Scheduling in Wireless Networks with Multiple Transmission Channels

Satish Damodaran and Krishna M. Sivalingam [*]
School of EECS, Washington State University, Pullman, WA 99164-2752

Abstract

This paper describes scheduling algorithms for scheduling traffic in wireless data networks with multiple channels per cell. The paper assumes that a reservation-based MAC protocol is used. The main objective of the scheduling algorithms is to reduce the computation time while maximizing the utilization of the network resources, thereby improving the system throughput. In this paper, we consider two type of systems – single priority and multiple priority. For each system, two different algorithms are considered depending on whether slots are allocated contiguously or not. A performance study that considers network utilization, computation time, and the throughput for 2 Mbps and 10 Mbps datastreams for the above algorithms is presented.

1 Introduction

Wireless Local Area Networks (WLANs) have become an increasingly important technology for today's computer and communications industry. The wireless channel is typically used as a shared medium because of the limited bandwidth. The channel can be shared among the nodes using either a random access method such as Slotted Aloha or by scheduling the channel to the users based on requests with or without priorities. Several scheduling algorithms [2,5–8] have been studied for organizing the traffic from the mobile nodes to the base stations and vice versa. However, the earlier research did not consider scheduling traffic in wireless data networks with multiple channels per cell. We assume that the mobile nodes have a tunable transmitter and a tunable receiver. The base station has a set of fixed frequency transmitters and receivers – one per channel. This paper addresses the scheduling of channels for such a system. The scheduling algorithms are typically used with a reservation based medium access control (MAC) protocol [9].

Note that multiple channels per cell have been considered for cellular networks, where a channel is allocated to a user in a circuit-switched mode. We focus on scheduling data traffic over multiple channels in a packet switched environment.

The scheduling algorithms are designed for two type of systems – single priority and multiple priority traffic systems. Two algorithms called the Contiguous Allocation Single Priority (CASP) algorithm and the Non-contiguous Allocation Single Priority (NCASP) algorithm are proposed for systems with single priority requests. Their performance is compared with the well-known MULTI-FIT algorithm [3]. These two algorithms are then extended to systems with multiple priority requests – referred to as the Contiguous Allocation Multiple Priority (CAMP) and the Non-contiguous Allocation Multiple Priority (NCAMP) algorithms. A performance study that considers network utilization, computation time, and the throughput for 2 Mbps and 10 Mbps datastreams for the above algorithms is presented.

2 Problem Formulation

This section describes the EC-MAC access protocol for the multiple-channel system and the scheduling problem statement.

The access protocol is defined for an infrastructure network with a single base station serving mobiles in its coverage area. EC-MAC was designed to minimize energy consumption and provide QoS support and is based on reservation and scheduling mechanisms. The EC-MAC protocol performance in [9] was studied with a single channel per cell. We are considering the protocol framework extended to multiple channels.

Transmission in EC-MAC protocol is organized by the base station into frames. Each frame has a fixed number of slots, where each slot equals the basic unit of wireless data transmission. The frame is divided into multiple phases as described below. In the first phase, a frame synchronization message (FSM) is transmitted on the downlink and it contains framing, synchronization and other control information. The request/update phase comprises the uplink request transmissions from the mobiles to the base station

[*]Corresponding Author. Part of the research was supported by Air Force Office of Scientific Research grants F-49620-97-1-0471 and F-49620-99-1-0125. Email: krishna@eecs.wsu.edu. The first author is presently with Cisco Systems, San Jose, CA 95134.

and operates in a collision-less mode. New mobiles that just entered the area covered by the base station register themselves with the base station in the New-User phase. The schedule is then computed by the base station during the schedule computation phase and the computed schedule is broadcast to all the mobiles. The data phase is composed of both uplink and downlink traffic.

With multiple channels available per cell, the frame synchronization, reservation, and schedule messages are sent over a fixed channel, referred to as the *control* channel. All mobiles tune into this channel during the appropriate phases. The downlink and uplink data phases can be frequency division duplexed by allocating a subset of channels to the uplink and downlink. Alternatively, the channels can be used in a time-division duplexed manner, with downlink and uplink transmissions separated in time, as in single-channel systems. We are considering the latter approach in this paper.

The notations used in the following sections are explained here. Let M denote the number of mobiles, C the number of *data* channels (in addition to the control channel), and P the number of priorities. The basic unit of transmission is a slot. Each request can be no larger than K slots. $\mathcal{L}$ denotes the schedule length, in slots, computed by a given algorithm, and Ω denotes the number of unused slots in a given schedule.

The problem considered is: Given traffic demand information obtained during the reservation phase, how do we efficiently schedule this traffic on multiple channels?

Scheduling Problem: The uplink requests made by the M mobiles to the base station and the downlink traffic from the base station to the M mobiles over the C channels are represented by a $M \times 2P$ matrix. This matrix is also referred to as the *traffic demand* matrix (or the *demand* matrix). The first P columns in the matrix indicate the slot requests from the M mobiles to the base station (uplink traffic). The second set of P columns indicate the downlink slot requests for transmission from the base station.

We now need to find a conflict-free assignment of the requests on the C channels such that the frame length is minimized. Since uplink transmissions and downlink transmissions are separated in time domain, they can be scheduled independent of each other. The rest of this paper considers only uplink transmissions with similar results applicable to the downlink.

This scheduling problem is similar to one of the basic, well-studied problems of scheduling theory, that of non-preemptively scheduling 'M' independent tasks on 'C' identical, parallel processors. The objective is to minimize the total time required to complete all the tasks. This problem is known to be NP-complete [10] and approximations to this problem such as MULTI-FIT [3] for finding near-optimal schedules have been studied earlier.

3 Scheduling for Single Priority Systems

We first consider scheduling algorithms for a system supporting single priority ($P = 1$) in this section. The data structures for the scheduling algorithm are:

1. Let $D = (d_i)$ denote the demand vector, and let d_i denote the request made by mobile i where $0 \leq d_i \leq K, i = 1, \ldots, M$.
2. Let $S = (S_i)$ denote the allocation vector, and let S_i denote the total amount of slots currently allocated to channel i where $0 \leq S_i, i = 1, \ldots, C$.

Definition 1 *The Optimal Schedule Length,* $\Phi = max(K, (\sum_i d_i/C))$.

Definition 2 *A channel C_k is said to overflow if $S_k > \Phi, k = 1, 2, \ldots, C$.*

Definition 3 *The channel capacity,* $\bar{\Phi} = (\sum_i d_i/C) + \min_j d_j$.

3.1 The Multi-Fit Algorithm [3]

The MULTI-FIT [3] algorithm uses the first fit decreasing (FFD) technique to sort the different requests made by the mobiles in decreasing order. An *expansion factor* is also defined (as the least amount by which the optimal schedule length can be enlarged). This guarantees that FFD will use all the C channels for balancing the demands of the M mobiles. Then a packing is built (while taking the expansion factor into account) by treating each element and assigning it to the lowest indexed channel, that does not violate the channel capacity constraint.

Theorem 1 *The time complexity of the MULTI-FIT algorithm as applied to this problem is $O(M \lg M + MC)$.*

Proof. Complexity of the MULTI-FIT algorithm is determined as follows:

1. Computing channel capacity, $\bar{\Phi}$ using the demand vector is of the order $O(M)$.
2. Sorting the demand vector is of the order $O(M \lg M)$.
3. Assigning an element from the demand matrix to one of the C channels is of the order of $O(C)$. Since we are assigning M elements to C channels we have, $O(MC)$.

This indicates that the overall time complexity of the MULTI-FIT algorithm is of the order $O(M \lg M + MC)$.
$\blacksquare$

```
Algorithm CASP
    For ($j = 1, 2, \dots, C$) $S_j = 0$ Endfor
    For ($i = 1, 2, \dots, M$)
        find $k$, such that $S_k \leq S_l, \forall l = 1, 2, \dots, C$.
            In the event of a tie, choose $k$ randomly.
        Allocate $d_i$ on channel $C_k$
        Update $S_k = S_k + d_i$
    Endfor
    Compute $\mathcal{L}$ = $\max_{l=1}^{C} S_l$; $\Omega$ = $C * \mathcal{L}$ - $(\sum_i d_i)$
End.
```

Figure 1. Contiguous allocation single priority (CASP) algorithm.

3.2 Contiguous allocation algorithm for single priorities (CASP)

The contiguous allocation algorithm for single priority systems (CASP) maintains the allocation vector S listed earlier. For a given request, the scheduling algorithm allocates the requests contiguously (without any splitting of the requests) on the channel which currently has the least partial sum of allocation. If more than one channel has the same least partial sum, one channel is chosen at random for the next allocation. The algorithm is described as pseudo-code in Fig. 1.

Theorem 2 *The time complexity of the CASP algorithm is* $O(M \lg C)$.

Proof. The C partial sums can be maintained as a min-heap data structure, so that we have $O(1)$ time for fetching the minimum element and $O(\lg C)$ for updating the min-heap. The total time spent in maintaining the heap for M requests is $O(M \lg C)$. ∎

Theorem 3 *Each channel overflows at most once. That is,* $\forall k, S_k > \Phi$ *at most once.*

Proof. If a channel is overflowing because of request allocation, there is at least one other channel that is underflowing. Therefore, the overflowing channel would not be among the next $C - 1$ channels which will be considered for resource allocation, i.e. $\exists k$, $S_k < \Phi$. ∎

Theorem 4 *An upper bound for the schedule length computed by CASP algorithm is* $\Phi + K$.

Proof. Follows from Theorem 3. Since $\forall k$, $S_k \geq \Phi$ at most once, the worst case will correspond to the largest entry, K. This implies that, $S_k \leq \Phi + K, \forall k$. Therefore, schedule length computed by this algorithm can never be greater than $\Phi + K$. ∎

From the theorem, it follows that the schedule length will be greater than the optimal value, by a factor of at most $\frac{K}{\Phi}$. If the standard deviation between the requests is small (all mobiles in the system approximately ask for the same number of slot requests - say x), then the percentage overflow will be $\frac{x}{Mx/C}\% = \frac{C}{M}$. This implies that if the number of channels is around 4 or 8 and the number of mobiles in the system is around 100, the percentage overflow will be around 10%.

```
Algorithm NCASP

    Calculate $\Phi$ = $max(K, \sum_i d_i/C)$
    For ($j = 1, 2, \dots, C$) $S_j = 0$ Endfor
    For ($i = 1, 2, \dots, M$)
        find $k$, such that $S_k \leq S_l, \forall l = 1, 2, \dots, C$.
            In the event of a tie, choose $k$ randomly.
            Allocate $d_i$ on channel $C_k$
        If ($S_k + d_i > \Phi$)
            Split an earlier assigned request
            in channel $C_k$ s.t. $S_k^{'} + d_i = \Phi$.
            after the split
            Find $l$ such that, $S_l \leq S_t, \forall t = 1, 2, \dots, C$
            and (t != k)
            Relocate the split request(s) to $C_l$
            Update $S_k = \Phi$
    Endfor
    Compute $\mathcal{L}$ = $\max_{l=1}^{C} S_l$
    Compute $\Omega$ = $C * \mathcal{L}$ - $(\sum_i d_i)$.
End.
```

Figure 2. Non-contiguous allocation single priority (NCASP) algorithm.

3.3 Non-contiguous allocation algorithm for single priorities (NCASP)

The non-contiguous allocation algorithm for single priority systems (NCASP) is very similar to the CASP algorithm except that the CASP algorithm allocates a request in contiguous slots, while NCASP algorithm uses non-contiguous allocations. When allocating a request to a channel, C_k, we compare it against Φ. If allocating this request will cause S_k to exceed Φ by a certain amount (the overflow), we try to avoid the situation by:

1. Allocating the current request in its entirety.

2. Splitting an earlier assigned request (if it is big enough to yield as much slots as the overflow) in the same channel so that the overflow in the current channel is reduced to zero.

3. If the above is not possible, i.e. a single request is not big enough to yield as much slots as the overflow, we need to walk back through the channel allocation and pick multiple relocatable requests, until we get enough slots to eliminate the overflow.

4. Relocating the selected requests to a next channel which will not overflow.

The algorithm is described as pseudo-code in Fig. 2.

Theorem 5 *The time complexity of the NCASP algorithm lies between $O(M \lg C)$ and $O(M^2 \lg C)$.*

Proof. Let β be the number of splits. As before, the C least partial sums can be maintained as a min-heap data structure, so that we have $O(1)$ time for fetching the minimum element and $O((M\beta) \times \lg C)$ for maintaining the min-heap. Therefore, the overall algorithm complexity is between $O(M \lg C)$ and $O(M^2 \lg C)$. The average case is at present not derived, but we are using empirical evidence from simulations to derive this result. ∎

One of the key requirements of the NCASP algorithm is that sub-requests assigned to two different channels do not overlap in time. This is needed because a mobile cannot tune to two different frequencies at the same time when transmitting or receiving data from the base station. This is the reason for the algorithm to relocate an earlier assigned request, instead of splitting the current request and assigning it to a different channel. We now prove that, split requests do not overlap in time.

Theorem 6 *For a request that has been split across two different channels, the split sub-requests do not overlap in time.*

Proof. Fig. 3 (a) shows a system with three channels C_1, C_2, C_3. The allocation vector S at this stage of the algorithm contains S_1, S_2, S_3. The next incoming request d_r is being assigned to C_2 as shown in Fig. 3 (b), since $S_2 < S_1$ and $S_2 < S_3$. Let $S_2 + d_r > \Phi$, so we split an earlier assigned request. The split, s, will be the excess over Φ i.e. $S_2 + d_r - \Phi$. s is relocated to C_1 and d_r is assigned to C_2 as shown in Fig. 3 (c). Channel C_1 is chosen because it has the next least partial sum.

Observe that, $S_2 < S_1$ when d_r was assigned to C_2. So any region from where s was split, would only cause that region to grow backwards in time. Any region to where s was assigned would be in one of the other channels and would have been greater than S_2 (S_1 in this case) before s was assigned to it. So this region grows forward in time, leaving no chance for an overlap to occur. ∎

This section presented the two algorithms, CASP and NCASP, for scheduling single priority traffic requests.

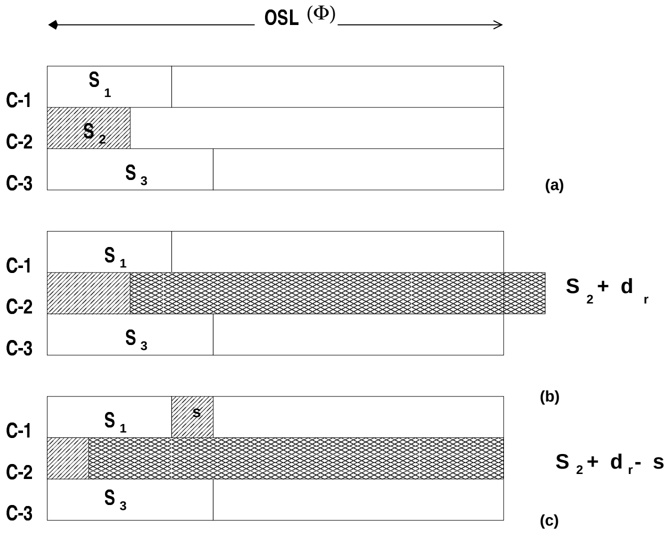


Figure 3. (a) Channels states before request d_r has been assigned. (b) Channel C_2 has been chosen for assigning request d_r. (c) An earlier assigned request has been split in channel C_2 and the split region has been relocated to C_1.

4 Scheduling for Multiple Priority Systems

In this section, we consider priority amongst packets which compete for slots in the C channels. A mobile may have multiple communication sessions from applications like telnet, ftp, email, Internet Phone, Web Browsing and Video Conferencing. Each of the applications may have different need and service characteristics. The Differentiated Services [1] model is currently being studied for Internet, based on such a priority mechanism. In this section, we extend the scheduling algorithms of Section 3 to accommodate multiple priority sessions of a mobile sharing the multiple channels. Each session traffic within a mobile is organized in the order of priority.

The data structures used are:

1. Let $D = (d_{ij})$ denote the demand matrix, and let d_{ij} denote the request made by mobile i for priority j traffic, $0 \le d_{ij} \le K$, $i = 1, \ldots, M$ and $j = 1, \ldots, P$, P represents the number of priorities. The actual value of P depends upon implementation of the priority scheme chosen.

2. Let $S = (S_i)$ denote the allocation vector as earlier.

The idea behind the scheduling algorithm is to schedule traffic of priority 1 first, followed by that of priority 2 and so on. More sophisticated scheduling mechanisms may be considered, which we defer to future work.

Contiguous allocation algorithm for multiple priorities (CAMP): The contiguous allocation algorithm for multiple priority systems (CAMP) uses the CASP algorithm to obtain the schedule for priority 1 requests. Once this is done, it computes the schedule length, $\mathcal{L}$, of the new schedule, initializes S_i = $\mathcal{L}$, ($i = 1, 2, \ldots, C$), and then continues by allocating the requests in the next priority queue.

Theorem 7 *The time complexity of the CAMP algorithm is* $O(PM \lg C)$.

Proof. The C partial sums can be maintained as a min-heap data structure, so that we have $O(1)$ time for fetching the minimum element and $O(\lg C)$ for updating the min-heap. The total time spent in maintaining the heap for M requests (per priority queue) is $O(M \lg C)$. So, for P priority queues, we have $O(PM \lg C)$. ■

Non-contiguous allocation algorithm for multiple priorities (NCAMP): The non-contiguous allocation algorithm for multiple priority systems (NCAMP) has the NCASP algorithm as the basis, and is extended to handle multiple priority queues within a mobile, as with CAMP.

Theorem 8 *The time complexity of the NCAMP algorithm is between* $O(PM \lg C)$ *and* $O(PM^2 \lg C)$.

Proof. Let β be the number of splits. As before, the C least partial sums can be maintained as a min-heap data structure, so that we have $O(1)$ time for fetching the minimum element and $O((M\beta) \times \lg C)$ for maintaining the min-heap. Therefore, the overall algorithm complexity is between $O(M \lg C)$ and $O(M^2 \lg C)$. Since we have P priority queues, the algorithm complexity lies in between $O(PM \lg C)$ and $O(PM^2 \lg C)$. The average case is unknown, but we are using empirical evidence from simulations to derive this result. ■

5 Performance Analysis

We study the performance analysis of the algorithms based on C-language implementation. The performance metrics studied include the actual schedule computation time obtained by executing the algorithms on an UltraSPARC workstation. The following sections describe the system parameters varied, the metrics studied, and the performance evaluation.

The system parameters varied include M: the number of mobiles, C: the number of channels, the transmission rate per channel, and K the maximum value over all entries in the traffic matrix.

The traffic matrix generation program is provided the following parameters as input: M, K and the number of matrices to be generated. Each entry in the matrix is a random number between 0 and K (both inclusive). The program generates the specified number of matrices, which are then applied as input to the algorithms. Effectively, the performance simulation is conducted for a system with heavy load. In the study below, the metrics are calculated using 1000 matrices per system parameter combination.

Performance Parameters Studied:

Schedule Length: represented by $\mathcal{L}$, denotes the number of slots in the data phase as determined by the scheduling algorithm.

Schedule efficiency: denoted by U, is defined as the number of slots actually utilized for packet transmission in a schedule. If a total of R slots was requested by all mobiles, and the schedule length is $\mathcal{L}$, efficiency is given by: $U = \dfrac{R}{\mathcal{L} * C}$

Computation Time: denoted by $\mathcal{T}$, is the time between the end of the reservation phase and the start of the data phase, spent in computing the schedule. Time is reported in microseconds, and is computed using *gettimeofday()* functions on an UltraSPARC/Solaris workstation.

Throughput: calculated in Megabits per second, is defined as the average number of useful bits transmitted per data frame, summed over *all* data channels. If L_{resv} denotes the size of the reservation packet in bits, L the size of the data packets, and S the transmission rate in Mbps, throughput is given by: $\Gamma = \dfrac{SRL_{data}}{(ML_{resv} + \mathcal{T}S + \mathcal{L}L)}$

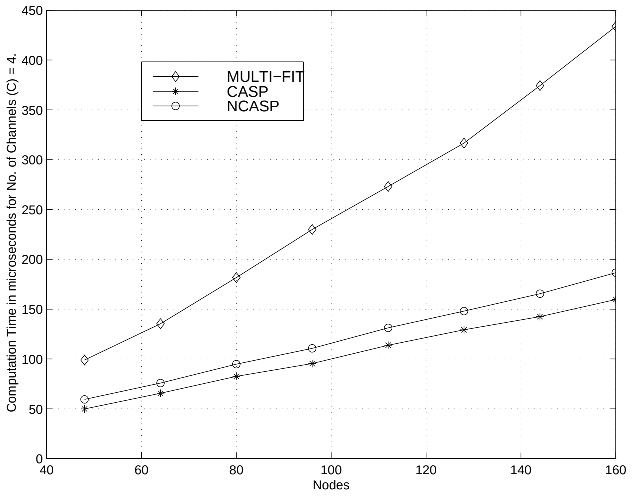
All the performance analysis described below has been done for varying $M \in \{48, 64, 80, 96, 112, 128, 144, 160\}$ and $K = 5$.

5.1 Single Priority Systems

This section provides the performance analysis of the algorithms for single-priority systems.

Fig 4 shows the computing time in microseconds, of algorithms MULTI-FIT, CASP and NCASP, for $C = 4$. The graphs show that CASP and NCASP require less time than the MULTI-FIT algorithm, as expected from the theoretical analyses. This is mainly because the MULTI-FIT algorithm sorts the input requests. Between CASP and NCASP, NCASP takes a longer time to compute the schedule because of the extra computing required to process splitting of requests.

Fig 4 also presents the schedule efficiency of the algorithms. The graphs show that the efficiency achieved by these algorithms increase with increasing values of M, showing that the architecture and the algorithms are best suited for systems with large M/C ratios. As expected, the NCASP gives much better utilization than both MULTI-FIT and CASP because of non-contiguous allocation. This leads to a more packed schedule with less wasted slots.



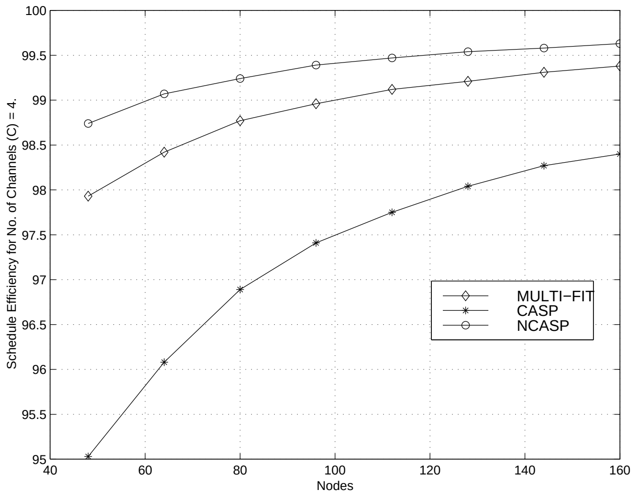
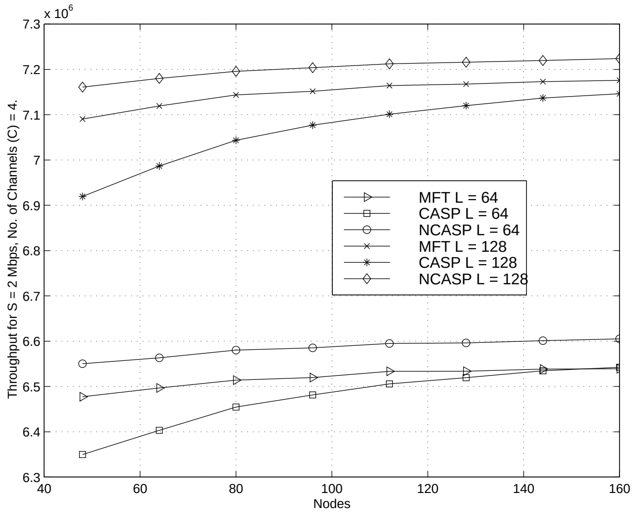


Figure 4. A comparison of the computation time and schedule efficiency in microseconds for MULTI-FIT, CASP and NCASP algorithms. $M \in \{48, 64, 80, 96, 112, 128, 144, 160\}$, $C = 4$, $K = 5$.

The above experiments were repeated for $C \in \{2, 8\}$ and showed similar performance trends among the algorithms. Increasing C resulted in increased computation time, and slightly reduced scheduling efficiency. For higher values of C, MULTI-FIT resulted in almost the same efficiency as NCASP. This is due to the inherent nature of the MULTI-FIT algorithm to perform better when the number of bins (here - the number of channels) increases.

The achieved throughput (Γ) for transmission rates of 2 Mbps and 10 Mbps is shown in Fig 5. The graphs show the throughput achieved for varying data packet lengths $L \in \{64, 128\}$ and similar parameters as before. For 2 Mbps and $C = 4$, the highest throughput is achieved by NCASP and is around 6.6 Mbps and 7.2 Mbps for $L = 64$ and $L = 128$ respectively. This may be compared to the theoretical maximum of $2 \times 4 = 8$ Mbps. This stresses the importance of the size of the data packet length to better utilize the available bandwidth. In this case, NCASP generates the highest throughput because of its better scheduling efficiency.



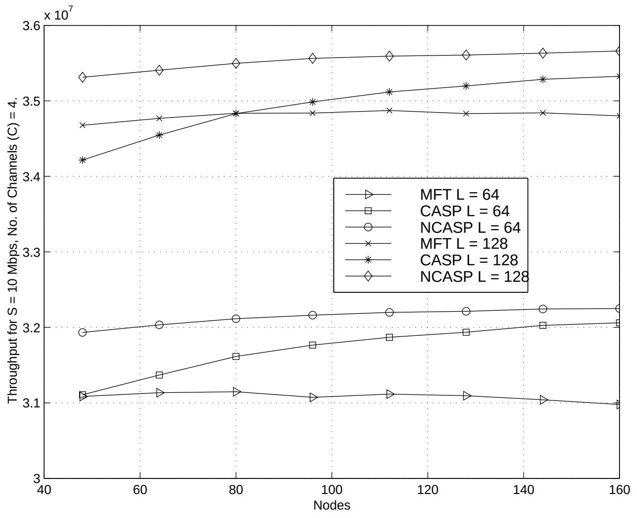
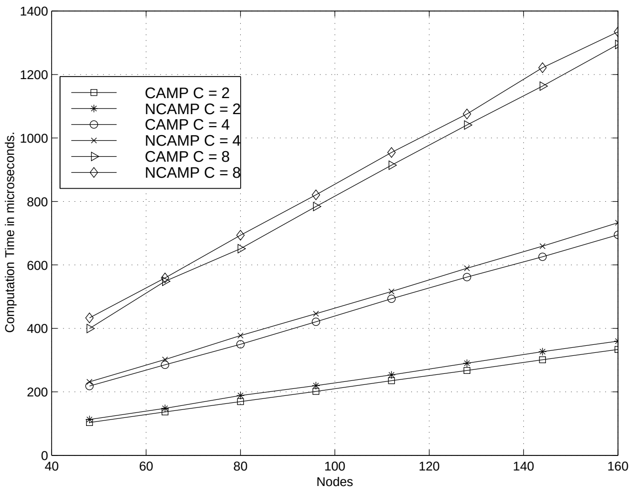


Figure 5. A comparison of throughput for MULTI-FIT, CASP and NCASP algorithms for $M \in \{48, 64, 80, 96, 112, 128, 144, 160\}$, $C = 4$, $K = 5$, Channel Speed $S \in \{2, 10\}$ Mbps, and $L \in \{64, 128\}$ bytes.

For 10 Mbps and $C = 4$, the highest throughput is still achieved by NCASP algorithm. This is around 32 Mbps and 35 Mbps for $L = 64$ and $L = 128$ respectively, compared to a theoretical maximum of $10 \times 4 = 40$ Mbps. At higher speeds, the packet transmission time reduces and the computation delay begins to dominate overall throughput. This is observed for $C = 4$ where CASP performs relatively better compared to MULTI-FIT algorithm for $M > 80$. This is because of the higher computation time of MULTI-FIT for $C = 4$.



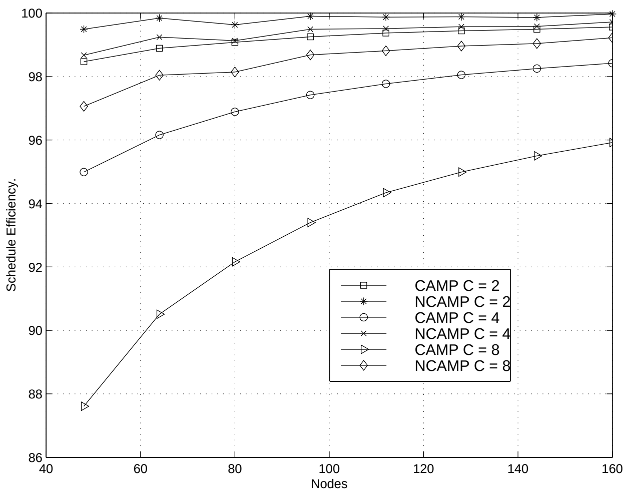


Figure 6. A comparison of the computation time (in microseconds) and scheduling efficiency for CAMP and NCAMP algorithms. $M \in \{48, 64, 80, 96, 112, 128, 144, 160\}$**,** $K = 5$**, and** $C \in \{2, 4, 8\}$**.**

The experiments above were repeated for $C \in \{2, 8\}$, and the trends were similar to $C = 4$. For higher C, the ratio of achieved throughput to theoretical maximum reduces due to increased number of wasted slots.

5.2 Multiple Priority Systems

This section provides the performance analysis of the algorithms for multiple-priority systems.

Fig 6 shows the computing time in microseconds, of the CAMP and NCAMP algorithms for $C \in \{2, 4, 8\}$ and other parameters as earlier. The graphs show that the running time of CAMP is almost similar to that of NCAMP. This is similar to the earlier case of single priority systems.

Fig 6 indicates that the schedule efficiency of the CAMP algorithm is much lower than that achieved by the NCAMP algorithm. The problem of more wasted slots with CAMP is compounded when multiple priorities are considered. Allocation using CAMP tends to follow a cascading effect, i.e. the wastage of slots at one stage of the algorithm computation is carried over to the next stage. Since NCAMP gives a more packed schedule for every stage, wasted slots are minimized when using NCAMP than when using CAMP. The graph also shows that increasing C results in lower scheduling efficiency due to more wasted slots, especially for CAMP.

Fig 7 presents the achieved throughput (Γ) for both algorithms for transmission rates of 2 Mbps and 10 Mbps and varying packet lengths $L \in \{64, 128\}$. For 2 Mbps and $C = 4$, the highest throughput is achieved by NCAMP and is around 7.5 Mbps and 7.7 Mbps for $L = 64$ and $L = 128$ respectively, compared to a theoretical maximum of 8 Mbps. For larger values of M, the throughput of CAMP seems to asymptotically approach that of CAMP because scheduling efficiency increases with M. For 10 Mbps, NCAMP again performs better than the CAMP algorithm. However, the ratio of highest throughput to maximum throughput decreases with increasing transmission speeds. For example, this ratio is 0.94 for $S = 2$ Mbps with NCAMP and 0.90 for $S = 10$ Mbps. This shows that for increasing speeds, the relative impact of computation time will increase.

6 Conclusion

This paper describes scheduling algorithms for a reservation-based medium access protocol for scheduling traffic in wireless data networks with multiple channels per cell. The time complexity of the MULTI-FIT algorithm adapted to this problem was $O(M \lg M + MC)$. The time complexities for CASP and NCASP were $O(M \lg C)$ while CAMP and NCAMP executed in $(PM \lg C)$ time, where P is the number of priority queues within a mobile. Simulation studies that measured computation time on an Ultra SPARC station are used to study the performance of the above algorithms. The higher schedule efficiency of the non-contiguous algorithms result in higher overall throughput as shown in the performance analysis. The throughput achieved increases with the number of mobiles in the network, indicating that these algorithms are suited for larger sized networks. Also, the system with eight channels

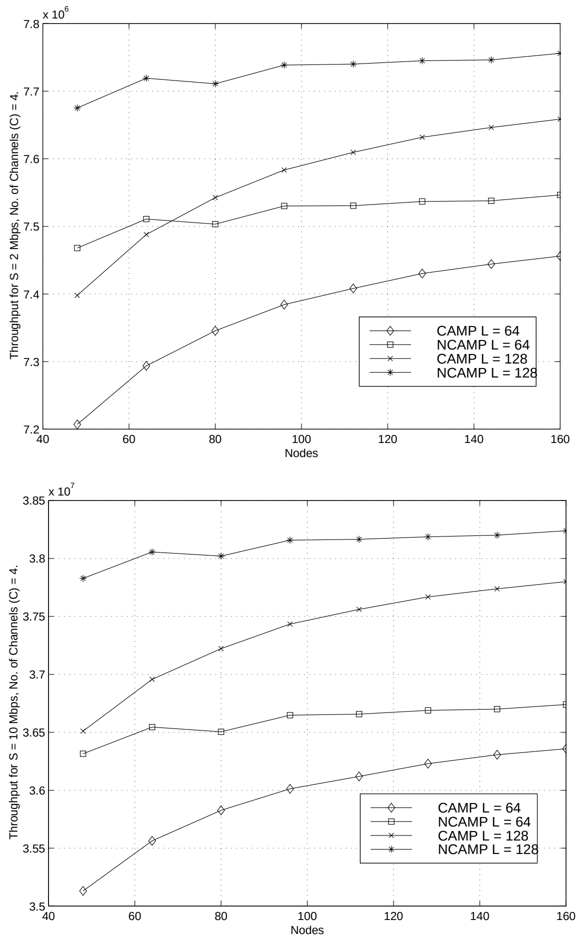


Figure 7. A comparison of throughput performance of CAMP and NCAMP algorithms for $M \in \{48, 64, 80, 96, 112, 128, 144, 160\}$, $K = 5$, Channel Speed $S \in \{2, 10\}$ Mbps, $C = 4$, and $L \in \{64, 128\}$ bytes.

has lower efficiency utilization than the system with four channels. This indicates that there is no linear improvement in performance with increasing number of channels. The results also demonstrate that the non-contiguous allocation algorithms perform better than the contiguous ones in most cases. This is despite the higher average running time of the non-contiguous algorithms. Also, network transmission speed was varied to show the relative impact of schedule computation time in higher speed networks.

A full version of the paper is available in [4].

References

[1] Y. Bernet, J. Binder, S. Blake, M. Carlson, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss. A Framework for Differentiated Services. IETF draft, draft-ietf-diffserv-framework-00.txt, May 1998.

[2] J.-C. Chen, K. M. Sivalingam, P. Agrawal, and R. Acharya. Scheduling Multimedia Services for a Low-Power MAC in Wireless and Mobile ATM Networks. *IEEE Transactions on Multimedia*, 1(2):187–201, June 1999.

[3] E. Coffman, M. R. Garey, and D. S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal of Computing*, 7:1–17, Feb. 1978.

[4] S. Damodaran and K. Sivalingam. Design and analysis of simple scheduling algorithms for wireless data networks with multiple channels. Technical Report TR99-DAWN-4, Washington State University, Email: krishna@eecs.wsu.edu, Sept. 1999.

[5] S. Lu, V. Bharghavan, and R. Srikant. Fair scheduling in wireless packet networks. In *Proc. ACM SIGCOMM*, pages 63–74, Cannes, France, Sept. 1997.

[6] S. Lu, T. Nandagopal, and V. Bhargavan. A Wireless Fair Service Algorithm for Packet Cellular Networks. In *Proc. ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 10–20, Dallas, TX, Oct. 1998.

[7] T. S. E. Ng, I. Stoica, and H. Zhang. Packet fair queuing algorithms for wireless networks with location-dependent errors. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, San Francisco, CA, Mar. 1998.

[8] P. Ramanathan and P. Agrawal. Adapting Packet Fair Queuing Algorithms to Wireless Networks. In *Proc. ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 1–9, Dallas, TX, Oct. 1998.

[9] K. M. Sivalingam, J.-C. Chen, P. Agrawal, and M. Srivastava. Design and Analysis of Low-Power Access Protocols for Wireless and Mobile ATM Networks. *ACM/Baltzer Mobile Networks and Applications*, 1998. (Accepted for Publication).

[10] J. D. Ullman. *Complexity of sequencing problems*, chapter 4. John Wiley, New York, 1976.