Real Time & Embedded Systems
Assessed Exercise
Lauren Norrie

1 Status

The implementation of the WFQ switch and the FIFO queue were completed. Experiments on varying degrees of cross traffic were undertook, ensuring that the output link was not overloaded. This report contains an overview of the implementation design, results of experiments carried out and a discussion of the results.

2 Design and Implementation

2.1 Class Overview

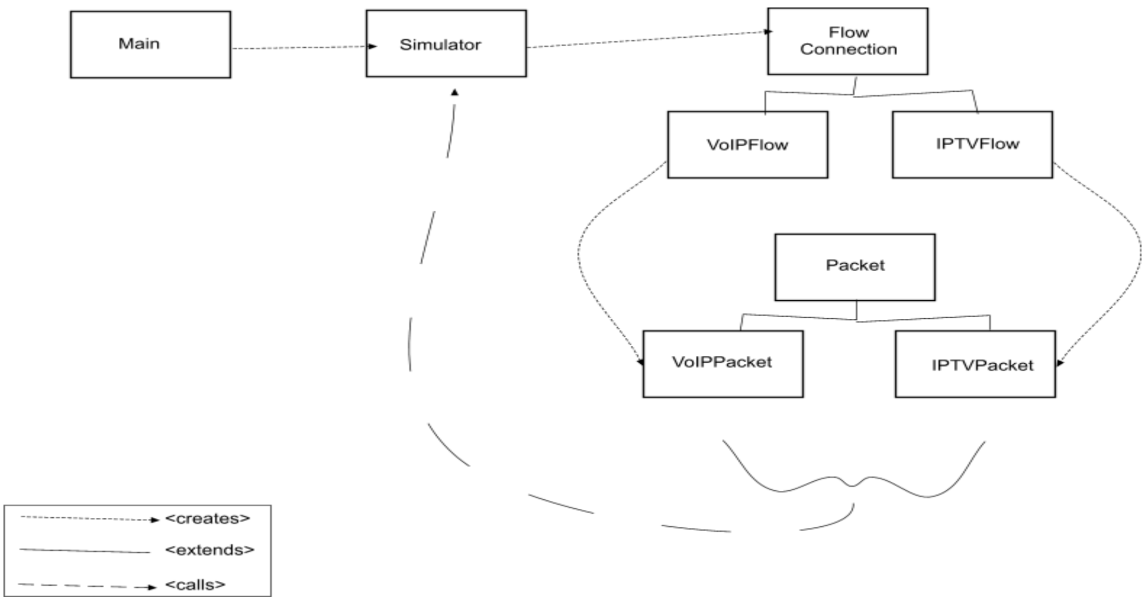

Illustration 1: Class Diagram

Main
Provides user with a UI to run the simulation with the WFQ or FIFO algorithm.
Creates an instance of the Simulator, initialised with an input file.

Simulator
Stores the FlowConnections and provides methods for both algorithms
Creates an outputQueue, used as the SFN queue or the FIFO queue for respective algorithms

FlowConnection
Stores the connection FIFO queue for the WFQ algorithm
Provides appropriate methods for storing/accessing flow information

VoIPFlow/IPTVFlow
Specific FlowConnection that generates VoIP/IPTV packets
Instantiates the appropriate size, period, rate for a VoIP/IPTV flow.

Packet

Stores information about a specific packet
Provides methods for calculating and accessing the delay and arrival times
Calls transmissionComplete() in Simulator

VoIPPacket/IPTVPacket
Adds header to recognise packet as a VoIP/IPTV packet

2.2 Algorithm Implementations

2.2.1 WFQ Algorithm

The output buffer is made up of FlowConnection FIFO queues and the Simulator SFN output queue.

Packets are generated in the flow connection depending on their period (Note: this may have been better if generated packets were read in from a file to make it more predictable). A timer is set for the packets to start generating depending on its start time and one to stop being generated after the flow duration. An acceptance test is performed on a new packet to ensure that the link is not overloaded – packets are discarded if congestion occurs. Though queues are supposed to be unbounded, this is used in the FIFO queue when the link should not be overloaded due to delays causing more packets to arrive than are being transmitted. If the link is idle, i.e. the FIFO queue is empty, then packetReady() in Simulator is called with the newly arrived packet. If the link is busy then the packet joins the end of the FIFO queue.

When packetReady() is called, the finish number is calculated. This is calculated depending on whether the link is idle, i.e. the output queue is empty; the link is busy but the packet's FlowConnection is idle; or both the link and the FlowConnection are busy.

Once the finish number is calculated a packet is inserted into the SFN queue using an insertion sort algorithm, sorted by shorted finish number. The first entry in the sorting algorithm is ignored since the WFQ algorithm is non-preemptive. If a packet is not currently transmitting, i.e. the boolean 'waiting' is set to false, then transmit() is called.

Transmit() in Simulator takes the packet at the head of the SFN queue and calls its 'transmit()' method, setting a timer to go off after its execution time. Transmission is done in a thread to simulate this process, as the Switch should continue to operate while transmission is being performed. When a transmission timer goes off, transmissionComplete() in Simulator is called, where the packet is removed from the queue and the appropriate procedures are executed depending on the status of the link and the flow; either setting the link idle or transmitting the packet at the head of the SFN and either setting the connection to idle or adding the next packet from the FIFO queue.

2.2.2 FIFO algorithm

The output queue in Simulator is used as the FIFO queue. Packets are generated as in the WFQ algorithm. When a packet arrives and is accepted, packetReady() is called immediately so that it may be added to the FIFO queue.

packetReady() takes the packet and calls addFIFOalg() where the utilisation is incremented and the packet added to the end of the FIFO queue. The transmission procedure is the same as in the WFQ algorithm.

When transmission complete is called, the total utilisation is decremented and the process repeats for the next packet in the queue, if one exists

3 Performance: Experiments

3.1 Data Collection

Experiments using both algorithms were carried out on a single VoIP packet with cross traffic made up of other VoIP flows, other IPTV flows, and with a mixture of VoIP and IPTV flows.

The data that was collected on the one VoIP packet were as follows:

- Disturbance: Time between the expected arrival, calculated from the period,
    - and the actual arrival, stored in the packet on creation
- Delay: Time between packetReady() and transmit() being called.
    - This was stored in the packet.
- Total utilisation at time VoIP packet is transmitted

The total utilisation was recorded to ensure that the link was busy enough. A IPTV flow was recorded to have a share of 10.04% of the total utilisation, while VoIP was calculated as 0.006%. Therefore a larger number of VoIP flows are required than IPTV flows to utilise the link. Additionally, the packet drop was recorded to make sure that the output link was not being overloaded by the chosen input. This was a problem for the FIFO implementation as the queue should grow unbounded. The experiments were performed ensuring that congestion did not occur to try control the effects using the threaded approach, though a consistent approach would have improved these results.

The parameters of the single VoIP packet were

- Start time: 0s
- Duration: 20s

These remained constant throughout the experiments and were the time boundaries for the data being recorded: between 0-20000ms. This bound was chosen as it is the least common multiple of the VoIP and IPTV periods (10ms and 20ms). Therefore the results show the performance of 100 packets.

3.2 Experiments

The experiments were carried out under the following conditions:

- No cross traffic
- VoIP cross traffic
- IPTV cross traffic
- VoIP & IPTV cross traffic

These are performed with constant start/durations for all flows and also performed with variation.

As the nature of the simulation design, random perturbation in arrival time was naturally existent and this was measured by calculating the expected start time, based on the period and packet number, and the actual arrival time, recorded as the packet was created.

3.2.1 Packet Distribution: No cross traffic

To create a baseline, the algorithms were both performed under no cross traffic such that the VoIP flow was the only connection through the switch.

Time Series Graphs

Histograms

All packets in the VoIP flow were subject to delays of < 20ms when there was no cross traffic.

Total Utilisation

A VoIP flow is allocated 0.06% of the outgoing link.

Perturbation of Arrival Times

Natural differences in arrival time were < 20ms

Results

In the case of no cross traffic it can be seen that delays between 0-10ms were normal and delays between 10-20ms are rare. This can be accounted for in the experiments.

3.2.2 Packet Distribution: VoIP cross traffic

Input Parameters

Flow ID	Start Time (ms)	Duration (ms)
[VoIP 0]	0	20000
[VoIP 1]	0	20000
[VoIP 2]	0	20000
[VoIP 3]	0	20000
[VoIP 4]	0	20000
[VoIP 5]	0	20000
[VoIP 6]	0	20000
[VoIP 7]	0	20000
[VoIP 9]	0	20000
...		
[VoIP 99]	0	20000

VoIP flows are given a share of 0.609% of the link. Theoretically, 164 VoIP flows would be required to utilise the link (as 164*0.609=99.876%). However, as delays cause queuing to build up, especially in the FIFO implementation where no packets are buffered in a connection queue, congestion would occur very quickly. A more suitable value for simulation was 100 flows, ensuring that the link would not be overloaded. It will be expected to run no more than 60.9%. All flows start immediately and finish at 20s.

Time Series

The WFQ graph shows 8 instances where the delay is > 80ms. Maximum delay was 2239ms.

The FIFO time series graph shows 25 instances where the delay is > 80 ms. Maximum delay was 186ms.

It may be noted that 26 packets were discarded in the FIFO performance test for an input of 100 VoIP flows.

Histogram

Delays > 180ms are shown as value 3000. This was to reduce the range of the graph.

Total Utilisation
Perturbation of Arrival Times

There was a maximum disturbance in WFQ packet arrival of 79ms.
Results shown

Looking at the total utilisation of FIFO with 100 VoIP flows, it is clear to see when the 26 packets were dropped. Though the FIFO queue was supposed to be unbounded, not dropping the packets would have over utilised the link, as in the FIFO implementation utilisation is incremented as packets are added to the queue. Instead of discarding the packet, the algorithm could have been implemented such that utilisation is incremented as the transmit method is called. Despite packets being dropped, it would be easy to imagine how the additional packets would have increased the delays further.

The WFQ total utilisation remains around 60% as expected. As packets are being buffered in the connection, only those in the SFN queue contribute to the total utilisation, and since only one packet per connection may be added to the SFN queue, it is the combined utilisation that is the bound.

At time 10000ms the utilisation of WFQ fluctuates after being constant. The reason for being constant is that the FIFO buffers in each connection are being emptied, to the utilisation does not decrement. All flows are backlogged due to each flow having equal parameters and so the WFQ algorithm allocates them finish numbers with equal priority, such that they perform as FIFO in the SFN queue. The fact that only one packet per connection avoids congestion occurring, and so the utilisation is constant while each flow is serviced; the finish number of each busy connection increments at a constant rate.

As the outgoing link sends at a faster rate than the incoming links, the SFN queue will have been cleared and SFN numbers starting to be recalculated, causing prioritisation to start to occur.

3.2.3 Packet Distribution: IPTV cross traffic

Input Parameters

Flow ID	Start Time (ms)	Duration (ms)
[VoIP 0]	0	20000
[IPTV 1]	0	20000
[IPTV 2]	0	20000
[IPTV 3]	0	20000
[IPTV 4]	0	20000
[IPTV 5]	0	20000
[IPTV 6]	0	20000
[IPTV 7]	0	20000
[IPTV 8]	0	20000

IPTV flows are given a share of 10% of the link. Therefore 8 IPTV flows were chosen to ensure that the link was not overloaded. It will be expected to run no more than 80.006%. 19.994% is reserved in the case of FIFO where delays may cause flows to miss their deadlines, i.e. a new packet arrives before the previous one sent. All flows start immediately and finish at 20s.

Time Series

WFQ was subject to more delays than FIFO when most traffic was IPTV.

Histogram

Total Utilisation

Perturbation of Arrival Times

Results shown

The WFQ algorithm performs worse than FIFO when most cross traffic is IPTV. IPTV traffic appears at a lower rate than VoIP packets; chunks of data is sent in one packet every 20ms, and so there would be at most 8 packets from the other flows in the queue every 20 ms. Therefore queuing would mainly be made up of its own packets. As FIFO is simpler than WFQ, packets are put onto the queue and transmitted straight away. Delays are more common in WFQ when finish numbers for the other packets are being computed, as the connection is busy when the VoIPs next packet is being computed it receives a higher finish number and is put to the end of the SFN queue while all other IPTV packets finish.

3.2.4 Packet Distribution: VoIP & IPTV cross traffic

Input Parameters

Flow ID	Start Time (ms)	Duration (ms)
[VoIP 0]	0	20000
[VoIP 1]	0	20000
[VoIP 2]	0	20000
…		
[VoIP 50]	0	20000
[IPTV 51]	0	20000
[IPTV 52]	0	20000
[IPTV 53]	0	20000
[IPTV 54]	0	20000
[IPTV 55]	0	20000

50 VoIP flows were chosen such that 50*0.61% = 30.5% of the total utilisation would be for VoIP flows, while 5*10.04% = 50.2% of the link would be IPTV traffic. Theoretically, the total utilisation should not exceed 80.7%.

Time Series

Maximum delay for WFQ was 1531ms.

Maximum delay for FIFO was 261ms.
19 packets were discarded in the FIFO experiment.

Histogram

Total Utilisation

Perturbation of Arrival Times
The maximum arrival time for WFQ was 44ms.

Results shown

Again, packets were discarded in the FIFO implementation due to packets being queued up faster than they were attended, though the link itself was not overloaded. The delays caused by the 19 discarded packets should also be taken into account.

Though WFQ experienced higher maximum delay, in general it performed better. The majority of packets experienced delay of below 10ms, which was seen to be a normal duration in the single VoIP test. Total utilisation does not exceed 80.7% in WFQ, while delays cause the FIFO queue to overload.

The rate shown in the WFQ utilisation between 0-10000ms is similar to the rate when all traffic was VoIP. The disturbance in this rate is due to the IPTV packets being added.

Looking at the Time Series graphs, it can be seen that FIFO causes packets to be delayed for a period of time before it catches up on packets, while WFQ may delay one packet for a longer period of time and quickly catch up. If you account for the small amount of large delays, and add in factors of packets being discarded in FIFO, WFQ performs better here when there is a variation in traffic.

3.2.5 Packet Distribution: VoIP & IPTV cross traffic with input variation

Input Parameters

Flow ID	Start Time (ms)	Duration (ms)
[VoIP 0] :	0	20000
[VoIP 1] :	1000	20000
[VoIP 2] :	2000	20000
[VoIP 3] :	3000	20000
[VoIP 4] :	4000	20000
[VoIP 5] :	3000	20000
[VoIP 6] :	4000	20000
[VoIP 7] :	5000	20000

[VoIP 8] :	6000	10000
[VoIP 9] :	7000	10000
…		
[VoIP 30] :	7000	10000
[IPTV 31] :	1000	20000
[IPTV 32] :	2000	20000
[IPTV 33] :	3000	20000
[IPTV 34] :	4000	20000
[IPTV 35] :	5000	20000
[IPTV 36] :	6000	10000

30 VoIP flows were chosen such that 30*0.61% = 18.3% of the total utilisation would be for VoIP flows, while 6*10.04% = 60.24% of the link would be IPTV traffic. Theoretically, the total utilisation should not exceed 78.54%.

Time Series

Maximum delay for WFQ was 1091ms.
Average delay for WFQ is 14.26ms.

Maximum delay for FIFO was 217ms.
Average delay for FIFO was 16.06ms.
33 packets were discarded in the FIFO experiment.

Histogram

Total Utilisation

Perturbation of Arrival Times
The maximum arrival time for WFQ was 155ms.

Results shown

Packets were discarded in the FIFO implementation due to packets being queued up faster than they were attended, though the link itself was not overloaded. The delays caused by the 33 discarded packets should also be taken into account.

The WFQ algorithm shows to perform better than FIFO under varied traffic, start times and duration. This is shown by the average delay in the time series; where one packet is delayed for a long period and arrives too late at the receiver, this would be preferred over many packets being delayed as the user of the application streaming IPTV or VoIP packets would be less likely to notice one packet than many. It can also be shown in the total utilisation, where in WFQ the algorithm is utilising the link such that each link is being attended to depending on its priority, while in FIFO performance is varied according to the traffic appearing on the link.

4 Performance: Discussion

The WFQ prioritises a particular connections traffic based on its share of the total utilisation and the utilisation of other busy connections. This balances the share of traffic that may consist of many small packets passing through a connection with a low send rate (e.g. the VoIP packets) or few large packets sent out on a higher rate connection (e.g. the IPTV packets).

The experiments showed traffic passing through the link under varying types of cross traffic. WFQ algorithm is similar to Earliest Deadline First as it schedules packets with a shorter period first, using the time between less frequent packets to catch up on delays. The parameters chosen for these were such that the output link would be close to 80-90% utilisation, while not overloading the link. Due to the implementation of FIFO, a lower number was used so that less packets would be discarded.

WFQ algorithm showed to perform better when there was varied traffic. Without varied traffic there was less to compute in terms of fairness; when all parameters were the same it would be better to use FIFO. As WFQ algorithm buffers incoming packets for each connection, only contributing one packet per connection to the total utilisation, this performed better. However, if FIFO was unbounded then it would not discard packets when it was subject to more traffic, and this could have been seen in the results.

The way that WFQ algorithm subjects some packets to a high delay would be acceptable for VoIP and IPTV traffic as packets would be containing soft real-time voice or video data where one packet received too late would only result in a decrease in quality for the user, who may not notice assuming the packet loss was at within a certain limit. The way that FIFO delays packets for longer periods of time would become larger enough to be noticed at the receiver, who would eventually have to wait for the packets to catch up in a buffer before playback would be resumed.

5 Further Work

This implementation used threads to generate packets. This caused the experiment results to vary on each execution. A better approach would have been to read in each step as a line in a file, containing the flowID of the newly arrived packet, along with its parameters. This would have allowed the performance to have been reasoned better as this would ensure that the parameters used in each experiment for the WFQ and FIFO algorithms were identical and would allow more accurate results to be shown.