

Providing QoS to Real and Interactive Data Applications
in WiMax Mesh Networks ¹

Vinod Sharma

Department of Electrical Communication Engg.,
Indian Institute of Science,
Bangalore 560012, INDIA.

Harish Shetiya

Ittiam Systems (P) Ltd
Richmond Road,
Bangalore 560025, INDIA

¹This work was partially supported by the DRDO-IISc program on Advanced Research in Mathematical Engineering

Contents

0.1	Introduction	1
0.2	System Model	4
0.3	Routing and Scheduling for Aggregate Traffic	6
0.4	QoS for Real Time Traffic	12
0.4.1	Scheduling of CBR traffic	13
0.4.2	Scheduling of VBR Traffic	15
0.5	QoS for TCP Traffic	16
0.6	Joint Scheduling of UDP and TCP Flows	20
0.7	Simulations	23
0.8	Admission Control	25
0.9	Conclusions	28
0.10	Acknowledgement	28

Abstract We consider the problem of centralized routing and scheduling for IEEE 802.16 mesh networks so as to provide Quality of Service (QoS) to individual real and interactive data applications. We first obtain an optimal and fair routing and scheduling decision for aggregate demands for different source-destination pairs. We then present scheduling algorithms which provide per flow QoS guarantees while utilizing the network resources efficiently. Our algorithms are also scalable: they do not require per flow processing and queueing and the computational requirements are modest. We also discuss admission control policies which ensure that sufficient resources are available. We have verified our algorithms via extensive simulations.

0.1 Introduction

IEEE 802.16 standard [1], also known as WiMax has been specifically designed to provide wireless broadband access in the Metropolitan Area Network (MAN), delivering performance comparable to traditional cable, DSL or T1 offerings. In order to provide the coverage and data rates envisioned, even on uneven terrain, the use of multihop communication seems desirable. Hence WiMax supports a Mesh mode (the other mode being point to multipoint) in which unlike the traditional cellular systems, the nodes can communicate without having a direct connection with the base station.

In a IEEE 802.16d Mesh network, a node that has a direct connection to backhaul services outside the Mesh network, is termed a Mesh Base Station (MBS). All other nodes of a Mesh network are termed Mesh Subscriber Stations (MSS). In IEEE 802.16d standards these nodes are stationary, i.e., the standards do not support mobility (see however 802.16e amendment [2] to the 802.16 standard which supports the mobility). The standard specifies a centralized scheduling scheme for mesh networks. Under this scheme, the MSSs notify the MBS their data transfer requirements and the quality of their links to their neighbours. The MBS uses the topology information along with the requirements of each MSS to decide the routing and the scheduling. The MAC scheme used is TDMA and the resource allocation is in terms of time slots within a frame. The standard does not specify an algorithm for scheduling of

the slots to different MSSs; neither does it specify any routing algorithm. Scheduling and routing will have significant impact on the performance of the system and will largely decide the end to end QoS to different users.

The WiMax standard also supports a distributed scheduling scheme in which each mesh node uses the local topology, channel and traffic information to decide which channel to use. The distributed approach is simple and robust as compared to the centralized approach. However it results in lower channel utilization and will provide less control over QoS. Thus it is recommended ([11]) that the distributed scheduling be used only for unlicensed spectrum while the centralized for licensed. The standard recommends the centralized scheduling for traffic entering/leaving the mesh while the distributed scheduling for Intra-net traffic. A part of the frame can be reserved for centralized scheduling and another for distributed scheduling and these can be configured. Since most of the traffic is expected to be Internet traffic we will concentrate on centralized mode.

In the following we survey the literature on scheduling and routing for wireless networks. Scheduling algorithms to provide QoS in single hop (point to multipoint) IEEE 802.16 networks are considered in [13], [18], [36] and [40]. See also ([19]) for a performance analysis.

The problem of scheduling and routing in adhoc multihop wireless networks has been extensively studied in recent years (see [4], [15], [26] for general surveys and tutorials). Scaling laws for fundamental limits on information transfer in multihop wireless networks are surveyed in [42]. The dominant MAC (multiple access control) protocols being considered for multihop networks are the CSMA/CA based IEEE 802.11 and the TDMA based IEEE 802.16. Since 802.11 technology is much more mature and cheaper, most of the mesh network deployments are based on it. However, due to co-channel interference it does not provide satisfactory performance ([28], [41]) while 802.16 can be much superior. See [21] for a recent contribution to provide QoS in 802.11 based mesh networks.

The studies on multihop 802.16 networks are [8], [9], [23], [34], [35] and [39]. In [39] a simple heuristic scheduling and a Tree routing algorithm are proposed to achieve efficient channel utilization. [8] and [23] provide fair access to all nodes and also efficient utilization of resources. In [35] also routing and scheduling algorithms are provided which are efficient

for the overall system but spatial reuse of the channels is not allowed (because the 802.16 standard at that time did not allow spatial reuse). In [34] also channel spatial reuse is not allowed but within this limitation they provide QoS to individual TCP and real time connections. The QoS guarantee to individual flows has not been provided in any other multihop wireless network study that we are aware of (all the other studies mentioned above provide scheduling and routing for the *aggregate* traffic generated at different nodes which as we will see is not sufficient to guarantee QoS to individual connections). In [9] the authors study the distributed scheduling.

In this paper, we present algorithms for centralized scheduling of real and non-real time traffic with the objective of providing QoS within the framework of the IEEE 802.16 mesh mode. We first obtain an optimal and fair routing and scheduling of the aggregate traffic generated at different nodes within the network. This way we fix the particular real time and TCP connections that pass through a particular link and also the slots in which the link transmits. Next we develop algorithms to decide how each link transmits the packets of different flows passing through it on the slots assigned to it so as to provide QoS to individual flows. The real time applications use UDP while data applications use TCP. TCP, being window flow controlled behaves very differently from UDP. First we develop algorithms for UDP and then for TCP. Finally we combine these algorithms to provide QoS in a network serving both real and data applications. To ensure sufficient resources we also discuss an admission control algorithm that can be used in our setup. Our algorithms use the network resources efficiently and fairly and can be used in real time by the MBS.

The organization of the paper is as follows. Section 0.2 describes the system model. We obtain an optimal and fair routing and link scheduling algorithm in Section 0.3. In Section 0.4 we develop scheduling algorithms to provide QoS to UDP connections. TCP connections will be studied in Section 0.5. In Section 0.6 we handle both UDP and TCP traffic together to provide QoS to each connection. Section 0.8 provides an admission control policy. Section 0.9 concludes the paper.

0.2 System Model

IEEE 802.16 supports two modes of operation: Point to multipoint (PMP) and Mesh mode. In PMP the traffic is transmitted directly between the BS and an SS. This is the common mode and current implementation efforts are directed at PMP. In the Mesh mode, the overall area is divided into meshes. Each mesh has a Mesh BS. The other nodes in a mesh are called Mesh subscriber stations (MSSs). A transmission can take place between two MSSs within a mesh or between two different meshes. The transmission between two MSSs within a mesh can occur via other MSSs within the mesh which may or may not involve the MBS. Transmission between two MSSs in two different meshes involves transmission from the source MSS to its MBS (possibly via other MSSs in the mesh), from MBS to BS, from BS to MBS of the receiver mesh and finally from this MBS to the receiver MSS.

In this paper we consider the mesh mode. We provide a brief overview of the IEEE 802.16 mesh mode of operation and present the system model that we use in our work.

The mesh mode supports two different physical layers, WirelessMAN-OFDM and WirelessHUMAN. Both of these use 256 point FFT OFDM TDMA/TDM for channel access and operate in a frequency band below 11GHz. The first operates in the licensed band but the second uses the unlicensed band. The standards also support adaptive modulation and coding where the burst profile of a link (i.e., modulation scheme and the coding rate) and hence the link rate is changed depending upon the channel conditions.

The mesh mode supports only Time Division Duplex (TDD) to share the channel between the uplink and the downlink. A Mesh frame consists of a control and a data subframe. The control subframe serves two basic functions. One is the creation and maintenance of cohesion between different stations. The other is the coordinated scheduling of data transfers between stations. The data subframe consists of MAC PDUs transmitted by different users. A MAC PDU consists of a generic MAC header, a Mesh subheader and optional data. The standards support both centralized and distributed scheduling of slots. Centralized scheduling is mainly used to transfer data between the MBS and the MSSs. Since this is the usual scenario, centralized scheduling is the dominant mode. The MBS periodically collects the channel

information and the resource (throughput) requests of all the nodes to draw up the schedule which it distributes to the nodes.

We consider the following scenario. Consider a Mesh network with M MSSs labeled $1, 2, \dots, M$. The MBS is labeled 0. We consider Uplink and Downlink *Centralized Scheduling* of the MSSs, which, according to the standards uses TDMA with spectral reuse. Also the data is directed either to or from the MBS. We assume that each node transmits at the maximum allowed power, if needed. (Although power control is also an important issue in performance of a mesh network, the standard currently does not emphasize it and hence we do not address this in this Chapter). As the channel condition on a link changes, the data rate is also changed so as to meet the desired BER. Let r_{ij} denote the rate and $E[r_{ij}]$ the average rate of the channel from node i to node j . Resource allocation is done by the MBS in units of (mini) time slots. One time slot consists of multiple OFDM symbols. Each allocation is valid for K frames consisting of N time slots (for simplicity of notation we will take $K=1$).

IEEE 802.16 supports real and nonreal time applications. The real time applications, e.g., IP telephony and video conferencing use UDP while data applications use TCP. Real time applications and interactive data (file transfer and web browsing) require QoS. To provide QoS to these applications will require careful routing and scheduling of traffic through the mesh network. We will consider these problems for both types of traffic. Since UDP traffic and real time QoS requirements are very different from TCP traffic and interactive data QoS requirements, we will consider these problems separately and then show how to integrate them in the same system.

To provide the QoS, we will generally follow the QoS-architecture developed in [34] since this seems to be the only architecture available for 802.16 mesh networks which guarantees per flow QoS. However, due to 802.16 mesh requirements at that time [34] did not consider spectral spatial reuse. This can be a serious limitation because spectral-reuse can provide significant capacity improvement. Thus, in our current proposal we will remove this restriction.

We will use a two step approach. In the first step we will provide routing and scheduling

for the *aggregate* traffic for each source-destination pair of MSSs (one of these MSSs will be the MBS). The aggregate traffic will be the *mean* total traffic of all the real and data applications between different source-destination pairs. This of-course does not guarantee the QoS to individual flows. In the second step we develop scheduling algorithms to share the long-term throughput guaranteed in step one between real and data applications of each source-destination pair to guarantee QoS to individual flows. We will justify the two step approach and will provide simulation results to verify the claims on QoS guarantees.

Section 0.3 provides the routing and scheduling for step 1 to satisfy the aggregate demands of source-destination pairs. In Sections 0.4-0.6 we detail our step 2 to ensure the QoS to individual flows.

0.3 Routing and Scheduling for Aggregate Traffic

The algorithms developed in this section can be used for uplink as well as downlink simultaneously. Let $\lambda(s, d)$ be the mean number of bits per slot to be transmitted from MSS s to MSS d . This is the sum of mean throughput required by all the real time and data connections transmitting from MSS s to MSS d . The calculation of mean throughput requirements for TCP connections is shown in Section 0.5. For the CBR connections, it is the traffic they generate per slot. For a VBR connection it equals the equivalent bandwidth (see Section 0.4.2; for calculation for this Section h there can be taken equal to the maximum distance between the MBS and a SS in the mesh). For downlink MSS s will always be the MBS and for uplink MSS d will be the MBS. We develop algorithms in this section which will decide the routes that $\lambda(s, d)$ will follow and also the slots in which each link will transmit.

We develop algorithms which provide routes and schedules that will be functions of $\lambda(s, d)$ and the mean link rates $E[r(i, j)]$ but otherwise would not vary with time. By exploiting the current queue lengths at different links and the channel states one could vary the routes and the schedules to obtain better performance ([33], [34]) but we will not do this. This is because, frequently varying routes and schedules in real time is computationally more complex and can also cause loops in the path traversed by a packet and resequencing problem at the receiver.

Furthermore, our QoS architecture requires reservation of resources at intermediate MSSs along a route. Thus frequent route variation is not desirable. In addition, in case of spectral spatial reuse, changing the routes and schedules is more complicated. Thus we will change the routes and schedules of link transmissions only when some of the $\lambda(s, d)$ and/or $E[r(i, j)]$ change drastically and/or some links/nodes fail. These algorithms will be run at the MBS and then the schedules broadcast to different nodes via Mesh Centralized Schedule messages.

The algorithms we develop will satisfy the traffic requirements $\lambda(s, d)$ of each source-destination pair (s, d) if possible. If not, then we will provide a “fair” solution which is also efficient. When it is possible to satisfy certain (fair) traffic requirements of all source-destination pairs, we will provide routing and scheduling which optimizes a cost function.

In this section we use the approach developed in [31] which in turn was partly motivated by [24].

In [34], where spatial reuse is not allowed it was shown that the routing and scheduling problems can be decoupled and that a Tree structure can be optimal for routing. In the present general scenario this may not be true (although the 802.16 standard seems to prefer the Tree structure ([8], [39])).

The cost function to optimize will be a sum of the link cost functions $f(\Gamma(i, j)n(i, j))$ where $\Gamma(i, j)$ is the total mean traffic rate per slot and $n(i, j)$ is the fraction of slots assigned to link (i, j) . Better cost functions can be obtained as a function of higher moments of traffic arriving at link (i, j) but higher moments are difficult to obtain and handle. Thus it is desirable to use only the first moments. But even then f will often be a nonlinear function. For example, using Kleinrock’s independence assumption ([38]) or approximating the queues at each link by an $M/M/1$ queue, we get

$$f(\Gamma(i, j), n(i, j)) = \frac{\Gamma(i, j)}{n(i, j)E[r(i, j)] - \Gamma(i, j)} \quad (1)$$

as the mean queue length at the link (i, j) and $[n(i, j)E[r(i, j)] - \Gamma(i, j)]^{-1}$ as the mean delay. Similarly we can consider packet loss probability if the buffer lengths are small. Using Lagrange multipliers one can accommodate constrained optimization (see [31] for more details).

The cost functions provided above may not be very good approximations of mean delay and queue lengths. Better approximations are provided in [31]. However it is an important direction for future research.

We consider the following joint routing and scheduling problem:

Find $n(i, j)$ and $\alpha_p(s, d)$ that minimizes

$$\sum_{(i,j) \in \mathcal{L}} f(\Gamma(i, j), n(i, j)) \quad (2)$$

subject to

$$\Gamma(i, j) = \sum_{(s,d)} \sum_{p: (i,j) \text{ is on } p} \alpha_p(s, d) \lambda(s, d) \leq n(i, j) E[r(i, j)], \quad (3)$$

$$0 \leq \alpha_p(s, d) \text{ for each } p, (s, d) \quad (4)$$

and

$$\sum_p \alpha_p(s, d) = 1 \text{ for each } (s, d) \quad (5)$$

where $\alpha_p(s, d)$ is the fraction of (s, d) traffic on route p , \mathcal{L} is the set of links and the inner summation in (3) is over all possible routes for (s, d) . The condition (3) is required to satisfy the stability condition at each link (i, j) .

Obtaining the optimal solution in (2)-(5) can be very time consuming because of the nonlinear cost function. Also, if it is not possible to satisfy the $\lambda(s, d)$ requirements of each (s, d) , the above optimization problem may not provide any solution. Thus in the following we first develop algorithms which will check for feasibility of the demands $\lambda(s, d)$. If these are not feasible, then we provide a solution which may be “fair” to all (s, d) pairs. Finally we obtain a solution which is fair to all (s, d) pairs and optimizes the nonlinear cost function.

Consider the following optimization problem:

$$\max \lambda \text{ such that} \quad (6)$$

$$\sum_p \alpha_p(s, d) \geq \lambda \text{ for all } (s, d) \quad (7)$$

and (3) and (4) are satisfied where the summation in (7) is over all possible paths p in the network. A solution to the above optimization problem can be considered “fair” and efficient. This is because if there is a routing and scheduling algorithm which satisfies all the traffic requirements $\lambda(s, d)$ then λ will be ≥ 1 . If not, it provides the largest fraction of traffic that can be satisfied for each (s, d) . This concept of fairness has also been considered in [25], [34] and [36]. Furthermore unlike (2)-(5), this problem is a linear program (LP) and hence can be solved much faster than the nonlinear problem (2)-(5).

In addition to (3)-(7) the network should also satisfy some transmission constraints. These constraints occur due to wireless nature of the links. In the 802.16 standard these are given by stating that two links can be scheduled for transmission in the same slot if they are 1, 3 or 7 hops away from each other. However in a practical system it may or may not be possible to schedule two links in a slot depending upon the power of transmission, the distance between the receiving nodes and other geographical factors. This can be decided in a particular scenario by actually taking measurements and finding the SINR (Signal to Interference and Noise Ratio) at different nodes. In the following we will assume that this has been done for the network under consideration. Sometimes we can write these constraints as necessary and/or sufficient inequality constraints. For example, if no spatial channel reuse is allowed (as was done in the 802.16 standard in 2004 or if in the current standard we choose the option that spatial reuse is allowed only with a hop distance of 7, in which case it may effectively be no spatial reuse) then necessary and sufficient conditions are

$$\sum_{(i,j)} n(i, j) \leq 1. \quad (8)$$

It is shown in [31] that if our transmission constraints are such that a node can receive successfully if and only if one of its neighbouring nodes transmits in a slot and that a node can transmit only on one of its links at a time, then the necessary and sufficient conditions are

$$\sum_{j:(i,j) \in \mathcal{L}} n(i, j) \leq 1 \text{ for all } i$$

$$\text{and } \sum_{i=(i,j) \in \mathcal{L}} n(i,j) \leq 1 \text{ for all } j. \quad (9)$$

If we put the constraint that only one incoming or outgoing link at a node can be active at a time, then it is shown in [8] that necessary and sufficient conditions, in the context of WiMax mesh networks are

$$\sum_{j:(i,j) \in \mathcal{L}} n(i,j) + \sum_{j:(i,j) \in \mathcal{L}} n(i,j) \leq 1 \text{ for all nodes } i. \quad (10)$$

It is argued in [8] that by using directional antennas and careful placement of nodes, these constraints can be quite realistic in the WiMax.

Our general setup can work with transmission constraints of the type [9], [10] along with the optimization problem considered above. The problem of transmission constraints has also been studied in [5], [17] and [25]. Sometimes corresponding to these constraints, one may only be able to get only *sufficient* inequality constraints. In the following we will assume the transmission constraints have been put as linear inequality constraints and call them (T).

As mentioned above, a solution $n(i,j)$ and $\alpha_p(s,d)$ satisfying (3), (4), (6), (7) and (T) will be considered efficient and fair. However observe that the service provider will frequently need to run an algorithm in real time to obtain a solution and hence complexity of the algorithm will be an important issue. In general the scheduling problem is NP hard because the $n(i,j)$ need to be integer valued (should be the number of slots in a frame assigned to link (i,j)). However if we ignore the integrality of $n(i,j)$ and consider them as non-negative fractions as considered above, (3), (4), (6), (7), (T) becomes an LP problem which is computationally much more tractable. Once a solution $n(i,j), \alpha_p(s,d)$ is found then one can find an appropriate frame length T (in number of slots) such that $n(i,j)T$ is (approximately) integer valued. Then obtaining a schedule for the links is not difficult (see [31] for an algorithm).

If the number of nodes in the mesh is large, then complexity of the above LP can also be of concern because the number of variables $\alpha_p(s,d)$ can be exponential in number of nodes. However in that case this LP can be reformulated in term of link flows ([5], [3]) and this

problem can be taken care of. Another way to handle it, in terms of $\alpha_p(s, d)$ itself is by solving the dual problem as discussed in [24].

If λ obtained from the above optimization problem is ≥ 1 then there is a route and schedule for all (s, d) pairs and links which can satisfy the traffic requirements of all users. If $\lambda < 1$ then, λ is the largest fraction of traffic requirements of all (s, d) pairs that can be satisfied by the network. Our solution of the above problem can also provide $\lambda > 1$. When this happens, the network has more BW/throughput than needed to satisfy the current QoS requirements of all the users. Then the above solution allocates the extra resources to different (s, d) pairs in a fair way. The extra resources can be used by the TCP connections usefully because in our QoS requirements we have only specified the minimum mean throughput a TCP desires.

Next we find a solution that minimizes the cost function

$$\sum_{(i,j) \in \mathcal{L}} f(\Gamma(i, j), n(i, j)) \quad (11)$$

while satisfying (3), (4), (T) and

$$\sum_p \alpha_p(s, d) \geq \lambda \text{ for all } (s, d) \quad (12)$$

where λ is the optimal solution obtained from LP (3), (4), (6), (7) and (T). This is a nonlinear optimization problem and can be quite computationally intensive. Depending upon the actual form of f one can try to obtain efficient algorithms to compute an optimal solution. In [31] several algorithms have been identified that can be useful for functions of the form (1) (see, e.g., [10], [12]). Furthermore an *LP* can also be used if instead of minimizing *sum* of link costs (11) we use minimization of $\max_{(i,j)} f(\Gamma(i, j), n(i, j))$ [31].

One can further improve the efficiency of the system if the optimal λ in (6) is less than 1. In [31] a method is suggested where the fraction of demands satisfied for some of the (s, d) pairs can be increased without decreasing the fraction for other (s, d) pairs below the optimal λ obtained above.

The routing and scheduling provided above will ensure that the slot assignment $n(i, j)$ for

link (i, j) is such that its average rate $n(i, j)E[r(i, j)]$ is sufficient to carry the overall traffic passing through it. However, it will not ensure that the throughput (rate) seen by traffic of a pair (s, d) will indeed get its required share of throughput. This may partly happen because the TCP connections passing through fewer hops tend to get more throughput than the other TCP connections sharing links with it (see [7], [34]). Thus to ensure that the traffic of some (s, d) pairs does not hog most of the throughput at a link, we will store the traffic of different (s, d) pairs in different queues at each link and provide the required throughput to each queue via WRR (Weighted Round Robin). This will ensure that traffic of each (s, d) pair will get its share of throughput at each link on its routes.

In Sections 0.4-0.6 we show how the aggregate allocation of BW to the traffic of different (s, d) pairs will be used so as to ensure end-to-end QoS to individual flows. Section 0.4 considers the real time traffic. Section 0.5 provides details for the TCP connections. Section 0.6 shows how to combine the real time and data traffic to provide QoS to individual connections. In this section we also verify via simulations that our QoS architecture actually indeed provides the QoS. Our QoS architecture will be scalable in the sense that it will not require per flow processing at intermediate nodes.

We will observe in Section 0.6 that in order to accommodate the QoS architecture of Sections 0.4-0.6, some optimality of the solution provided above will be lost.

0.4 QoS for Real Time Traffic

In this section we design scheduling algorithms to guarantee QoS to individual UDP connections. Two important real time applications are IP telephony and video conferencing. For these applications, the end to end delay of a packet should not exceed (say) 150 msec. If a packet exceeds this delay, it will be dropped. For satisfactory performance the fraction of packets dropped for an application should be less than (say) 2%. To satisfy these QoS requirements, we propose that at the end of a (scheduling) frame we drop the packets which could not be transmitted through the wireless network. This will ensure a maximum delay of about 40 msec (for 4 frames of 10 msec each) in the wireless network (the rest of the

delay margin is left for the remaining part of the network that a packet may have to travel). We develop algorithms which will ensure that a particular user will not experience drop probability greater than 2%.

Packets generated by audio encoders (in IP telephony) usually generate a constant bit rate (CBR) traffic. But a video encoder (say MPEG) one may use in video conferencing generates variable bit rate (VBR) traffic (although downloading a stored video may arrive as a CBR traffic). To satisfy the QoS requirements of these two types of applications efficiently we require different considerations. Therefore we consider these two cases separately. We consider scheduling of CBR traffic in Section 0.4.1 and VBR traffic in Section 0.4.2.

Based on the routing and scheduling algorithm of Section 0.3, we know the fraction $\alpha_p(s, d)$ of total average traffic requirement $\lambda(s, d)$ of each pair (s, d) passing through a route p . Then, as we will detail in Section 0.6, based on the *average* throughput requirement of each UDP and TCP connection of (s, d) , we will decide which of the CBR, VBR and TCP connections of (s, d) will pass through which route. Knowing the route that each connection will take, we decide the QoS architecture in the following to provide the QoS to each connection.

0.4.1 Scheduling of CBR traffic

Let X (a constant) be the total amount of traffic generated during a frame by different CBR connections of a particular (s, d) pair following a particular route denoted by links p_1, p_2, \dots, p_h (this will be known based on the algorithm in Section 0.3). As mentioned above, in order to provide delay guarantees to these flows, we propose dropping of data that cannot be transmitted at the end of the scheduling frame. Now, the scheduler has to ensure that the amount of data dropped conforms to the QoS requirements of the flow. Let the upper bound required on the drop probability of the packets of these flows be ϵ . (For simplicity of notation we are taking this upper bound same for all CBR applications. If different flows have different requirements then ϵ is the minimum of these requirements. Of course one can handle the general case in the same way for each flow separately).

The scheduling problem for this CBR-UDP traffic is to calculate the number of slots

$n_j, j = 1, \dots, h$ required at link p_j such that X units of data can be transmitted to the MBS per scheduling frame and the end to end drop probability is bounded by ϵ .

We decompose the drop probability ϵ into $\{\epsilon_j, j = 1, \dots, h\}$ such that $\prod_{j=1}^h (1 - \epsilon_j) \geq (1 - \epsilon)$. At link p_j the number of slots allocated for these flows has to ensure that the drop probability is upper bounded by ϵ_j . We use n_j for the allocation of slots for link p_j , $r(j)$ for $r(p_j)$ and X_j for $\prod_{k=1}^j (1 - \epsilon_k) X$ to simplify the notation. Then, n_j has to satisfy

$$\lim_{n \rightarrow \infty} \frac{\sum_{k=1}^n (X_j - n_j r_k(j))^+}{n X_j} \leq \epsilon_j. \quad (13)$$

This reduces to $E((X_j - n_j r(j))^+) \leq \epsilon_j X_j$. We can rewrite it as

$$\int_0^{\frac{X_j}{n_j}} (X_j - n_j r) f_j(r) dr \leq \epsilon_j X_j \quad (14)$$

where $f_j(\cdot)$ is the pdf of the link rate $r(j)$ which is assumed to be known. The quantity on the left in (14) is a non-increasing function of n_j and hence it is easy to compute the required n_j .

Since the drop probability ϵ is small, we don't lose much in optimality if in the above calculation we replace X_j with X . Also, instead of arbitrarily choosing the values $\{\epsilon_j, j = 1, \dots, h\}$, we can consider the optimization problem

$$\min \left\{ \sum_{j=1}^h n_j \right\}$$

subject to

$$\prod_{j=1}^h (1 - \epsilon_j) \geq (1 - \epsilon) \quad \text{and}$$

$$\int_0^{\frac{X_j}{n_j}} (X_j - n_j r) f_j(r) dr \leq \epsilon_j X_j, \quad j = 1, \dots, h.$$

The above allocation of slots to satisfy the QoS was proposed in [34]. However, it was observed in [34] that even though one can obtain the required number $n(j)$ of slots needed to satisfy the QoS as above, in practice, due to very small probabilities ϵ , the number of

slots needed actually becomes $X/r_{min}(j)$ where $r_{min}(j)$ is the minimum rate supported by link p_j . By taking $r_{min}(j)$ as the minimum rate supported in the standard, $n(j)$ becomes independent of statistics of $r(j)$. This may be much easier to do and will entail only a small loss of optimality. Similar comments will hold for VBR scheduling in Section 0.4.2.

0.4.2 Scheduling of VBR Traffic

Consider K VBR flows generated at an (s, d) that will follow the same route p_1, p_2, \dots, p_h . Let $D_n(k)$ be the amount of data generated by flow k in frame n . We assume that the arrival process $\{D_n(k), n \geq 0\}$ for each $k = 1, \dots, K$ is stationary and ergodic with known statistics. We also assume that the arrival processes from the various sources are mutually independent. As in the case of CBR traffic, we provide delay guarantees to the VBR flows by dropping the data not transmitted by the end of each frame. The problem is to calculate the number of slots required by this VBR traffic on each node on its route in order to bound the drop probability by ϵ . Again obtain $\epsilon_j, j = 1, \dots, h$ as in Section 0.4.1.

The amount of resources required utilizes the statistics of the arrival process. Since the data not transmitted at the end of a frame is dropped we need to consider only the *marginal* distribution of $D_n(k)$ to calculate the amount of resources required at the first node. Also, since the drop probability is typically small, we can assume that the statistics of the arrival process is not distorted after flowing through the first node. Hence we can use the same analysis for each of the nodes along the route. Choose ϵ_j^b and ϵ_j^d such that $(1-\epsilon_j^b)(1-\epsilon_j^d) \geq (1-\epsilon_j)$. Now, find C_j such that

$$P\left(\sum_{k=1}^K D_1(k) > C_j\right) \leq \epsilon_j^b. \quad (15)$$

This C_j/K is called the *equivalent bandwidth* of the VBR source [38]. If we take ϵ_j^b to be same for all j , C_j becomes independent of j which simplifies the further design. Therefore we do that in the following and denote it by C . The MBS can treat a VBR source as a CBR flow generating C/K units of data per frame and calculate the number of slots required to satisfy the drop probability requirement of ϵ^d as in Section 0.4.

In practice, the exact statistics of a VBR arrival process may not be available. The statistics that is generally available is the maximum, the minimum and the average data rates. In order to satisfy the QoS requirements of the flows, we can calculate the value of C by using a source model that has all the known characteristics of the original source but has the worst case behaviour (i.e., gets the largest equivalent BW). It is shown in [20] that for the case of K independent, homogeneous, stationary sources with arrivals in a slot taking values in a finite set (this class covers Markov modulated sources modulated by finite state Markov chains) the worst case drop probability is obtained by replacing these sources by i.i.d. ON-OFF sources having the same maximum, minimum and average rates.

Simulation results in [34] show that the slot allocation provided in Sections 0.4.1 and 0.4.2 does indeed provide QoS to individual flows.

0.5 QoS for TCP Traffic

This section develops scheduling algorithms which can guarantee the QoS for TCP connections. Some TCP applications, e.g., email do not require any QoS. However web traffic and file transfer may require certain minimum response time. We try to satisfy these QoS requirements by providing adequate minimum mean throughput to individual connections. But, if there is insufficient bandwidth to satisfy the minimum mean throughput of all the TCP connections, then again there is the question of how should we do the allocation in a fair way. In this case unlike the UDP connections where we recommend admission control, one other option is to give these connections less bandwidth than requested (see more comments on this in Section 0.9).

Initially we will consider the case of persistent TCP connections. These are long lived connections which need to send a large file. The QoS requirement for these connections is the maximum response time. Later on we will also consider TCP-ON-OFF connections (see [16] for details on this model) which model the web traffic using HTTP 1.1. In this model, a TCP connection transfers multiple files. Between transfer of two files, a TCP connection may not have a file to transfer (OFF period) for sometime. This is the dominant traffic type

in the current Internet. An appropriate QoS requirement for these connections is the mean file download time.

For both of the above TCP types the QoS can be satisfied if we ensure a minimum mean throughput to each connection. In the following we provide scheduling schemes to ensure this. First we compute the average throughput these connections need to satisfy their QoS. These computations can be used also to compute the total average throughput requirement of all TCP connections for any particular (s, d) that we need in Section 0.3.

We consider TCP persistent connections first. Let N^P persistent TCP connections of an (s, d) be passing through a particular route. Let λ_j^P be the minimum throughput requirements (in packets/sec) and s_j^P the packet lengths (in bits) of the j^{th} persistent TCP connection. Thus the total average throughput requirement of the persistent TCP connections is $\lambda^P = \sum_{j=1}^{N^P} \lambda_j^P s_j^P$ bits/sec.

We now consider TCP-ON-OFF traffic. Let N^O TCP-ON-OFF flows of (s, d) be following the same route as the N^P persistent connections mentioned above. For simplicity assume all of them to have the same mean download time requirement of T^{on} (this is the mean time that will be taken to download a file by such a connection) and the same mean number of packets D to be downloaded during an ON period. Let the packet size of a connection be s^O . Let the mean time between two downloads be T^{off} . Then the throughput required by such a TCP flow to satisfy its QoS requirement is $\lambda^m = \frac{D s^O}{T^{on}}$. Also the long term average throughput required by this flow is $\lambda^a = \frac{D s^O}{(T^{on} + T^{off})}$. The probability of a connection being ON is $\frac{T^{on}}{T^{on} + T^{off}}$. Making the *assumption* that the ON-OFF processes of different connections are independent, the mean number of connections ON at anytime is $\frac{N^O T^{on}}{T^{on} + T^{off}}$. Now since each connection requires a throughput of λ^m when ON, the total throughput requirement of ON-OFF traffic at node i is

$$\lambda^O = \left(\frac{N^O T^{on}}{T^{on} + T^{off}} \right) \lambda^m = N^O \lambda^a.$$

The above approximation improves as the number of connections N^O increases.

The overall average throughput needed for the persistent and ON-OFF connections is

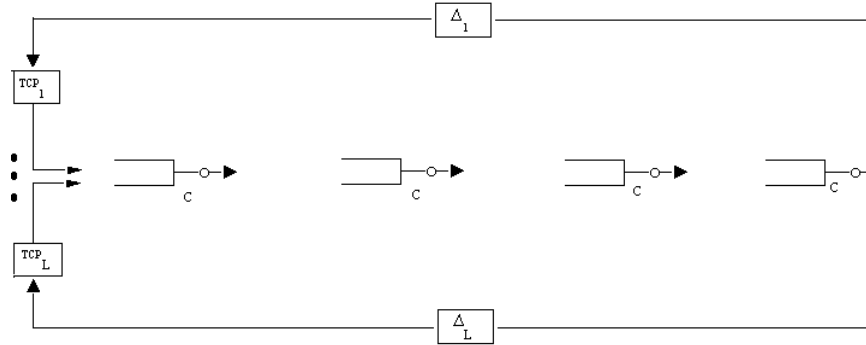


Figure 1: Multiple TCP flows through multiple queues with fixed rates

$$\lambda_T = \lambda^P + \lambda^O \text{ bits/sec.}$$

Let $L = N^P + N^O$ TCP connections of an (s, d) be passing through a particular route as decided by the algorithm in Section 0.3. Suppose they have been guaranteed a mean throughput of λ_T bits/sec at each node on its route (say via WRR as mentioned before). We will see below that this is not sufficient to guarantee minimum throughput to individual TCP connections passing through the same set of links. This will require extra care. First we consider persistent connections.

Consider the system shown in Fig.1. Let the N^P TCP connections are passing through (say) four queues. TCP_i has window size W_i (assume it is fixed) and propagation delay Δ_i (representing delays in the rest of the network). At each queue the link speed is $c = \lambda^P$ bps (ensured say, by WRR discussed above). In this scenario the packets/acks of different TCPs will be either at the first queue (and not at the other three queues) or propagating in the rest of the network (in propagation pipes Δ_i s). If TCP_i gets a throughput of λ_i packets/sec, then by Little's law ([37], [38]) it has on the average (under stationarity, proved in [16]) $\lambda_i \Delta_i$ packets in the propagation pipe. Thus the average queue length in the first queue is $\sum_{i=1}^{N^P} (W_i - \lambda_i \Delta_i)$ (actually it will be a little less than this; some packets might be getting serviced at the other queues). Since TCP_i has packet length s_i^P (in bits) the mean queueing delay in the first queue is $\sum_{i=1}^{N^P} (W_i - \lambda_i \Delta_i) s_i^P / c$. As there is no queueing delays in the other queues, the total mean round trip time of TCP_i is approximately $\frac{1}{c} \sum_{j=1}^{N^P} (W_j - \lambda_j \Delta_j) s_j^P +$

$\frac{3s_i^P}{c} + \Delta_i$. Thus the total throughput obtained by TCP_i is

$$\frac{W_i c}{\sum_{j=1}^{N^P} (W_j - \lambda_j \Delta_j) s_j^P + 3s_i^P + \Delta_i c} \text{ packets/sec.} \quad (16)$$

To provide a desired throughput (or minimum throughput) to different TCPs, one needs to adjust c , W_j , s_j^P appropriately in (16) such that it becomes equal to the desired throughput for each i (which may be different for different TCP connections). The bandwidth c (in bits/sec) should equal $\sum_{i=1}^{N^P} \lambda_i^P s_i^P$ (if λ_i^P packets/sec is the minimum mean throughput desired by connection i). It is possible that a service provider may not have the freedom to choose W_i and s_i^P for different TCP connections (these are selected by the receivers and the networks through which the connections are passing). However even though the maximum window size W_i may not be controllable, if it is large enough, its mean size can be reduced to an appropriate size by dropping its packets in a controlled way (say) via RED [14] at the nodes in the mesh such that the required throughput can be provided to each TCP. We explain this in the following.

Let us fix a desired queueing delay of d^* sec in the first queue. Define for each i , $\widetilde{\Delta}_i = \Delta_i + \frac{3s_i^P}{c}$. We fix the desired mean window size of $E[W_i]$ such that

$$\frac{E[W_i]}{d^* + \widetilde{\Delta}_i} = \lambda_i^P \quad \text{for each } i. \quad (17)$$

Now we use RED control for each TCP connection i and specify its RED parameters such that at average queue length $d^* c$, it will drop the packets of TCP_i with probability p_i , where

$$p_i = \frac{8}{(3(E_\pi[W_i] + 4)^2 + 5)}$$

for each i . (This has been used in [34] and is based on [27]. It provides a reasonable approximation for small values of p_i . We are working on better approximations.) Then it can be shown (see [32] and [33]) that this system will operate under steady state such that the first queue will have the mean queue length $d^* c$ and each of the TCPs will have their mean window size $E[W_i]$ satisfying the above requirements. Furthermore, TCP_i will get the

throughput λ_i^P packets/sec. We will verify these claims via simulations in Section 0.7.

To include also the N^O TCP-ON-OFF connections in the above setup, we make $c = \lambda_T$ bps. Furthermore, we use RED to control the window size of a ON-OFF connection also where its mean window size $E[W]$ should satisfy (17) with λ_i^P replaced by λ^m .

It is shown in [22] and [30], that the TCP connections can be grouped such that one needs only a few RED parameters to take care of the throughput requirements of different TCPs and per flow processing is not required.

Once we have ensured that the overall TCP traffic originating at the different nodes gets the bandwidth it requires at each node on its route, to ensure that the different persistent and ON-OFF TCP connections in it get the throughput they want, we set the window sizes according to (17). To provide the needed mean window size, as explained above we can use RED at the bottleneck node along the route of the TCP connections. If the link rates are all fixed, then after ensuring that these flows get their required throughputs at each link, the node through which the TCP flows enter the mesh network is the bottleneck. However due to the random variation of the link rates in wireless links, any link along the route can momentarily turn into a bottleneck (whenever its channel state is poor). To make our scheme work, one method is to implement RED control at all nodes along the path of the TCP flows. However this involves difficulties in practical implementation since every node has to acquire the RED parameters of all the flows passing through it. An alternate method is to force the ingress node to be the bottleneck by providing about 3-5% extra bandwidth to the flows at the other nodes along the route. This extra bandwidth whenever not used by these TCP flows, can be provided to the best effort traffic.

0.6 Joint Scheduling of UDP and TCP Flows

In this section we address the problem of scheduling in presence of both UDP and TCP traffic. The requirements of UDP traffic were discussed in Section 0.4 and of TCP traffic in Section 0.5. From the arguments in these sections, in order to provide QoS to UDP we had to

consider the worst case channel conditions whereas for TCP we had to consider the average channel rates. Thus there is a huge difference between the average bandwidth requested and the average bandwidth provided to guarantee the QoS of the CBR and VBR connections. Here we utilize this extra bandwidth for scheduling of TCP flows.

We provide priority to UDP traffic over TCP traffic in the network. It has been observed in [22], [30] and [34] that by doing this the delays experienced by UDP flows can be drastically reduced without affecting the throughput of the TCP flows. In the present context, it will allow us to save resources to provide QoS.

Let $\lambda_U(s, d)$ and $\lambda_T(s, d)$ be the *average* throughput required by the total UDP (for a CBR connection it is its rate, for a VBR connection, it is its equivalent bandwidth computed by taking h to be the maximum number of hops of a node from MBS) and TCP traffic generated by (s, d) . Then we define $\lambda(s, d) = \lambda_U(s, d) + \lambda_T(s, d)$ as the average requirement of (s, d) . We use this requirement in Section 0.3 to obtain the routing and scheduling for all the pairs. It is possible that the overall traffic of (s, d) is split over several routes. Let $\alpha_p(s, d)$ be the fraction of (s, d) traffic on route p .

If $\alpha_p(s, d) < 1$ then we will send $\alpha_p(s, d)$ fraction of $\lambda_U(s, d)$ and $\lambda_T(s, d)$ on route p . On a link (i, j) on p out of a total $n(i, j)$ slot assignment in Section 0.3, a fraction $n'(i, j, s, d)$ would be assigned for $\lambda(s, d)\alpha_p(s, d)$ traffic of (s, d) which is $\geq \frac{\lambda(s, d)\alpha_p(s, d)}{E[\tau(i, j)]}$.

This slot assignment is sufficient to satisfy the average throughput requirement of (s, d) traffic but may not be sufficient for the QoS of the real time traffic. For this we do the following. First, from the above assignment of UDP and TCP traffic on route p we assign a number of UDP and TCP flows of (s, d) to the route p . Since we want to send the traffic of a particular flow on a single route (it is advisable not to split the traffic of a particular flow on more than one route), we may need to change a little bit, the fraction $\alpha_p(s, d)$ of UDP and TCP traffic on route p so as to obtain an integral number of UDP and TCP flows of (s, d) on p . In the process, if an $\alpha_p(s, d)$ is very small, one may just make it zero. This can compromise a bit on the optimality of the solution but due to other benefits we would prefer such a solution.

Once the UDP and TCP flows of (s, d) to be routed on route p have been identified, we can compute the fraction of slots $n_U(i, j, s, d)$ needed on a link (i, j) on route p to satisfy the QoS of those CBR and VBR connections via the methods detailed in Section 0.4. Finally we assign $n(i, j, s, d) \triangleq \max(n_U(i, j, s, d), n'(i, j, s, d))$ fraction of slots of link (i, j) for the traffic of (s, d) passing through it. Also, we give priority to the real time traffic of (s, d) over the TCP traffic of (s, d) on each link (i, j) . Thus, since $n(i, j, s, d) \geq n_U(i, j, s, d)$, the QoS of the real time traffic will be satisfied. Also, because $n(i, j, s, d) \geq n'(i, j, s, d)$, the long term average rate of TCP and UDP traffic of (s, d) on (i, j) will be satisfied and hence the TCP traffic gets its share of throughput on each link (i, j) . Often $n(i, j, s, d)$ will be much less than $n_U(i, j, s, d) + \frac{\lambda_T(s, d)\alpha_p(s, d)}{E[r(i, j)]}$ (the number of slots needed on link (i, j) to satisfy the QoS of TCP and UDP flows of (s, d) if we do not give priority to UDP flows) and hence this multiplexing of UDP and TCP traffic of (s, d) on (i, j) provides significant gains in resource requirement. At present the traffic mix in Internet is such that the real time traffic makes less than 10% of the overall traffic. It is likely to be so in near future. In that case $n(i, j, s, d)$ will most likely be close to $n'(i, j, s, d)$ and hence the optimal solution obtained in Section 0.3 will not need to be modified.

In Fig. 2 we show our overall QoS architecture when there are three nodes. Node i has N_{iU} UDP flows and N_{iT} TCP flows entering the Mesh. Flows from nodes 1 and 3 enter node 2. At node 2 the flows from the three nodes are stored in separate queues and are provided their required throughput via WRR. In each queue the UDP packets are given priority over

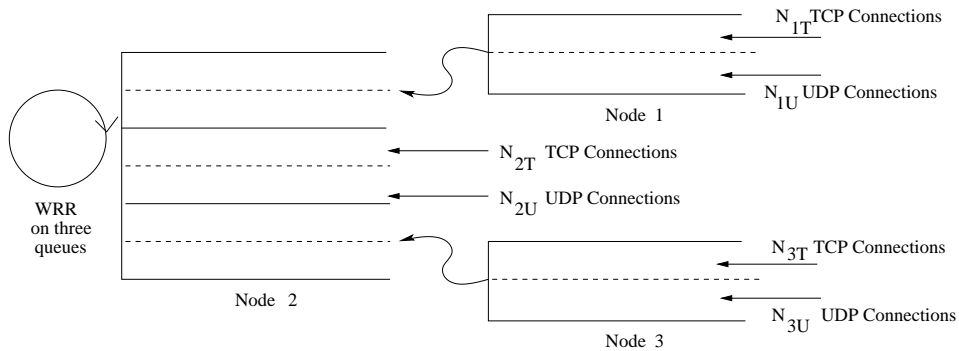


Figure 2: Overall Scheduling of TCP and UDP flows to provide QoS. UDP flows get priority over TCP flows in each queue. RED applied on TCP flows at entry nodes. (We have separated UDP flows and TCP flows in each queue by a dotted line indicating each queue might be implemented via two queues.)

the TCP packets. The TCP flows are controlled via RED algorithms at their entry nodes.

We comment on the scalability of our overall scheme. At each node we need one queue for each source node whose traffic is passing through this node. In a mesh the number of nodes will not be large (it will be a few tens of nodes) and hence even in a large overall network this will not cause any problem. We also commented earlier that the RED control for TCP traffic can be used only at the node it enters. Also as in [22] and [30], we could classify TCP connections according to the mean window size (equivalently, their probability of loss) they need. Thus, even at the entry node we do not need to provide different RED parameters for each TCP but rather for each TCP class. The number of TCP classes needed will not be more than twelve (as shown in [22], [30] although one can work with even fewer). Furthermore, the computational complexity of our algorithms is rather low (at least if we ignore the nonlinear optimization in Section 0.3) and hence these can be implemented easily in real time for a reasonably large network.

0.7 Simulations

We consider the network shown in Figure 3. The network characteristics are summarized in Tables 1 and 2. The frame duration is 10ms and scheduling is done over 3 frames. The channel rates of the channels vary randomly from one scheduling period (of 3 frames) to another independently. The link parameters shown in 2 are the mean data rate per slot expressed in terms of the burst profile given in Table 2.

We consider the transmission constraints (9). For uplink, from each MSS node data needs to be transmitted to the MBS. Each MSS node is sending traffic from 3 CBR sources with rates 16kbps, 32kbps and 64kbps. All MSSs (except MSS 5 and 1) are transmitting traffic from 3 VBR sources each transmitting with mean rate 256 kbps. The MSSs 5 and 6 have four VBR sources each, 2 transmitting at rate 256 kbps and 2 at rate 128kbps. Each VBR source is a Markov modulated source with transition matrix

$$\begin{pmatrix} 0.4 & 0.3 & 0.2 & 0.1 \\ 0.2 & 0.4 & 0.2 & 0.2 \\ 0.2 & 0.3 & 0.2 & 0.3 \\ 0.1 & 0.4 & 0.2 & 0.3 \end{pmatrix}$$

and with rates 73 kbps, 146 kbps, 293 kbps and 586 kbps in the four states for a source with 256 kbps. For a source with 128 kbps in each state, the rate is half of that of a 256 kbps source. For uplink, for all MSSs (except MSSs 5,11) there are 24 persistent TCP connections with 1 requiring 16 kbps, 7 requiring 64 kbps, 8 requiring 128 kbps and the rest 256 kbps. For the MSSs 5 and 11 there are 25 TCPs with 2 requiring 16 kbps, 1 requiring 32 kbps, 6 requiring 64 kbps, 8 requiring 128 and the rest 256 kbps. For the downlink, each MSS is getting the same TCP traffic as for the uplink. There is no UDP traffic in the downlink. The packet sizes of the CBR connections are 100 Bytes, of the VBR connections are 1500 Bytes and of the TCP connections are 1000 Bytes. The TCP connections also have an extra propagation delay of 0.05 sec.

From the above traffic details, the MSSs 1, 2, 3, 4, 8, 9 and 10 generate an average data of 4.5928 Mbps and the MSSs 5, 6, 7 and 11 generate respectively 4.5768, 4.592, 4.592 and 4.5768 Mbps in the uplink. Also the MSSs 1, 2, 3, 4, 8, 9 and 10 receive 3.7128Mbps and MSSs 5, 6, 7, 11 receive respectively 3.712, 3.6968, 3.6968 and 3.712 Mbps in the downlink. In computing these averages we have also included the acknowledgement traffic generated in the uplink and the downlink for the TCP traffic . The ack traffic in each queue is also given higher priority (along with the CBR and the VBR traffic) than the TCP data packets. For these average traffic requirements, we ran the LP (6), (3), (4), (7) and (9) and obtained the $n(i, j)$ and $\alpha_p(s, d)$. Due to lack of space we are not reporting these here. The λ obtained was equal to 1, i.e., the network is able to satisfy the traffic demands of all (s, d) pairs. From $\alpha_p(s, d)$, we identified the CBR, VBR and TCP connections of (s, d) that will use the route p . Also given the $n(i, j)$ for each link (i, j) , using the algorithm in [31] we obtain the allocation of slots to different links in each scheduling period of three frames. Finally we obtain via the WRR, the bandwidth allocation on each link for the traffic of each (s, d) passing through it

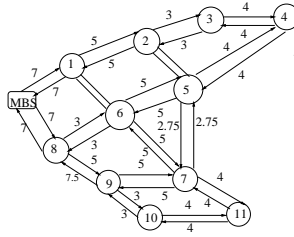


Figure 3: Network used in simulations

and also the mean window size for each TCP connection.

The results obtained from these simulations are provided in Tables 3 and 4. Table 3 provides the maximum drop probability of 0.0 for the CBR and the VBR connections while the maximum average delay is 28.09 msec. Table 4 provides the error= Achieved average throughput - minimum required throughput for the different TCP connections. From these results we see that the QoS of the real and data traffic is being met very well. This has happened when most of the links are heavily loaded.

We have simulated several other networks with different traffic mixes and have made similar observations.

0.8 Admission Control

To provide QoS guarantees to different connections, admission control is required: if a new connection request arrives and the network does not have sufficient resources to provide it the QoS requested, the service provider should reject the request. A more relaxed rule would be: limit admission control decision (to reject) to applications with real time hard constraints, e.g., IP telephony and video conferencing. For other requests (e.g., audio/video streaming, web browsing) if there are insufficient resources, one can provide throughput less than requested by them. Of course here also a service provider may decide to reject an admission request if he is unable to provide beyond a certain fraction of requested throughput. In the following we will limit ourselves to the first option: reject a request if its QoS cannot be satisfied.

Table 1: Physical Layer Parameters

Bandwidth	20 MHz
Number of Subcarriers	256
Frame Duration	10ms
No. of OFDM symbols / frame	844
No. of OFDM symbols / minislot	4
Total No. of minislots / frame	211
No. of minislots / frame for uplink Centr. Sched.	194

Table 2: Burst Profiles

burst profile No.	Modulation	Coding Rate	Uncoded bytes per OFDM symbol	Uncoded bytes per minislot
1	QPSK	1/2	24	6
2	QPSK	3/4	36	144
3	16QAM	1/2	48	192
4	16QAM	3/4	72	288
5	64QAM	2/3	96	384
6	64QAM	3/4	108	432

Table 3: Performance of UDP Flows

	CBR Flows	VBR Flows
Max Avg Delay	28.01 ms	28.09 ms
Max Drop Prob	0.000%	0.000%

Table 4: Performance of TCP Flows

Percent Error	Number of Flows
< -25	0
-20 to -10	2
-10 to 0	85
0 to 10	351
10 to 20	77
20 to 30	9
> 30	1

Our admission control rule is simple and based on only the mean throughput requirements of the new TCP/UDP connection (for the VBR connection it is the equivalent BW). When a new connection request comes (say at (s, d)), it is sent to the MBS. The MBS chooses one of the routes p for which $\alpha_p(s, d) > 0$ at present (picking p with highest $\alpha_p(s, d)$ will be desirable). Then if λ bits/slot is the mean throughput requirement of the new request and (i, j) is a link on p , we compute $\bar{n}(i, j) = \lambda/E[r(i, j)]$, the number of slots required on link (i, j) to satisfy the mean throughput requirement. These requirements are summed up for all (i, j) on p and the MBS decides if it has the necessary number of slots (beyond the requirements of existing connections) to support the new request. If it has, the request is accepted. If not, then the MBS tries other routes with $\alpha_p(s, d) > 0$ till it gets one where the requirements are satisfied. If not, the request is denied. On first thought this procedure may look strange because one may expect that a route p on which $\alpha_p(s, d) = 0$ may also satisfy the QoS requirements. But this is unlikely because our routing and scheduling algorithm picks 'good' routes to send the traffic of each (s, d) . Of course the traffic of a new UDP connection will be given priority over the TCP connections of (s, d) on the route selected.

It is possible that if a routing and scheduling decision is being run for sometime, due to new connection arrivals or departures and other reasons, the current routing/scheduling may not be efficient. Thus, the MBS will occasionally run the routing and scheduling algorithm of Section 0.3 to reroute/reschedule the total traffic of different (s, d) pairs. It may periodically do it after every (say) M frames and/or when one of the following events happens:

- (a) A node/link fails.
- (b) The $E[r(i, j)]$ of some links (i, j) and/or the mean traffic requirements $\lambda(s, d)$ of some (s, d) pairs change drastically.
- (c) The admission control policy has been blocking too many calls in recent past.

0.9 Conclusions

In this paper we have designed efficient, fair and practically implementable algorithms for routing and centralized scheduling in IEEE 802.16 mesh networks. We provide end to end QoS to different flows in the network. For this, we first provide an optimal and fair joint routing and scheduling solution to satisfy the *aggregate* mean traffic requirements of different source-destination pairs. Then we do scheduling at individual links to provide QoS to each flow. For this, we have handled UDP and TCP traffic separately at first and then jointly. Our algorithms are able to provide QoS to real and nonreal time individual flows efficiently and fairly. We have also provided an admission control policy which is an important part of any QoS framework.

0.10 Acknowledgement

The simulations in this chapter have been done by Mr Anil Kumar and Mr Siddhartha Sankaran.

References

- [1] Air Interface for Fixed broadband Wireless Access Systems, *IEEE STD 802.16*, October 2004.
- [2] Air Interface for Fixed and mobile broadband Wireless Access Systems”, *IEEE P802.16e/D12*, February 2005.
- [3] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, “Network flows: Theory, Algorithms and Applications”, *Englewood Cliffs: Prentice Hall*, 1993.
- [4] I. F. Akyildiz, X. Wang and W. Wang, “Wireless mesh networks: A survey”, *Computer Networks*, Vol 47, No.4, 2005, pp. 445-487.
- [5] M. Alichery, R. Bhatia and L. Li, “Joint channel assignment and routing for throughput optimization in multiradio wireless Mesh networks”, *in Proc. Conf. Mobicom*, 2005.
- [6] M. Andrews and L. Zhang, “Routing and scheduling in multihop wireless networks with time-varying channels”, *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, Jan 2004.
- [7] C. Barakat, E. Altman and W. Dabbous, “On TCP performance in heterogeneous networks: A survey”, *IEEE Communication Magazine*, Vol 38, Jan 2000, pp. 40-46.
- [8] M. Cao, V. Raghunathan and P.R. Kumar, “A tractable algorithm for fair and efficient uplink scheduling of multi-hop WiMax mesh networks”, *Preprint, To appear in Proceedings of WiMesh 2006: Second IEEE Workshop on Wireless Mesh Networks*, Sept. 2006, Reston, VA.
- [9] M. Cao, W. Ma, Q. Zhang, X. Wang and W. Zhu, “Modelling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode”, *ACM MobiHoc*, 2005.
- [10] D. Z. Chen, O. Daescu, Y. Dai, N. Katoh, X. Wu and J. Xu, “Efficient algorithms and implementations for optimizing the sum of linear fractional functions with applications”, *J., Combinatorial Optimization*, Vol 9, 2005, pp. 69-90.
- [11] W-P. Chen, C-F Su, R. Rabbat and T. Hamada, “Radio resource management under fixed mobile convergence architecture”, .
- [12] W. Dinkelbach, “On nonlinear fractional programming”, *Management Science*, 1967.
- [13] M. Ergen, S. Coleri and P. Varaiya, “QoS aware adaptive resource allocation techniques for fair scheduling in OFDMA based broadband wireless access”, *IEEE Transactions on Broadcasting*, Vol. 49, No. 4, Dec 2003.
- [14] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance”, *IEEE/ACM Trans. Networking*, Vol. 1, 1993, 397-413.
- [15] L. Georgiadis, M.J. Neely and L. Tassiulas, “Resource allocation and cross-layer control in wireless networks”, *Foundations and Trends in Networking*, 2006, Vol 1, No.1, pp.1-144, Now Publishers.
- [16] A. Gupta and V. Sharma, “A Unified approach for analyzing persistent, non-persistent and ON-OFF TCP sessions in the Internet”, *Performance Evaluation*, Vol. 63, 2006, 79-98.
- [17] R. Gupta, J. Musacchio and J. Walrand, “Sufficient rate conditions for QoS flows in adhoc networks”, *Preprint*, 2004.
- [18] M. Hawa and D. W. Petr, “Quality of service scheduling in cable and broadband wireless access systems”, *10th IEEE International Workshop on Quality of Service*, May 2002, pp. 247-255.
- [19] C. Hoymann, “Analysis and performance evaluation of the OFDM-based metropolitan area network IEEE 802.16”, *Computer Networks*, Vol 49, 2005, pp. 341-363.
- [20] I. Hsu and J. Walrand, “Admission control for multiclass ATM traffic with overflow constraints”, *Computer Networks and ISDN Systems*, Vol. 28, 1996, 1739-1751.
- [21] H. Jiang, W. Zhuang, X. Shen, A. Abdrabou and P. Wang, “Differentiated services for wireless mesh backbone”, *IEEE Communication Magazine*, Vol 44, No.7, July 2006, pp. 113-119.

- [22] V. Kamble and V. Sharma, "A simple approach to provide QoS and fairness in Internet", in *Proc. International Conf. on Signal Processing and Communications (SPCOM 2004)*, Bangalore, Dec. 2004.
- [23] D. Kim and A. Ganz, "Fair and efficient multihop scheduling algorithm for IEEE 802.16 BWA systems", in *IEEE Broadnets'05* Boston, 2005.
- [24] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in wireless multi-hop networks: The joint routing and scheduling problem", *ACM Mobicom*, September 2003.
- [25] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio, multichannel wireless mesh networks", in *Proc. ACM Mobicom*, 2005.
- [26] X. Lin, N. B. Shroff and R. Srikanth, "A tutorial on cross-layer optimization in wireless networks", *IEEE Journal on Selected Areas in Communication*, Vol 24, August 2006, pp. 1-12.
- [27] J. Pandhaye, V. Firoiu, D. Towsley and J. Kurose, "Modelling TCP throughput : A simple model and its empirical validation", *ACM Proc. SIGCOMM'98*, 1998.
- [28] V. Raghunathan and P. R. Kumar, "A counter-example in congestion control of wireless networks", *ACM Conference MSWiM'05*.
- [29] S. Ramanathan and E. L. Llyod, "Scheduling algorithms for multihop radio networks", *IEEE/ACM Transactions on Networking*, Vol. 1, Apr. 1993, pp. 166-177.
- [30] V. Reddy, V. Sharma and M. B. Suma, "Providing QoS to TCP and real time connections in the Internet", *Queueing Systems*, Vol. 46, 2004, 461-480.
- [31] S. R. Sandeep, "Joint routing and scheduling for multihop wireless networks", *M.E. Thesis, Electrical Communication Engineering Dept, Indian Institute of Science, Bangalore* June 2006, (Also presented in poster session of *Conference on Stochastic Networks*, Urbana, IL, June 2006).
- [32] V. Sharma and P. Punyaslok, "Stability and analysis of TCP connections with RED control and exogenous traffic", in *Queueing Systems*, Vol 48, 2004, 193-235.
- [33] H. Shetiya, "Efficient routing and scheduling algorithms for IEEE 802.16 mesh networks", *M.E. Thesis, Dept. of ECE, Indian Institute of Science, Bangalore*, 2005.
- [34] H. Shetiya and V. Sharma, "Algorithms for routing and centralized scheduling to provide QoS in IEEE 802.16 mesh networks", in *Proc. of 1st ACM workshop on Wireless Multimedia Networking and Performance Modelling (WMuNeP 2005) in 7th International Symposium on Modelling, Analysis and Simulation of Wireless and Mobile systems (ACM/IEEE MSWiM 2005)*, October 2005.
- [35] H. Shetiya and V. Sharma, "Algorithms for routing and centralized scheduling in IEEE 802.16 Mesh Networks", in *Proc IEEE Wireless Communications and Networking Conference (WCNC06)*, April 2006.
- [36] V. Singh and V. Sharma, "Efficient and fair scheduling of uplink and downlink in IEEE 802.16 OFDMA Networks", *IEEE Wireless Communications and Networking Conference (WCNC06)*, April 2006.
- [37] Jean Walrand, "An Introduction to Queueing Networks", Prentice Hall, 1988.
- [38] J. Walrand and P. Varaiya, "High Performance Communication Networks", *Morgan Kaufmann*, San Francisco , 1996.
- [39] H-Y. Wei, S. Ganguly, R. Izmailov and Z. J. Haas, "Interference aware IEEE 802.16 WiMax mesh networks", in *61th IEEE Vehicular Technology Conf., (VTC 2005)*, Stockholm, 2005.
- [40] I. C. Wong, Z. Shen, B. L. Evans and J. G. Andrews, "A low complexity algorithm for proportional resource allocation in OFDMA systems", *Signal Processing Systems (SIPS)*, 2004.
- [41] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless adhoc networks ?", *IEEE Communication Magazine*, Vol 39, No.6, June 2001, pp.130-137.
- [42] F. Xue and P.R. Kumar, "Scaling laws for adhoc wireless networks: An Information

theoretic approach”, *Foundation and Trends in Networking*, Vol. 1, No.2, 2006, pp.145-270, Now Publishers.