Credit-Based Slot Allocation for Multimedia Mobile Ad Hoc Networks

Hsi-Lu Chao, *Student Member, IEEE* and Wanjiun Liao, *Member, IEEE*

***Abstract*—This paper studies resource management for multimedia mobile ad hoc networks (MANET). In particular, we focus on providing fair scheduling with quality-of-service (QoS) support for MANET. We consider two types of flows: guaranteed and best effort flows. The goal is to satisfy the QoS requirements of guaranteed flows and to provide global fairness for best effort flows. In this paper, a credit-based fair scheduling mechanism called credit-based slot allocation protocol (CSAP) is proposed. In CSAP, nodes are logically grouped into clusters, each with a scheduler. Each scheduler assigns time slots to nodes in its cluster based on the first tier algorithm. The node scheduled to send at the next time slot then in turn assigns the time slot to a relayed flow determined by the second-tier algorithm. Each multihop flow is treated as multiple single-hop flow segments. These segments are then correlated such that a downstream segment will not be allocated a slot unless the upstream segments have all been allocated. We evaluate the performance of CSAP by simulations. The results show that CSAP meets the QoS requirements of guaranteed flows, provides global fairness for best effort flows, and improves overall system throughput.**

***Index Terms*—Ad hoc networks, fair scheduling, mobility, quality-of-service (QoS).**

I. Introduction

An ad hoc network is a self-organizing wireless network comprised only of mobile nodes. In such a network, nodes are interconnected by multihop paths without the support of any preexisting wired infrastructure and may move in and out of the transmission ranges of one another. A multihop path is built with an ad hoc routing algorithm such as [1] and [2]. The constructed path may be broken when a node on a path has moved away. When this occurs, some rerouting procedure is initiated to reconnect the path. In an ad hoc network, each node plays both roles of a terminal and a router. Due to the fully distributed characteristics of the network, collisions may occur and the system throughput may be low without proper coordination among the transmitting nodes.

Resource management for multimedia traffic in packet networks has been an active research topic. To date, research in ad hoc networks has mostly been concerned with best effort traffic. In particular, the major focus has been placed on fair scheduling for ad hoc wireless networks [3]–[6]. Fairness is an important criterion of resource sharing in best effort packet networks, especially when there is competition for resource due to unsatisfied demands [7]. Fair scheduling allows all participating flows to share resource fairly. With technological advances in consumer electronics and increasing demand of multimedia applications, one must address how to support quality-of-service (QoS) for multimedia traffic in ad hoc networks. Providing QoS in ad hoc networks is a challenge, given their peculiar characteristics compared with conventional wired or single-hop wireless networks. In wired or single-hop networks, only routers or base stations are involved in making scheduling decisions. In ad hoc networks, all nodes may be involved in making decisions. The major problem of a fully distributed scheduling mechanism is that it may cause serious collisions and significantly degrade network throughput without proper coordination among transmitting nodes. This problem is further exacerbated with node mobility, which may cause the constructed QoS routes to fail. Existing work focuses mainly on QoS routing [8], [9], which finds multihop paths to meet the desired service levels of flows. However, QoS routing alone cannot guarantee QoS requirements of multimedia traffic. This paper studies a companion QoS issue for multihop, multimedia mobile networks. We consider a mix of guaranteed and best effort flows and investigate fair scheduling with QoS support for the flows. Our work is motivated as follows. In the future, while guaranteed flows (i.e., multimedia flows) will become the most important traffic in the network, best effort traffic will still play an important role. Since fairness is the key of resource sharing for best effort traffic and QoS is the major performance metrics for guaranteed traffic, it is important and timely to study fair scheduling with QoS support for mobile ad hoc networks.

The rest of this paper is organized as follows. Section II states the related work of fair scheduling in ad hoc networks. Section III describes the proposed credit-based slot allocation protocol (CSAP). Section IV shows the simulation results. Finally, the conclusion is drawn in Section V.

II. Related Work

According to the decision metrics, existing scheduling disciplines for best effort ad hoc networks can be classified into two categories: *timestamp-based* [3]–[5] and *credit-based* [6]. In [3]–[5], timestamp-based fair scheduling mechanisms are proposed for ad hoc wireless networks. They all have to first convert a node graph into a flow graph. In a flow graph, a vertex indicates a flow, and an edge is added when two flows are contending for resource. The mechanisms in [3] and in [4] work similarly. Each newly received packet is assigned two tags: one is the start tag and the other is the finish tag, as in [10]. Either tag can be used as the service tag. The packet with

Manuscript received October 15, 2002; revised April 26, 2003. This work was supported in part by the Ministry of Education (MOE) Program for Promoting Academic Excellence of Universities under Grant 89E-FA06-2-4-7 and in part by the National Science Council, Taiwan, under Grant NCS91-2213-E-002-057.

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: wjliao@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/JSAC.2003.815232

the least service tag is transmitted first. Let S_f^j and F_f^j denote the start tag and the finish tag for the jth packet of flow f, respectively. The tag is assigned according to (1)

$$\begin{cases} S_f^j = \max\left\{v\left[A\left(p_f^j\right)\right], F_f^{j-1}\right\}, & j \geq 1 \\ F_f^j = S_f^j + \dfrac{l_f^j}{w_f}, & j \geq 1 \end{cases} \tag{1}$$

where p_f^j, l_f^j and $A(p_f^j)$ are the jth packet of flow f, the length of p_f^j, and the arrival time of p_f^j, respectively; $v\lfloor A(p_f^j)\rfloor$ denotes the virtual arrival time of p_f^j; w_f is the flow weight of flow f, indicating a certain percentage of link capacity to be shared by the flow.

In [5], the scheduling decision is made according to the contending power, instead of the service tag as in [3] and [4]. The contending power of a flow, say f, is defined as follows. For each fully connected subgraph in which flow f is involved, say G_f, in the flow graph, the flow weights of all the flows in G_f are first summed. The contending power of flow f (denoted as $P(f)$) is then set to the maximum over the summation of the flow weights in each subgraph, i.e., $P(f) = \max_{\forall G_f}\{\sum_{\forall f_i} w_{f_i}\}$, where f_i denotes the flows in the set G_f.

The timestamp mechanisms in [3]–[5] suffer from the following two problems: 1) packets in the queue are required to sort increasingly based on their service tags and 2) their virtual clock cannot be reset unless the queue has become empty [11]. To avoid these problems, a credit-based scheme is proposed in [6]. Unlike the credit-based mechanisms [12] and [13] which use accumulated "credit" values for fair scheduling in the wired line networks, the mechanism in [6] makes the decision according to the excess value of credit usage, where the excess value is equal to the actual usage minus the accumulated credit. The flow with less excess in usage value has higher transmission priority. As shown in [6], this credit-based mechanism can achieve the same performance as timestamp-based schemes without suffering their problems.

None of these existing mechanisms, namely, [3]–[6], consider QoS support when fair scheduling is performed. They would starve best effort flows if applied directly to support QoS for guaranteed flows in a network with a mix of both types of flows. The study in [14] shows that the timestamp-based mechanisms tend to distribute resource equally among all flows even when QoS support is taken into account. This characteristic may be desirable for best effort traffic, but definitely not for guaranteed flows with different QoS requirements. Credit-based mechanisms, on the other hand, are easier to modify to support QoS for guaranteed flows while ensuring fair share of residual bandwidth for best effort flows.

In this paper, we propose a new mechanism called the CSAP, which provides fair scheduling with QoS support for multimedia mobile ad hoc networks. We consider a mix of best effort packets and guaranteed flows, and use "bandwidth" as the QoS metric due to its importance for real-time applications. In [11], bandwidth requirement in time-slotted network is measured in terms of a fraction of one time slot. As in [11], we use a fraction of one slot time to denote the bandwidth requirement of each guaranteed flow. The goal is to guarantee the bandwidth requirements of guaranteed flows, while ensuring fair share of residual bandwidth for best effort flows. Extended from [6], CSAP has the following advantages over timestamp-based schemes when QoS is provided with fair scheduling for mobile ad hoc networks.

- The mechanisms in [3]–[5] require very complicated calculations to determine the flow weight for (1), i.e., w_f. To extend them to allow QoS, an intuitive way is to assign different flow weights to flows with different QoS requirements. However, if the flow weight is assigned based on the flow requirement, best effort flows may be starved because each best effort flow has a zero flow requirement and, thus, an infinite service tag according to (1). To avoid such starvation, a centralized mechanism may be used to calculate proper flow weights, but may be not practical for ad hoc networks. On the other hand, each node in CSAP accumulates flow requirements as its "credit" independently, and slots are then allocated to each node proportionally according to its credit.
- The mechanisms in [3]–[5] need to convert a node graph into a flow graph before service tags are assigned. As a result, they may incur frequent recomputations for such conversions when node mobility is allowed. On the other hand, the impact of mobility for CSAP is only on the reset of "credit" values for affected flows, without extra calculations for graph conversions.
- To provide global fairness, the mechanisms in [3]–[5] need a global scheduling window defined as the maximum number of packets allowed to send for a flow in a predefined period of time. This may incur large overhead due to intensive message exchanges for mobile ad hoc networks. On the other hand, CSAP does not need such a scheduling window and still maintains global fairness for the network.

III. Credit-Based Slot Allocation Protocol (CSAP)

In this section, the proposed CSAP is described. We consider a mix of best effort and guaranteed flows. The features of CSAP are summarized as follows.

- A slot allocation mechanism, which supports QoS requirements for guaranteed flows and ensures global fairness for best effort flows in mobile ad hoc networks.
- A cluster-based mechanism to achieve spatial channel reuse
- A two-tier hierarchy to allocate time slots for better coordination among transmitting nodes
- A credit-based mechanism to avoid the defects of timestamp-based mechanisms.

Note that our mechanism works with different residual bandwidth allocation schemes, including fair scheduling for best effort flows only, for a mix of best effort and guaranteed flows, or for a mix of flows but with an upper limit on the resource share for guaranteed flows. In this paper, we just demonstrate the second case, i.e., for a mix of flows without an upper bound on each flow.

A. Assumptions

- Wireless media may exhibit time-varying errors due to co-channel interference, fading, link errors, and collisions. In this paper, we only consider the errors caused by collisions as in [3]–[5].
- We assume a time-division multiple-access (TDMA)-based system on a single channel shared by all nodes. To avoid interference among different clusters, we further assume TDMA is overlaid on top of a CDMA system, as in existing work [9]. A code assignment algorithm is assumed running in the lower layer of our system. Each TDMA frame contains a fixed number of time slots. The network is synchronized on a frame and slot basis.
- A mobile node cannot transmit and receive packets simultaneously as in [3] and [4].

B. Cluster-Based Scheduling

CSAP is a cluster-based mechanism. All mobile nodes in the network are logically divided into several clusters. Clusters may be overlapped, each with a designated code to avoid interferences. CSAP can cooperate with any existing clustering algorithm (e.g., [15], [16]) to form clusters. For example, a cluster could be formed as follows. Each mobile node periodically broadcasts beacons to nodes located within its transmission range. Based on the received beacons, mobile nodes learn of their neighbors and related information, such as node ID, node stability, etc., for operation. According to the selected criterion, one node is elected as the scheduler per cluster. Each scheduler periodically advertises itself to its neighbors, from which newly arriving mobile nodes learn where to register. Those nodes which can hear the same scheduler form a logical cluster. Each node can hear several schedulers, but can only be registered with one scheduler at a time.

C. Scheduling Tables

Each node on any flow path maintains a table called flow allocation table (FAT) for flow scheduling. The nodes selected as schedulers need to maintain an extra table called Node Allocation Table (NAT) for node scheduling.

1) Node Allocation Table (NAT): Maintained at the scheduler; each entry is for a (node ID, service type) pair, and has seven fields: a node ID, a service type, a *Resv*, a *Num*, a *Credit*, a *Usage*, and an *Excess*. The (node ID, service type) pair is to identify each entry; the *Resv* field, used in the guaranteed service entry, specifies the total reserved resource accounting for all guaranteed flows relayed by that node; the *Num* field, used in the best effort service entry, records the number of best effort flows managed by the corresponding node; the *Credit*, *Usage*, and *Excess* fields are used for the first-tier scheduling.

2) Flow Allocation Table (FAT): Maintained at the path node (i.e., a node on a flow path); each entry is for a (flow, service type) pair and has eight fields: a flow ID, a service type, a scheduler ID, a *Resv*, a *Credit*, a *Usage*, an *Excess*, and a Q-*size*. The (flow ID, service type) pair is to identify each entry; the *Resv* field is to specify the requested bandwidth requirement for the flow; the *Credit*, *Usage*, *Excess*, and Q-*size* fields are used for the second-tier scheduling. Note that the *Resv* value is used to denote the QoS requirement of a flow, and is represented as a fraction of one time slot. For a guaranteed flow, this value is a positive real number between zero and one; for a best effort flow, this value is always zero.

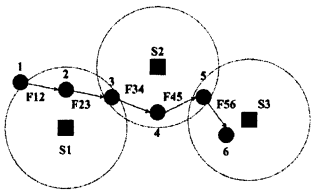


Fig. 1. An example to explain Q-*size*.

D. Scheduling Parameters

In CSAP, three parameters are defined for slot allocation: *Credit*, *Usage*, and *Excess*. For a scheduler, these three parameters are used for registered nodes, and for a node, they are for relayed flows. To accommodate both cases, we use a "request" to represent a node entry in NAT and a flow entry in FAT.

1) Credit: The cumulative time slots required for a guaranteed request, or the remaining slot quota per iteration[1] for a best effort request. For a guaranteed entry, the initial *Credit* value is set to a *Resv* and is increased by an amount of *Resv* at each time slot. For a best effort entry, the initial *Credit* value in NAT is set to a *Num* and is decremented by one when a slot is assigned to the entry; but in FAT, the *Credit* value is always set to zero for each best effort service entry.

2) Usage: The cumulative time slots assigned to a request. The *Usage* value is initialized to zero and is increased by one when a time slot is allocated to the request.

3) Excess: This value is used to determine to whom the next time slot is assigned. The next time slot is assigned to the request with the least "*Excess*" value. For guaranteed entries, the *Excess* values are equal to "*Usage* minus *Credit*," and are updated at each time slot. For best effort entries, the *Excess* values in FAT are also equal to "*Usage* minus *Credit*" and are updated at each time slot; but in NAT, the *Excess* values are initialized with zero, and are increased by one at each "iteration." In other words, the *Excess* value of a best effort entry in NAT is updated only when the corresponding *Credit* value counts down to zero. After the *Excess* value has been updated, the *Credit* value is reset to a *Num* value and a new iteration starts.

In CSAP, a multihop flow is modeled as multiple single-hop flow segments. For example, flow F in Fig. 1 is comprised of five single-hop flow segments, i.e., $F_{12}, F_{23}, F_{34}, F_{45}$, and F_{56}. The source and the destination of flow F are nodes 1 and 6, respectively. The sender of each flow segment is registered with a scheduler. The registered schedulers may not be the same. For example, nodes 1 and 2 are registered with scheduler S1, while nodes 3 and 4 are registered with S2. In CSAP, each scheduler works independently. As a result, even though a downstream flow segment has been assigned a slot, it may not have packets queued to be transmitted unless the upstream flow segments have all been allocated a slot. To correlate the segments belonging to a flow, a new parameter is defined, called Q-*size*. Each flow has a corresponding Q-*size* maintained in the FAT of each node on the flow path. This parameter is initialized with

[1]An "iteration" is a cycle in which the value of a *Num* is count down from the original value to zero.

zero at all nodes except for the sender, which has a nonzero Q-*size* depending on the number of packets to be transmitted. When a node receives a packet sent from its previous node, the Q-*size* value corresponding to this flow is increased by one. Similarly, the Q-*size* value is decremented by one when this node transmits a packet to its next node. Thus, we can avoid a downstream segment being allocated a slot but without a packet ready to be transmitted. Thus, when a node is assigned a slot by its scheduler, it will further allocate the slot to a flow with the least Excess value and nonzero Q-*size*.

E. Service Registration

Before a transmission starts, the sender of the flow must first determine a flow path, irrespective of whether it is a best effort or guaranteed (i.e., QoS) flow. Best effort flows do not place any resource requirement on the path nodes (i.e., the nodes along the path). However, for a QoS flow, the determined path needs to meet the desired QoS level of the flow. CSAP can be easily integrated with any existing ad hoc routing protocol for path construction. For best effort flows, ad hoc routing is used (e.g., [1], and [2]); for guaranteed flows, QoS routing is required (e.g., [8] and [9]). Below, we simply use an existing QoS routing mechanism as an example to illustrate how CSAP works.

1) Path Construction: To construct a path, the sender first broadcasts a route REQuest (RREQ) message with a service type, a *Resv*, and other information such as routing messages to its neighbors. Each node receiving this RREQ to relay the flow's packets will verify if the summation of the reservation levels (i.e., *Resv*) of all relayed flows, including the newly arriving one, is less than or equal to the target link utilization. If the verification fails, the request is denied and the RREQ packet is discarded; otherwise, the request is tentatively accepted and the RREQ packet is rebroadcast to its neighbors after a temporary entry[2] for the flow is created in FAT. Duplicated RREQ messages are dropped when received.

When an RREQ reaches the flow destination, a route REPly (RREP) message is sent along the reverse path of the RREQ back to the source. Upon receiving an RREP message at a node, the node learns which node is the next hop on the path. Suppose that node N receives an RREP from node M. Node N then performs service registration as follows.

1) Node N sends a resource Allocation ReQuest (ARQ) message, including the service type and the cumulative *Resv* value, to a scheduler which can serve both nodes N and M. The cumulative *Resv* includes the *Resv* values of all entries in its FAT, including temporary entries.
2) The scheduler then verifies if the total *Resv* values of all entries including the new ARQ are still less than or equal to the target link utilization. If the verification fails, the request is denied and an NAK is sent back to the node; otherwise, the scheduler updates its NAT and an ACK is replied. Note that the NAT is updated as follows. The scheduler creates a new entry in the NAT for the request if the node has made requests for the first time; otherwise, the scheduler modifies the *Resv* value of the corresponding entry in the NAT.
3) If an ACK is received, node N switches the corresponding entry in FAT from temporary to regular, and relays the RREP message to its previous node on the path. If an NAK is received, node N tries to register with another eligible scheduler (i.e., from which the advertisement can be heard). If all attempts fail, a new path should be built and all the constructed entries in both NAT and FAT for the flow should be deleted.

When this RREP message reaches the source node, a qualified QoS route is constructed and service registration is completed.

2) Path Remedy: Rerouting is used to deal with the problem of path disconnection due to node mobility. From periodic beacons, each node on the flow path can detect if its previous node or next node has moved away from its transmission range. Once a broken path is detected, the nodes detecting the breakage each sends an NAK (i.e., path broken) message to the sender or the destination following the flow path.[3] The nodes, say node N, on the path receiving this NAK releases the reserved resources, and updates the tables as follows.

a) Remove the corresponding flow entry from its FAT.
b) Inform the scheduler. There are two cases:
   - Send an NAK (i.e., service deactivation) to the scheduler if node N has no more flows with that service type registered with the scheduler. This will make the scheduler delete the corresponding entry from its NAT;
   - Send an ACK (i.e., *Resv/Num* update) to the scheduler if node N still has some other flows with that service type registered with that scheduler. This will make the scheduler update the corresponding entry in its NAT. For guaranteed service entries, this updated *Resv* is equal to the original *Resv* value (i.e., the sum of *Resv* values from all managed flows) minus this flow's *Resv* value; for best effort entries, the updated *Num* is equal to the original *Num* value minus one.

Once this NAK (path broken) message reaches the sender, the path construction and service registration procedure as described in Section III-E is reinitiated.

F. Two-Tier Slot Allocation

In CSAP, slot allocation is based on a two-tier mechanism. The scheduler assigns time slots to nodes based on the first tier slot allocation mechanism. The node to whom the next time slot is assigned then in turn assigns the time slot to a flow based on the second-tier slot allocation mechanism.

1) Table Initiation: Suppose that there are m entries in the NAT of a scheduler. These m entries can be divided into two sets: S_g and S_b, indicating guaranteed and best effort service entries, respectively. Each entry of NAT is initialized as follows. For each entry x in set S_g, $Credit(x) \leftarrow Resv(x)$, $Usage(x) \leftarrow 0$, and $Excess(x) \leftarrow Usage(x) - Credit(x)$; for each entry y

[2]A temporary entry is not considered in the scheduling at the node. Such an entry will become regular once a confirmation is received within a predefined timeout. If the timer expires before a confirmation arrives, the entry is deleted.

[3]Note that depending on which ad hoc routing is used, the path can be reconnected from the broken point, i.e., at the node which has detected the breakage, or from the sender. In this example, we just demonstrate how rerouting is performed from the sender.

in set S_b, $Credit(y) \leftarrow Num(y)$, $Usage(y) \leftarrow 0$, $Excess(y) \leftarrow 0$.

FAT entries are initialized similarly. Suppose that there are n entries in the FAT of a node, say node A. Each entry of FAT is initialized as $Credit(z) \leftarrow Resv(z)$, $Usage(z) \leftarrow 0$, $Excess(z) \leftarrow Usage(z) - Credit(z), z = 1, 2, \ldots, n$, and $Q\text{-}size(z) \leftarrow 0$ if node A is not the sender of flow z.

2) Slot Allocation: The first tier slot allocation mechanism works as follows.

a) The scheduler assigns the next time slot to the node with the smallest *Excess* value in the NAT at the decision time.

The scheduler then updates the values of *Credit*, *Usage*, and *Excess* in all entries of the NAT, as follows.

b) The *Usage* value of the node scheduled to send at the next time slot, say node A, is incremented, and all the others are left intact, i.e., $Usage(A) \leftarrow Usage(A) + 1$, where $Excess(A) = \min\{Excess(x) | x = 1, 2, \ldots, m\}$.

c) The *Credit* value of each guaranteed-service entry is increased by a *Resv*, i.e., $Credit(x) \leftarrow Credit(x) + Resv(x), x \in S_g$. The *Credit* values of all nodes belonging to set S_b remain intact. However, if node A to whom the slot is assigned is in set S_b, the *Credit* value of node A will be decremented by one, i.e., $Credit(A) \leftarrow Credit(A) - 1$, and the other best effort entries will still remain intact.

d) The *Excess* value of each guaranteed-service entry is updated as $Excess(x) \leftarrow Usage(x) - Credit(x), x \in S_g$, and that of each best effort entry stays unchanged. However, if node A to whom the slot is assigned is in set S_b and the *Credit* value of node A is zero, the *Excess* value of node A will be increased by one and the *Credit* value of node A will be reset to the value of a *Num*, i.e., if $Credit(A) = 0$, then $Excess(A) \leftarrow Excess(A) + 1$ and $Credit(A) \leftarrow Num(A)$.

The second-tier slot allocation algorithm works as follows.

a) A node, say node A, cannot schedule any flow packets unless it has been assigned a time slot. Once node A is assigned the time slot, it assigns the slot to a flow with nonzero Q-*size*. Suppose that p of the n entries in the FAT have nonzero Q-*size* values, where $p \leq n$. The time slot is assigned to flow F if it satisfies the following three conditions.
   - $F \in p$.
   - Flow F has the smallest *Excess* value among the p flows.
   - The service type of flow F matches the service type of node A designated by the scheduler.

   Node A then updates the values of *Credit*, *Usage*, *Excess*, and Q-*size* in all entries of the FAT, as follows.

b) Increment the *Usage* value of the flow scheduled to send at the next time slot and leave the others intact. That is, $Usage(F) \leftarrow Usage(F) + 1$, where $Excess(F) = \min\{Excess(y) | y \in p\}$.

c) Update the *Credit* values of all entries in the FAT as $Credit(y) \leftarrow Credit(y) + Resv(y), y = 1, 2, \ldots, n$.

d) Update the *Excess* values of all entries in the FAT as $Excess(y) \leftarrow Usage(y) - Credit(y), y = 1, 2, \ldots, n$.

e) Update the Q-*size* values of flow F and leave the others intact. In other words, the Q-*size* value of flow F is decremented, while that of the flow receiving a packet from its previous node is incremented, i.e., $Q\text{-}size(F) \leftarrow Q\text{-}size(F) - 1$ if node A is the sender of flow F; $Q\text{-}size(F) \leftarrow Q\text{-}size(F) + 1$ if node A is the receiver of flow F.

3) Table Updates for Broken Path Remedy: When a flow route is broken due to node movements, a path remedy process is initiated. Depending on the companion routing protocol, path remedy may be initiated by the sender or the path node discovering the broken path. After the route is reconnected, those nodes no longer on the path delete the corresponding entries in their FAT's. Meanwhile, the scheduling parameters at all nodes on the reconnected path are updated. Thus, for each node, say node N, on the reconnected path of flow F, flow F's entry in its FAT is reset as follows:

a) $Credit(F) \leftarrow Resv(F)$;
b) $Usage(F) \leftarrow 0$;
c) $Excess(F) \leftarrow Usage(F) - Credit(F)$;
d) $Q\text{-}size(F) \leftarrow 0$ if node N is not the sender of flow F.

The NAT of node N's scheduler is also modified. There are two cases.

a) Update the *Resv* value for node N as $Resv(N) \leftarrow Resv(N) + Resv(F)$, or $Num(N) \leftarrow Num(N) + 1$, depending on the service type, and keep other parameters intact if node N has other flows and has been registered with the scheduler.

b) Create a new entry for node N, and set the parameters as follows if node N is registered with the scheduler for the first time. If the service type is guaranteed, the setting is $Resv(N) \leftarrow Resv(F)$, $Credit(N) \leftarrow 0$, $Usage(N) \leftarrow 0$, and $Excess(N) \leftarrow Usage(N) - Credit(N)$; if it is a best effort entry, the parameter setting becomes $Num(N) \leftarrow 1$, $Credit(N) \leftarrow Num(N)$, $Usage(N) \leftarrow 0$, and $Excess(N) \leftarrow 0$.

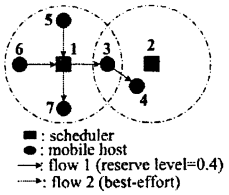
Note that the schedulers of path nodes may move away. In this case, the affected path nodes have to reinitiate the scheduler selection process as mentioned in Section III.B so as to select a new scheduler, even though the path is not broken. Let node S denote the move-away scheduler, and S^* denote the newly selected scheduler. The corresponding tables of these two nodes are updated as follows.

a) If node S has not received beacons from some the registered nodes for a predefined time interval, the corresponding entries in its NAT are deleted. Once all the entries of its NAT become empty, node S switches its role to a normal node.

b) For node S^*, a new entry is created for each registered node in its NAT and the table is updated accordingly.

G. An Example

Figs. 2 and 3 show how CSAP works. Fig. 2(a) shows a seven-node ad hoc network. These nodes are logically divided into two clusters, each with one scheduler (represented as a square). Nodes 1 and 2 are the schedulers; nodes 3, 5, 6, 7 can



Flow ID	Service type	Resv.	Path
1	Guaranteed	0.4	6-1-3-4
2	Best-effort	0	5-1-7

(a) (b)

Fig. 2. An example for CSAP operations. (a) Network topology. (b) Flow information.

hear scheduler 1; nodes 3 and 4 can hear scheduler 2. There are two flows currently active in this network, i.e., flows F1 and F2. Fig. 2(b) summarizes the flow information. Suppose that each flow has 12 packets to be sent. Thus, the Q-*size* values of nodes 5 and 6 are both initialized with 12, and those of the other nodes are set to zero. Both flows are multihop flows. Thus, node 1 has four entries in its NAT: two for guaranteed service and two for best effort service. Node 2 has only one entry in its NAT, which is a guaranteed entry.

Fig. 3(a) and (b) shows the slot allocations of the schedulers (i.e., nodes 1 and 2) to the registered mobile nodes. The formats of K(b) and K(r) denote the entries of the NAT for node K with best effort and guaranteed service types, respectively. C, U, and E represent *Credit*, *Usage*, and *Excess*, respectively. Initially (i.e., at time zero), the scheduler copies the *Resv* values of ARQs as the *Credit* values in the corresponding entries, sets the value of *Usage* zero in all entries, and calculates the *Excess* value as "*Usage* minus *Credit*" for each entry. For scheduler node 1, the *Excess* values of N1(r), N1(b), N5(b), and N6(r) are $-0.4, 0, 0, -0.4$, respectively. The first slot can be assigned to either entry of N1(r) or N6(r) due to a tie on the least *Excess* value. Suppose the first slot is assigned to N6(r). Therefore, the *Usage* of N6(r) becomes one, and the new *Excess* value is $1 - 0.4 = 0.6$. The second time slot is assigned to N1(r) due to the least *Excess* value. The updates of the NAT's at scheduler nodes 1 and 2 are based on the first tier scheduling in a five-slot time span.

Fig. 3(c), (d), and (e) shows the slot allocations of mobile nodes to the relayed flows. To better examine the allocation for a multihop flow, the FAT updates at the path nodes of a flow are depicted. We use F1 as an example, and show the FAT updates at nodes 6, 1, and 3. Since the first slot of scheduler node 1 is allocated to node 6, node 6 then allocates the slot to a flow based on the second-tier slot allocation. Since, flow F1(r) is the only flow with a nonzero Q-*size* managed by node 6, the slot is assigned to F1(r). Thus, the Q-*size* of F1(r) at node 6 (i.e., sending node) is decremented by one, i.e., $12 - 1 = 11$ [Fig. 3(c)], and that of F1(r) at node 1 (i.e., receiving node) is incremented by one, i.e., $0 + 1 = 1$ [Fig. 3(d)]. The other fields of F1(r) in node 6's FAT are updated accordingly. The second slot of scheduler node 1 is allocated to node 1. Node 1 then in turn allocates the slot to flow F1(r) due to the least *Excess* value and a nonzero Q-*size* value. The Q-*size* values of F1(r) at nodes 1 (sending node) and 3 (receiving node) are updated accordingly. Note that in Fig. 3(e), while the first two slots of scheduler node 2 is allocated to node 3, node 3 does not further allocate these two slots to F1(r), the only flow managed at node 3, due to the zero Q-*size* value of F1(r) at node 3. Thus, these two slots are wasted. Flow F1(r) at node 3 is allocated a slot at the third slot, when the Q-*size* of F1(r) is nonzero.

H. Bottleneck Consideration

In CSAP, a multihop flow is treated as multiple single-hop flows called flow segments. Each multihop flow is associated with a Q-*size* so that a downstream segment will not be allocated a slot unless the upstream segments have all been assigned a slot. The use of the Q-*size* parameter can only correlate those segments belonging to a multihop flows to some extent. Each segment may be registered with a different scheduler engaging a different number of flow segments. Some may be involved in more segments and some may in less. As a result, each segment may not be allocated with the same number of time slots. The more segments managed by a scheduler in a cluster, the less residual bandwidth each segment can be allocated and the less throughput each flow can have. Thus, the bottleneck cluster of a multihop flow, i.e., the cluster with the largest number of flow segments on the flow path, will determine the flow throughput.

Fig. 4 shows an example to explain how the bottleneck cluster affects the throughput of a multihop flow. Fig. 4(a) shows a 14-node mobile ad hoc network, in which the squares represent schedulers and the circles are mobile nodes. There are five multihop flows active in the network. The dashed arrows are for best effort flows and the solid arrows are for guaranteed flows. Fig. 4(b) summaries flow information. Suppose Fx is a multihop flow with path $\mathrm{a} \rightarrow \mathrm{b} \rightarrow \cdots \mathrm{m} \rightarrow \mathrm{n}$. The segments of flow Fx are denoted as $\mathrm{Fx_{a-b}}, \mathrm{Fx_{b-c}}, \ldots, \mathrm{Fx_{m-n}}$, respectively. Fig. 4(c) shows the schedulers and the senders of flow segments registered to the schedulers. It also shows segment information including the sender of each segment (i.e., indicated by NID), the service type, and the *Resv* value.

We introduce a recursive methodology to estimate flows' throughputs. Suppose that the network is partitioned into m clusters. Without loss of generality, the m clusters are assumed in a descending order of the congestion degree[4], i.e., $Resv(1) > Resv(2) > \cdots > Resv(m)$. Let N_i denote the number of flow segments in cluster i, and $N_{i,j}$ denote the number of flow segments in both clusters i and j. Let S_i denote the set of flows in cluster i, and $S_{i,j}$ denote the set of flows in both clusters i and j. Let x_j^i be the target share of residual bandwidth for flow j in cluster i. Thus, the target share of residual bandwidth for each flow, say f, in cluster 1 is in (2)

$$x_f^1 = \frac{1 - \sum_{j \in S_1} Resv_j}{N_1}, \quad f \in S_1. \tag{2}$$

The target share of residual bandwidth for each flow f in cluster 2 but not in cluster 1 is (3)

$$x_f^2 = \frac{1 - \sum_{j \in S_{1,2}} x_j^1 - \sum_{k \in (S_2 - S_{1,2})} Resv_k}{N_2 - N_{1,2}} \quad f \in (S_2 - S_{1,2}). \tag{3}$$

Therefore, to find the share of flow i in cluster j, we should first recursively calculate the share of each flow segment in cluster j, but bottlenecks are in clusters $1, 2, \ldots, j-1$.

[4]The congestion degree of a cluster is defined as the total reservation level recorded by the scheduler.

Slot / Node	0			1			2			3			4			5		
	C	U	E	C	U	E	C	U	E	C	U	E	C	U	E	C	U	E
N1(r)	0.4	0	−0.4	0.4	0	−0.4	0.8	1	+0.2	1.2	1	−0.2	1.6	1	−0.6	2.0	2	+0.0
N1(b)	1	0	+0.0	1	0	+0.0	1	0	+0.0	1	0	+0.0	1	0	+0.0	1	0	+0.0
N5(b)	1	0	+0.0	1	0	+0.0	1	0	+0.0	1	1	+1.0	1	1	+1.0	1	1	+1.0
N6(r)	0.4	0	−0.4	0.4	1	+0.6	0.8	1	+0.2	1.2	1	−0.2	1.6	2	+0.4	2.0	2	+0.0

(a) NAT updates at scheduler node 1

Slot / Node	0			1			2			3			4			5		
	C	U	E	C	U	E	C	U	E	C	U	E	C	U	E	C	U	E
N3(r)	0.4	0	−0.4	0.4	1	+0.6	0.8	2	+1.2	1.2	3	+1.8	1.6	4	+2.4	2.0	5	+3.0

(b) NAT updates at scheduler node 2

Slot / Flow	0				1				2				3				4				5			
	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q
F1(r)	0.4	0	−0.4	12	0.4	1	+0.6	11	0.8	1	+0.2	11	1.2	1	−0.2	11	1.6	2	+0.4	10	2.0	2	+0.0	10

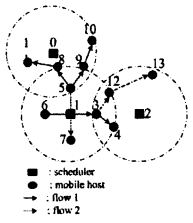
(c) FAT updates at node 6

Slot / Flow	0				1				2				3				4				5			
	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q
F1(r)	0.4	0	−0.4	0	0.4	0	−0.4	1	0.8	1	+0.2	0	1.2	1	−0.2	0	1.6	1	−0.4	1	2.0	2	+0.0	0
F2(b)	0.0	0	+0.0	0	0.0	0	+0.0	0	0.0	0	+0.0	0	0.0	0	+0.0	1	0.0	0	+0.0	1	0.0	0	+0.0	1

(d) FAT updates at node 1

Slot / Flow	0				1				2				3				4				5			
	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q	C	U	E	Q
F1(r)	0.4	0	−0.4	0	0.4	0	−0.4	0	0.8	0	−0.8	1	1.2	1	−0.2	0	1.6	1	−0.6	0	2.0	1	−1.0	1

Fig. 3. CSAP operations.



(a)

FID	Service type	Resv.	Path
1	Guaranteed	0.2	6-1-3-4
2	Best-effort	0	5-1-7
3	Guaranteed	0.2	5-8-11
4	Guaranteed	0.2	5-9-10
5	Best-effort	0	3-12-13

(b)

Scheduler	NID	Service type	Resv.	Single hop flows
0	5	Guaranteed	0.4	F35-8, F45-9
	8	Guaranteed	0.2	F38-11
	9	Guaranteed	0.2	F49-10
1	1	Guaranteed	0.2	F11-3
	1	Best effort	0	F21-7
	3	Guaranteed	0.2	F13-4
	3	Best effort	0	F53-12
	5	Best effort	0	F25-1
	6	Guaranteed	0.2	F16-1
2	12	Best effort	0	F512-13

(c)

Fig. 4. An example of a bottleneck cluster. (a) Network topology. (b) Flow information. (c) Scheduler registrations.

For example, as shown in Fig. 4(c), the most congested cluster is managed by node 0 and the residual bandwidth shared by each single-hop flow is $(1-0.8)/4 = 0.05$. Thus, F3 and F4 are each allocated the resource in a fraction of $0.2 + 0.05 = 0.25$. The second congested cluster is managed by node 1. The residual bandwidth shared by each flow segment is $(1-0.6)/6 = 0.067$. Thus, F1, F2, and F5 share the resource in a proportion of (0.267, 0.067, 0.067). The least congested cluster is managed by node 2 and the residual bandwidth shared by each flow is supposed to be $(1-0)/1 = 1$. However, F5 spans over two clusters and its bottleneck cluster is the second one (i.e., managed by node 1). Thus, it can only be allocated the resource with a fraction of 0.067 in this cluster (i.e., $\mathrm{Min}\{0.067, 1\}$).

IV. Simulation

In this section, the performance of CSAP is evaluated via simulation. The simulation environment is described as follows. There are 20 mobile nodes randomly distributed in a 670-m × 670-m area. The transmission range of each node is 250 m. We randomly select nodes, some as flow sources and some as flow destinations. Each flow may be a best effort or guaranteed flow. The duration of each slot is set to 577 μs and each frame is of eight slots. Each packet is assumed to occupy one time slot, and is of fixed packet length.

In the simulation, a cluster is formed as follows. Each node can learn of its neighbors from periodic beacons. The node with the smallest node ID becomes the scheduler. The scheduler then periodically broadcasts a message to advertise this information. Those nodes, which can hear the same scheduler, form a logical cluster. Therefore, clusters may be overlapped and a node may hear multiple schedulers. Once the scheduler of a node has moved away, the scheduler selection process is reinitiated.

A. Performance Metrics

We define several metrics to better evaluate the performance of CSAP as follows.

1) Flow Throughput (υ): The fraction of bandwidth allocated to each flow. The flow throughput of flow F is defined as $\upsilon(F) = (Slot(F)/total\ slots)$, where $Slot(F)$ denotes the number of packets received at the destination of flow F. Flow F is satisfied with the QoS demand if $\upsilon(F)$ is larger than its reserved level, i.e., $\upsilon(F) > Resv(F)$.

2) Share Degree (ϕ): The fraction of residual bandwidth shared by each flow. Here, residual bandwidth means the bandwidth left after the bandwidth up to the minimum requirements requested by all QoS flows have been allocated. The share degree of flow F is defined as the slot allocation minus its reservation level, i.e., $\phi(F) = \upsilon(F) - Resv(F)$.

3) Satisfaction Index (ρ): This parameter indicates how well all QoS flows are satisfied. ρ is defined in a way similar to the definition of the fairness index in [17]. The parameter x_i indicates if a QoS flow has been satisfied. If the time slot usage of a QoS flow is larger than or equal to its minimum

requirement, x_i is set to be one; otherwise x_i indicates the insufficient portion. Thus, $\rho = (1/m)\sum_{i=1}^{m}(x_i/x_f)$, where

$$x_i = \begin{cases} 1, & \text{if } \phi(i) \geq 0 \\ 1 - \alpha \times \dfrac{|\phi(i)|}{Resv(i)}, & \text{if } \phi(i) < 0 \end{cases}$$

α is a weight, $\alpha \geq 1, m$ is number of QoS flows and $x_f = (\sum_{i=1}^{m} x_i^2)/(\sum_{i=1}^{m} x_i)$.

4) Fairness Index (κ): This parameter indicates how fair the residual bandwidth is shared by all flows. Again, this definition is similar to the fairness index defined in [17]. The difference is that our definition is based on the share degree ϕ, instead of flow throughput υ. Thus, $\kappa = (1/n)\{\sum_{i=1}^{n}(y_i/y_f)\}$, where

$$y_i = \begin{cases} \phi(\mathrm{i}), & \text{if } \phi(i) \geq 0 \\ 0, & \text{if } \phi(i) < 0 \end{cases}$$

n is the number of flows, and $y_f = (\sum_{i=1}^{n} y_i^2)/(\sum_{i=1}^{n} y_i)$.

5) Network Throughput: Defined as $\sum_{i \in N} \upsilon(F_i)$, where N is the set of all flows in the ad hoc network.

B. Multihop Flows Without Mobility Support

We randomly generate five flows in a 20-node network, two are best effort flows and three are guaranteed flows. These 20 nodes are logically partitioned into four clusters. There are no node movements during the simulation period. Fig. 5(a) shows the node topology and Fig. 5(b) describes the flow information, including the flow route, the service type and *Resv* value, the corresponding single-hop flow segments, and the registered scheduler of each segment. The *Resv* values of guaranteed flows are randomly generated between zero and one to indicate the QoS requirements. Fig. 5(c) shows the flow throughput of each flow. The black bars are for CSAP, and the white bars are for the respective *Resv* values. We see that CSAP provides the minimum requirement for all guaranteed flows.

We then calculate the estimated the share degree of each flow.

1) The residual bandwidth shared by each segment in each cluster is roughly derived as follows.
   a) Scheduler 2 manages three flow segments, each with a share of $(1 - 0.459 - 0.459)/3 = 0.0273$.
   b) Scheduler 8 manages five segments, each with a share of $(1 - 0.467)/5 = 0.1066$.
   c) Scheduler 1 manages one segment, with a share of $(1 - 0.397) = 0.603$.
2) The descending order of clusters according to the congestion degree is: $\mathrm{cluster2} > \mathrm{cluster8} > \mathrm{cluster1}$.
3) The estimated share degree is then
   a) Flows 0 and 4 are bottlenecked by cluster 2, and each has a share degree of 0.0273.
   b) For those segments (except from flow 0) managed by scheduler 8, the share for each becomes $(1 - 0.467 - 0.0273)/4 = 0.126\,425$. Thus, each of flows 1 and 3 has a share degree of 0.126 425.
   c) Flow 2 is the only flow managed by scheduler 1 and has a share degree of 0.603.

Fig. 5(d) shows how these five flows share the residual bandwidth. In this example, CSAP has a share degree slightly lower than the estimated value because some slots may be assigned to nodes which have no flows with nonzero Q-*size*. Thus, some slots are wasted. Fig. 5(e) shows the other performance indices, including the satisfaction index, fairness index, and network throughput. Since each guaranteed flow is satisfied with the requested resource, the satisfaction index is one. The low fairness index is again due to the effect of clustering on a nonmobile network and the high network throughput is thanks to the spatial channel reuse from clustering.

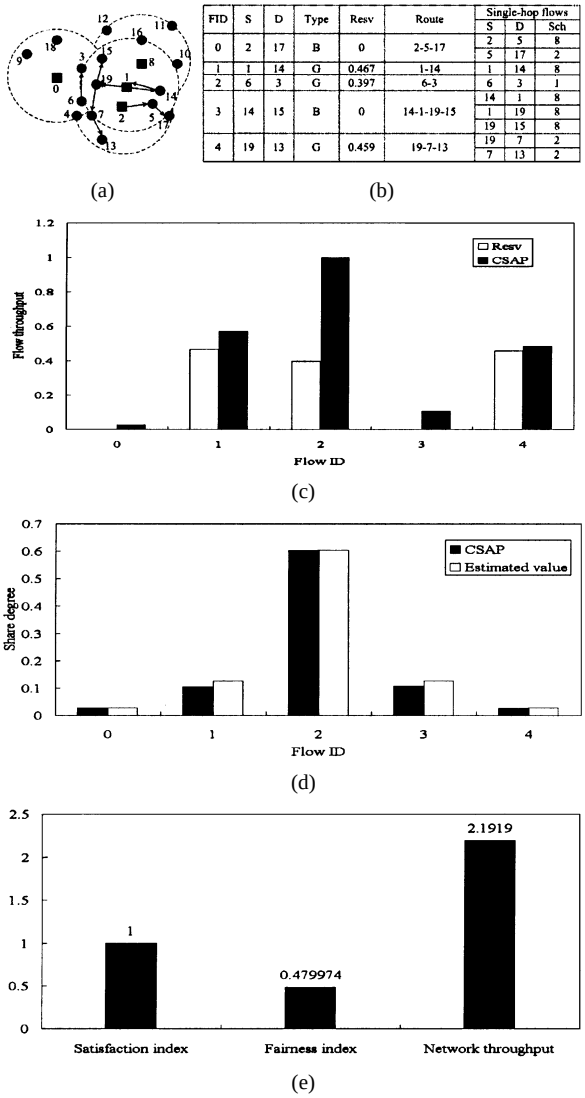


Fig. 5. Mixed flows without mobility support. (a) Node topology. (b) Flow information. (c) Flow throughput. (d) Share degree. (e) Other performance index.

C. Multihop Flows With Mobility Support

This simulation evaluates the performance of CSAP when node mobility is allowed. The mobility pattern of each node follows the random waypoint model [18]. Each node randomly selects a target position and speed to move. The speed a mobile node selects is within (0 and Max) meters per second. After arriving at the target position, the mobile node stays in that position for a predefined period of time called pause time. The mobile host then randomly selects a new target position and a speed, and move again.

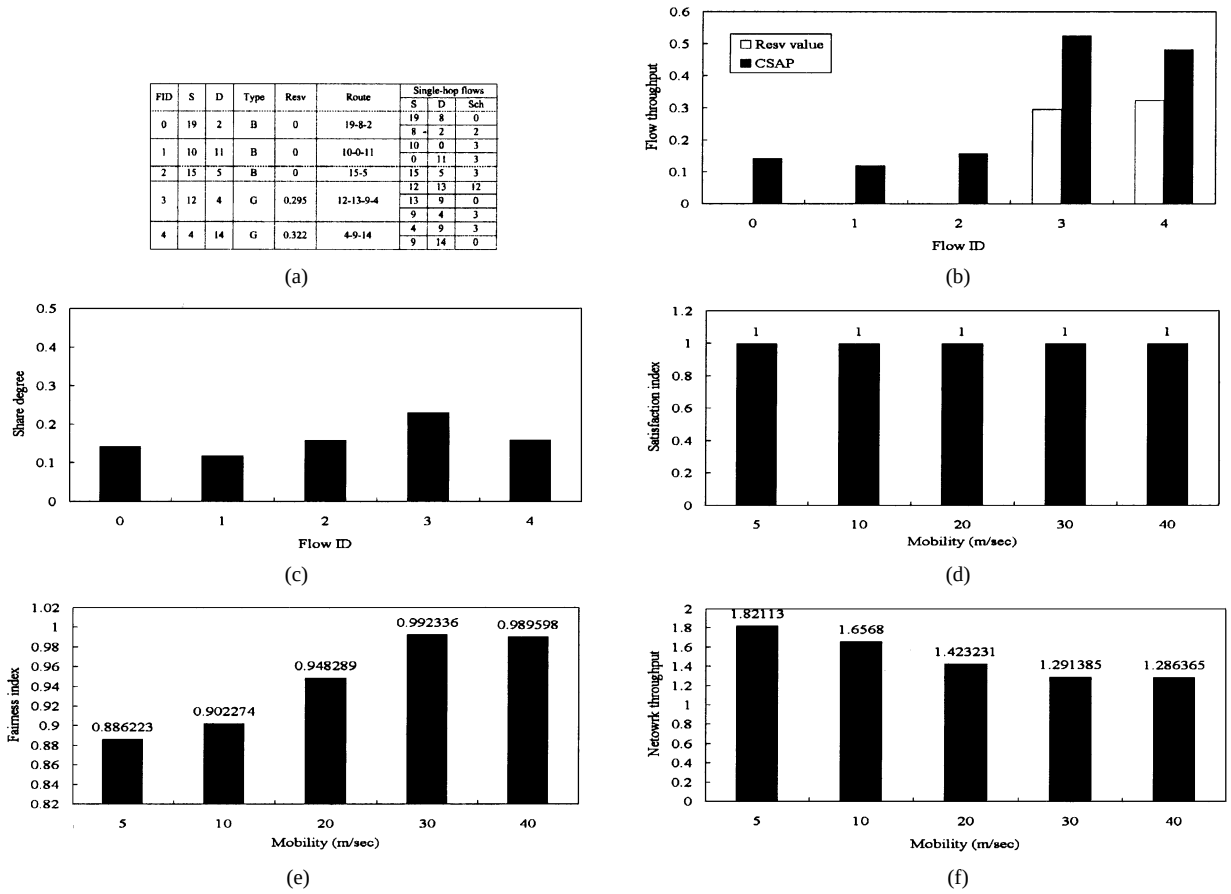


Fig. 6. Mixed flows with mobility support. (a) Flow information. (b) Flow throughput. (c) Share degree. (d) Satisfaction index versus mobility. (e) Fairness index versus mobility. (f) Network throughput versus mobility.

We randomly generate five flows: three are best effort flows and two are guaranteed flows. The network is initially partitioned into five clusters. Flows 0, 1, and 2 are best effort flows; flows 3, and 4 are guaranteed flows with a *Resv* of 0.295 and 0.322, respectively. Fig. 6(a) shows the flow information. Fig. 6(b) shows the flow throughput of each flow with a setting of 20 m/s maximum speed and 1-s pause time in the mobility model. The black bars are for CSAP and the white ones are for the requested *Resv* values. Again, CSAP can satisfy each QoS flows' minimum bandwidth requirement. Fig. 6(c) shows the share degree of each flow. Again, it has a high fairness index of 0.948 289. Fig. 6(d) and (e) shows the impact of mobility on the satisfaction index and the fairness index for CSAP. We find that our mechanism meets the minimum requirement of each flow at any speed (i.e., the values of the satisfaction index are all one) and fairly share residual resource among flows especially at a higher speed. However, as the moving speed is very fast (40 m/s in this example), flows with frequently broken routes are allocated less number of slots. Thus, the fairness index degrades. Finally, the network throughput, with the maximum speed varying from 5 m/s to 40 m/s, is shown in Fig. 6(f). It is above one in all cases. Again, due to frequent rerouting, the network throughput decreases as the speed increases.

V. Conclusion

In this paper, we have proposed a credit-based slot allocation protocol (CSAP) with mobility support, which provides fair scheduling with QoS support for multihop, multimedia ad hoc wireless networks. Based on a two-tier scheduling algorithm, CSAP can cope with frequent node movements and the fully distributed nature of ad hoc networks, which distinguishes CSAP from other credit-based solutions for wired or single-hop wireless networks. CSAP models each multihop flow as multiple single-hop flows and uses a *Q-size* to correlate the segments belonging to the same flow. We have studied the performance of CSAP via simulation, both with and without node movements. The simulation results show CSAP improves system throughput and ensures global fairness among all flows while satisfying the requirements of guaranteed flows.

There are still many related issues to be studied. For example, in this paper, we only consider "bandwidth" as the QoS metric for guaranteed flows. In the future, we will investigate the same issue but with different metrics such as delay/latency. Besides, we will verify our results but based on different fairness indices such as min–max fairness.

References

[1] C. E. Perkins and E. M. Royer, "Ad-hoc on demand distance vector routing," in *Proc. 2nd IEEE Workshop Mobile Computing Systems Applications*, Feb. 1999, pp. 90–100.

[2] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. ACM SIGCOMM*, Sept. 1994, pp. 234–244.

[3] H. Luo and S. Lu, "A topology-independent fair queueing model in ad hoc wireless networks," in *Proc. IEEE Int. Conf. Network Protocols*, Nov. 2000, pp. 325–335.

[4] ——, "A self-coordinating approach to distributed fair queueing in ad hoc wireless networks," *Proc. IEEE INFOCOM*, pp. 1370–1379, Apr. 2001.
[5] X. Wu, Y. Gao, H. Wu, and B. Li, "Fair scheduling with bottleneck consideration in wireless ad-hoc networks," in *Proc. IEEE Int. Conf. Computer Communications and Networks*, Oct. 2001, pp. 568–572.
[6] H. L. Chao and W. Liao, "Credit-based fair scheduling in wireless ad hoc networks," *Proc. IEEE Vehicular Technology Conf.*, Sept. 2002.
[7] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti, "Congestion control mechanisms and the best effort service model," *IEEE Network*, vol. 15, pp. 16–26, May/June 2001.
[8] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad hoc networks," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1488–1505, Aug. 1999.
[9] C. R. Lin and J.-S. Liu, "QoS routing in ad hoc wireless networks," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1426–1438, Aug. 1999.
[10] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Networking*, vol. 5, pp. 690–704, Oct. 1997.
[11] E. Hossain and V. K. Bhargava, "A centralized TDMA-based scheme for fair bandwidth allocation in wireless IP networks," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 2201–2214, Nov. 2001.
[12] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE/ACM Trans. Networking*, vol. 4, pp. 375–385, June 1996.
[13] B. Bensaou, D. H. K. Tsang, and K. T. Chan, "Credit-based fair queueing (CBFQ): A simple service-scheduling algorithm for packet-switched networks," *IEEE/ACM Trans. Networking*, vol. 9, pp. 591–604, Oct. 2001.
[14] H. L. Chao and W. Liao, "Fair scheduling with QoS support in ad hoc networks," in *Proc. IEEE Conf. Local Computer Networks*, Nov. 2002, pp. 502–507.
[15] M. Gerla and J. T.-C. Tsai, "Multicluster, mobile, multimedia radio network," *ACM J. Wireless Networks*, vol. 1, no. 3, pp. 23–29, 1995.
[16] H.-C. Lin and Y.-H. Chu, "A clustering technique for large multihop mobile wireless networks," in *Proc. IEEE Vehicular Technology Conf.*, Sept. 2000, pp. 1545–1549.
[17] R. K. Jain, D. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," DEC, Tech. Rep. DEC-TR-301, 1984.
[18] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison study of multi-hop wireless ad hoc network routing protocols," in *Proc. ACM/IEEE Int. Conf. Mobile Computing and Networking*, Oct. 1998, pp. 85–97.

Hsi-Lu Chao (S'02) received the B.S. degree in electronic engineering from Fengchia University, Taichung, Taiwan, in 1992 and the M.S. degree in electrical engineering from University of Southern California, Los Angeles, in 1996. She is currently working toward the Ph.D. degree in electrical engineering, National Taiwan University, Taipei, Taiwan.

Her research interests include wireless communication and QoS issues in ad hoc networks.

Wanjiun Liao (S'96–M'97) received the B.S. and M.S. degrees from National Chiao Tung University, Hsinchu, Taiwan, in 1990 and 1992, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1997.

She joined the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, Taiwan, as an Assistant Professor in 1997. Since August 2000, she has been an Associate Professor. Her research interests include wireless networks, optical networks, and broadband Internet.

Dr. Liao is actively involved in the international research community. Most recently, she has been appointed as the Newsletter Editor of IEEE ComSoc Asia Pacific Board (2002–2003) and the Tutorial Chair of IEEE INFOCOM 2004. She is currently an Associate Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. She has received many research awards. She was a recipient of the Outstanding Research Paper Award in Electrical Engineering at the University of Southern California in 1997. Two papers she coauthored with her students received the Best Student Paper Award at the First IEEE International Conferences on Multimedia and Expo (ICME) in 2000 and the Best Paper Award at the First International Conference on Communication, Circuits and Systems (IEEE Communication Press) in 2002. She was elected as one of Ten Distinguished Young Women in Taiwan in 2000 and is listed in the *Marquis Who's Who* in 2001–2003 and the *Contemporary Who's Who* in 2003.