SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks

Gahng-Seop Ahn, Andrew T. Campbell, Andras Veres and Li-Hsiang Sun

Abstract—**We propose SWAN, a stateless network model which uses distributed control algorithms to deliver service differentiation in mobile wireless ad hoc networks in a simple, scalable and robust manner. We use rate control for UDP and TCP best-effort traffic, and sender-based admission control for UDP real-time traffic. SWAN uses explicit congestion notification (ECN) to dynamically regulate admitted real-time traffic in the face of network dynamics brought on by mobility or traffic overload conditions. We use the term "soft" real-time services to indicate that real-time sessions could be regulated or dropped due to mobility or excessive traffic overloading at mobile wireless routers. SWAN is designed to limit such conditions, however. A novel aspect of SWAN is that it does not require the support of a QOS-capable MAC. Rather, soft real-time services are built using existing best effort wireless MAC technology. Simulation, analysis, and results from an experimental wireless testbed show that real-time applications experience low and stable delays under various multi-hop, traffic and mobility conditions. The wireless testbed and ns-2 simulator source code are available from the Web (comet.columbia.edu/swan).**

I. INTRODUCTION

There is a growing need to support quality of service (QOS) in mobile ad hoc networks [1], however, this is challenging. Wireless ad hoc networks represent distributed systems, which interconnect wireless mobile nodes without the need for fixed infrastructure. The interconnection between remote nodes relies on peer wireless and mobile nodes that operate as routers on behalf of source-destination pairs. Rerouting among mobile devices causes topology and network load conditions to change dynamically, making it difficult to support real-time applications with appropriate QOS.

Another challenge in supporting QOS for real-time applications is associated with the design of the medium access control (MAC) protocol. The dynamic nature of wireless ad hoc networks makes it difficult to dynamically assign a central controller to maintain connection state and reservations. Because of this, best effort distributed MAC controllers are widely used in existing wireless ad hoc networks. The IEEE 802.11 Distributed Coordination Function (DCF) is a good example of a best effort distributed MAC. Recently, there have been a number of proposals to support service differentiation at the MAC layer using distributed control schemes [2] [3].

In this paper, we assume a best effort MAC and propose a simple, distributed, and stateless network model called SWAN that uses feedback-based control mechanisms to support soft real-time services and service differentiation in wireless ad hoc networks. We use rate control for UDP and TCP best-effort traffic and sender-based admission control for UDP real-time traffic. We use explicit congestion notification (ECN) to dynamically regulate admitted real-time traffic in the face of network dynamics such as mobility and temporary traffic overload. Intermediate nodes do not keep per-flow state information in SWAN wireless networks. As a result, there is no need for signaling or complex control mechanisms to update, refresh, and remove per-flow state information, as is the case with "stateful" mobile ad hoc networks found in the literature [1] [3]. Changes in topology and network conditions, even node and link failure do not affect the operation of the SWAN control system. This makes the system simple, robust, and scalable. Instead of depending on state information, SWAN uses feedback information from the network. A rate control mechanism uses the MAC delay measurements from packet transmissions as feedback, while a source-based admission control mechanism uses rate measurements from aggregated real-time traffic as feedback.

In order to ensure that the bandwidth and delay requirements of real-time UDP traffic are met, rate control of TCP and UDP best effort traffic is performed at every mobile node in a fully distributed and decentralized manner. Rate control is designed to restrict best effort traffic yielding the necessary bandwidth required to support real-time traffic. Rate control also allows the best effort traffic to efficiently utilize the bandwidth that is not utilized by the real-time traffic at any moment. The total rate of all best effort and real-time traffic transported over each local shared media channel should be maintained below a certain "threshold rate", limiting any excessive delays that might be experienced. SWAN adopts engineering techniques that attempt to set the admission threshold rate at mobile nodes to operate under the saturation level of the wireless channel.

Addressing these goals is challenging in mobile wireless ad hoc networks because of the dynamic nature of the wireless channel, and the unpredictability of mobile devices. In SWAN, we adapt the well-known additive increase multiplicative decrease (AIMD) rate control mechanism to address some of these challenges. AIMD algorithms are widely used by a number of transport protocols. For example, the TCP congestion control mechanism uses AIMD window based control, while WTCP [4] uses AIMD rate control. In [10] AIMD control is applied to real-time UDP traffic. TCP and WTCP use AIMD control to improve the performance of TCP traffic. In contrast, we propose to use AIMD rate control to improve the performance of real-time UDP traffic. TCP attempts to avoid network congestion collapse by using packet loss as feedback. We propose to control the rate of TCP traffic more conservatively to avoid excessive delays of real time UDP traffic by using packet delay as a feedback to local rate controllers. Figure 1 illustrates the general behavior of a TCP-like congestion controlled system [8]. The congestion control algorithm ensures that the system works around, or preferably close to, the "cliff", which ensures maximum system throughput, but at the cost of larger queues, and therefore larger average delays. SWAN's AIMD control algorithm, on the other hand, keeps the system at the delay "knee", where the system throughput is almost the same as at the cliff, but the buffers are significantly less loaded, so the delay is close to the minimum. SWAN achieves this by using the MAC delay as a feedback instead of packet loss. The reason for doing this is that loss typically happens at the cliff, while delays start to increase at the knee, as illustrated in Figure 1.

This paper is structured as follows. Section II presents the SWAN network model. Section III describes distributed control algorithms for rate control, source-based admission control and dynamic regulation, respectively. Section IV analyzes the MAC delay and the busy probability of a wireless network with and

Gahng-Seop Ahn, Andrew T. Campbell, Andras Veres and Li-Hsiang Sun are with the COMET Group, Columbia University, New York, e-mail: ahngang@comet.columbia.edu.

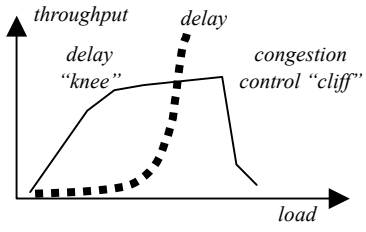

Fig. 1. General behavior of a congestion controlled system.

without rate control. Sections V and VI present the performance evaluation of SWAN using the ns-2 simulator and an experimental wireless testbed, respectively. Section VII presents some concluding remarks.

II. SWAN MODEL

The SWAN model includes a number of mechanisms used to support rate regulation of best effort traffic, as illustrated in Figure 2. A classifier and a shaper operate between the IP and MAC layers. The classifier is capable of differentiating real-time and best effort packets, forcing the shaper to process best effort packets but not real-time packets. The shaper represents a simple leaky bucket traffic shaper. The goal of the shaper is to delay best effort packets in conformance with the rate calculated by the rate controller, as illustrated in Figure 2.

There is no flow or session state information maintained at intermediate nodes in support of end-to-end communications between source destination pairs. Furthermore, when a session is admitted there is no admission control decision taken at intermediate nodes. Rather, the admission control test to determine if a new session should be admitted or not is conducted solely at the source node. A key operation of the admission controller, which is based at every mobile device, is to efficiently estimate local bandwidth availability. The admission controller located at the source node probes the network between the source and destination to determine the instantaneous end-to-end bandwidth availability. Based on the results of a *request*/*response* probe the session admission controller located at source node makes a decision to admit a new real-time flow or not. Once a session is admitted as a real-time session its packets are marked as RT (for real time service) otherwise they are considered as best effort packets. We use the DS (DiffServ) code word to maintain this packet state information in our SWAN wireless testbed and ns-2 simulation environment. Typically, a bandwidth probe is sent at the beginning of a session or, as discussed later, when mobility or channel load conditions force a real-time session to re-establish its end-to-end service quality.

Once a session is admitted it is desirable to maintain service quality for the lifetime of the session. Because our wireless network model takes a conservative approach when allocating bandwidth to real-time traffic, small scale violations of service quality can be tolerated without impacting application level QOS. These small-scale violations may occur because of bursty real-time sources or unpredictable traffic patterns. In a static wireless ad hoc network there is little need for further control algorithms above and beyond the rate control of best effort traffic and admission control of real-time traffic. One could even argue that under low mobility conditions this approach is sufficient for the delivery real-time performance. However, the bandwidth availability and dynamics of a wireless channel may change rapidly in the case of moderate to higher levels of mobility. Larger-scale violations may occur when real-time flows are admitted or dynamically re-routed. In the former case, multiple source nodes could simultaneously send new session probes that may traverse common intermediate nodes facilitating the

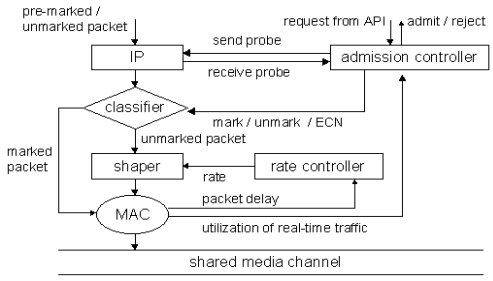

Fig. 2. SWAN Model.

admission of new sessions. This in turn could overload these common intermediate nodes. There is a need for additional SWAN mechanisms that can help resolve these issues.

We propose to regulate real-time sessions when a mobile node observes violation of real-time sessions; for example, due to mobility or source-based admission control. We adopt a regulation mechanism based on ECN, which was originally proposed for controlling and improving TCP traffic performance in IP networks. To allow for experimentation with IPv4, two bits (i.e., ECN-Capable Transport and Congestion Experienced bits) have been set-aside in the IP header for ECN [5]. In another proposal [6] ECN is used to control both TCP and non-TCP traffic in a wireline DiffServ architecture. We propose to use ECN to control and regulate UDP real-time traffic in the case of traffic violations most likely brought on by the re-routing of real-time sessions. By regulation, we mean that the ECN mechanism forces real-time flows to re-establish their real-time service. Under such conditions an existing flow would either be able to re-establish its original service quality or be dropped. We do not consider bandwidth adaptation of real-time sessions in this paper. Rather, we assume that real-time flows attempt to re-establish service at their original bandwidth levels.

It should also be noted that SWAN's ECN control system also serves to regulate real-time sessions in the case of persistent congestion, or due to potential overloading associated with source-based admission control. If multiple sources "read" (via request/response probes) the state of the network at the same time they may erroneously admit more real-time sessions than an intermediate node can support. We call this condition "false admission" and discuss it further in Section III-C. If false admission occurs, then the ECN control system will randomly regulate real-time flows, forcing the control system back to an operating point that is within the admission threshold of the wireless network. SWAN's design philosophy is not to add more protocol complexity or state information to resolve any perceived unfairness among real-time flows that are randomly selected for dynamic regulation. Rather, we rely on the existing SWAN distributed control algorithms to resolve such issues.

III. DISTRIBUTED CONTROL ALGORITHMS

A. Rate Control of Best Effort Traffic

Each node in the mobile ad hoc network independently regulates best effort traffic. The rate controller determines the departure rate of the shaper using the AIMD rate control algorithm based on feedback from MAC. This feedback measure used by the rate controller represents the packet delay measured by the MAC layer. In our model we use existing best effort MAC technology as part of SWAN. Packet delay for the IEEE 802.11 DCF mode MAC, for example, can be measured rather simply. When a packet arrives at the MAC layer, the MAC listens to the channel and defers access to the channel according to the CSMA/CA algorithm. When the MAC gets access to the channel then RTS-CTS-DATA-ACK packets are exchanged. The reception of an ACK packet at the transmitter indicates that a packet was

received successfully. The packet delay represents the time it took to send the packet between the transmitter and receiver including the total deferred time (including possible collision resolution) plus the time to fully acknowledge the packet. This is simply measured at the source node by subtracting the time that a packet is passed to the MAC layer (from the upper layer) from the time an ACK packet is received from the receiver.

SWAN's AIMD rate control algorithm is shown in Figure 3. Every T seconds, each mobile device increases its transmission rate gradually (additive increase with increment rate of **c** Kbps) until the packet delays become excessive. The rate controller detects excessive delays when one or more packets have greater delays than the threshold delay **d** sec. As soon as the rate controller detects excessive delays, it backs off the rate (multiplicative decrease by **r%**). The threshold delay d is based on the real-time delay requirements of applications in wireless network, as discussed in our previous work [2]. The shaping rate is adjusted every **T** second. The period T should be small enough to be responsive to the dynamics of mobile ad hoc networks [1] [11] [12].

```
Procedure update_shaping_rate ( )
/* called every T second period */
Begin

if (n > 0)                          /* one or more packets have delays
                                       greater than the threshold delay d sec */
   s ← s * (1 – r / 100)            /* multiplicative decrease by r% */

else

   s ← s + c                        /* additive increase by c Kbps */

if ( (s – a) > a * g / 100)         /* difference between actual rate and shaping
                                       rate is greater than g% of actual rate */
   s ← a * (1+ g / 100)             /* adjust shaping rate to match actual rate */

end
```

Fig. 3. SWAN AIMD Rate Control Algorithm.

If there is a large difference between the shaping rate and the actual transmission rate then a mobile device is capable of transmitting a burst without due control, hence, potentially limiting the performance of real-time traffic. To resolve this problem, the rate controller monitors the actual transmission rate. When the difference between the shaping rate and the actual rate is greater than **g%** of the actual rate, then the rate controller adjusts the shaping rate to be g% above the actual rate. This "gap" (i.e., g%) allows the best-effort traffic to increase its actual rate gradually.

In this paper, we argue that bandwidth and delay bound requirements of real-time traffic can be adequately supported by using rate control based on our simple SWAN AIMD rate control algorithm, while best effort traffic can efficiently utilize any remaining bandwidth. However, to fully support real time traffic, rate controlling best effort traffic is insufficient. There is also a need to support "edge" admission control.

B. Source-based Admission of Real-Time Traffic

Using a shared wireless channel allows mobile hosts to listen to packets sent within radio transmission range. An admission controller uses this feature to measure local resource availability. At each node, the admission controller measures the rate of real-time traffic in terms of bits per second. Note that in order to smooth out small-scale traffic variations, the admission controller uses a running average (e.g., weighted moving average) of these measures.

If we know the threshold rate [2] that would trigger excessive delays, then bandwidth availability in a local shared media channel is simply the difference between the threshold rate and current rate of the real-time traffic. However, it is difficult to estimate the threshold rate accurately because the threshold rate may change dynamically depending on traffic patterns. It is not desirable to admit real-time traffic up to the threshold rate for a number of reasons. First, best effort traffic would be starved of resources should the real-time traffic consume bandwidth up to such a threshold rate. Best effort traffic is rate controlled to yield the bandwidth required for real-time traffic and to keep the total traffic, both real-time and best effort, below the threshold rate. Second, there would be no flexibility to tolerate channel dynamics, as previously discussed. The total rate of aggregated real-time traffic may be dynamic due to changes in traffic patterns and node mobility. Due to node mobility, for example, intermediate nodes may need to maintain real-time traffic in excess of resources set-a-side for real-time traffic. There are a number of possible ways to address this problem.

We take a simple approach and admit real-time traffic up to a rate that is more conservative than the threshold rate. Therefore, the estimated available bandwidth of a local shared media channel is the difference between this conservative admission control rate and the current rate of the real-time traffic. With such a policy, we can use fixed, coarse, and statistically approximated values for the admission control rate. Even though the measure is conservative the utilization of the network is not limited because any remaining unutilized bandwidth will be potentially absorbed by the best-effort traffic. This approach is simple and flexible and allows bandwidth sharing between real-time and best-effort traffic.

The process of admitting a new session is as follows. The admission controller located at the source node sends a probing request packet toward the destination node to estimate the end-to-end bandwidth availability. The probing request packet is a UDP control packet that contains a "bottleneck bandwidth" field. Each intermediate node between the source-destination pair intercepts the probing request packet and updates the bottleneck bandwidth field if the bandwidth availability at the node is less than the current value of the field. Therefore, if the local bandwidth availability is different for each hop along the path between the source and destination then the value of the bottleneck field at the destination node represents the bottleneck bandwidth along the path. The destination node sends a probing response packet back to the source node with the bottleneck field copied from the probing request. There is no need for this probe response message to follow a reverse path back toward the source node.

Once the source node receives the probing response packet it can execute the simple source-based admission control by comparing the end-to-end bandwidth availability and the bandwidth requirement for the new real-time session. Note that no bandwidth request is carried in the probe message, no admission control is executed at intermediate nodes, nor are there any resources allocated or reserved on behalf of the source node during the lifetime of the session. Rather, the probe instantaneously "reads" the state of the network path presented to it by the routing protocol and makes a local source based admission decision based on the probe response. What makes such a stateless approach work is that all nodes independently regulate best effort traffic and each source uses admission control for real-time sessions. When a new real-time session is admitted, the packets associated with the flow are marked as RT (real-time packets/traffic). The classifier looks at the marking, and if the packet is marked as RT, the packet will bypass the shaper mechanism, remaining unregulated. Here there is an implicit assumption that the source node regulates its real-time sessions based on its admission control decision.

C. Dynamic Regulation of Real-Time Traffic

Impact of Mobility and False Admission

Mobility and "false admission" represent two conditions that violate this simple approach to source-based admission control thereby complicating the delivery of soft-QOS assurances in stateless wireless networks. Node mobility is harmful because real-time flows admitted along a certain path are dynamically re-routed due to node mobility. Because nodes are unaware of mobility and flow re-routing, resource conflicts can arise and persist. Source nodes, for example, that have been previously admitted flows are unaware of node mobility and the re-routing of flows through new intermediate nodes that may have insufficient resources to support previously admitted traffic. False admission is a result of multiple source nodes simultaneously initiating admission control at the same instance and sharing common paths and nodes between source-destination pairs. Because intermediate nodes do not maintain state information and admission control is conducted at the edge/source node in a fully decentralized manner, each source node may receive a response to their probe message indicating that resources are available when in fact they are not. However, the source node is unaware of this fact and falsely admits a new flow and starts transmitting real-time packets under the assumption that resources are available to meet the flows needs. Consider the following false admission scenario. Four source nodes want to establish video flows at a rate of 200 Kbps and probe the network. A common intermediate node only has resources to support 200 Kbps of real-time traffic, sufficient to support only a single video flow. However, in the case of false admission all the flows are admitted erroneously because all the nodes "see" a reservation that can support 200 Kbps. This results in the four source nodes pumping data into the wireless network with an aggregate rate of 800 Kbps, destined toward the common node under discussion. If left unresolved, the re-routing of admitted flows and false admission can cause excessive delays in real-time traffic, because, the utilization of the admitted or falsely admitted real-time traffic can violate the admission control rate and exceed the threshold rate by a significant margin. To resolve these problems, we augment the SWAN AIMD rate control and source-based admission control algorithms with dynamic regulation of real-time traffic when congestion/overloading is experienced by nodes due to the re-routing of admitted flows and false admission.

Source and Network-Based Regulation Algorithms

The ECN-based regulation of real-time sessions operates as follows. Each node continuously, and independently, measures the utilization of its real-time traffic to estimate the local available bandwidth, as discussed in Section III-B. Each mobile node can detect violations (i.e., congestion/overload conditions) using this periodic traffic measurement. When a node detects such a violation, it starts marking the ECN bits in the IP header of the real-time packets. The destination node monitors the ECN bits and notifies the source using a regulate message. When the source node receives a regulate message, it initiates re-establishment of its real-time session based on its original bandwidth needs. To reestablish a real-time session a source node follows the same process as setting up a new session by sending a probing request toward the destination. A source node terminates the session if the estimated end-to-end bandwidth indicated in the probing response packet cannot meet its existing session needs. This is one of the reasons why we call our approach to service differentiation in mobile ad hoc networks "soft" because an admitted real-time flow may encounter both periodic violations in its bandwidth requirements and, in the worst case, may have to be "dropped" or live with degraded best effort delivery.

If the nodes in a congested or overloaded condition were to mark all packets with CE (Congestion Experienced) then all sessions traversing these nodes would be forced to be re-established their real-time service at the same time. Such an approach is inefficient and would lead to erroneous behavior. For example, all sources may re-probe the network, "see" network resources over utilized and drop all their flows accordingly. This clearly is not the best policy. More systematic approaches may only penalize a small number of sources. To address the problem, we consider two approaches and analysis their suitability and trade-offs:

Source-Based Regulation (SWAN-1). When an intermediate node experiences overload or congested conditions it marks all flows with CE. When destination nodes encounter packets with the CE bit marked they send regulate messages to the appropriate source nodes to force the re-establishment of flows that have previously been successfully admitted. However, in this case the source node does not immediately initiate re-establishment upon receipt of a regulate message. Rather, it waits for a random amount of time before initiating the re-establishment procedure. Under such a regime, source regulation would be staggered thereby avoiding flash-crowd conditions, where, a number of sources simultaneously initiate regulation (i.e., re-establishment of service) at the same time, "see" the path overbooked and drop their real-time sessions accordingly. Under a staggered regime, the rate of the real-time traffic will gradually decrease until it reaches below the admission control rate. At that point, congested or overbooked nodes will stop marking packets. Because flows can be admitted by mistake, due to false admission, source nodes need to be capable of differentiating between regulation associated with false admission and regulation due to mobility. Nodes can do this by keeping some state information about newly admitted flows versus on-going flows. This allows a source node to take immediate corrective action in the case where it receives a regulation message for a flow that it just admitted, albeit falsely. A disadvantage of this approach is that sources which regulate earlier than other sources (i.e., wait the shortest period of time before initiating re-establishment) are more likely to find the path overbooked and be forced to drop their sessions. An advantage of this scheme, however, is that it is purely source-based.

Network-Based Regulation (SWAN-2). Rather than marking all packets with CE, congested/overloaded nodes randomly select a "congestion set" of real-time sessions and only mark packets associated with this set. This can be done using a hash function without keeping any per-flow state at the intermediate nodes. A congested node marks the congested set for a period of time T seconds and then calculates a new congested set. As in the case of the previous algorithm, nodes stop marking packets "congested" when the measured rate of the real-time traffic drops below the admission control rate. Under such an approach the rate of the real-time traffic will gradually decrease until it reaches below the admission control rate. However, there is a need for intermediate routers to distinguish between flows that have been falsely admitted or not. In this case, source nodes could use an additional bit in the TOS field to indicate if a RT session is new or old (namely RT-new, RT-old). When a flow is newly admitted, packets are marked as RT-new for a period of time before being marked as RT-old. A disadvantage on this scheme is that it requires some intelligence at intermediate nodes to manage the congested sets and determine if a flow is new or old in order to correctly respond to false admission.

Performance Considerations and Trade-offs

There are a number of trade-offs between the source-based and network-based regulation schemes. Figure 4 compares the two approaches and shows how a combination of admission control and regulation can manage the rate of real-time traffic under

overload conditions. The results were obtained from an ns-2 simulation of SWAN that is further discussed in Section V. To observe how regulation works, we consider an extreme scenario where a number of real-time sessions are suddenly rerouted in the network through a certain hop. The re-routed sessions have all previously been successfully admitted. As shown in Figure 4, due to mobility re-routed flows start to be routed over the shared media channel (without admission control) at 20 sec into the simulation scenario. Also a number of source nodes simultaneously perform admission control for new real-time sessions creating conditions for false admission at 20 sec into the scenario. Note, that the proposed admission control mechanism allows small-scale violation up to the threshold rate for ECN-based regulation. In this simulation, the admission control rate is set at 2 Mbps and the threshold rate is 3.5 Mbps. The channel bandwidth is 11 Mbps. For a more detailed discussion on the setting of these system parameters, which are determined by considering the requirements of the admission control and threshold rates, see our previous work [2] on the analysis of distributed MAC delays. As soon as the rate exceeds the threshold rate, the mobile nodes in the shared media channel begin to mark the ECN field of the packets of all real-time flows (SWAN-1) or the active set (SWAN-2) depending on the scheme.

Figure 4, shows the results for both schemes. For both algorithms flows are dropped gradually after intermediate nodes start to mark the ECN field in the packets of real-time flows. Note that the dropped flows may not necessarily be the re-routed sessions but existing sessions that were admitted previously. This may seem unfair but our approach is stateless and there is no mechanism for congested nodes to differentiate between existing and re-routed sessions. Furthermore, it is likely that most real-time sessions would be re-routed multiple times during the lifetime of their sessions. In this case, there is little benefit in attempting to discriminate between the existing and re-routed flows when statistically all sessions should be treated in the same manner on average. We used a random number between (0, 7) to support the back-off and set selection functions required by the SWAN-1 and SWAN-2 schemes, respectively.

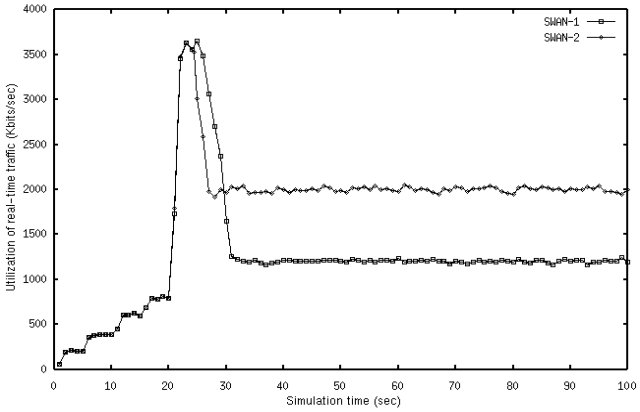

Fig. 4. Comparison of the source-based delayed regulation (SWAN-1) and network-based set regulation (SWAN-2) schemes. Trace of the actual rate of the real-time traffic for both schemes.

From Figure 4, we observe that the rate of the real-time traffic is gradually reduced until it reaches the admission control rate for the SWAN-2 approach. We observe that this took approximately 4 seconds for SWAN-2. The simulation results indicate that after re-routing, nineteen sessions were traversing the node under congested conditions. Five of these sessions were traversing the node prior to overloading. Eight sessions were re-routed sessions and six sessions were the product of false admission. The regulation process resulted in eleven sessions being successfully regulated with eight sessions dropped. In the scenario discussed above, all falsely admitted sessions are dropped prior to other flows being dropped, with one of the dropped flows traversing the node before flows were re-routed. The response of the source-based regulation scheme (i.e., SWAN-1) took longer (2 seconds) than the network-based approach (i.e., SWAN-2). This additional latency caused more real-time traffic to be dropped over that experienced by SWAN-2. In the case of SWAN-1, the network operates below the admission control rate indicating that the scheme results in under utilization of resources at the congested node, as shown in Figure 4. In summary, the SWAN-2 approach performs better than SWAN-1 with the cost of an additional 1-bit in the packet header for marking flows new/old, and more intelligence at intermediate nodes to manage the sets to mark.

IV. ANALYSIS

We analyze the MAC delay and the probability that mobile devices find themselves in a backlogged state in IEEE 802.11 wireless networks. We use the terms "original system" and "proposed system" to refer to IEEE 802.11 wireless networks with and without rate control, respectively. In section IV-A, we show through analysis that the proposed system performs better than the original system in terms of MAC delay. Section IV-B explains why this is the case. We show that by controlling the probability of mobile nodes being in a backlogged state, the target MAC delay of the real-time traffic can be maintained. This result confirms that the SWAN approach is feasible and effective.

A. Analysis of MAC Delay

Assume there are two classes of mobile devices in the shared channel environment. Class 1 and Class 2 represent real-time UDP traffic and best effort TCP traffic, respectively. Each of the n_1 Class 1 mobile devices has an active UDP session, and each of the n_2 Class 2 mobile devices has an active TCP session. We define an idle mobile device as a mobile device whose MAC layer is idle and interface queue empty. If a mobile device is not idle then it is busy. Denote the portion of time that a class i mobile device is busy as $p_{on,i}$. From [7], a busy class i mobile device's transmission probability in each time slot is represented as,

$$\tau_i = \frac{2\cdot(1-2p_i)}{(1-2p_i)(W+1)+p_iW(1-(2p_i)^m)} \qquad (1)$$

where p_i is the collision probability for a class i mobile device at each time slot. W is the initial back-off window, and $W2^m$ is the maximum back-off window in the IEEE 802.11 protocol. By following the procedure in [7], the collision probability can be represented as,

$$\begin{aligned} p_1 &= 1-(1-p_{on,1}\tau_1)^{n_1-1}(1-p_{on,2}\tau_2)^{n_2} \\ p_2 &= 1-(1-p_{on,1}\tau_1)^{n_1}(1-p_{on,2}\tau_2)^{n_2-1} \end{aligned} \qquad (2)$$

The probability that one or more packets are sent to the channel at each slot is then,

$$P_{tr} = 1-(1-p_{on,1}\tau_1)^{n_1}(1-p_{on,2}\tau_2)^{n_2}, \qquad (3)$$

and the probability of a successful transmission each slot is,

$$\begin{aligned} P_s = &\frac{n_1 p_{on,1}\tau_1(1-p_{on,1}\tau_1)^{n_1-1}(1-p_{on,2}\tau_2)^{n_2}}{P_{tr}} \\ &+\frac{n_2 p_{on,2}\tau_1(1-p_{on,1}\tau_1)^{n_1}(1-p_{on,2}\tau_2)^{n_2-1}}{P_{tr}} \end{aligned} \qquad (4)$$

The total throughput of the system (in packets/sec) can be represented as,

$$S = \frac{P_s P_{tr}}{(1-P_{tr})\sigma + P_{tr}(P_s T_s + (1-P_s)T_c)} \qquad (5)$$

where σ is the length of a time slot, which is 20μs in all our simulations. T_s and T_c are the times needed to send un-collided

and collided packets, respectively, on the channel. T_s and T_c are calculated from the packet length distribution, taking into account the overhead of the MAC and physical layers (i.e., SIFS, DIFS, ACK, header, and preamble). The length of collided packets is approximated as the maximum length of two collided packets. The overall average MAC delay can be simply calculated using Little's formula as,

$$d = \frac{p_{on,1}n_1 + p_{on,2}n_2}{S} \qquad (6)$$

We carried out an ns-2 simulation for 4 video (Class 1) mobile devices and 8~32 FTP (Class 2) mobile devices. In this simulation, video sessions are modeled as CBR sources and the FTP sessions have an infinitely long file sizes that lasts for the whole simulation period. We denote the uncontrolled system as the original system, and the system with the proposed feedback control as the proposed system. Because it is difficult to use a simple model to characterize the flow control of TCP/IP, coupled with a queuing system on top of the MAC layer, and the MAC layer and traffic shaper (for the proposed system), we record the quantity $p_{on,1}, p_{on,2}$ during the simulation as an approximation of this complex system. With $p_{on,1}, p_{on,2}, n_1, n_2$ known, we jointly solve (1), (2) for p_1, p_2, τ_1, τ_2, then from (3), (4), (5), (6), the overall average delay is computed. The results are plotted in Figure 5. It is shown that the analytical delay approaches closer to the overall average delay than the MAC delay of the video sessions for the original system.

It can be shown that the proposed control mechanism is equivalent to a rate control mechanism that controls the probability of the interface queue system (including the MAC layer) being busy. Because the input to the analysis, $p_{on,1}, p_{on,2}$ are different for the original and proposed systems, this results in quite different delay performance, as shown in Figure 5. The analysis and simulation results show that the proposed system performs well for a growing number (#) of TCP sources.

B. Analysis of Busy Probability

We now analyze the proposed system from another perspective and try to find the value of $p_{on,2}$ (i.e. the probability of a Class 2 mobile device being in backlogged state) that Class 2 mobile devices must achieve so that the target average MAC delay can be maintained. We assume no packet loss due to buffer overflow for Class 1 mobile devices. Because Class 1 mobile devices carry UDP real-time sessions with known data rates (S_1 packets/sec) from the application layer, we set our target MAC delay as d, so $p_{on,1}$ can be acquired from the following:

$$p_{on,1} = d \cdot S_1 \qquad (7)$$

We use a procedure called Equilibrium Point Analysis (EPA) [9] in calculating the collision probabilities. In this approach, probabilities are calculated in terms of the equilibrium point of the system, and the average number of mobile devices in each state is often chosen as the point. Here we choose the average number of backlogged Class 2 mobile devices $\bar{n}_2$ as the equilibrium point. We modify (2) as,

$$p_1 = 1-(1-p_{on,1}\tau_1)^{n_1-1}(1-\tau_2)^{\bar{n}_2} \qquad (8)$$
$$p_2 = 1-(1-p_{on,1}\tau_1)^{n_1}(1-\tau_2)^{\bar{n}_2-1}$$

Note that we can still use (2) to find $p_{on,2}$ but slightly worse analytical results are produced. We modify (3) accordingly as,

$$P_{tr} = 1-(1-p_{on,1}\tau_1)^{n_1}(1-\tau_2)^{\bar{n}_2} \qquad (9)$$

The probability of transmission of Class 1 mobile device being successful, conditioned on at least one mobile device transmitting is given by,

$$P_{s1} = \frac{n_1 p_{on,1}\tau_1(1-p_{on,1}\tau_1)^{n_1-1}(1-\tau_2)^{\bar{n}_2}}{P_{tr}} \qquad (10)$$

The probability of a packet transmission being successful, conditioned on at least one mobile device transmitting is then given by,

$$P_s = \frac{n_1 p_{on,1}\tau_1(1-p_{on,1}\tau_1)^{n_1-1}(1-\tau_2)^{\bar{n}_2}}{P_{tr}} + \frac{\bar{n}_2\tau_1(1-p_{on,1}\tau_1)^{n_1}(1-\tau_2)^{\bar{n}_2-1}}{P_{tr}} \qquad (11)$$

As in (5), the throughput (in packets/sec) of Class 1 mobile device is,

$$S_1 = \frac{P_{s1}P_{tr}}{(1-P_{tr})\sigma + P_{tr}(P_s T_s + (1-P_s)T_c)} \qquad (12)$$

We choose the Class 1 MAC delay in the proposed system observed in the simulation-video curve of Figure 5 as our target delay d. We put (7), (9), (10), (11) to (12) and solve (1), (8), (12) jointly for $\bar{n}_2, \tau_1, \tau_2, p_1, p_2$. Following this, we calculate the probability of a Class 2 mobile device being busy as,

$$\bar{p}_{on,2} = \frac{\bar{n}_2}{n_2} \qquad (13)$$

Figure 6 shows analytical results of $p_{on,2}$ in comparison to the measured $p_{on,2}$ from our simulation of the proposed system. In order to achieve the target delay d for Class 1 mobile devices, we need to keep the probability of Class 2 being busy to be less than $p_{on,2}$. The input parameters of the above analysis are d (desired average delay), S_1 (the throughput of Class 1 mobile devices), n_1, and n_2 (the number of Class 1 and 2 mobile devices, respectively) and the output is $p_{on,2}$ (the probability of Class 2 mobile devices being busy). With S_1, n_1, n_2 fixed, the delay statistics are positively related to $p_{on,2}$, where a delay violation implies a violation of $p_{on,2}$. This prompts us to correct this situation by minimizing the possibility that a mobile device will

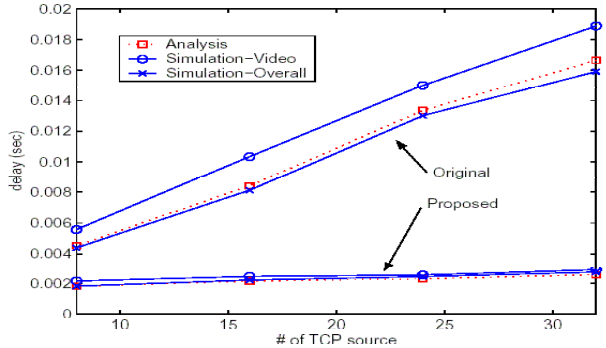

Fig. 5. A Delay Comparison of Original and Proposed Systems.

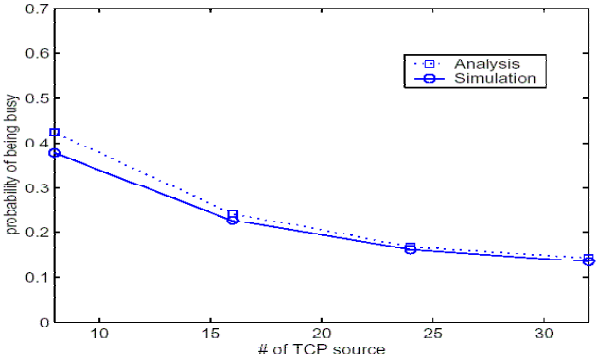

Fig. 6. Comparison of Busy Probability.

be still in a backlogged state. In our proposed system, this is achieved by the multiplicative decrease procedure when the delay violation is observed. On the other hand, when the delay is small, $p_{on,2}$ is also small. The system is under-loaded. The additive increase procedure increases $p_{on,2}$ by gradually increasing the packet arrival rate. The AIMD mechanism with delay feedback is thus an automatic procedure to keep $p_{on,2}$ on a desirable level, so the bandwidth is effectively utilized, but the system is not overloaded. As a result, TCP traffic has reasonable throughput, while UDP traffic achieves the desirable delay performance. Figure 6 compares the analytical result of the desirable $p_{on,2}$ for achieving the target MAC delay, with the measurement from the simulations of the proposed system. The simulation curve closely matches the analysis curve. This result confirms that AIMD rate control is capable of keeping $p_{on,2}$ at a desired level, and thus maintaining the target MAC delay.

Figure 6 shows that it is exactly what the SWAN AIMD rate control mechanism does. The simulation curve closely matches the analysis curve. This result confirms that AIMD rate control is capable of minimizing $p_{on,2}$ and thus maintaining the MAC delay under a certain boundary.

V. EVALUATION

We implemented SWAN using the ns-2 simulator and its wireless extensions developed at CMU [12]. The SWAN ns-2 extensions include the AIMD rate controller, admission controller, packet delay measurement mechanism, local utilization monitoring, probe protocol for bandwidth availability estimation, and explicit congestion notification. As mentioned in the previous section, we use the terms "proposed system" and "original system" to refer to wireless ad-hoc networks with and without the SWAN mechanisms, respectively. Throughout the simulation, each mobile node has a transmission range of 250 meters and shares an 11 Mbps radio channel with its neighboring nodes. The simulation includes a two-ray ground reflection model and IEEE 802.11 MAC protocol.

A. Performance of a Single Shared Channel

To best understand the characteristics of SWAN's rate control and admission control mechanisms we first study a wireless ad hoc network that comprises a single shared wireless channel. The simulated network has a square shape of 150m x 150m where all wireless ad hoc mobile nodes share a single radio channel of 11 Mbps. The source and destination nodes associated with flows are distributed among the mobile nodes in the wireless ad hoc network. We ran a large set of simulations using different values for the AIMD rate controller parameters, c (increment rate, Kbps/sec), r (decrement rate, %), and g (gap between actual rate and shaping rate, %) to understand the characteristics, trade-offs and performance of our rate control mechanism.

Gap Control Parameter (g) Analysis

We use 4 TCP connections to see how the SWAN AIMD rate controller controls TCP traffic. In this simulation, all TCP flows are greedy FTP type of traffic with packet size of 512 bytes. Figure 7 shows a trace of a TCP traffic flow that exhibits some inefficiency under our rate control regime. Note, that the actual and shaping rates do not match, and as shown in the trace, the shaping rate keeps growing while the actual rate of the TCP stalls and cannot follow the computed shaping rate. Such a disparity between the actual and shaping rates can be harmful. If left unresolved the "gap" can affect the performance of real-time traffic and other best-effort traffic. Such a condition arises naturally for TCP. For example, TCP traffic cannot follow the computed shaping rate when TCP traffic reaches the maximum throughput of the network or backs-off due to packet losses, etc. To resolve the mismatch of the actual and shaping rates, we introduce a gap control algorithm with parameter g, as described in Section III-A. Figure 8 illustrates the trace of the TCP traffic with the gap control algorithm activated. From Figure 8, we can observe that the actual rate closely follows the shaping rate providing better rate control for the TCP traffic; that is, gap control prevents a TCP from transmitting an uncontrolled, excessive burst of data, which could happen without gap control, as illustrated in Figure 7.

To understand the characteristic of parameter g, we measured the fairness between TCP flows (see Figure 9). The definition of fairness in [8] is used in measuring the fairness between TCP flows:

$$F(x) = \frac{\left(\sum x_i\right)^2}{n\left(\sum x_i^2\right)},$$

where x_i is the fraction of the bandwidth allocated for i-th flow and n is the number of flows. F becomes 1 when all flows share the exactly same fraction. The x-axis in Figure 9 represents the value for parameter g (gap between actual rate and shaping rate, %). As shown in Figure 9, fairness tends to decrease as the value of g increases except when the value of g is less than 10%. A mobile device may have more chance of transmitting a burst as the value of parameter g increases. If a mobile device is allowed to transmit a burst then this burst may limit the performance of other TCP flows. As a result, the fairness decreases as the value of g increases. If the value of g is too small, the gap between the

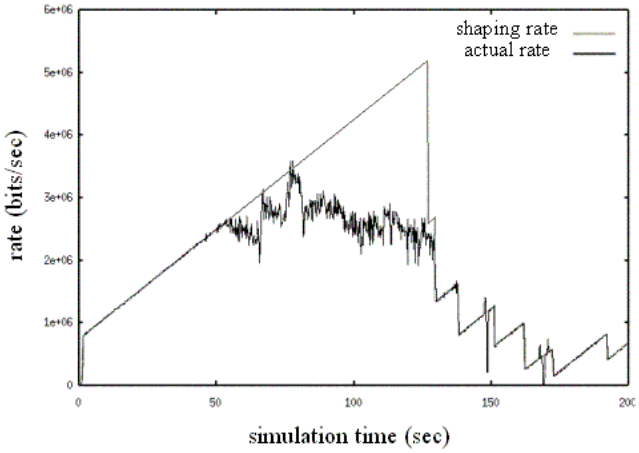

Fig. 7. Trace of the shaping rate and the actual rate of the best effort TCP traffic without the gap control algorithm.

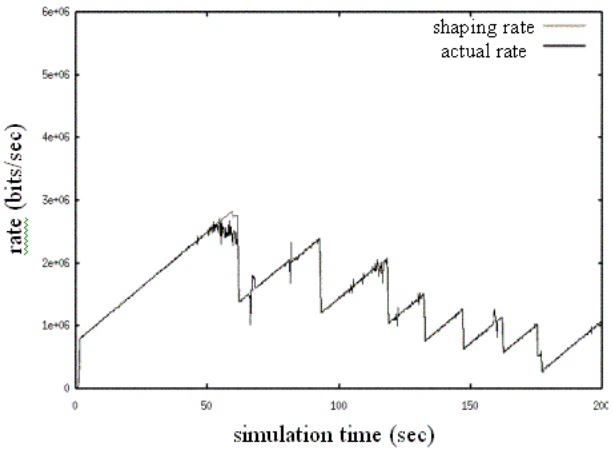

Fig. 8. Trace of the shaping rate and the actual rate of the best effort TCP traffic with the gap control algorithm.

actual rate and the shaping rate will be greater than g% most of the time, and the gap control algorithm will set the shaping rate to closely match the actual rate rather than increasing the shaping rate with increment rate c. So the additive increase of the AIMD algorithm may not be able to operate efficiently. Thus, TCP flows with smaller traffic rates may have little chance to grow their rate with the result that fairness may eventually decrease.

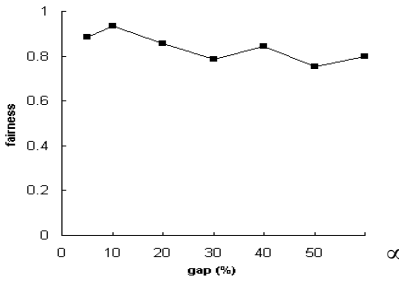

Fig. 9. Fairness between best-effort TCP flows vs. parameter **g.**

AIMD Parameter (c, r) Analysis

To better understand the properties of the SWAN AIMD rate control parameters c and r, we consider two scenarios for background TCP best-effort traffic. The first scenario has 8 TCP flows and the second has 32 TCP flows. In both scenarios, all TCP flows are greedy FTP type of traffic with packet size of 512 bytes. TCP flows are rate controlled with parameter c and parameter r, while voice and video flows are not rate controlled once admitted through the source-based admission control process. During the simulation, 4 voice flows and 4 video flows are active and monitored for the duration of 200 seconds representing real-time traffic. Voice traffic is modeled as 32 Kbps constant rate traffic with a packet size of 80 bytes. Video traffic is modeled as 200 Kbps constant rate traffic with a packet size of 512 bytes.

We measured the average MAC delay of real-time traffic (see Figures 10 and 12) and the total throughput of best-effort traffic (see Figures 11 and 13). The x-axis of Figures 10 and 11 represent the value for parameter c (increment rate, Kbps/sec). The x-axis in Figures 12 and 13 represents the value for parameter r (decrement rate, %). It is shown in Figure 10 that the value of parameter c does not have much impact on the average delay of real-time traffic. The average delay grows very slowly with the increasing value of parameter c. In contrast, the total throughput of best-effort TCP traffic is noticeably decreased when a small value of parameter c is chosen, as shown in Figure 11. When the increment rate is 5 Kbps/sec, throughput is reduced by about 10% for the 8 TCP flow scenario and by 13% for the 32 TCP flow scenario in comparison to the original system. For an increment rate of 20 Kbps/sec or larger, the TCP throughput becomes almost constant with less than 3% reduction in throughput.

The value of parameter r has significant impact on the average delay of the real-time traffic, as shown in Figure 12. When the decrement rate is set to 10 %, the average delay becomes almost as large as the average delay in the original system. The average delay becomes smaller as the value of parameter r increases. It is observed in Figure 13 that total throughput of the best-effort TCP traffic is also sensitive to the value of parameter r. The proposed system shows the best and worst-case performance in terms of the total throughput of best-effort TCP traffic when the value of parameter r is 25% and 75%, respectively. When the value of c is 35 and the value of r is 50, the average delay of the real-time traffic is reduced by more than 60% with 8 background TCP flows, and by 75% with 32 background TCP flows. These results demonstrate that we can achieve a reduction of 60-75% in the average delay of real-time traffic with a 2% loss of TCP

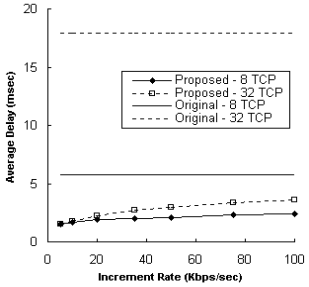

Fig. 10. Average delay of real-time traffic vs. increment rate.

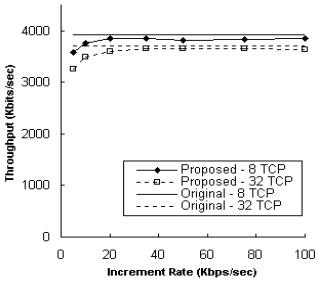

Fig. 11. Total throughput of best-effort TCP traffic vs. increment rate.

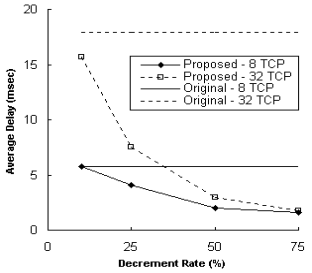

Fig. 12. Average delay of real-time traffic vs. decrement rate.

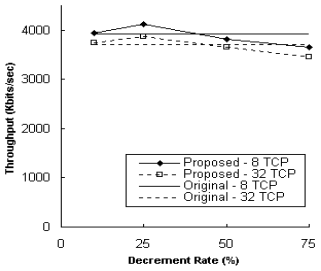

Fig. 13. Total throughput of best-effort TCP traffic vs. decrement rate.

throughput using the SWAN AIMD rate control algorithm. This is a promising result.

Figures 14 and 15 show the average delay of real-time traffic with a growing number of TCP flows and web sources, respectively. TCP flows are greedy FTP type of traffic with packet size of 512 bytes. The web sources are modeled as short TCP file transfers with random file size and random silent period between transfers. The file size is driven from a Pareto distribution with a mean file size of 10 Kbytes and a shape parameter of 1.2. The length of the silent period between two transfers is also Pareto in distribution with the same shape parameter with a mean of 10 seconds. This creates a highly bursty background best-effort traffic load over multiple time-scales. Web traffic represents micro flows, whereas, FTP traffic corresponds to macro flows. The real-time traffic is modeled in the same manner as discussed in the previous simulation using 4 voice flows of 32 Kbps and 4 video flows of 200 Kbps. From Figure 14, we observe that the average delay of real-time traffic in the original system is the same as the proposed system without background TCP flows. The average delay of real-time traffic in the original system grows linearly from 3 to 17 msec when the number of background TCP flows increase from 4 to 32 flows. In contrast, the average delay of real-time traffic in the proposed system remains less than 3 msec.

Figure 15 shows that the average delay for real-time traffic in

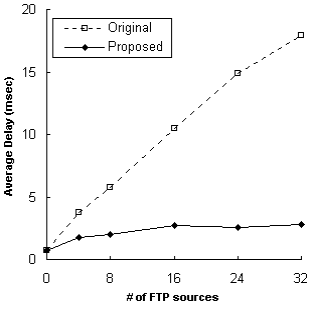

Fig. 14. Average delay of real-time traffic vs. number of FTP macro-flows.

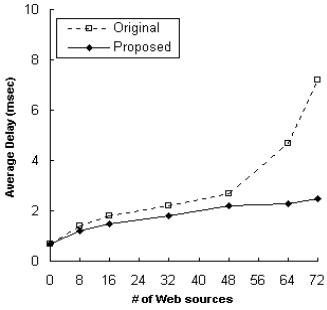

Fig. 15. Average delay of real-time traffic vs. number of web micro-flows.

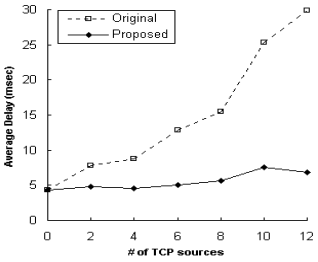

Fig. 16. Average delay of real-time traffic vs. number of TCP flows.

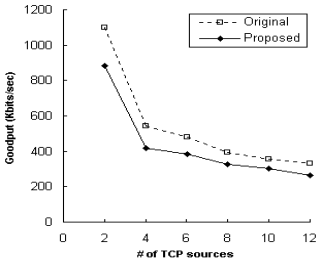

Fig. 17. Average "goodput" of TCP best-effort traffic vs. number of TCP flows.

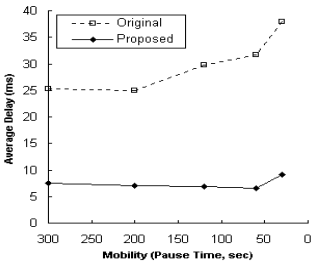

Fig. 18. Average delay of the real-time traffic vs. mobility.

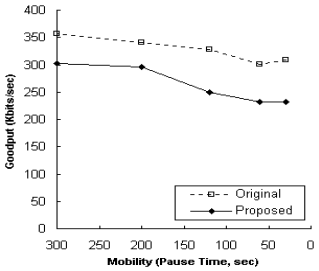

Fig. 19. Average goodput of the best–effort TCP traffic vs. mobility.

the original and proposed systems is similar for up to 48 web sources. The average delay of real-time traffic in the original system grows from 2 to 7 msec when the number of web sources increases from 48 to 72 micro flows. In contrast, the average delay of the real-time traffic in the proposed system remains around 2 msec. It is observed that the average delay in the proposed system is less than 4 msec even when web traffic is of high intensity.

The results presented in this section imply that the proposed system can support soft real-time traffic with consistently low delay by controlling the rate of best-effort TCP and web traffic for a single shared media channel. In the next section, we investigate the performance of the proposed system that considers multi-hops and varying device mobility.

B. Performance of Multi-hop Scenarios with Mobility

In this section, we consider a simulated multi-hop network with 50 mobile ad hoc nodes. The network area has a rectangular shape of 1500m x 300m that minimizes the effect of network partitioning. AODV [11] is used for routing in the simulated network. The real-time traffic is modeled in the same manner as discussed previously. The background TCP traffic is modeled as a mixture of FTP and web traffic. Typically, flows traverse 2-5 hops (3 hops on average) between source-destination pairs.

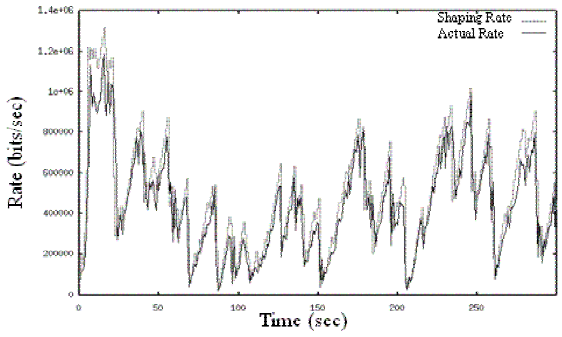

Fig. 20. Trace of the shaping rate and the actual TCP transmission rate.

Figures 16 and 17 show the average end-to-end delay for real-time traffic and TCP best effort traffic for an increasing number of background TCP traffic, respectively. We observe that the packet loss of the real-time traffic is less than 1% in both the original and proposed systems. However, the average delay of the real-time traffic shows a significant difference between the original and proposed systems. The average end-to-end delay of the real-time traffic in the original system grows linearly from 8 to 30 msec, as the number of TCP flows increase from 2 to 12 flows, respectively. In contrast, the average delay of real-time traffic in the proposed system remains around 5 to 7 msec. The average "goodput" of TCP traffic in the proposed system is about 15-20% less than the original system. By adopting the proposed control mechanisms, we observe a 38-77% reduction in the average delay of the real-time traffic at a cost of 15-20% loss of TCP goodput. In addition, the average delay of the real-time traffic remains consistently below 8 msec in the proposed system while the average delay in the original system grows above 30 msec.

The impact of mobility is illustrated in Figures 18 and 19. The simulated network is the same as the previous multi-hop scenarios with the addition of the introduction of mobility. We use a random way-point mobility model [12]. Each mobile node selects a random destination and moves with a random speed up to a maximum speed of 72 km/hr and pauses for a given "pause time" when the destination is reached. When the pause timer expires, the mobile node picks another random destination and moves at another random speed. The real-time traffic is modeled in the same manner as discussed previously. The number of best-effort TCP flows comprises 5 FTPs and 5 web micro-flows.

As shown in Figure 18, the average end-to-end delay of the real-time traffic in the original system increases slowly as mobility increases and the average end-to-end delay of the real-time traffic in the proposed system grows only for the highest mobility scenarios. We observed in the simulations that the throughput of the real-time traffic decreases slowly from 99% to 95% (i.e., the packet loss increases from 1% to 5%) as mobility increases in both the original and proposed systems. The impact of mobility on delay and throughput is due to route discovery latency and congestion along the new route. However, the end-to-end average delay of the real-time traffic in the proposed system remains under 10 msec in all cases while the average delay in the original system grows to 38 msec. The average goodput of best-effort TCP traffic in the proposed system is about 15-25% less than the original system, as shown in Figure 19.

In the proposed system, the average end-to-end delay of the real-time traffic is reduced by 70-75% with 15-25% loss of best-effort TCP goodput. The average end-to-end delay of the real-time traffic in the proposed system stays consistently below 10 msec while the average delay in the original system grows to 38 msec.

VI. WIRELESS TESTBED RESULTS

In what follows, we describe our experimental results from the SWAN wireless testbed, which is based on Linux notebooks using Aironet IEEE 802.11b wireless interfaces. The rate controller is implemented by modifying the Aironet device driver. We also modified the device driver to measure packet delay. The packet delay is measured by calculating the difference between the time the device driver feeds a new packet into an Aironet card and the time the Aironet card acknowledges back to the device driver that the transmission of the packet was successful. We implemented a traffic shaper driver between the kernel and the Aironet card device driver to control the rate of TCP traffic.

The utilization monitor and probe protocol are implemented using the Berkeley Packet Filter's Packet Capture library (PCAP). PCAP is designed to capture packets for statistical purposes but it can also be used to forward packets to the network interface. PCAP is used to capture every UDP packet transmitted within the

radio contact range of a wireless mobile host. The admission controller reads the IP header of captured UDP packets and estimates the local bandwidth availability. We also use PCAP to capture and forward probe signals. The admission controller estimates the end-to-end bandwidth availability when a source node probes the network path, as previously discussed.

The results presented in this section were obtained from a SWAN wireless ad hoc testbed, which consists of five mobile hosts using Aironet 11 Mbps IEEE 802.11b PCMCIA cards. The configuration of the testbed is as follows. Four mobile hosts generated TCP traffic and one mobile host generated UDP traffic. The source and the destination nodes associated with each flow were distributed among the mobile hosts. The UDP host generated packets every 20 msec at 32 Kbps with the rate of the TCP flows being controlled by the rate controller.

Figure 20 shows a trace of the shaping rate controlled by the rate controller and the actual TCP transmission rate. The actual TCP rate is well controlled by the shaper, as shown in Figure 20. When all four TCP flows were rate controlled, we measured the delay of each packet in a UDP real-time flow. Figures 21 and 22 show the delay of each packet when the TCP flows are regulated and unregulated, respectively. By comparing Figures 21 and 22, we can observe that the measured delay is improved when TCP flows are rate controlled. The average measured delay is 2.3 msec and 3.3 msec in Figure 21 and 22, respectively. The measured delay in Figure 21 remains below a certain boundary most of the time, while the delay in Figure 22 reaches significantly higher values. Figure 23 shows a simple distribution of the measured UDP real-time packet delay for the original (i.e., the wireless testbed without rate control) and proposed systems (i.e., the wireless testbed with rate control). We can observe that proposed system has more packets with packet delays smaller than 2 msec, and fewer packets with measured delays exceeding 4 msec. 66% of the packets have delays of less than 2 msec in proposed system, in comparison to 52% for the original system. In the proposed system only 11% of the packets have delays greater than 4 msec in comparison to 24% for the original system.

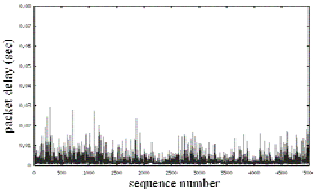

Fig. 21. The delay of each packet in a UDP real-time flow from the SWAN wireless testbed without rate control.

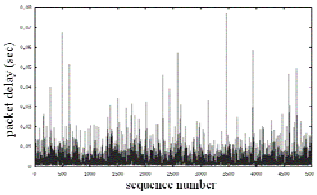

Fig. 22. Delay of each packet in a UDP real-time flow from the SWAN wireless testbed without rate control.

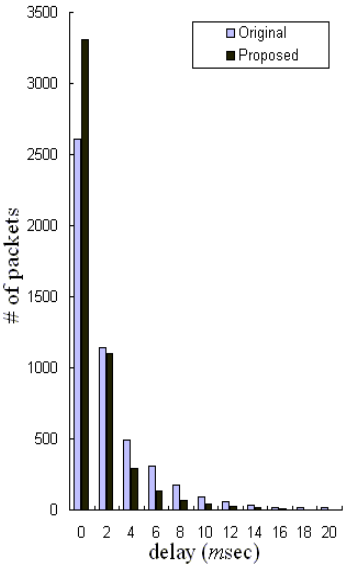

Fig. 23. The distribution of packet delay.

VII. CONCLUSION

In this paper, we proposed a simple, distributed and stateless network model called SWAN that uses distributed control algorithms to support soft real-time services and service differentiation in wireless ad hoc networks. An important benefit of SWAN is that it is independent of the underlying MAC layer, and can be potentially suited to a class of physical/data link wireless standards. We presented the performance evaluation of SWAN using the ns-2 simulator, and analyzed the MAC delay and busy probabilities. Simulation, analysis, and results from our experimental wireless testbed show that real-time applications experience low and stable delays under various multi-hop, traffic and mobility conditions. The SWAN testbed and ns-2 source code are available from the Web (comet.columbia.edu/swan).

ACKNOWLEDGEMENT

The SWAN Project is sponsored in part by the ARO Award DAAD 19-99-10287, and with support from Ericsson Research. The authors would like to thank Jaekwon Oh and Il-Jun Hwang for helping to build the SWAN wireless testbed, and Professor Mischa Schwartz for his great input.

REFERENCES

[1] S-B. Lee, G-S Ahn, X. Zhang, and A.T. Campbell, "INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks", *Journal of Parallel and Distributed Computing* (Academic Press), Special issue on Wireless and Mobile Computing and Communications, Vol. 60 No. 4 pp. 374-406, April 2000.

[2] A. Veres, A.T. Campbell, M. Barry and L-H. Sun, "Supporting Service Differentiation in Wireless Packet Networks Using Distributed Control", *IEEE Journal of Selected Areas in Communications*, Special Issue on Mobility and Resource Management in Next-Generation Wireless Systems, Vol. 19, No. 10, pp. 2094-2104, October 2001.

[3] J. L. Sobrinho and A.S. Krishnakumar, "Quality-of-Service in Ad hoc Carrier Sense Multiple Access Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, pp. 1353-1368, August 1999.

[4] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks", in *Proc. ACM ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom),* August 1999.

[5] K. Ramakrishnan, S. Floyd, and D. Black, "An Addition of Explicit Congestion Notification (ECN) to IP", *Internet Draft* <draft-ietf-tsvwg-ecn-03.txt>, work-in-progress, March 2001.

[6] S. Kalyanaraman, D. Harrison, S. Arora, K. Wanglee, and G. Guarriello, "A One-bit Feedback Enhanced Differentiated Services Architecture", *Internet Draft* <draft-shivkuma-ecn-diffserv-01.txt>, work-in-progress, March 1998.

[7] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function", *IEEE Journal on Selected Areas in Communications*, Vol. 18, March 2000.

[8] D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Computer Networks*, 1989.

[9] S. Nanda, D. J. Goodman, U. Timer, "Performance of PRMA: A Packet Voice Protocol for Cellular Systems", *IEEE Trans. Veh. Technology,* Vol. 40, August 1991.

[10] D. Bansal and H. Balakrishnan, "TCP-friendly Congestion Control for Real-time Streaming Applications", *Technical Report*, MIT-LCS-TR-806, MIT Laboratory for Computer Science, May 2000

[11] C. E. Perkins and E. M. Royer, "Ad-hoc On Demand Distance Vector Routing", *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, February 1999

[12] J. Broch, D. Maltz, D. Johnson, Y-C Hu, and J. Jetcheva, "A Performance Comparison of Multihop Wireless Ad Hoc Network Routing Protocols", *Proc. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, October 1998.