Controlled Bandwidth Allocation for Wireless Networks

Dr. Dheeraj Sanghi
dheeraj@cse.iitk.ac.in

Ankur Gupta
ankur@cse.iitk.ac.in

Nitin Gupta
guptan@cse.iitk.ac.in

Indian Institute of Technology
Department of Computer Science and Engineering
Kanpur 208 016, INDIA

Abstract

The problem of controlled bandwidth allocation has been extensively studied for wired domain. But very little attention has been paid for wired-cum-wireless network topologies. This paper focuses on issues that makes a wireless scenario different from a wired one and proposes novel mechanisms for ensuring user-defined shares to different traffic flows. We show that most of the existing scheduling policies used for fair share that work very well in wired networks fail to achieve reasonable fairness in case of wired-cum-wireless domain. A whole spectrum of control mechanisms that can be used and a detailed comparison of each of the mechanisms, with simulation results, is presented. A novel way of quantifying fairness by means of fairness numbers is presented and the mechanisms are compared based on the fairness number they achieve.

Keywords: Wired-cum-wireless networks, Bandwidth control mechanisms, Fairness number and Goodput.

1. Introduction

The Internet carries traffic of widely varying bandwidth requirements. In order to provide service differentiation between these different traffic classes, network bandwidth needs to be managed efficiently. The mechanism that facilitates this management is the buffer management and scheduling schemes used by routers or gateways.

With the advent of novel applications like video conferencing, Internet telephony etc the need for having a control over link share has become a must. In fact, control over Internet traffic is so important that even if we have to sacrifice a small fraction of bandwidth for it, it is acceptable. Up till now, focus has been on providing link sharing in wired networks. How well will these mechanisms perform in wired-cum-wireless networks has not been studied. Our initial experiments showed that most of the existing scheduling policies used for fair share that work very well in wired networks fail to achieve reasonable fairness in case of wired-cum-wireless domain. This motivated us to look at the issues that makes the wireless case different from the wired one. In a wireless environment even if two mobile nodes are in communication range of the base station, it is possible that the delay encountered in successfully delivering the packets to the nodes be different. Moreover, these differences in delays are dynamic (due to mobility of nodes as well as local channel errors). As a result differences in RTT values result [1] which fluctuates a lot which an intermediate router fails to take into account. These differences lead to unfair allocations.

Our simulation results showed that scheduling policies alone are not enough for controlled link sharing. We have used scheduling policies in conjunction with either (1) dropping packets or (2) delaying acknowledgments or (3) feedback from bandwidths used by acknowledgments to control TCP traffics. These mechanisms achieve a high degree of fairness. In order to quantify fairness, we have defined a quantity called fairness number. The higher the fairness number, the more the mechanism is successful in achieving fairness. Note that, here, fairness doesn't necessarily mean equal bandwidth to flows.

[1] For reasons discussed later, we assume one-way TCP flows from wired domain to wireless domain.

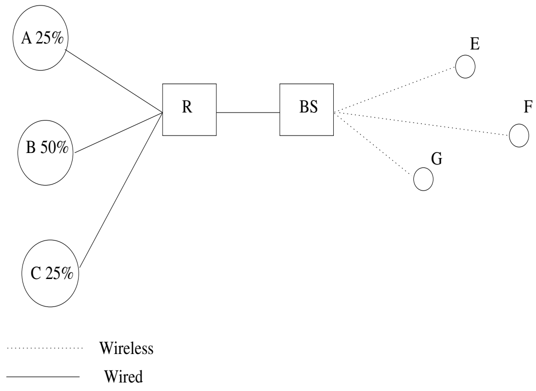


Figure 1. A typical scenario

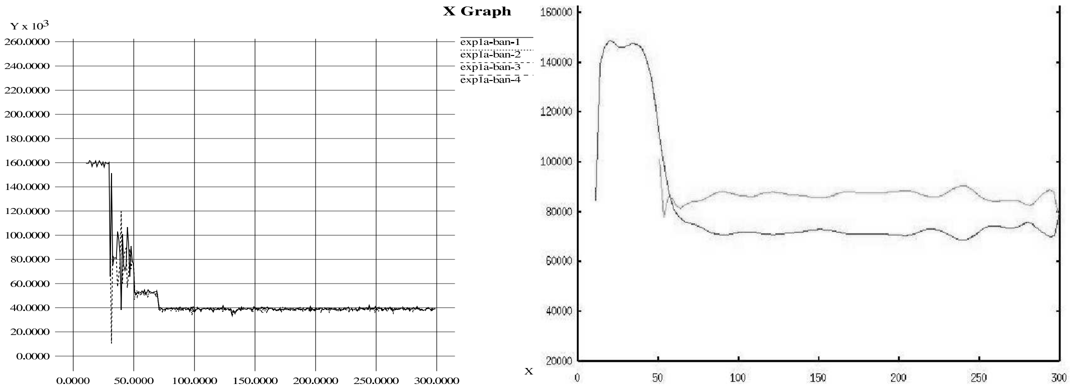


Figure 2. Throughput Vs Time (1) Wired Topology (2) Wired-Wireless Topology

2. Problem Description

We consider a set of mobile nodes and an access point connected to a wired network. The mobile nodes are in the communication range of the access point. We consider TCP flows (UDP flows are also considered in some cases) with varying RTT values and assume one-way traffic from wired to wireless side. The other scenario, in which traffic goes from wireless to wired side is less interesting as the bottleneck link is the wireless link and our scenario reflects a more realistic scenario as, generally, the mobile nodes request information from wired network where request packets are of small size and the data packets from the wired network are large. The mobile nodes are at different distances from base station and have different error probabilities. We desire a user-defined bandwidth share for each of the flows. We assume that 802.11 is used as the underlying MAC layer protocol on the wireless side. Figure 1 shows a possible bandwidth allocation to three TCP flows.

In order to motivate the problem, we simulated a simple wired-cum-wireless topology where two wired nodes send TCP traffic to different mobile nodes that are at different distance from the base station and have different error probabilities. The scheduling policy used was DRR [7]. Figure 2 shows that DRR which performs quite well in case of a wired topology fails to do fair allocation in case of wireless. Sally Floyd *et al.* [3], proposed link-sharing guidelines that can be used for controlled bandwidth allocation. We implemented the guidelines and the scheduler [2] they used and found similar discrepancy in fair bandwidth allocation as in the case of DRR.

[2]The TCP flows that we considered always had sufficient demand, hence a simplistic version of their scheduler was implemented.

3 Related Buffer Management and Scheduling policies

Many rate based message scheduling schemes used at runtime packet transmission have been proposed/investigated, among which (1) Virtual Clock and its variations Self-Clocked Fair Queuing (SCFQ) [4] and Leap Forward Virtual Clock (LFVC) [9], and (2) Generalized Processor Sharing (GPS or termed elsewhere Weighted Fair Queuing) and its realistic implementations Packet-by-Packet Generalized Processor Sharing (PGPS) and Worst-case Fair Weighted Fair Queuing (WF2Q) have received the most attention. Although realized with different approaches, all the above schemes aim to develop service disciplines that can provide worst-case performance guarantees (e.g., bounded delay and minimum effective service rate) and fairness (in some sense) to connections, and/or prevent interference among connections. Moreover, all the above schemes are centered upon the notion of service share or requested service rate: each connection at a node is assigned a service share or can request a service rate. The message scheduler then serves connections in proportion to the service share (or rate), independent of the queuing load presented by the connections.

Deficit Round Robin (DRR) [7] is another scheduling policy that serves flows in a round robin fashion. DRR works by associating a time quantum with each queue that is to be serviced. Backlogged queues are serviced in a round-robin order, and on each round, the amount of data sent from a queue cannot exceed the time quantum for that queue. If a packet cannot be completely serviced on a round without violating the quantum requirement, its transmission is deferred to the next round, but the portion of the quantum for the queue that was unused in the current round is saved, and can be added to and reused with the quantum for the next round (hence the term "deficit" round-robin).If a queue ever becomes empty, then it cannot carry over any deficit, thereby ensuring that the accumulated deficit can never exceed the length of a maximum sized packet.Note that although DRR is fair in terms of throughput for wire-line networks, it lacks any reasonable delay bound.

The random drop notion of RED [2] was motivated by the systematic discrimination against some connections by Drop Tail gateway in a TCP/IP network with strongly periodic traffic. The decision to accept an incoming packet in RED is based on whether the low pass filtered average queue size exceeds or conforms to predetermined thresholds. As long as the average size is between min_th and max_th, an arriving packet is dropped with probability that increases linearly with average queue size. Flows are penalized roughly in proportion to their buffer occupancy. While RED tries to ensure per flow fairness without per flow state having to be maintained in the router, it turns out to be unfair to low rate TCP flows. The randomness of the drop decision combined with the dropping of the arriving packet could cause the congestion window of a flow to be halved irrespective whether it exceeds its fair share of the buffer. Flow RED was proposed to alleviate some of RED's limitations. FRED [5] monitors buffer occupancies on a per flow basis with min and max threshold being defined for each flow. This rigid allocation of thresholds is not effective in managing excess bandwidth when the network is not congested.

Apart from this, there are schemes which have higher packet drop rates with longer queues. The instantaneous queue size is monitored for each flow. These schemes unlike RED and its variants do not drop the incoming packet. When the total buffer occupancy is exceeded, a packet is chosen to be discarded from a connection whose current occupancy exceeds its allocation. These do not try to predict the onset of congestion by monitoring the queue size and use the instantaneous queue size to determine the drop queue. By waiting until congestion occurs, it is highly likely that multiple packet drops will occur with high temporal locality leading to degradation of throughput of a source or flow synchronization. So it is better to predict the onset of congestion and also maintain per flow information (where possible as in our case) for taking dropping decisions.

4 General Principles

We now study some general principles that all control mechanism use. Fairness can be achieved either by using (1) effective packet scheduling policy or (2) buffer management policies (3) or feedback from up-link traffic to control down-link traffic. Each of these is elaborated below.

4.1 Scheduling Policy

Scheduling Policy is the most essential part of a bandwidth allocation control mechanism. The Scheduling policy can preferentially select packets of some flows to provide them greater bandwidth shares so that their fair share can be achieved. A variety of scheduling policies have been proposed depending upon the network requirements like FIFO, policies that emulate GPS like WFQ [8] and SFQ [6] , DRR [7] etc.

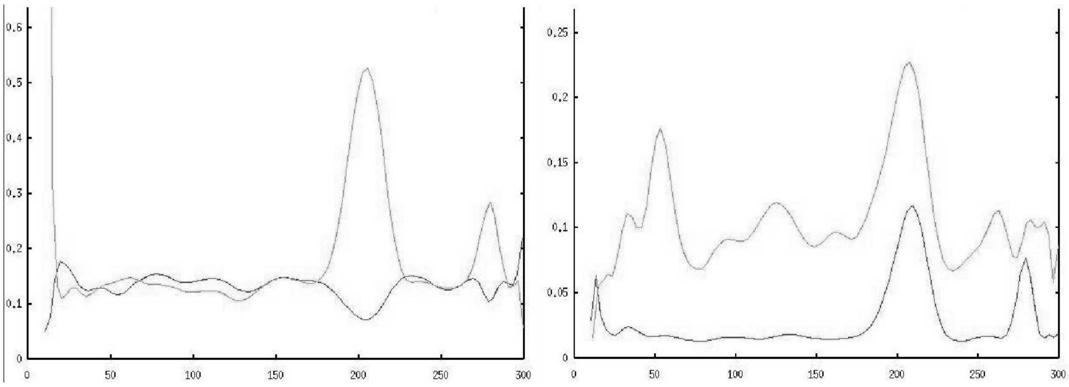


Figure 3. Forward Delay Vs Time **Backward Delay Vs Time**

4.2 Buffer Management Policy

The buffer management policy can ensure controlled bandwidth allocations in two ways.

1. It can preferentially drop (possibly in a probabilistic way) some data packets of over-limit flows to control shares when buffers are full.

2. The policy can ensure that at each instant the buffer space occupied by each flow is proportional to its desired bandwidth by dropping packets even when the buffer is not full.

4.3 Feedback from up-link and Regulation of up-link traffic

This control mechanism is applicable to the one-way TCP scenario we have considered. Since we have assumed that the traffic is primarily from the wired side to the wireless side which is usually the case, so acknowledgments constitute the up-link traffic. Therefore a possible control mechanism is to monitor the up-link traffic and provide feedback to the scheduling policy on the down-link side. This is called Feedback from the up-link side. An alternative approach is to control the flow of acknowledgments on the up-link side in order to control the down-link flow. This is called Regulation of up-link traffic. The following paragraph discusses the motivation behind this control mechanism.

We collected extensive data for two ftp flows to two different mobile nodes with DRR as the scheduling policy for a simple wired cum wireless scenario. The topology on the wired side was identical for both the flows. However, on the wireless side the nodes were at different distances from the base station. DRR was successful in making the forward delays of packets of the flows (which were going to two different mobile nodes with different error probabilities) equal. However, on the up-link side because of MAC layer induced differences (the nodes were at different distances and under different error probabilities) the delays experienced by acknowledgments were different. So the two flows had different RTTs and hence we observed a difference in the bandwidth allocation. On the up-link side the traffic was underflow since we only had acknowledgments. From this we concluded that in case of TCP flows we can also take into account the flow of acknowledgments on the up-link side for controlling bandwidth shares. Figures 3 plot the backward delay versus time and forward delay versus time for the two flows.

There are various possible ways in which regulation and feedback can be achieved. Since the buffer on the up-link side is never full so scheduling policy either can delay/drop these acknowledgments even when the bandwidth is available. This is the simplest way of regulation of up-link traffic. The second possibility is to get some feedback related to the rate of acknowledgments on the up-link side and use it while using control mechanisms on the down-link side. Figure 5 illustrates the various control mechanisms.

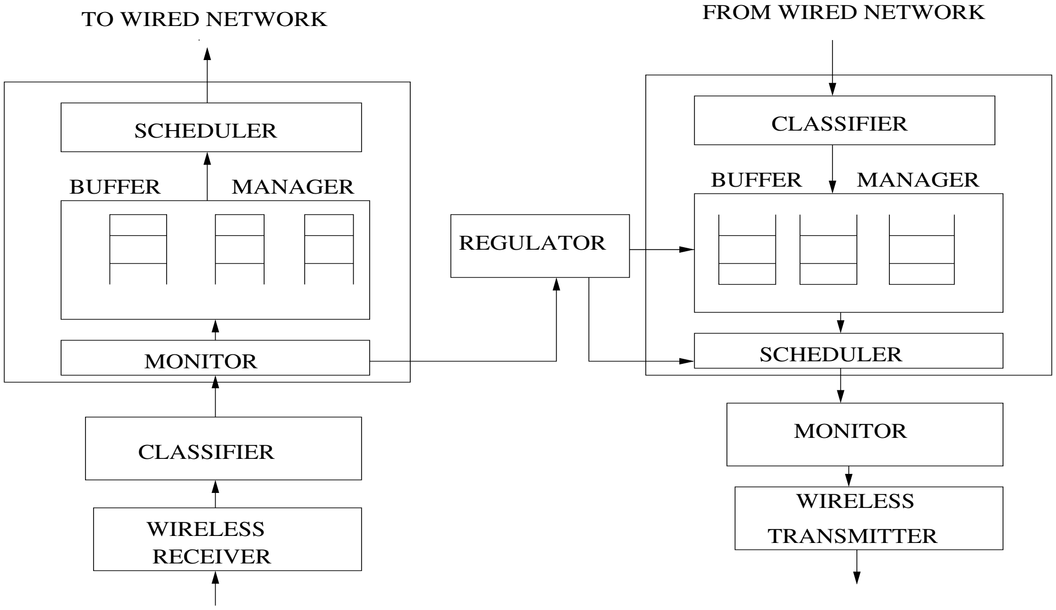


Figure 4. A: Up-link Side Control Mechanism B: Down-link Side control mechanism

5 Algorithms

In this section we discuss various scheduling policies, buffer management algorithms and feedback/regulation algorithms.

5.1 Inter-Arrival Time Packet Scheduling (ITS)

The algorithm[3] calculates the expected inter-arrival of packets for each flow that ensures the given bandwidth distribution. Next compare this with the actual inter-arrival time of each flow. Calculate the difference between the latter and the former. A positive difference means the flow is using less than its allotted share (i.e., it is under-limit) and a negative difference means the flow has used more than its share (i.e., it is over-limit). While scheduling, schedule packet from that active flow which is lagging the most. To avoid unnecessary jitters, instead of using instantaneous difference between actual and expected inter-arrival time, the scheduler uses exponential weighted (EWMA scheme) averages to select the flow. The complete algorithm is presented below.

Parameters :

τ	weight factor used in EWMA
LinkBan	Total link bandwidth for the bottleneck link

Variables :

$LastSent_i$	time at which the packet of flow i was last dequeued
$ActiveFlow_i$	set to 1 if the ith flow is active at the instant
$AvgDiff_i$	represents the average difference value of flow i
w_i	fraction of bandwidth for the ith flow
ActiveFlows	set of flows active over past t_0 seconds
f	expected inter-arrival times according to allocations
CurrentActiveFlows	set of flows that have at least one packet in the queue at the instant

[3]This algorithm is based on ideas from [3].

Algorithm

1. Whenever a new packet arrives, enqueue it in its proper queue
2. Dequeue packet belonging to *i th* flow where $Max_{j \epsilon CurrentActiveFlows}(AvgDiff_j) = AvgDiff_i$
3. After Dequeuing, do the following

f	$\leftarrow$	$\frac{packetSize*w_i}{(\Sigma_{j \epsilon ActiveFlows} w_j)*LinkBandwidth}$
t	$\leftarrow$	currentTime - $LastSent_i$
diff	$\leftarrow$	t - f
$AvgDiff_i$	$\leftarrow$	τ * $AvgDiff_i$ + (1-τ) * diff
$LastSent_i$	$\leftarrow$	currentTime

5.2 Probabilistic Proportional drop

This scheme attempts to share the buffer among the flows in proportion to the desired bandwidths. Parameters, desired buffer occupancy b_i, and the current buffer occupancy, q_i, are maintained for each flow i. Whenever the buffer is near full, a packet is dropped with some probability ζ from that flow for which the buffer occupancy factor $\frac{q_i}{b_i}$ is maximum.

Parameters :

ζ	probability with which to drop a packet
ψ_0	threshold buffer load factor
b_i	$\frac{B*(desired\ bandwidth\ for\ flow\ i)}{total\ bandwidth}$

Variables :

$AvgBuff$	average buffer size over the past few seconds
ψ	current buffer load factor
BuffSize	total buffer size
q_i	current buffer occupancy for flow i

Algorithm :
Whenever a packet arrives do the following:

1. $\psi \leftarrow \frac{AvgBuff}{BuffSize}$
2. If $\psi \geq \psi_0$, then
   - *i* where $Max_{j \epsilon CurrentActiveFlows}(q_j/b_j) = q_i/b_i$
   - Drop packet of flow i with probability ζ

Note that PPD is a buffer management policy. Scheduling decisions are made on basis of the scheduling policy used.

6 Control Mechanisms

We now describe a series of control mechanisms that we actually implemented. Our preliminary simulations indicated that ITS alone is not sufficient for controlling over-limit traffics. Therefore, all the control mechanisms presented below use some other measures to obtained the desired effect.

6.1 Stochastic Inter-arrival Time Packet Scheduling (SITS)

SITS uses the scheduling policy as ITS and uses a buffer management policy that we describe now. The motivation for the policy came from the pro-active measures taken by RED to avoid buffer overflow. This can be exploited to probabilistically drop packets of the over-limit flows even before the buffers are full. SITS is essentially same as ITS except that when we enqueue a new packet in its flow queue, we do the following:

Parameters :

ζ	probability with which to drop a packet
ψ_0	threshold buffer load factor
ϵ	drop reduction factor

Variables :

$AvgBuff$	average buffer size over the past few seconds
ψ	current buffer load factor
BuffSize	total buffer size

1. $\psi \leftarrow \frac{AvgBuff}{BuffSize}$
2. If $\psi \geq \psi_0$ and some flow is over-limit, then
   - *i th* flow where $Min_{j \epsilon CurrentActiveFlows}(AvgDiff_j)$ = $AvgDiff_i$
   - Drop packet of flow i with probability ζ
   - $AvgDiff_i \leftarrow AvgDiff_i$ * ϵ

The intuition behind the step 2 above is that a flow that has negative $AvgDiff_i$ value is over-limit and has been using more than its share. So we penalize that flow by dropping its packet. If all the flows have positive $AvgDiff_i$ value then the router is not congested and therefore no dropping of packets is required. Also note that after dropping packet of flow i, we reduce the AvgDiff for flow i, so that multiple packets for that flow are not dropped simultaneously (in our simulations, ϵ = 0.5).

6.2 ITS_PPD

ITS_PPD is a combination of the ITS scheduling policy and the PPD buffer management policy.

6.3 ITS_FEED

ITS_FEED is only for controlling TCP traffic. Here the router computes AvgDiff (refer 5.1) for the TCP flow and also AvgDiff for acknowledgments. It schedules packet based on the feedback (in the form of the AvgDiff of acks) and the AvgDiff for the TCP flows. The complete algorithm is presented below:

Parameters :

γ	scaling factor to be multiplied to acks' AvgDiff

Variables :

$TcpAvgDiff_i$	$AvgDiff_i$ for the tcp packets of flow i
$AckAvgDiff_i$	$AvgDiff_i$ for the acks of flow i

Algorithm :

1. $GloablAvgDiff_i \leftarrow TcpAvgDiff_i$ + γ * $AckAvgDiff_i$
2. Schedule packet of flow i, where $Max_{j \epsilon CurrentActiveFlows}(GlobalAvgDiff_j)$ = $GlobalAvgDiff_i$

6.4 ITS_DACK

ITS_DACK combines ITS with delaying of acks. ITS is used for scheduling both acks and tcp packets. The acknowledgments of over-limit flows are delayed whenever such flows are too over-limit.

Parameters :

ψ_0	Threshold for over-limit flows
ϵ	drop reduction factor

Variables :

$TcpAvgDiff_i$	$AvgDiff_i$ for the tcp packets of flow i
$AckAvgDiff_i$	$AvgDiff_i$ for the acks of flow i

Algorithm :

1. For tcp packets, schedule using ITS.
2. For acks, select the flow i whose acks needs to be scheduled (as governed by ITS).
3. If $AckAvgDiff_i \geq \psi_0$ then
   - $AvgDiff_i \leftarrow AvgDiff_i * \epsilon$
   - Do not schedule the ack
4. else Schedule the ack.

7 Simulation Details and Results

Simulations were carried using the NS-2 [1] simulator. We simulated wireless-cum-wired topologies with traffic from the wired side to the wireless side. The experiments conducted can be broadly categorized as follows:

1. Delays on the wired side were different and fair allocation was sought.
2. Wireless nodes were having different errors and fair allocation was sought.
3. The Delays on both sides were equal and a non uniform distribution according to some weights was sought.
4. Miscellaneous experiments with different delays on both sides and non uniform allocation was sought.

The above experiments were carried out for each of ITS_FEED, ITS_DACK, SITS and ITS_PPD mechanisms. For TCP flows, FTP application and for UDP flows, CBR traffic was used. In all experiments there are two domains: wired and wireless with a base station in between. The wireless bandwidth is the bottleneck link. The TCP sources implement TCP Reno with the Fast Retransmit and Recovery algorithms. The TCP maximum window size is set to be sufficiently large so that there is always sufficient demand. The MAC layer on the wireless side was 802.11. The wireless error rates varied with distance. The time for simulations was 300 sec.

7.1 Performance Measures

We evaluate the performance of the various schemes using the goodput and a quantity (described below) called, Fairness Number. Goodput defined as the average rate of data received by the application. The total goodput is obtained by summing the goodput obtained for all connections.

In order to measure how good a control mechanism performs, we define a mathematical quantity called Fairness Number. Fairness number is calculated as follows:

Variables:

χ_i	actual goodput obtained for flow i
ω_i	desired bandwidth fraction for flow i
N	total number of flows

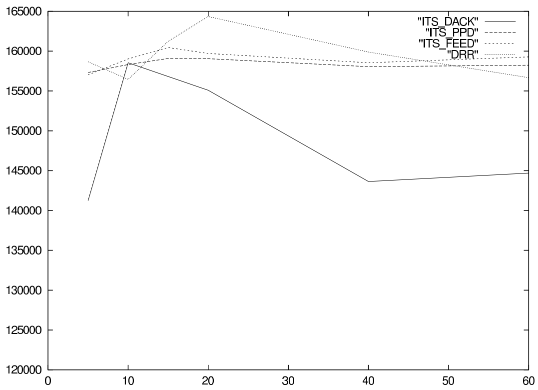
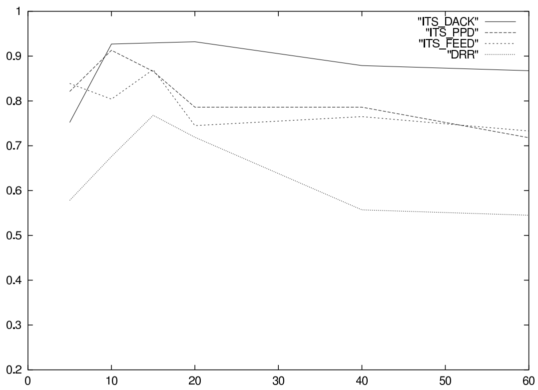


Figure 5. Total Goodput Vs Buffer Size



Fairness Number Vs Buffer Size

Fairness Number Computation:

$$
\begin{array}{lll}
\text{avg} & \leftarrow & \frac{\Sigma_i(\frac{x_i}{\omega_i})}{N} \\
\text{sd} & \leftarrow & \sqrt{\frac{\Sigma_i(\frac{x_i}{\omega_i} - avg)^2}{N}} \\
\Upsilon \text{ (fairness no.)} & \leftarrow & 1 - \frac{sd}{avg}
\end{array}
$$

As is obvious, the Fairness Number Υ always lies between 0 and 1. The higher the fairness number of a control mechanism, the more fair the mechanism is.

7.2 Performance Results

We did a comparison study of all the control mechanisms and calculated the total goodput and fairness number for each scheme. The figure 5 show the variation of total goodput and fairness with buffer sizes. The range of buffer sizes is chosen to capture regions in which queuing delay is insignificant in comparison to the RTT as well regions in which the queuing delay is significant and is comparable to RTT values. We had three TCP flows (FTP application). The aim was to allocate 50 percent, 25 percent and 25 percent shares respectively to the flows. The delays on the wired side as well as the wired side were equal.

As seen from the figures above , the control mechanisms proposed perform better in terms of fairness numbers than DRR which acquires high fairness in case of wired networks. At the same time, they donot result in any appreciable deterioration of goodput values. The graph also shows their relative performances.ITS_DACK was able to achieve the desired allocation with very high fairness numbers. However, the total goodput obtained deteriorated in this scheme as compared to other schemes which is expected as delaying acknowledgments will result in lowering of goodput. ITS_PPD gave very reasonable goodput values but fairness deteriorated when we had very large buffer sizes. This again can be explained by the fact that with very large buffer sizes the drop threshold is not achieved too often and the policy reduces to ITS which by itself does not result in good fairness numbers. ITS_FEED performs reasonably well with high fairness and throughputs.

There were some other secondary observations as well. It was seen that in case of ITS_DACK, increasing the weight parameter resulted in smooth goodput versus time graphs and therefore, less jitter. However, it reduced the sensitivity of the dropping mechanism, as a result the delay threshold has to be incresed. The delay threshold, however cannot be increased indefinitely. If it becomes too small the packets of the under limit flows also start getting dropped aggressively which lowers goodput drastically. So we must strike a balance between the delay threshold and the weight. In our case, the weight was set to be 0.91 which gave reasonably smooth variation of goodput versus time.

We now present Goodput vs Time graphs for some simulation cases. These cases are as follows:

1. There are three one way TCP traffics (from wired to wireless side). Error probabilities for the the receivers (wireless nodes) are same by keeping them stationary and at same distances. The path lenghts and link delays in the wired domain are also same for all the flows. Desired bandwidth distribution ratio is 2:1:1. The policies used are ITS_DACK and ITS_PPD respectively. (refer figure 6-7).

2. There are two one way TCP traffics and one UDP traffic. Error probabilities for the the receivers (wireless nodes) are same. The path lengths and link delays on the wired side are also same. The desired bandwidth distribution is 2:1:1 and policy used is ITS_PPD. (refer figure 8).

3. We have two one-way TCP traffics. Error probabilities for the the receivers (wireless nodes) are significantly different with equal path lengths and link delays on the wired side. We use the ITS_DACK mechanism and equal bandwidths are sought. (refer figure 9).

4. There are two one-way TCP traffics. Error probabilities for the the receivers (wireless nodes) are same but wired links to which senders are connected have very different delays (2:52). We use ITS_DACK and equal goodputs are desired. (refer figure 10).

5. We have three one-way TCP traffics. The wired side link delays as well as the wireless side error probabilities are different. We use SITS as the control mechanism. (refer figure 11).

As can be seen in each of these cases, the bandwidth allocations intended were achieved with high fairness numbers. SITS gave only a satisfactory result with high standard deviation. ITS_DACK resulted in very high fairness numbers and is a good policy if the buffer size is moderate (set to 20 for all the above cases).

Finally, we enumerate some interesting observations:

- ITS_DACK, ITS_PPD and ITS_FEED achieved good fairness in the case when it is desired to share the bandwidth equally.
- Non-uniform distribution for TCP flows can be achieved using either ITS_DACK or ITS_PPD. But they work only when the bandwidth distribution ratio is not too large.
- Non-uniform distribution, when both TCP and UDP flows are present, can be achieved using ITS_PPD only. Again, the bandwidth distribution ratio should not be too large.
- The mechanisms achieve good fairness for moderate buffer size. The mechanisms deviate significantly from ensuring fairness when buffer size is too small or too large.

  We conclude that ITS_DACK is a good control mechanism for bandwidth shares when we only have TCP flows. However, it has the drawback of lowering goodput although the deterioration is very small. ITS_FEED is also a good mechanism that gives high fairness and reasonable goodput values. ITS_PPD mechanism can be used even in the presence of UDP flows.

8 Conclusion

Existing wire-line policies for controlling bandwidth allocations cannot be used in the presence of wireless environments. As most of the existing mechanisms do not take into account the nature of the link to which they are forwarding packets, their efforts to ensure fairness, they fail significantly for wired-cum-wireless network topologies. We showed that scheduling policies used alone are not a good control mechanism and hence the use of feedback and regulation of up-link traffic as a strategy for controlling shares is a must. We proposed four different control mechanisms for controlling bandwidth shares in a wired-wireless topology. ITS_DACK used delaying of acknowledgments, ITS_PPD and SITS probabilistic dropping of packets and ITS_FEED a feedback from the up-link side along with the scheduling policy. These mechanisms gave better fairness numbers than DRR and Fair Queuing algorithms with almost no deterioration of goodput values.

References

[1] K. Fall and K. Varadhan. The ns Manual (formerly ns Notes and Documentation). *http://www.isi.edu/nsnam/ns/doc/*, page The VINT Project, 2002.

[2] S. Floyd and V. Jacobson. Random Early detection for congestion avoidance. *IEEE/ACM transactions on Networking*, July 1993.

[3] S. Floyd and V. Jacobson. Link Sharing and resource management models for packet networks. In *IEEE Workshop*, pages 77–82, August 1995.

[4] S. Golestani. A Self-Clocked Fair Queuing Scheme for High Speed Networks. *INFOCOM*, 1994.

[5] D. Lin and R. Morris. Dynamics of Random Early Detection. *SIGCOMM*, 1997.

[6] P. Mckenney. Stochastic Fair Queuing. In *Internetworking: Reserach and Experience, Vol. 2*, pages 113–131, January 1991.
[7] M. Shreedhar and G. Varghese. Efficient Fair Queuing with Deficit Round Robin. *SIGCOMM*, 1995.
[8] R. S. Songwu Lu, Vaduvur Bharghavan. Fair Scheduling in Wireless Packet Networks. *SIGCOMM*, September 1997.
[9] S. Suri, G. Varghese, and G. Chandranmenon. Leap Forward Virtual Clock. *INFOCOM*, 1997.

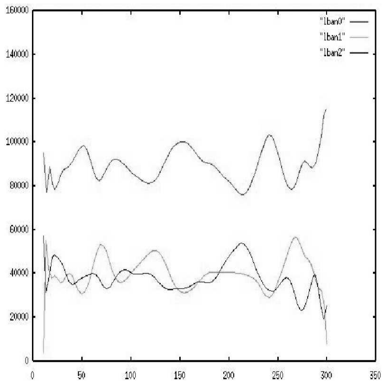


Figure 6. ITS_DACK: TCP:TCP:TCP:2:1:1

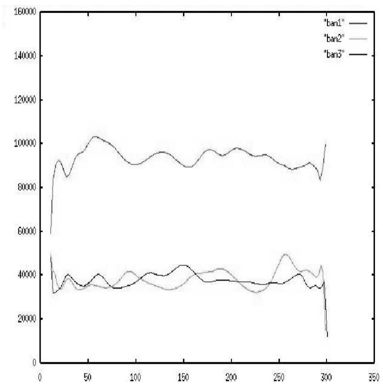


Figure 7. ITS_PPD: TCP:TCP:TCP:2:1:1

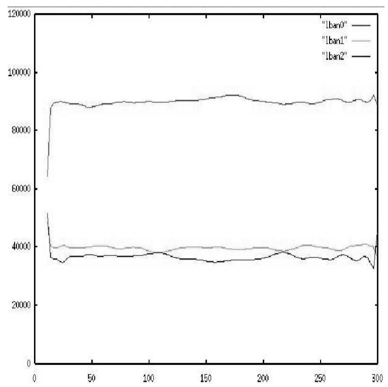


Figure 8. ITS_PPD: TCP:TCP:UDP:2:1:1

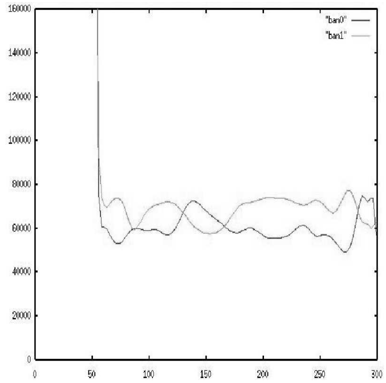


Figure 9. ITS_DACK: TCP:TCP:1:1 Different Error Probabilities

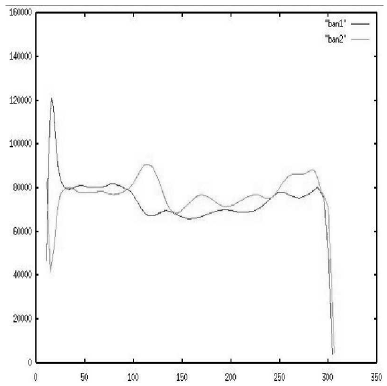


Figure 10. ITS_DACK: TCP:TCP:1:1 Different Wired Delays

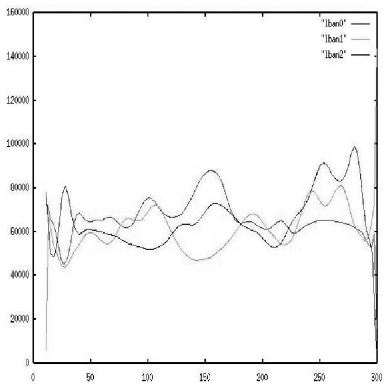


Figure 11. SITS: TCP:TCP:TCP:1:1:1