End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks

Saswati Sarkar* and Leandros Tassiulas†

Abstract

Sharing the locally common spectrum among the links of the same vicinity is a fundamental problem in wireless ad-hoc networks. Lately some scheduling approaches have been proposed that guarantee fair share of the bandwidth among the links. What really affects the quality of service perceived by the applications though, is the effective end-to-end bandwidth allocated to the different network sessions that span several links. We propose an algorithm that provides fair session rates in that context. The algorithm is based on a combination of a link scheduling method to avoid local conflicts, a fair session service discipline per link and a hop-by-hop window flow control scheme. It can be shown that the long term rates allocated to the different sessions are maxmin fair. All the stages of the algorithm are implementable based on local information only, except the link scheduling part that needs some network-wide coordination. Some numerical study is performed to evaluate the impact of various parameter choices on the performance of the algorithm.

1 Introduction

Link transmission scheduling in multihop wireless networks attracted a lot of attention over the last twenty years. The earlier work on the subject was focused mostly on guaranteeing end-to-end connectivity whenever that was feasible [1, 8, 11]. As the applications became more and more bandwidth hungry as well as sensitive to the perceived quality of service, there has been a lot of effort lately to obtain transmission scheduling algorithms that provide some guarantees on the effective rates enjoyed by each individual link [3, 6, 7, 15, 12, 17]. What really affects the quality of service perceived by the applications though, is the effective end-to-end bandwidth allocated to the different network sessions that span several links. We address the objective of providing maxmin fair end-to-end bandwidth to sessions.

Providing end-to-end rate guarantees in wired networks has been studied extensively [4, 10]. We combine some features of network control approaches for wired networks with wireless link scheduling techniques to design a provably fair rate allocation algorithm. The algorithm is based on a combination of a link scheduling method to avoid local conflicts, a fair session service discipline per link and a hop-by-hop window flow control scheme. Most of the stages of the algorithm are implementable based on information that is available locally in the node where the scheduling is performed, except the link scheduling part that needs to compute a maximum weighted matching of the network topology graph. When the latter computation is replaced by an approximate distributed link scheduling algorithm, then we may have a fully distributed solution with suboptimal performance.

We explain the fairness objectives and the network model in Section 2. We present scheduling strategies which attain maxmin fairness and their analytical performance guarantees in Section 3. We investigate the impact of the choice of certain algorithm parameters via simulation in Section 4. We discuss several implementation related features of our algorithms in Section 5. Refer to technical report [14] for proofs.

2 Fairness Objectives and Network Model

We consider a wireless network with V nodes, E links and N multihop sessions. The session routes are predetermined. Every node can transmit one packet per unit time, and has one radio unit. Thus, a node can be involved in a single transmission at a time, i.e., it can either transmit one packet or receive one packet or remain idle. Every node has a locally unique frequency. Thus, multiple transmissions can proceed simultaneously without any interference as long as they do not have a common node. Hence, the links which are active at any time must constitute a matching. For example, a bluetooth network satisfies the above assumptions [9]. We do not consider channel errors.

A bandwidth allocation $(r_1, \ldots, r_N)$ can be attained if there exists a scheduling sequence which attains the

*S. Sarkar is with the department of Electrical and Systems Engineering in the University of Pennsylvania, Philadelphia, PA, USA. Email: swati@ee.upenn.edu. Her work was supported in part by NSF grants ANI01-06984 and NCR02-38340.

†L. Tassiulas is with the Computer Engineering and Telecommunications department in the University of Thessaly, Voulos, Greece. Email: leandros@inf.uth.gr

corresponding rates. Hajek *et al.* [5] showed that if the network is a bipartite graph* then a bandwidth allocation $(r_1, \ldots, r_N)$ is attainable if and only if the sum of the bandwidth of all sessions traversing a node is less than or equal to 1. Many wireless networks, e.g., bluetooth networks are bipartite graphs. For non-bipartite graphs, a bandwidth allocation $(r_1, \ldots, r_N)$ can be attained if the sum of the bandwidth of all sessions traversing a node is less than or equal to $2/3$† [5]. In practice, bandwidth is allocated so as to utilize the bandwidth capacity of the nodes only partially, and leave the rest for protection against transients and overloads. Thus, combining the two cases, we will consider a bandwidth allocation $(r_1, \ldots, r_N)$ to be attainable if the sum of the bandwidth of all sessions traversing a node is less than or equal to α, where α is the desired bandwidth utilization factor ($\alpha \leq 2/3$ for non-bipartite graphs and $\alpha \leq 1$ for bipartite graphs). We refer to this constraint as the *node capacity constraint.*

If a session i generates packets at rate ρ_i, then its bandwidth r_i is upper bounded by ρ_i, $r_i \leq \rho_i$. We refer to this constraint as the *demand constraint.*

A bandwidth allocation is feasible if and only if it satisfies both the demand and the node capacity constraints.

Every session i has weight G_i, $G_i > 0$. *A feasible bandwidth allocation $(r_1, r_2, \ldots, r_N)$ is maxmin fair, if it satisfies the following property w.r.t. any other feasible bandwidth allocation $(s_1, \ldots, s_N)$: if there exists a i such that $r_i < s_i$, then there exists a j such that $r_j/G_j \leq r_i/G_i$ and $s_j < r_j$.*

Unequal weights allow allocation of bandwidth on the basis of the quality of service requirements. If all the weights are equal, then under maxmin fair allocation, the sessions can have unequal bandwidths only because of different congestions in their paths and different packet generation rates. Under unequal weights, a session with a higher weight can have a higher bandwidth than another even if the latter travels the same path and generates packets at the same rate.

A node is a bottleneck node of a session i if the weighted bandwidth allocated to session i (r_i/G_i) is the maximum among the weighted bandwidth allocated to all sessions traversing the node and the sum of the bandwidth of the sessions traversing a node is equal to the bandwidth utilization factor α. We present a necessary and sufficient condition for maxmin fairness of multihop sessions, which is similar to the one that holds for wireline sessions [2].

*A bipartite graph is one where the vertex set can be partitioned in two sets such that there is no edge connecting the vertices in the same set.

†The result is sufficient, but not necessary, i.e., a bandwidth allocation can be attained even if this condition is not satisfied.

Lemma 1 *A bandwidth allocation is maxmin fair if and only if the following holds: for every session i, either the bandwidth allocated to session i is equal to ρ_i or the session has a bottleneck node.*

3 A back-pressure based fair bandwidth allocation algorithm

We propose a two-tier approach for attaining maxmin fairness for multihop sessions. The first step computes the maxmin fair bandwidth share of each session in each node on its path, and releases packets for transmission in accordance with these fair shares. The second step schedules the transmission of the released packets so as to attain the fair shares. This modularization enables us to use different algorithms for attaining different fairness objectives (e.g., maxmin fairness with different weights) in the first step, and attain the bandwidth shares computed as per the desired objective using the existing the maximum difference backlog scheduling [16] in the second step. This scheduling can stabilize the network for any feasible arrival proces. Since the packet release process is fair and hence feasible, the overall framework attains maxmin fairness. We present the basic algorithm and its performance guarantees in subsection 3.1. We consider generalizations in subsections 3.2 and 3.3.

3.1 Basic Algorithm

In this subsection we consider the special case that all sessions have equal weights, and the source node of each session has an infinite supply of packets ($\rho_i = \infty$, $\forall \ i$,). The algorithm has been presented in Figure 1. Here, we describe each part. Fair bandwidth is computed by a token generation process. Every node generates tokens for all sessions traversing the node. The token generation process is so designed that the tokens are generated for each session at its maxmin fair rate. Whenever a new token is generated for a session at its source node, the source node releases a new packet for transmission. Thus the packet release process is maxmin fair as well. Only the "released" packets are eligible for transmission.

We now describe the token generation process, and explain why the rate of token generation equals the maxmin fair rate. The maxmin fair rate of a session is determined by the bandwidth offered by its bottleneck node, which happens to be the most congested node on its path. Intuitively, the token generation rate for a session at any node on its path should equal that at the bottleneck node. The challenge is to attain this equality at every node on the path of a session without explicit information about the bottleneck node. A node learns this information implicitly by relating the token generation process for a given session to that at

Procedure Token Generation (node m)
begin
Sample each session traversing node m in round robin order.
Let session i traverse node m and
nodes l, n be adjacent to node m in the path of session i,
When session i is sampled in slot t:
if $C_{i,m}(t) < C_{i,l}(t) + W$ and $C_{i,m}(t) < C_{i,n}(t) + W$ **then**
Generate a token for session i in slot t ($C_{i,m}(t+1) = C_{i,m}(t)+1$;
else
Do not generate token for session i ($C_{i,m}(t+1) = C_{i,m}(t)$) and
Sample the next session in the round robin order
end

Procedure Packet Release (source i)
begin
Release a new session i packet for transmission at session i source when a new token is generated at the source.
end

Procedure Packet Scheduling For Transmission (link e)
begin
Let link e be between nodes m and n,
$\mathcal{L}_e$ be the set of sessions traversing link e and
$P_{i,n}(t)$ be the number of released packets of session i waiting at node n at time t.
$W_e(t) = \max_{i \in \mathcal{L}_e} (P_{i,m}(t) - P_{i,n}(t))$ (/*Weight of link is the maximum difference in backlog across the link */)
Schedule the links which constitute a maximum weighted matching
If link e is scheduled then,
transmit a packet of session j from node m to node n if
$P_{j,m}(t) - P_{j,n}(t) = \max_{i \in \mathcal{L}_e} (P_{i,m}(t) - P_{i,n}(t))$ /*session j has the maximum difference of backlog across e*/
end

Figure 1: Pseudo code of the fair bandwidth allocation algorithm for saturated sessions

the adjacent nodes on the path of the session.

We describe the token generation process for a session i at node m. We consider slotted time. Node m samples in round robin order every session traversing it. Node m samples one session in a slot. Let l and n be the nodes adjacent to node m on the path of session i (i.e., one is the immediate upstream and the other is the immediate downstream node). Let $C_{i,p}(t)$ be the total number of tokens generated for session i at node p in the interval $[0, t]$. Let node m sample session i in slot t. Then, m generates a token to session i in slot t if and only if $C_{i,m}(t) < \min(C_{i,l}(t), C_{i,n}(t)) + W$. Thus, session i receives a token unless the number of tokens for session i at node m substantially exceeds that at the adjacent nodes. This "prohibitive" difference is a window parameter W. Note that the source(destination) node of a session has only one adjacent node for the session. Thus such a node takes the token generation decisions based on the number of tokens at only one adjacent node. Tokens are never removed from a node.

It follows from the token generation process that the number of tokens for a session at two adjacent nodes on the path of the session differ by W or less at any time t, and the difference is at most LW for any two nodes on the path of a session, where L is the number of hops in the session path. Thus the rates of token generation for a session are equal at any two nodes on the path of the session. Since the bottleneck node is the most congested, the sampling rate for a session is the least at its bottleneck node, and hence the token generation rate at each node is upper bounded by this sampling rate. It turns out that this sampling rate is maxmin fair, and furthermore the token generation rate of a session exactly equals the sampling rate at its bottleneck node. Thus the token generation rate is maxmin fair for each session as well.

Lemma 2 *Let $r_1, r_2, \ldots r_N$ be the maxmin fair rates of the sessions. Let $C_{i,n}(t)$ be the number of tokens for session i at node n at time t. Let $W \geq W_0$, where W_0 is a constant whose value depends on the system parameters. Then, in any interval (x, y)*

$$|C_{i,n}(y) - C_{i,n}(x) - r_i(y - x)| \leq \varrho$$

Here, ϱ is a constant whose value depends only on the topology and not on the interval (x, y).

The implicit discovery of the bottleneck information from the bandwidth allocation process at neighboring nodes has been motivated by fair bandwidth allocation algorithms in wireline networks [4]. This is commonly termed as "back-pressure."

Whenever the source node of a session receives a new token, it releases a new packet. The maximum backlog based scheduling [16] transmits the released packets along the pre-specified routes to the desired destinations. The maximum backlog based scheduling assigns a weight to each link as follows. The difference in backlog of a session in a link is equal to the difference between the number of released packets of the session waiting at the source node of the link and that at the destination node of the link. The weight of a link is the maximum difference in backlog of a session in the link. Note that only the source node may have packets which have not been released. The links which constitute a maximum weighted matching are scheduled for service. When a link is scheduled, a packet of the session with the maximum difference in backlog in the link is served. It has been shown that the maximum backlog based scheduling stabilizes a network as long as the packet arrival process is feasible [16]. The packet arrival process in the current network is the packet release process. Tokens and hence packets are released for each session at the maxmin fair rate (Lemma 2) which is feasible by definition. Thus the stability result indicates that the system attains the maxmin fair bandwidth.

Theorem 1 *Let $r_1, r_2, \ldots r_N$ be the maxmin fair rates of the sessions. Let $D_i(t)$ be the number of packets for session i which have reached destination by time t. Then if $W \geq W_0$ in any interval (x, y),*

$$|D_i(y) - D_i(x) - r_i(y - x)| \leq \kappa$$

Here, W_0 and κ are constants whose values depend only on the topology, and not on the interval (x, y).

Theorem 1 shows that in any interval the total number of packets of a session delivered to the destination differs from the maxmin fair number by at most a constant. Thus, the long term rates are maxmin fair.

3.2 Generalization for addressing the unsaturated case

We mention the necessary modifications to address the case when the source nodes do not always have packets for transmission ($\rho_i < \infty$ for some i). Only the packet release process and the token generation at the source node need to be altered. The source node of a session may not have a new packet to release whenever it generates a new token for the session. So it stores such "unused tokens" for release of future packets. The "used tokens" are those which have been used for packet release. When a packet is generated at the source node, it is released for transmission if there is an outstanding unused token for the session, and the status of the corresponding token becomes used. If there is no unused token, then the packet waits for the generation of a new token. If a source has W unused tokens for a session, then the token generation process does not release a new token for the session. Also, the node immediately downstream of the source considers only the total number of tokens for the session at the source node while deciding whether or not to generate a token for the session. Refer to Figure 2 for the modification.

If a session has a low rate of packet generation, then the system generates fewer tokens for the session at the source because of the restriction on the number of unused tokens. Hence, the session receives fewer tokens at other nodes as well, since back-pressure upper bounds the difference between the number of tokens for a session in any two nodes by a constant LW. Thus the session obtains fewer transmission opportunities, and thus less bandwidth as required by the definition of maxmin fairness.

The analytical service guarantees presented in Lemma 2 and Theorem 1 hold for pseudo-deterministic(ρ, σ) arrival processes, i.e., if the total number of packets arriving for any session i in any interval of length t is upper bounded by $t\rho_i + \sigma_i$ and lower bounded by $t\rho_i - \sigma_i$ for arbitrary t. Here, ρ_i is the long term arrival rate of a session i and σ_i is the burstiness.

Procedure Token Generation at Source Node (source i)
begin
Let node n be the source of session i.
Sample each session traversing node n in round robin order.
Token generation procedure is similar for all sessions other than i.
Let node l be the immediate downstream of node n in the path of session i.
When session i is sampled in slot t:
if $C_{i,n}(t) < C_{i,l}(t) + W$ and $C_{i,n}^{\text{unused}}(t) < W$ **then**
Generate a token for session i in slot t ($C_{i,n}(t+1) = C_{i,n}(t)+1$);
if no session i packet is waiting for transmission at node n, **then**
Increment the number of unused tokens of session i ($C_{i,n}^{\text{unused}}(t+1) = C_{i,n}^{\text{unused}}(t) + 1$)
else
Release a session i packet for transmission
else
Do not generate token for session i ($C_{i,n}(t+1) = C_{i,n}(t)$) and Sample the next session in the round robin order
end

Procedure Handling Packet Generation (source i)
begin
When a new packet is generated at the source n of session i,
if there is an unused token for session i ($C_{i,n}^{\text{unused}}(t) > 0$) **then**
Release the new packet for transmission
Decrement the number of unused tokens ($C_{i,n}^{\text{unused}}(t+1) = C_{i,n}^{\text{unused}}(t) - 1$)
else
Store the new packet for future release
end

Figure 2: The figure shows the pseudo code of the token generation process at the source nodes for systems with un-saturated sessions.

3.3 Generalization for addressing the case with unequal weights

The sampling procedure in the basic algorithm presented in Section 1 must be altered to attain the maxmin fair rates when the sessions have unequal weights. Let the weight of session i be G_i. Node n samples the session which has the minimum weighted number of tokens, i.e, the minimum value of $C_{i,m}(t)/G_i$ among all sessions i traversing the node. Thus, the sessions with higher weights are sampled more often. The rest of the algorithm remains the same, As before, back-pressure is used to improve the bandwidth allocation of the less congested sessions without reducing the bandwidth of the more congested ones. Theorem 1 and Lemma 2 hold.

4 Performance Evaluation

We now examine using simulations (a) the time required for convergence of the computed rates to the maxmin fair rates and (b) the impact of the choice of the window parameter (W) on the convergence result. We have designed a simulator in C for this purpose. The first investigation has been motivated by the fact that we do not have a tight analytical bound on the

convergence time. The lower bound on W needed to guarantee the converge results (Theorem 1), depends on the system parameters like the number of sessions, the length of session paths and arrival parameters, etc. Thus, this bound is impossible to compute without explicit knowledge of the network topology. This motivates the investigation of the sensitivity of the convergence towards the choice of the window parameter W.

We present simulation results for a network of 21 nodes and 14 sessions as shown in Figure 3. Here, $W = 5$. We focus on the token generation procedure only. The maximum backlog based scheduling [16] has been known to attain any feasible rate as long as the packet arrival process is feasible. We consider the relative difference between the long term token generation rate for each session i at its source $(C_{i,n}(t)/t)$ and the maxmin fair rate (r_i). The relative difference, which we call relative error, at time t for session i is $|1 - \frac{C_{i,n}(t)}{r_i t}|$. We plot the maximum and average relative errors over all sessions as a function of time t in Figure 3. In Figure 3, the second and fourth figures consider the case when all the sessions are saturated. The third figure considers the case when all sessions are saturated except session 7 which receives packets at the rate 0.1 per unit time. All figures consider sessions with equal weights, except the fourth figure, where $G_i = 2$ if $i = 7$ and $G_i = 1$, if $i \neq 7$.

We make the following observations from Figure 3. The average relative error decays fast, e.g., it is less than 5% within 500 slots. The maximum relative error decays somewhat slower indicating that a few sessions experience slower convergence. The rates of token generation converge to the maxmin fair rates even though $W = 5$, while the lower bound W_0 for guaranteed convergence is 2^{70}! [14]. We observed similar trends for several other topologies. The following are the conclusions. The token generation rate converges to the maxmin fair bandwidth rapidly on an average. Also, in practice convergence is not sensitive to the choice of W and moderate values of W in the range of 5 to 10 ensure convergence. Thus, small window sizes can now be used to control the delay and buffer requirements.

5 Discussion and Conclusion

The computation of the fair bandwidth via token generation and the scheduling can operate in parallel. A sequential operation increases the overall delay in attaining the desired bandwidth allocation.

A dynamic scenario can be accommodated where sessions can join and leave any time, as the overall scheme need not restart for any such change, and the analytical guarantees hold.

The performance guarantees hold even when a node knows the number of tokens at its neighbors only at a previous time instant, as long as the time lag is upper bounded. We have shown in [13] that the rates obtained by a similar back-pressure technique converges irrespective of the feedback delay.

A node can execute the token generation and the packet release processes with the knowledge of the status of its one-hop neighbors only. However, the maximum difference backlog based scheduling is a centralized procedure. The execution of the token generation and the packet release processes are not tightly coupled to this particular scheduling. Thus, future research will be directed towards the investigation of a distributed scheduling strategy which can be used in conjunction with the token generation scheme.

The system does not remove any token. Thus, the register storing the number of tokens may overflow. The performance guarantees hold if the tokens are removed without affecting the difference in the number of tokens for a session in any two nodes in its path. The removal process can be executed by exchanging synchronization information periodically. The additional system overhead is marginal as the periods can be long.

References

[1] D. J. Baker and A. Ephremides. The architectural organization of a packet radio network via a distributed algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, November 1981.

[2] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1987.

[3] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair-queueing algorithm. *Proceedings of ACM SigComm 1989*, 19(4):1–12, September 1989.

[4] E. Hahne. Round-robin scheduling for max-min fairness in data networks. *IEEE Journal on Selected Areas in Communications*, 9(7):1024–1039, September 1991.

[5] B. Hajek and G. Sasaki. Link scheduling in polynomial time. *IEEE Transactions on Information Theory*, 34(5):910–917, September 1988.

[6] X.L. Huang and B. Bensaou. On max-min fairness and scheduling in wireless ad-hoc networks: Analytical framework and implementation. In *Proceedings of IEEE/ACM MobiHoc, 2001*, pages 221–231, Long Beach, CA, October 2001.

[7] N. Johansson, U. Korner, and L. Tassiulas. A distributed scheduling algorithm for a bluetooth scatternet. In *Proceedings of the 17th International Teletraffic*

Congress, 2001, Salvador da Bahia, Brazil, December 2001.

[8] J. Ju and V.O.K. Li. An optimal topology transparent scheduling method in multihop packet radio networks. *IEEE Transactions in Networking*, 6(3):298–306, June 1998.

[9] B. Miller and C. Bisdikian. *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications*. Prentice-Hall, 2000.

[10] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control - the single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[11] M. Post, P. Sarachik, and A Kershenbaum. A biased greedy algorithm for scheduling multihop radio networks. *Proceedings of 19th Annual Conference on Information Sciences and Systems, Johns Hopkins Univ.*, March 1985.

[12] A. Racz, G. Miklos, F. Kubinszky, and A. Valko. A pseudo random coordinated scheduling algorithm for bluetooth scatternets. In *Proceedings of IEEE/ACM MobiHoc, 2001*, pages 193–203, Long Beach, CA, October 2001.

[13] S. Sarkar and L. Tassiulas. Back pressure based multicast scheduling for fair bandwidth allocation. In *Proceedings of INFOCOM, 2001*, pages 1123–1133, Anchorage, Alaska, May 2001.

[14] S. Sarkar and L. Tassiulas. End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks. *University of Pennsylvania, Technical Report, (http://www.seas.upenn.edu/∼swati/publication.htm)*, 2002.

[15] X. Gao T. Nandagopal, T. Kim and V. Bharghavan. Achieving mac layer fairness in wireless packet networks. In *Proceedings of ACM Mobicom, 2000*, pages 87–98, Boston, MA, August 2000.

[16] L. Tassiulas and A. Ephremidis. Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):165–168, 1992.

[17] L. Tassiulas and S. Sarkar. Maxmin fair scheduling in wireless networks. In *Proceedings of INFOCOM'2002*, pages 763–772, New York, NY, June 2002.

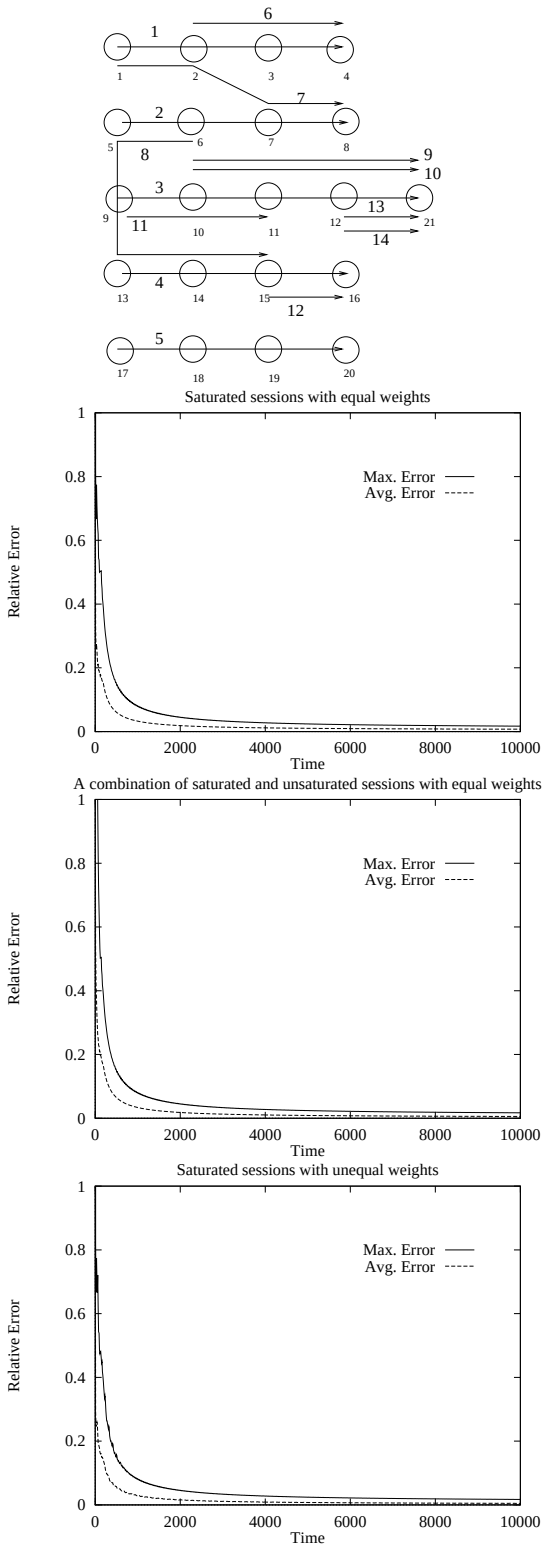

Figure 3: The first figure shows the topology with 21 nodes and 14 sessions which is used in the simulations. The second and the fourth figures consider the case when all the sessions are saturated. The third figure considers the case when all the sessions are saturated, except session 7 which receives packets at the rate 0.1 per unit time. All sessions have weight 1 except in the fourth figure, where session 7 has weight 2.