OPTIMISING THE POLLING SEQUENCE IN EMBEDDED ROUND ROBIN WLANS

LACHLAN L.H. ANDREW AND RAVINDRA S. RANASINGHE
ARC Special Research Centre for Ultra-Broadband Information Networks (CUBIN),
Department of Electrical and Electronic Engineering
The University of Melbourne, VIC 3010, Australia
{l.andrew,r.ranasinghe}@ee.mu.oz.au
Tel. +61 3 8344 3816 Fax. +61 3 8344 6678

Transmitting variable-bit-rate real-time data on the uplink of a polled wireless local area network requires careful scheduling to achieve satisfactory performance and capacity. This is a "blind" scheduling problem, since the number and arrival times of packets at each remote station are not known by the scheduler. Embedded round robin (ERR) was recently proposed to address the problem of the high cost of polling idle stations. This paper presents an enhancement of ERR, least-recently-used-ERR, which outperforms the original ERR, especially when network load is high.

1 Introduction

In wireless local area networks, bandwidth management (i.e., packet scheduling) is crucial to providing acceptable quality-of-service (QoS) for real-time sessions having diverse traffic characteristics[1]. Real-time multimedia services are expected to form a major component of future network traffic. The IEEE 802.11 wireless local access network (WLAN) standard[2] specifies that real-time data be transmitted by polling, but does not specify the order in which stations are to be polled.

Scheduling downlink packet streams is essentially a centralized task, and is analogous to scheduling in wired networks. Many centralised scheduling algorithms have been proposed to achieve different QoS characteristics such as low delay[3,4], low delay jitter[5], fairness[6,7,8,9,10], and maximum end user satisfaction[11]. All of these schemes require the scheduler to know details pertaining to each active session (i.e., packet arrival times, queue lengths, packet lengths etc.). These algorithms can be easily implemented in a centralised queueing environment since the required information is available.

The uplink of wireless networks presents particular difficulties for packet scheduling, especially in the presence of non-uniform load. When a distributed queueing system is controlled using a centralised scheduler, the obvious mechanism of granting channel access to mobile stations is polling. One possible polling strategy is round robin[12] (RR), which polls stations cyclically, regardless of the state of their queues. When RR is used to schedule service from a centralized queue, the scheduler can efficiently bypass stations that have no data to transmit. In contrast, the scheduler of a wireless network must poll a station in order to determine that it has no data to transmit. This overhead can be significant; for example, in IEEE 802.11 wireless LANs such a null poll takes about 5% of the time of a maximum length packet transmission[2].

If the scheduler is able to poll queues non-uniformly depending on their current load then it can save bandwidth. Most existing non-uniform polling schemes require explicit messaging. (See for example Reference 13.) This not only introduces transmission overhead but is also incompatible with many WLAN standards[2]. Other MAC protocols

have been proposed[14,15,16] to extend to wireless networks some of the fair queueing and deadline-ordered scheduling schemes popular in wired networks. These again generally achieve QoS at the expense of explicit messaging.

Several schemes have been proposed to avoid null polls, notably embedded round robin[17] (ERR). This algorithm is the basis for the wireless dual queue, which implements the dual queue algorithm[11] for transient congestion avoidance.

This paper will explore the potential of the embedded round robin paradigm to provide high quality of service under a range of conditions. After reviewing the original ERR algorithm, the least-recently-used ERR (LRU-ERR) algorithm will be described and its performance will be investigated under a range of conditions. The LRU-ERR algorithm aims to increase the proportion of packets being served without undue delay in the presence of high loads.

2 Embedded Round Robin

One of the major problems associated with scheduling distributed uplink queues is the lack of information available to the scheduler. Therefore the scheduler may poll potentially idle stations, wasting bandwidth. By knowing whether or not a station had further packets to transmit the previous time that it was polled, the scheduler can identify stations which are are guaranteed not to yield null polls. This concept is the basis for the embedded round robin (ERR) scheme[17].

Most polling protocols provide a single bit of feedback from remote stations indicating the presence of more data to transmit. For example, it is supplied by the the `more data` bit of the IEEE 802.11 header[2]. Based on this feedback, traditional exhaustive round robin repeatedly polls a single station until it has no more data, before advancing to the next station in round robin order. This only risks polling an idle station when there are no stations known to have data, and hence minimises the number of null polls. A polling strategy with no null polls (and no packet discards) is "work conserving", and all work conserving service disciplines have equal average packet delay, provided that the order of serving packets is independent of their lengths. By minimising the number of null polls, exhaustive round robin provides the minimum average packet delay of all schedulers. However, the average delay is not the primary performance criterion for real-time services. Of greater interest is the proportion of packets which arrive within a particular "good service" delay, t_G. This is particularly so when a jitter buffer is used by the application. A jitter buffer will discard all packets delayed by more than t_G, and delay all packets received with lower delay, to ensure that all retained packets have an equal delay of t_G. Although exhaustive round robin minimises the average packet delay, there is scope to deliver a higher proportion of packets with low to moderate delays. This is the goal of embedded round robin (ERR).

Define "busy" stations to be those known to have data to transmit, and "clear" stations to be those which may be idle. The ERR scheduler services clear stations in round robin order. However, between consecutive services of clear stations, it performs an "embedded" round robin cycle of the busy stations. This reduces the time wasted on polling idle or low-rate stations. When there are only a few busy stations, they receive most of the bandwidth, hence their backlog is soon cleared and polling of clear stations is not unduly delayed.

However, when the number of busy stations increases, the backlog can remain for some time, and substantially increase the duration of the outer round robin cycle. This

```
Variables:
    N_max      = max. consecutive polls of busy stations
    busy_count = number of busy stations
=========================================================
i := next clear station in RR order
Poll station i
IF (station i had more data)
    Mark station i as busy
END IF

FOR k := 1 to min (N_max, busy_count)
    j := next busy station in RR order
    Poll station j
    IF (station j had no more data)
        Mark station j as clear
    END IF
END FOR
```

Algorithm 1. One iteration of embedded round robin

could cause packets arriving at "clear" stations to receive unnecessarily poor service. To prevent this, the busy round robin cycle may have to be interrupted to service the next "clear" station. In order to accomplish this, at most $N_{\max}$ busy stations are served between each pair of busy stations. Pseudocode for ERR is given as Algorithm 1.

A key advantage of ERR over exhaustive RR is that exhaustive RR has no way of knowing which stations other than the current station have data to transmit, whereas ERR does. It is this property which allows the wireless dual queue algorithm[17] to detect and combat transient congestion, without causing excessive null polls.

3 Improving High Load Performance

When the number of stations is large, the time taken to cycle through the "clear" list may be excessive if every station in the "busy" list is polled between each pair of polls to clear stations. If the proportion of busy stations remains constant, the cycle time of the clear list grows quadratically with the number of stations. To avoid this, a maximum of $N_{\max}$ stations are served from the busy list between polls of the clear list. The maximum time to cycle once through the clear list is then

$$c(e + N_{\max} f) \tag{1}$$

where c is the number of clear stations, e is the polling overhead for sending an empty packet, and f is the time required to transmit a full (maximum length) packet, including the polling time. However the maximum time to cycle once through the busy list becomes

$$(e + N_{\max} f) b / N_{\max} \tag{2}$$

where b is the number of busy stations, compared with

$$bf + ce \tag{3}$$

for standard round robin.

As originally proposed, ERR removes backlogged stations from the main "clear" round robin list when it adds them to the "busy" list. Comparing (2) and (3) shows that when $b > N_{\max}c$, busy stations are actually served less often under ERR than under round robin, rather than more. This condition says that the busy list contains significantly more entries than the clear list, due to network congestion. This phenomenon may be avoided by simply reverting to simple round robin service in this case. However it is exactly at these times that the greater efficiency of ERR is required. An alternative is to leave busy stations in the "main" list when they are added to the busy list. (The term "clear" list is no longer appropriate.) When the busy list is short, this will make minimal difference to the polling order. However, it will ensure that the majority of polls are still sent to busy stations, even when the busy list is long, which improves the network throughput.

When a station is removed from the busy list, it should be shifted to the end of the main list, since it is the station with the lowest probability of having data to transmit (zero probability).

4 Least-recently-used ERR

The algorithm hinted at in the previous section can be defined very simply in terms of a single list in which station are sorted in order of how recently they have been polled. This algorithm, given as Algorithm 2, will be called least-recently-used embedded-round-robin (LRU-ERR). Note that this algorithm is much simpler than Algorithm 1, as the data structures it uses are much simpler.

To see how LRU-ERR achieves the objectives set out above, note first that when there are no "busy" stations, it becomes a simple round robin scheduler. This is because the first `IF` always fails, and the station selected is `callsLRU[1]`, the least recently polled station. This is the next station in round robin order. This provides the "outer" round robin loop of ERR.

Next, consider the case when the system is lightly loaded, so that every station is polled frequently, and the `ELSE IF` always fails. The "busy" stations will be polled in LRU (round robin) order, which corresponds to the inner, or embedded, round robin loop of ERR.

The round robin polling of "busy" stations will continue until it is interrupted by one of two events. Unless the `ELSE IF` succeeds, all packets in all queues will be served, causing the "busy" stations one by one to become "clear". When all stations are clear, `found` will not be set by the `FOR` loop, causing the final `IF` to trigger polling of the next "clear" station in round robin order. If the inner round robin cycle always ends this way, then the next station to report `more_data` will be the only "busy" station, and thus the inner loop will poll it until it has no more packets. That implies that if `thresh` is so large that the `ELSE IF` always fails, then LRU-ERR becomes exhaustive round robin.

The `ELSE IF` prevents LRU-ERR from degenerating to exhaustive round robin by interrupting the inner loop when it is in danger of causing another station to receive bad service. The least recently polled station is in the most imminent danger; if a packet had arrived immediately after its last poll, then it must be polled again within the "good service" time, t_G. Moreover, it must be polled in time for all fragments of the original application layer packet to be received before t_G. For this reason, the threshold used is reduced by a

```
Variables:
    calls         = number of current connections
    callsLRU[]    = array of calls in order of most
                    recent access, [1] the earliest
    meanClearTime = mean time required to poll a
                    clear station
    goodSrvThresh = maximum desired delay
    margin        = "safety margin" (See text.)
=========================================================
thresh := goodSrvThresh - margin
found := false
FOR i := 1 to calls
    IF (callsLRU[i] is "busy")
        found := i
        BREAK from FOR loop
    ELSE IF (time since callsLRU[i] polled > thresh)
        found := 1
        BREAK from FOR loop
    ELSE
        thresh := thresh - meanClearTime
    END IF
END FOR

IF (found = false)
    found := 1
END IF

poll station for  callsLRU[i]
move  callsLRU[i]  to end of  callsLRU
```

Algorithm 2. One iteration of least-recently-used embedded round robin

margin, t_M, which may be tuned to give good performance.

In the implementation used here, `meanClearTime` was taken to be the time required for a null poll. For that case, the `ELSE IF...ELSE` can be taken outside the loop, since the least recently polled station is the only one which can cause it to succeed.

5 Performance Results

In order to evaluate the performance of LRU-ERR, it was simulated and compared with ERR, round robin (RR) and exhaustive RR. The simulation set up consists of a single cell infrastructure WLAN. The polling mechanism is essentially that defined in the IEEE 802.11 standard[2] for the "contention free" mode. Although the standard requires that this mode be punctuated by periods of contention based operation, this was omitted in our simulations.

Thirty stations continuously transmit VBR video on the uplink, and the downlink is

Table 1. Parameters used in simulations.

Parameter	value
Good service threshold, t_G	75 ms
Packet expiry timeout, t_e	500 ms
Time to poll an idle station	0.456 ms
Time to transmit a maximum length packet (including polling)	2.83 ms

idle. The lengths of the video frames were taken from the real MPEG traces of several standard video sequences, as used in Reference 18. The 30 sequences were chosen randomly (with replacement) from the 20 available sources, and the starting points within the sequences were chosen randomly. The overall network load was set by scaling the frame sizes by a factor of α, which was significantly less than 1 to model the higher compression needed for wireless transmission. Video frames were segmented into packets of size less than or equal to 2312 bytes, as specified in the 802.11 standard.

The raw bit rate was 7.5Mbps. Packets delayed more than $t_e = 500$ ms were discarded by the remote stations. The MAC header and inter-frame spacing were set according to the IEEE 802.11 standard[2]. Other system parameters are given in Table 1.

The performance of LRU-ERR was tested under two assumptions for the arrival process of video frames. First, the algorithm will be studied when frames arrive periodically at a rate of 25 frames/sec, after which Poisson arrival will be assumed.

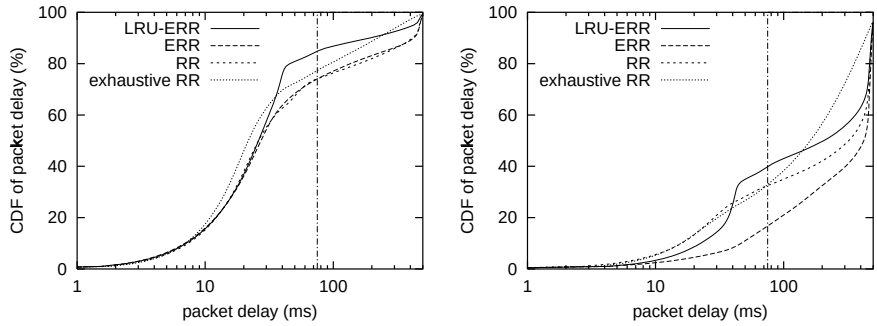
5.1 Periodic frame arrivals

For average loads of $\alpha = 0.3$ and 0.33, ten independent simulations were performed. Since different video traces were used for each simulation, the performance of each run was substantially different. The packet delays from all runs for each average load were combined and their cumulative distribution functions (CDFs) are shown in Figure 1. For comparison, the results for the same traffic are shown for the original ERR[17], standard round robin and exhaustive round robin.

When arrivals are periodic with period T, the scheduler can know that the first poll of a station after an interval of T will not be a null poll. Thus there is no throughput benefit associated with having `thresh` of Algorithm 2 any larger than T. In these experiments the margin, t_m, was set such that `thresh` was marginally above $T = 40$ ms.

The value of $N_{\max}$ used for which the original ERR was 6, which was empirically found to maximise the proportion of packets received within the good service time. The proportion of packets served by LRU-ERR within the good service threshold was clearly greater than all three other polling strategies.

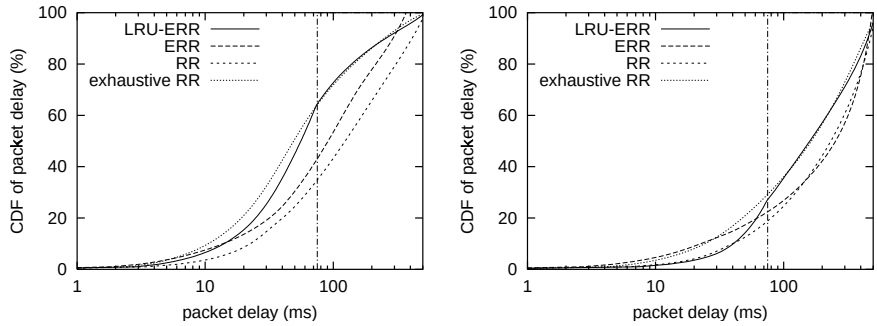
LRU-ERR attempts to maximise the proportion of packets served within time `thresh` by placing a higher priority on packets about to exceed that time. Because this must increase the delay of the other packets, it is expected that there will be a sharp rise in the CDF immediately before `thresh`, which is indeed observed.



(a) Relative load $\alpha = 0.3$.

(b) Relative load $\alpha = 0.33$.

Figure 1. Cumulative distribution of packet delays for $t_G = 75$ ms with periodic arrivals.



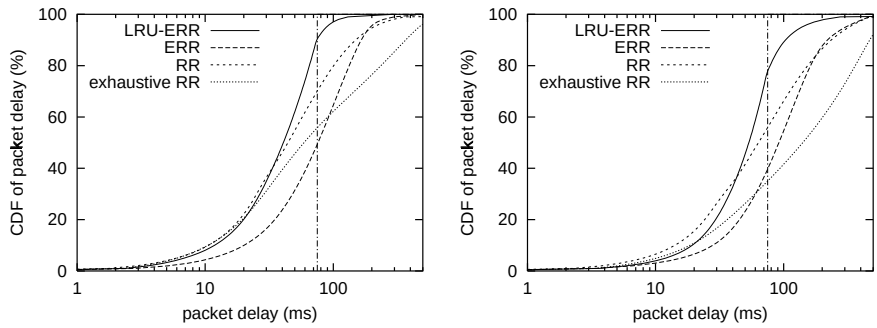
(a) Relative load $\alpha = 0.3$.

(b) Relative load $\alpha = 0.33$.

Figure 2. Cumulative distribution of packet delays for $t_G = 75$ ms with Poisson arrivals.

5.2 Poisson arrivals

When video frames arrive according to a Poisson process rather than periodically, the performance of all four scheduling disciplines deteriorates, but exhaustive RR suffers the least. As a result, both exhaustive RR and LRU-ERR perform similarly, and are considerably better than ERR and RR. This can be seen in Figure 2. For these experiments, $N_{\max} = 3$ was found to be optimal for ERR. However, it is well know that exhaustive round robin suffers from fairness problems in the presence of a small number of very heavy sources. This case will be considered in the following section.



(a) Relative load $\alpha = 0.225$.

(b) Relative load $\alpha = 0.25$.

Figure 3. Cumulative distribution of packet delays for lightly loaded stations, with $t_G = 75$ ms, Poisson arrivals and asymmetric loads.

5.3 Asymmetric load

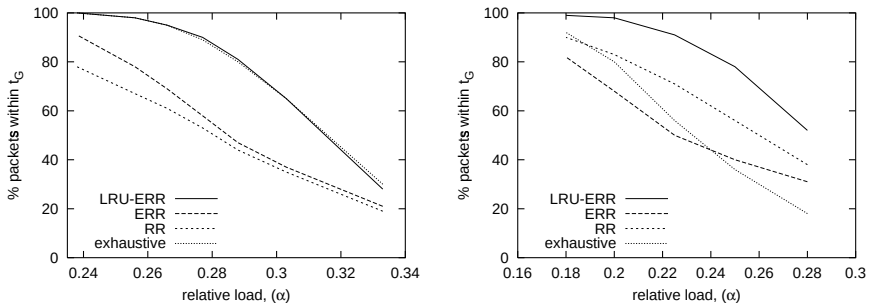
In order to test the fairness of LRU-ERR, it was simulated in a network of 30 nodes, of which one produced 10 times the traffic of the others. The delay of packets from the lightly loaded stations was measured and its CDF is shown in Figure 3, when the lightly loaded stations have $\alpha = 0.225$ and 0.25. Under exhaustive round robin, the heavily loaded station has a dramatic adverse effect on the remaining stations. In contrast, the round robin nature of LRU-ERR distributes bandwidth fairly, as well as efficiently, causing the performance of the majority of stations to be significantly higher than for the other three disciplines.

The proportion of packets delivered within the good service time, $t_G = 0.75$ ms, is shown in Figure 4 for a range of loads for both symmetric and asymmetric loads. This again shows that both LRU-ERR and exhaustive round robin effectively eliminate null polls, and consequently give high efficiency for symmetric loads, but that LRU-ERR avoids the unfairness problems inherent to exhaustive round robin.

6 Conclusion

Least-recently-used embedded-round-robin (LRU-ERR) has been presented as an enhancement to the original ERR recently proposed for IEEE 802.11 wireless LANs. This algorithm retains the benefits of ERR demonstrated for light to moderate loads for periodic traffic arrivals, while at the same time improving its performance under high loads and for bursty arrivals.

Exhaustive round robin is optimal for avoiding the waste caused by polling idle stations, and for Poisson frame arrivals and symmetric traffic, LRU-ERR achieves performance of the same high standard. However, LRU-ERR outperforms exhaustive round robin under conditions of asymmetric load.



(a) Symmetric load

(b) Asymmetric load

Figure 4. Proportion of packets (from lightly loaded stations) received within $t_G = 75$ ms with Poisson arrivals.

References

1. G. Anastasi, L. Lenzini, E. Mingozzi, "MAC protocols for wideband wireless access: evolution towards wireless ATM," IEEE Personal Commun. , 53–64 (1998).
2. P802.11, Wireless medium access control (MAC) and physical layer (PHY) working group, IEEE 802.11 draft standard, "Wireless medium access control (MAC) and physical layer (PHY) specifications," IEEE Stds. Dept., July 1996, D5.
3. D. Ferrari, D. Verma, "A scheme for real-time channel establishment in wide-area networks," IEEE J. Select. Areas Commun. **8(4)**, 368–379 (1990).
4. N.R. Figueira, J. Pasquale, "Leave-in-time: a new service discipline for control of real-time communications in a packet-switching network," in *Proc. ACM SIGCOMM'95*, August 1995, pp. 207–218.
5. D. Verma, H. Zhang, D. Ferrari, "Delay jitter control for real-time communication in a packet switching network," in *Proc. IEEE TriCom '91*, April 1991, pp. 35–43.
6. A. Demers, S. Keshav, S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM 89*, Sept. 1989, vol. 19, no. 4, pp. 1–12.
7. A.K. Parekh, R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," IEEE/ACM Trans. Networking **1(3)**, 344–357 (1993).
8. S.J. Golestani, "A self-clocked fair queuing scheme for broadband applications," in *Proc. IEEE INFOCOM '94*, 1994, pp. 636–646.
9. D. Saha, S. Mukherjee, S.K. Tripathi, "Carry-over round robin: a simple cell scheduling mechanism for ATM networks," IEEE/ACM Trans. Networking **6(6)**, 779–796 (1998).
10. M. Shreedhar, G. Varghese, "Efficient fair queueing using deficit round robin," IEEE/ACM Trans. Networking **4(3)**, 375–385 (1996)
11. D.A. Hayes, M. Rumsewicz, and L.L.H. Andrew, "Quality of service driven packet scheduling disciplines for real-time applications: looking beyond fairness," in *Proc. IEEE INFOCOM'99*, 1999, vol. 1, pp. 405–412.

12. J. Nagle, “On packet switches with infinite storage,” IEEE Trans. Commun. **COM-35**, 435–438 (1987).
13. N. Movahhedinia, G. Stamatelos, H.M. Hafez, “Polling-based multiple access for indoor broadband wireless systems”, in *Proc. PIMRC 95*, 1995, vol. 3, pp. 1052–1056.
14. R. Kautz, A. Leon-Garcia, “A Distributed self-clocked queueing architecture for wireless ATM networks,” in *Proc. PIMRC’97*, 1997, vol. 1.1, pp. 189–193.
15. C.-S. Chang, K.-C. Chen, “Guaranteed quality-of-service wireless medium access by packet-by-packet generalized processor sharing algorithm,” in *Proc. IEEE Int. Conf. Commun. (ICC)’98*, June 1998, vol. 1, pp. 493–497.
16. R.S. Ranasinghe, D. Everitt, L.L.H. Andrew, “Fair Queueing scheduler for IEEE 802.11 based wireless multimedia networks,” Multiaccess, Mobility and Teletraffic in Wireless Communications (MMT’99), Oct. 1999, vol. 4, pp. 241–251.
17. R.S. Ranasinghe, L.L.H. Andrew, D.A. Hayes and D. Everitt, “Scheduling disciplines for multimedia WLANs: Embedded round robin and wireless dual queue,” in *Proc. IEEE Int. Conf. Commun. (ICC) ’01*, Helsinki, Finland, June 2001.
18. O. Rose, “Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems,” in *Proc. 20th Annual Conf. Local Comp. Network*, Minneapolis, CA, Oct. 1995. pp. 397–406.