be set to zero.

- When a packet from flow *i* is scheduled or completes service
  ```
  memset(flow[i].credits, 0, sizeof(flow[i].credits));
  ```

Preliminary experiments have shown that the second strategy provides the best performance for small packet loads competing with large packet high throughput workloads.

A sample workload

```
sys
{
   maxclock 1000000
}
chan
{
   rate 1.0
}
chan
{
   rate 1.0
}
flow
{
   arrv_dist exp
   serv_dist exp
   arrv_time 10.0 0.0
   serv_time 7.0 0.0
   maxq 40
   chan_id 0 1.0  0.5
   chan_id 1 1.0  0.5
}
flow
{
   arrv_dist exp
   serv_dist exp
   arrv_time 10.0 0.0
   serv_time 7.00 0.0
   maxq 40
   chan_id 1  1.0 0.5
   chan_id 0  1.0 0.5
}

flow
{
   arrv_dist det
   serv_dist det
   arrv_time 1.0  0.0
   serv_time 0.1  0.0
   maxq 400
   chan_id 0  1.0 0.5
   chan_id 1  1.0 0.5
}
```

Workload characteristics

The workload is designed to compare the effect of pitting two bulk flow type workloads against a low volume CBR flow.

Flows 0 and 1 and each consume 70% of a channel and have exponentially distributed interarrival times and packet sizes.

Flow 2 consumes 10% of a channel and has deterministic interarrival and service times. The average packet on Flows 0 and 1 is (a somewhat unrealistic) 70 times as large as a Flow 2 packet.

Both flows can use both channels and so channel utilization is expected to be 75% .   In the following pages the results of simulating the four flavors of WTB,   WFQ, WCFWFQ, FCFS and RR are shown.

Strict priority scheduling

The best performance the CBR workload can possibly get with a work conserving scheduler is obtained by giving it strict priority over the bulk flows. These results can be used as a base line against which other schedulers can be compared.

```
Flow  0 W 1.48e+06 R 1.48e+01 Q =     7.84 SQ =    13.80  N     1.48
C    99888 D        0  X  9.99e-02 VR 2.56e+07 SC   2.78   0.00
Tx by channel:    49759    50130

Flow  1 W 1.46e+06 R 1.46e+01 Q =     7.55 SQ =    12.52  N     1.46
C   100290 D        0  X  1.00e-01 VR 2.22e+07 SC   5.99   3.97
Tx by channel:    49989    50302

Flow  2 W 2.25e+06 R 2.25e+00 Q =     2.15 SQ =     3.19  N     2.25
C   999996 D        0  X  1.00e+00 VR 1.02e+07 SC   5.24   1.50
Tx by channel:   497471   502525

Chan  0 U     0.75 S 1.26e+00 B 751062.90 GPS-R 538944.75
Chan  1 U     0.75 S 1.25e+00 B 751042.91 GPS-R 538944.75
```

Decoding the output.

The first line identifies the scheduler. That is followed by a section for each of the three flows and finally by a line for each channel

Flow data begins with the word *Flow* followed by the flow id. *W* = integral *n(t)* where *n(t)* is the flow's population (including packets in service) at time *t*. *C* = number of packets that completed transmission on the flow. *R* = *W/C* is the average system (queue + tx) time spent by the packet. *N* = *W/T* (where T = total simulation time) is the average system population. *Q* is the average time spent in the queue. It is equal to *R - average Tx time*. *SQ* is the standard deviation of the observed Q's. D is drops. X = N / R is the throughput in packets per second.

In the channel data U = channel utilization, S = average service time, B = total time the channel was busy.

Weighted token bucket

The four flavors of WTB are presented in order of best performance to the CBR workload.

Reset all credits at end of service

```
Using scheduler wtb
Flow  0 W 1.47e+06 R 1.47e+01 Q =     7.70 SQ =    13.91  N     1.47
C    99888 D        0  X  9.99e-02 VR 2.60e+07 SC   3.33   3.33
Tx by channel:    49776    50113

Flow  1 W 1.46e+06 R 1.45e+01 Q =     7.54 SQ =    12.89  N     1.46
C   100290 D        0  X  1.00e-01 VR 2.33e+07 SC   1.49   1.49
Tx by channel:    50015    50276

Flow  2 W 2.42e+06 R 2.42e+00 Q =     2.32 SQ =     3.45  N     2.42
C   999996 D        0  X  1.00e+00 VR 1.19e+07 SC   1.00   1.00
Tx by channel:   499423   500573

Chan  0 U     0.75 S 1.25e+00 B 750541.72 GPS-R 538946.63
Chan  1 U     0.75 S 1.25e+00 B 751564.09 GPS-R 538946.63
```

Reset only used channel at end of service

```
Using scheduler wtb
Flow  0 W 1.47e+06 R 1.47e+01 Q =     7.75 SQ =    13.94  N     1.47
C    99888 D        0  X  9.99e-02 VR 2.60e+07 SC   2.74   2.04
Tx by channel:    49736    50153

Flow  1 W 1.46e+06 R 1.46e+01 Q =     7.54 SQ =    12.87  N     1.46
C   100290 D        0  X  1.00e-01 VR 2.32e+07 SC   7.74   2.43
Tx by channel:    49900    50391

Flow  2 W 2.46e+06 R 2.46e+00 Q =     2.36 SQ =     3.50  N     2.46
C   999996 D        0  X  1.00e+00 VR 1.23e+07 SC   5.32   1.50
Tx by channel:   498508   501488
Chan  0 U     0.75 S 1.25e+00 B 750640.28 GPS-R 538927.91
Chan  1 U     0.75 S 1.25e+00 B 751465.52 GPS-R 538927.91
```

Reset all credits at start of service

```
Using scheduler wtb
Flow  0 W 1.46e+06 R 1.46e+01 Q =     7.61 SQ =    13.50  N     1.46
C    99888 D        0  X  9.99e-02 VR 2.49e+07 SC   0.00   0.00
Tx by channel:    49866    50023

Flow  1 W 1.45e+06 R 1.45e+01 Q =     7.49 SQ =    12.83  N     1.45
C   100290 D        0  X  1.00e-01 VR 2.32e+07 SC   3.97   3.97
Tx by channel:    49976    50315

Flow  2 W 2.51e+06 R 2.51e+00 Q =     2.41 SQ =     3.59  N     2.51
C   999996 D        0  X  1.00e+00 VR 1.29e+07 SC   1.50   1.50
Tx by channel:   498886   501110

Chan  0 U     0.75 S 1.25e+00 B 750828.71 GPS-R 538929.53
Chan  1 U     0.75 S 1.25e+00 B 751277.09 GPS-R 538929.53
```

Reset only used channel at start of service

```
Using scheduler wtb
Flow  0 W 1.47e+06 R 1.47e+01 Q =     7.70 SQ =    13.65  N     1.47
C    99888 D        0  X  9.99e-02 VR 2.53e+07 SC   2.74   0.00
Tx by channel:    49780    50109

Flow  1 W 1.45e+06 R 1.44e+01 Q =     7.44 SQ =    12.45  N     1.45
C   100290 D        0  X  1.00e-01 VR 2.22e+07 SC   5.95   4.08
Tx by channel:    50021    50270

Flow  2 W 2.80e+06 R 2.80e+00 Q =     2.70 SQ =     3.98  N     2.80
C   999996 D        0  X  1.00e+00 VR 1.58e+07 SC   5.32   1.50
Tx by channel:   498667   501329

Chan  0 U     0.75 S 1.25e+00 B 750792.13 GPS-R 538916.64
Chan  1 U     0.75 S 1.25e+00 B 751313.68 GPS-R 538916.64
```

Priority weighted token bucket

This scheduler uses strict priority scheduling when a channel becomes available but breaks ties by using the *wtb* credit scheme. The result obtained are those presented earlier.

```
Flow  0 W 1.48e+06 R 1.48e+01 Q =     7.84 SQ =    13.80  N     1.48
C    99888 D        0  X  9.99e-02 VR 2.56e+07 SC   2.78   0.00
Tx by channel:    49759    50130

Flow  1 W 1.46e+06 R 1.46e+01 Q =     7.55 SQ =    12.52  N     1.46
C   100290 D        0  X  1.00e-01 VR 2.22e+07 SC   5.99   3.97
Tx by channel:    49989    50302

Flow  2 W 2.25e+06 R 2.25e+00 Q =     2.15 SQ =     3.19  N     2.25
C   999996 D        0  X  1.00e+00 VR 1.02e+07 SC   5.24   1.50
Tx by channel:   497471   502525

Chan  0 U     0.75 S 1.26e+00 B 751062.90 GPS-R 538944.75
Chan  1 U     0.75 S 1.25e+00 B 751042.91 GPS-R 538944.75
```

Results with wfq and wcfwfq

```
Using scheduler wfq
Flow  0 W 1.53e+06 R 1.53e+01 Q =     8.29 SQ =    14.62  N     1.53
C    99889 D        0  X  9.99e-02 VR 2.72e+07 SC   0.00   0.00
Tx by channel:    49849    50040

Flow  1 W 1.50e+06 R 1.50e+01 Q =     7.98 SQ =    13.33  N     1.50
C   100289 D        0  X  1.00e-01 VR 2.37e+07 SC   6.51   6.51
Tx by channel:    50289    50002

Flow  2 W 2.37e+06 R 2.37e+00 Q =     2.27 SQ =     3.33  N     2.37
C   999998 D        0  X  1.00e+00 VR 1.11e+07 SC   0.50   0.50
Tx by channel:   500297   499701

Chan  0 U     0.75 S 1.25e+00 B 750912.03 GPS-R 538943.99
Chan  1 U     0.75 S 1.25e+00 B 751196.30 GPS-R 538943.99


Using scheduler wcfwfq
Flow  0 W 1.57e+06 R 1.57e+01 Q =     8.72 SQ =    15.20  N     1.57
C    99888 D        0  X  9.99e-02 VR 2.81e+07 SC   3.97   3.97
Tx by channel:    50037    49852

Flow  1 W 1.57e+06 R 1.57e+01 Q =     8.69 SQ =    14.25  N     1.57
C   100290 D        0  X  1.00e-01 VR 2.54e+07 SC   2.22   2.22
Tx by channel:    49958    50333

Flow  2 W 2.39e+06 R 2.39e+00 Q =     2.29 SQ =     3.33  N     2.39
C   999993 D        0  X  1.00e+00 VR 1.11e+07 SC   2.00   2.00
Tx by channel:   498588   501405

Chan  0 U     0.75 S 1.25e+00 B 750691.82 GPS-R 538926.70
Chan  1 U     0.75 S 1.25e+00 B 751413.69 GPS-R 538926.70
```

Other possible scheduling discplines

FCFS extends to channel bonding groups in a straightforward way. Each packet is time stamped when it enters its scheduling queue. At scheduling time, all flows eligible to use a channel are examined and the one containing the packet with the earliest time stamp is selected.

As expected, the CBR flow experiences significantly degraded performance under FCFS scheduling, and the performance of the bulk flows is comparable to that provided by WFQ but not as good as WTB.

```
Using scheduler fcfs
Flow  0 W 1.53e+06 R 1.53e+01 Q =     8.32 SQ =    12.42  N     1.53
C    99888 D        0  X  9.99e-02 VR 2.03e+07 SC   2.70   0.00
Tx by channel:    49751    50138

Flow  1 W 1.54e+06 R 1.53e+01 Q =     8.31 SQ =    12.45  N     1.54
C   100290 D        0  X  1.00e-01 VR 2.05e+07 SC   5.97   3.57
Tx by channel:    50167    50124

Flow  2 W 8.40e+06 R 8.40e+00 Q =     8.30 SQ =    12.45  N     8.40
C   999988 D        0  X  1.00e+00 VR 1.55e+08 SC   6.12   1.85
Tx by channel:   499105   500883

Chan  0 U     0.75 S 1.25e+00 B 750699.22 GPS-R 538878.11
Chan  1 U     0.75 S 1.25e+00 B 751405.78 GPS-R 538878.11
```

Round robin

There are at least two feasible ways to extend round robin to bonded channels. Neither of the obvious ways is effective.

Global round robin

There is a system wide “last flow scheduled” variable. When a channel becomes available, a scan begins at the last flow scheduled + 1 and continues until a flow is found that (a) can use the channel and (b) has non-zero queue length. This flow becomes the “last flow scheduled”. If there are no packets to schedule, last flow scheduled is not changed.

Local round robin

Each channel possesses a local “last flow scheduled” variable. When a channel becomes available, a scan begins at its last flow scheduled + 1 and continues until a flow is found that (a) can use the channel and (b) has non-zero queue length. This flow becomes the channel's “last flow scheduled”.

These results were obtained with *local round robin.* Unlike any of the other methods, local round robin produced drops in the CBR workload (which has a maximum queue size of 400). It also produced queuing delays more than 10 times longer than WTB, WFQ, and WCFWFQ. However, local round robin did create the minimum delays for the bulk flows.

```
Using scheduler rr
Flow  0 W 1.41e+06 R 1.41e+01 Q =     7.11 SQ =    12.03  N     1.41
C    99954 D        0  X  1.00e-01 VR 1.95e+07 SC   2.00  11.05
Tx by channel:    49738    50218

Flow  1 W 1.40e+06 R 1.40e+01 Q =     6.98 SQ =    11.89  N     1.40
C   100166 D        0  X  1.00e-01 VR 1.90e+07 SC   0.33   1.27
Tx by channel:    50345    49821

Flow  2 W 2.63e+07 R 2.63e+01 Q =    26.22 SQ =    53.23  N    26.27
C   998032 D     1846  X  9.98e-01 VR 2.83e+09 SC   2.05  11.10
Tx by channel:   500680   497352

Chan  0 U     0.75 S 1.25e+00 B 751306.68 GPS-R 537483.12
Chan  1 U     0.75 S 1.26e+00 B 751303.03 GPS-R 537483.12
```

Dealing with overloaded systems

The *weight* component of the weighted scheduling algorithms affects clearly affects queue population and response, but it does so in ways that are not possible to quantify analytically. In contrast, when the offered load of all flows exceeds the capacity of the channel, the weighting mechanism is designed to distribute the throughput achieved by each load according to its relative weights.

In this example, the load offered by both flows, is sufficient to create an unbounded backlog. Flow 0 has a weight of 3.0 and Flow 1 a weight of 1.0. Therefore 3/4 of the total throughput should go to flow 0 and 1/4 to flow 1.

```
sys
{
   maxclock 1000000
}

chan
{
   rate 1.0
   sched wtb
}

flow
{
   arrv_dist exp
   serv_dist exp
   arrv_time 1.0 0.0
   serv_time 1.0 0.0
   chan_id 0 3.0 1.0
}

flow
{
   arrv_dist exp
   serv_dist exp
   arrv_time 1.0 0.0
   serv_time 1.0 0.0
   chan_id 0 1.0 1.0
}
```

Results

All three weighted scheduling schemes produce identical and correct throughput distribution. WCFWFQ produces the minimum queuing delay, but the differences in delays are insignificant. However, WCFWFQ produces the highest standard deviation in the sample queuing delays.

```
Using scheduler wcfwfq
Flow  0 W 1.76e+07 R 2.35e+01 Q =    22.50 SQ =     7.25  N    17.60
C   749043 D   250290  X  7.49e-01 VR 4.01e+07 SC 278.45   0.00
Tx by channel:   749044

Flow  1 W 1.99e+07 R 7.92e+01 Q =    78.22 SQ =    18.12  N    19.89
C   251063 D   749553  X  2.51e-01 VR 8.27e+07 SC -278.45   0.00
Tx by channel:   251063

Chan  0 U     1.00 S 1.00e+00 B 1000000.13 GPS-R 250695.28

Using scheduler wfq
Flow  0 W 1.77e+07 R 2.37e+01 Q =    22.69 SQ =     7.14  N    17.74
C   748954 D   250607  X  7.49e-01 VR 3.95e+07 SC -15.34   0.00
Tx by channel:   748955

Flow  1 W 1.99e+07 R 7.95e+01 Q =    78.51 SQ =    17.94  N    19.92
C   250593 D   749752  X  2.51e-01 VR 8.24e+07 SC  16.83   0.00
Tx by channel:   250593

Chan  0 U     1.00 S 1.00e+00 B 999998.38 GPS-R 250415.27

Using scheduler wtb
Flow  0 W 1.77e+07 R 2.36e+01 Q =    22.60 SQ =     7.19  N    17.68
C   749318 D   250708  X  7.49e-01 VR 4.02e+07 SC   1.60   0.00
Tx by channel:   749319
Flow  1 W 1.99e+07 R 7.93e+01 Q =    78.29 SQ =    17.90  N    19.89
C   250915 D   748690  X  2.51e-01 VR 8.16e+07 SC  -0.49   0.00
Tx by channel:   250915
Chan  0 U     1.00 S 1.00e+00 B 999998.53 GPS-R 250454.24
projects/sched-3.4 ==>
```