Impact of Queuing Discipline on Packet Delivery Latency in Ad Hoc Networks

Josiane Nzouonta, Teunis Ott, Cristian Borcea * [a]

[a]Department of Computer Science,
New Jersey Institute of Technology, Newark, NJ 07102, USA

Delivering live multimedia streaming over ad hoc networks can improve coordination on battlefields, assist in disaster recovery operations, and help prevent vehicular traffic accidents. However, ad hoc networks often experience congestion faster than wired networks, leading to high end-to-end delays and jitter even for moderate traffic. This paper describes a partial remedy that applies to delay sensitive but loss tolerant applications such as live streaming. We find that under relatively high UDP traffic load, the Last–In–First–Out (LIFO) with Frontdrop queuing discipline achieves less than half the delay of the commonly used First–In–First–Out (FIFO) with Taildrop, while maintaining similar jitter. In low traffic situations, FIFO and LIFO have similar delays, but FIFO with Frontdrop has the lowest jitter. The results can be applied to an adaptive queuing mechanism that changes the queuing discipline at nodes function of the locally observed traffic load. The advantage of such an approach is that it does not require new protocols and does not incur any network overhead.

Keywords: Ad hoc networks, Live streaming, Queuing disciplines, Delay and jitter

1. Introduction

The vision of mobile ad hoc networks helping soldiers communicate on battlefields or aiding rescuers explore regions affected by natural disasters has started to become reality. For instance, unmanned vehicles (aerial, terrestrial, and aquatic) with autonomic operation of a few hours can be sent to regions where human presence is deemed dangerous [1,2], and they can form networks on the fly to report observations to command and control centers. The decrease in price of WiFi-enabled smart phones and the appearance of DSRC (Dedicated Short Range Communication [3])-enabled vehicles are expected to accelerate the integration of ad hoc networks into everyday activities. Future vehicular networks, for example, will be able to monitor the traffic conditions on the roads ahead without requiring a network infrastructure.

In this study, we are interested in improving the performance of delay sensitive but loss-tolerant applications, such as live video streaming, in an ad hoc network. These applications can be used for video surveillance, entity tracking, survivor discovery in damaged buildings, or traffic monitoring near an accident on the road. Unfortunately, their performance degrades quickly in ad hoc networks even for moderate traffic. For many reasons, including varying connectivity and interference between links, ad hoc networks experience problematic effects of congestion, such as high end-to-end delay and jitter, faster than well-designed wired networks. Figure 1 illustrates this problem with two networks, one wired and one wireless (IEEE 802.11), having the same topologies and queuing discipline (FIFO with Taildrop). Under identical traffic load, the wired network results in an average end-to-end delay of 87ms while the wireless network experiences more than 2sec average delay. The difference between these results can be partly explained by the contention algorithm used in the wireless network, which prevents any two nodes from the set ($S1$,$S2$, $S3$, $S4$, $n1$, $n2$) from transmitting data simultaneously. Consequently, packets spend longer time in transit.

To better understand the combined effect of contention and queuing discipline on end-to-end delay, we ran simulations in a mobile ad hoc network with 250 wireless vehicles moving at normal speeds on city roads forming a grid layout (the simulation settings are similar to those described in Section 4.3). We observed that during high contention periods the IEEE

*Corresponding author: borcea@cs.njit.edu. The other authors emails: jn62@njit.edu and teun@teunisott.com. This material is based upon work supported by the National Science Foundation under Grants No. CNS-0520033, CNS-0834585, and CNS-0831753. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

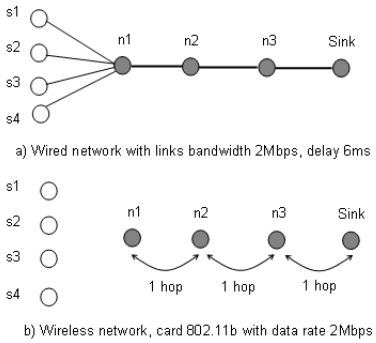


Figure 1. The shared nature of a wireless network creates contention and leads to congestion faster than in wired networks. Hosts ($S1$, $S2$, $S3$, $S4$) transmit UDP traffic to a sink host at a rate of 20 packet/sec for a duration of 100sec. The resulting average end-to-end delay is greater than 2sec in the wireless network topology (case b) while the wired network topology (case a) has an average end-to-end delay of 87ms.

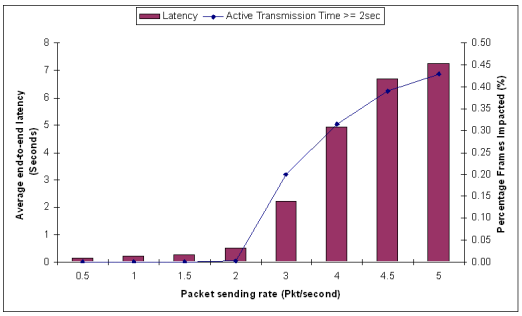


Figure 2. Average end-to-end delay as function of percentages of frames experiencing transmission delays greater than 2 seconds. The results were obtained using RBVT-R routing [4] and 15 source-destination pairs exchanging packets at rates varying from 0.2 to 5 packets/sec.

802.11 exponential back-off mechanism can make a frame, ready to be transmitted, wait more than 2 seconds at a node queue before completing the transmission. Although the number of frames that experience these unusually high transmission delays is relatively low, their effect propagates to all the frames in the node queue. Figure 2 shows that a very low percentage of "problem" frames could lead to more than one order of magnitude increase in the average end-to-end delay. This observation follows from the characteristics of the FIFO Taildrop queuing discipline. When the queue becomes full, newly arrived packets are dropped. On the other hand, the old packets that experienced long queuing delays are delivered, contributing to the substantial increase in the end-to-end delay.

The main questions drawn from these preliminary experiments are: Can ad hoc networks achieve better delay and jitter with a different queuing discipline? Is there a queuing discipline that performs better in all cases or different disciplines work better for different network conditions? The choice of queuing discipline has been already shown to impact the performance of data transfers in wired IP networks [5], where TCP was proved to perform better under congestion when routers use FIFO with Frontdrop instead of FIFO with Taildrop or RED [6].

This paper analyzes the relative performance of FIFO versus LIFO and of Taildrop (TD) versus Frontdrop (FD) in all four possible combinations, namely FIFO-TD, FIFO-FD, LIFO-TD and LIFO-FD. The analytical results for single node queue show that FIFO disciplines perform better in terms of expected waiting time and variance of the expected waiting time for non-congested situations (i.e., packet arrival rate is less than packet service rate), while LIFO-FD performs better for congested situations (i.e., packet arrival rate is greater than packet service rate). Based on these results, we ran simulations with FIFO-TD (the conventional queuing discipline) and LIFO-FD to understand their impact on end-to-end performance in large scale ad hoc networks.

The results of simulations in a static wireless ad hoc network show that LIFO-FD leads to less than half the UDP delay of FIFO-TD under high traffic load, while maintaining similar jitter. When the traffic load is low, the four disciplines present comparable end-to-end delays, but LIFO shows greater jitter than the other disciplines (FIFO-FD has the lowest jitter). For

TCP, LIFO has the disadvantage of leading to out of order delivery which may result in re-sequencing delays at the destination. We find that under moderate traffic, this is of minimal effect as LIFO-FD shows jitter and fairness comparable to FIFO-TD. Finally, we compared the two disciplines in a vehicular ad hoc network; the results showed that the end-to-end delay using LIFO-FD decreased as much as 45% for higher packet transmission rates, with a comparable packet delivery rate. The results of this study can be applied to an adaptive queuing mechanism that changes the queuing discipline at nodes function of the locally observed traffic load. The advantage of such an approach is that it does not require new protocols and does not incur any network overhead.

The rest of this paper is organized as follows. A brief overview of the service disciplines and drop policies studied as well as a discussion of related work are presented in Section 2. Section 3 contains the theoretical analysis and results for single node queuing. The simulation results are presented in Section 4. The paper concludes in Section 5.

2. Background

In this section, we first review current solutions aimed at improving the performance of delay sensitive applications in ad hoc networks. Then, we briefly go over the commonly perceived advantages and disadvantages of the queuing disciplines analyzed in this paper.

2.1. Related Work

Long queuing delays and packet losses are typical signs of congestion in wired networks. These problems are exacerbated in ad hoc networks by the shared medium contention, higher bit-error rates, changing channel quality during data transfers, and potential mobility. To improve the quality of delay sensitive applications, several solutions that adapt techniques dealing with congestion in wired networks to the conditions encountered in wireless networks have been proposed.

Rate-based approaches [7–9] reduce the sending rate when congestion is detected. In [7], the authors adapt TFRC (TCP Friendly Rate Control [10]) to wireless networks. With TFRC, the destination host continuously sends estimates of the loss rate to the source host, which adjusts its sending rate to these estimates. The solutions proposed in [9] point out that hop-by-hop techniques are able to react to congestion faster, providing improved performance compared to end-to-end schemes such as [7] and [8]. Indeed, in ad hoc networks, multi-hop communications often use routing protocols such as [11,12] which perform symmetric routing (the same path is used for communications to and from the source). Thus, the notification packets sent by the destination are likely to experience some delay as well. Furthermore, notification packets increase the amount of data in the network, thus adding to the channel contention [13].

A cross-layer approach presented in [14] proposes adaptations to all the layers of the TCP/IP protocol stack. At the link layer for example, the authors propose adapting the packets' lengths to the current SINR (signal to interference plus noise ratio) to decrease the packet error rate. The optimizations are carried out by each node individually, adapting to wireless link conditions and traffic flows. Cross-layer solutions are attractive in that they provide a holistic approach to the problem and can be applied locally at each node. The disadvantage of this is the corresponding complexity: a complete overhaul of the protocols on all the layers is needed.

A priority-content approach [15], focused on video transmissions, proposes to use a congestion-distortion scheduler to prioritize certain packets based on their content. For example, the I and SI frames of an MPEG-4 stream are given priority over B frames because the loss of I or SI frames impacts the decoder significantly compared to the loss of a B frame. Prioritizing certain packets can be accomplished at the original source of the transmission. However, congestion in a multi-hop ad hoc network may appear at any intermediate node which acts as a router for the packet and the quality of the transmission would still suffer. The solution proposed in [16] describes a hybrid Automatic Repeat Request (ARQ) algorithm which combines ARQ and Forward Error Correction (FEC) to improve unicast communications, while the one in [17] presents a connection-oriented unreliable transport protocol which operates like UDP with a congestion control scheme.

Different from all these solutions, our work demonstrates that end-to-end delay and jitter can be improved significantly by simply allowing the nodes to

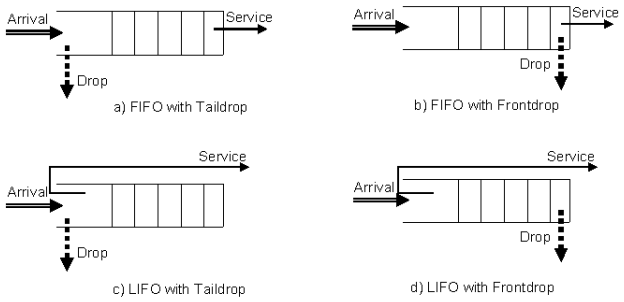


Figure 3. FIFO and LIFO with Taildrop and Frontdrop in all four possible combinations.

dynamically switch between different queuing disciplines. The decision is made locally as function of the observed traffic load. This solution does not generate network overhead, does not require changes to higher layer protocols, and can work in conjunction with optimizations at other layers.

2.2. Queuing Disciplines Overview

In the following, we present the perceived advantages and disadvantages of the queuing disciplines, specifically the service disciplines and the drop disciplines, considered in this paper as they pertain to TCP and UDP traffic. This preliminary discussion will help with the understanding of our experiments, results, and conclusions. Figure 3 illustrates how each discipline works.

2.2.1. Service disciplines

The FIFO service discipline selects the packet at the head of the queue as the next packet to service. FIFO allows packets forwarded along the same path to be delivered at the destination in the same order as they were produced at the source. Protocols, such as TCP, benefit from the in-order delivery of packets because the destination can remove data from buffers as soon as in-order data is received. FIFO also provides a sense of fairness because the packet which is serviced is the one which has been waiting the longer.

However, FIFO leads to long waiting times under congestion. Each newly received packet is inserted at the end of queue and will be serviced after all the packets already waiting in the queue at its arrival. When this happens over multiple queues and the wait time in a specific queue is not negligible, the total time spent by a packet in transit may increase considerably. In applications such as live streaming, this may result in packets arriving too late to be played, thus unnecessarily using network resources.

The LIFO service discipline, on the other hand, selects the packet at the tail of the buffer as the next packet to service. The advantage is that the newest packets will be served first. UDP performance for live streaming could be improved if the packets that stayed for too long in the queues, which will probably be dropped at destination anyway, do not affect the delivery of newer packets. More details about this idea will be presented in Section 4.

A number of issues might arise with LIFO. First, LIFO provides a sense of unfairness because newer packets are prioritized over packets which have been waiting longer. Additionally, while packet-switched networks do not expect in-order delivery of packets, it is expected that LIFO will result in more out-of-order packets than FIFO.

2.2.2. Drop disciplines

When the queues become full, the node/router applies the drop discipline (Taildrop or Frontdrop) to each newly received packet. Taildrop drops the newly received packet. The advantage is not only that the waiting time to drop a packet is null, but also this policy provides a sense of fairness by keeping the packets already admitted in the system. On the other hand, one might argue that dropping newly received pack-

ets, as Taildrop does, is a disadvantage because some packets may never get a chance to enter the queue and be serviced.

With Frontdrop, the oldest packet in the queue (the one at the head) is dropped, and the newly received packet is added to the tail of the queue. The advantage of this policy is that aged packets are dropped. However, it provides a sense of unfairness because packets which are close to being serviced are dropped from the queue.

3. Theoretical Analysis

In this section, we derive analytical expressions for the expected waiting time and the variance of waiting time for packets in a single queue under FIFO-TD, FIFO-FD, LIFO-TD, and LIFO-FD. The model used in this analysis is an exponential queue (M/M/1/N). While this model is not accurate for quantitative predictions of delays in a real-life wireless node (Section 3.2 discusses this issue), it is useful to provide qualitative insights into the differences between the four queuing disciplines. Specifically, it helps us understand which ones are the most promising and under what conditions. Considering that the analysis focuses on the performance of a single node queue, external parameters (e.g., wireless signal fluctuations, node mobility, etc.) are not explicitly studied in the model. Rather, their impact will be assessed through simulations.

3.1. System Model

Under the exponential model, if there is space for N waiting packets in the queue, the system has the states $\{-1, 0, \cdots, N, N+1\}$. In state (-1), the system is empty: no packet waiting or being served. In state (0), there is one packet being served and there are no waiting packets. In state (k) with $1 \leq k \leq N$, there is one packet being served and there are k waiting packets. State $(N+1)$ is not a real state but is used to facilitate the description: when a packet arrival occurs while the system is in state (N), the system first goes to the hypothetical state $(N+1)$ and then instantaneously drops a packet and goes back to state (N).

This model leads to a Markov Chain $X(t)$ on the state space $\{-1, 0, \cdots, N\}$, where $X(t)$ is the state at time t. The stationary distribution of this Markov chain does not depend on FIFO–LIFO or Taildrop–Frontdrop. The traffic intensity is defined as $\rho = \frac{\lambda}{\mu}$, where λ is the arrival rate and μ is the service rate. In other words, if the system (the sender) were permanently busy, then in a long period of length Υ it would serve about $\Upsilon\mu$ packets (assuming there are no re–transmissions or retries and no timeout for blocking). For all queuing disciplines, the stationary probabilities $\pi_k = P\{X(t) = k\}$ are: $\pi_{N+1} = 0$ and for $-1 \leq k \leq N$.

$$\pi_k = \rho^{k+1}\pi_{-1}, \;\; \pi_k = \rho^{k+1}.\frac{1-\rho}{1-\rho^{N+2}}. \tag{3.1}$$

It must be noted that we do *not* require that $\rho < 1$. The finite state space makes the process ergodic anyhow, and in fact, the problem is of most interest when ρ is close to 1 or even larger. If $\rho = 1$, then according to (3.1): $\pi_k = \frac{1}{N+2}$ for $-1 \leq k \leq N$. In addition to (3.1), we have that a fraction $1 - \pi_N$ of all arriving packets will eventually be served and a fraction π_N will not be served (i.e., they will be dropped). For all disciplines considered, if a packet arrives in state (k), $k < N$, then the state changes to $(k+1)$: all packets already in the system keep their position, and the new packet goes into position $k+1$. If there is a service completion in state (k) (this implies $k \geq 0$), then the state changes to $(k-1)$. Under FIFO, the packet in position j $(0 \leq j \leq k)$ moves to position $j-1$. Under LIFO, the packet in position k moves to position 0, and all the other packets stay in the position they were in.

If a packet arrives while the system is in state (N), the system remains in this state; as explained, the system moves to state $(N+1)$ for an infinitely small amount of time, then moves back to state (N). Under Taildrop, the arriving packet is dropped and all the other packets remain in their old positions. Under Frontdrop, the packet in position 1 is dropped, all the other packets already in the system move up one position $(j \rightarrow j-1)$, and the arriving packet goes to position N.

Let T denote the time from arrival of a packet until either being dropped or having service start. We define "eventually being dropped" as "failure" and "eventually being served" as "success". We define the probabilities of success and failure as follows:

$$\eta_S(s) = E[e^{-sT}\chi(Success)],$$
$$\eta_F(s) = E[e^{-sT}\chi(Failure)], \tag{3.2}$$

where $\chi(.)$ is the indicator function.

We further define the probabilities of success and failure given that the packet arrives when the system is in state k as follows:

$$\eta_{k,S}(s) = E[e^{-sT}\chi(Success)| \text{ packet arrives while system in state } (k)],$$
$$\eta_{k,F}(s) = E[e^{-sT}\chi(Failure)| \text{ packet arrives while system in state } (k)], \quad (3.3)$$

such that

$$\eta_S(s) = \sum_{k=-1}^{N} \pi_k \eta_{k,S}(s),$$
$$\eta_F(s) = \sum_{k=-1}^{N} \pi_k \eta_{k,F}(s), \quad (3.4)$$

where for all disciplines considered

$$\eta_{-1,S}(s) \equiv 1, \;\; \eta_{-1,F}(s) \equiv 0. \quad (3.5)$$

Section 3.3 shows how the expressions for $\eta_{k,S}(.)$ and $\eta_{k,F}(.)$ are derived for all the four disciplines. While standing for the same concepts, the expressions are different for the four disciplines considered.

3.2. On the Exponential Model Used

An important question is whether the exponential model is appropriate for packet queues. Before we discuss this question further, we point out that the modeling stage does not lead to a choice of parameters for the actual queuing disciplines to be used, but to broad categories of queuing disciplines that are the most promising. These broad categories will be further investigated through simulations.

We use the $M/M/1/N$ model to get a qualitative insight in the trade-offs between the four disciplines used: FIFO-TD, FIFO-FD, LIFO-TD, and LIFO-FD. It must be understood that the $M/M/1/N$ model is not suitable to quantitatively predict the behavior of a specific discipline in a node. Rather, it is strictly used to get a qualitative insight into the circumstances under which individual disciplines perform better. This insight is quantitatively verified through simulations.

The $M/M/1/N$ model is unsuitable for quantitative prediction of the behavior of nodes under the 4 disciplines for various reasons, including:

- The assumption of Poisson arrivals: In a superposition of UDP streams, we expect a somewhat deterministic behavior over short periods, with randomly varying hold rates if the number and types of flows change. Additionally, TCP feedback from mechanisms such as RED would cause wide deviations from any Poisson character of the arrival process.
- The assumption of exponential service times: It is to be expected that a large fraction of packets are either exactly one MSS (for their specific flow), or, in case of TCP, are acknowledgments.
- The assumption of independence: While not as untenable as the first two assumptions, it is still questionable. For example, if old flows terminate and new flows start up, we may see a different mix of packet sizes.

It is conceivable that a $M/G/1/N$ model might be a better model. However, there are two problems: **(1)** which service time to use and **(2)** it is far from clear that getting a good grip on the service times is more important than getting a good grip on the arrival process. In addition, in a node, the same packet may be retransmitted several times and loss probabilities may depend on the packet size.

Therefore, we have opted to use the simple $M/M/1/N$ model. This means of course that it should be used only to pinpoint the likely better candidates as function of traffic intensity and buffer size, but not to quantitatively predict performance of specific disciplines for specific traffic intensities and buffer sizes. The simulations in fact bear out that the model does a good job in predicting which discipline and under what circumstances is better.

3.3. Analytical Results

3.3.1. FIFO with Taildrop

In the stationary situation, an arriving packet finds, with probability π_k as in (3.1), that the system is in state (k). If $k = -1$ the arriving packet does not need to wait: waiting time is zero. Under FIFO with Taildrop, if the arriving packet finds the system in state (N), the arriving packet is dropped (and encounters no waiting). If the arriving packet finds the system in state (k), with $0 \leq k \leq N-1$, the wait is the sum of $(k+1)$ independent random variables, each

exponentially with parameter μ. We therefore have

$$\eta_S(s) = \sum_{k=-1}^{N-1} \pi_k \left(\frac{\mu}{\mu+s}\right)^{k+1}. \quad (3.6)$$

$$\eta_F(s) = \pi_N. \quad (3.7)$$

3.3.2. FIFO with Frontdrop

We consider pairs (k, L) with $0 \leq k \leq L \leq N$, where $(L) = X(0^+)$ is the state of the system (number of waiting packets) and k is the location of a "marked" packet, both at time 0. "Success" means that the marked packet is eventually served, and "Failure" means that the marked packet is eventually dropped. Dropped means the packet makes it to location 1 and then (so close to success) is dropped. Let $T_{k,L}$ be the time until the marked packet either starts being serviced or is dropped. We define

$$\zeta_{k,L,S}(s) = E[e^{-sT_{k,L}}\chi(Success)],$$
$$\zeta_{k,L,F}(s) = E[e^{-sT_{k,L}}\chi(Failure)]. \quad (3.8)$$

Thus, $\zeta_{k,L,S}(0)$ is the probability the marked packet (eventually) gets served, and $\zeta_{k,L,F}(0)$ is the probability that the marked packet eventually gets dropped. Also, for $0 \leq k \leq N-1$

$$\eta_{k,S}(s) = \zeta_{k+1,k+1,S}(s),$$
$$\eta_{k,F}(s) = \zeta_{k+1,k+1,F}(s), \quad (3.9)$$

and

$$\eta_{-1,S}(s) = 1, \quad \eta_{-1,F}(s) = 0,$$
$$\eta_{N,S} = \zeta_{N,N,S}(s), \quad \eta_{N,F} = \zeta_{N,N,F}(s). \quad (3.10)$$

The expression of $\zeta_{k,k,S}(s)$ is computed in Appendix A.2.

3.3.3. LIFO with Taildrop

As before, if the arriving packet finds the system in state (-1), the packet is served and there is no waiting. If the packet finds the system in state (N), the packet is dropped and there is no waiting.

If the arriving packet finds the system in state (k), with $0 \leq k \leq N-1$, the packet takes position $k+1$ and will certainly be served; in fact, it will be served before all already waiting packets. Therefore, it is as if we have a system with a buffer for $N-k$ waiting packets, the just arriving packet has position 1 in that new system which starts in state (1), and the waiting time of that packet is the first passage time to state (0) in that new system.

We therefore have

$$\eta_S(s) = \pi_{-1} + \sum_{k=0}^{N-1} \pi_k \cdot \left(\sigma_{1,N-k}(s) + \frac{\phi_{1,N-k}(s)\sigma_{N-k,N-k}(s)}{1-\phi_{N-k,N-k}(s)}\right) \quad (3.11)$$

As previously with Taildrop:

$$\eta_F(s) = \pi_N. \quad (3.12)$$

The entities $\sigma_{k,N}(.)$, $\phi_{k,N}(.)$, $\sigma_{k,N} = \sigma_{k,N}(0)$, $\phi_{k,N} = \phi_{k,N}(0)$ are derived in Appendix A.1.

3.3.4. LIFO with Frontdrop

This case is somewhat similar to the previous case. The difference is that every time a drop occurs, the packet in location 1 is dropped and all the other packets go to one lower location. Also, while a packet which arrives in state (k), with $0 \leq k \leq N-1$, goes into location $k+1$, a packet which arrives while the system is in state (N) goes into position N. A packet in position $k+1$, while the state is $(k+1)$, either is served before the next drop or moves to position k, state N, at the next drop. Hence, for $0 \leq k < N$:

$$\eta_{k,S}(s) = \sigma_{1,N-k}(s) + \phi_{1,N-k}(s) \cdot \left(\sum_{j=1}^{k}\left(\prod_{\nu=1}^{j-1}\phi_{N-k+\nu,N-k+\nu}(s)\right)\cdot .\sigma_{N-k+j,N-k+j}(s)\right) \quad (3.13)$$

And

$$\eta_{N,S}(s) = \eta_{N-1,S}(s). \quad (3.14)$$

In (3.13), the sum is for j from 1 to k, because the $(k+1)$-th packet to drop before the packet starting in position $k+1$ is served would be that packet itself.

3.4. Numerical Results

Using the expressions derived in Section 3.3, we assess the expected waiting time as well as the variance of waiting times of packets eventually served for different traffic intensities using all four queuing disciplines. The plots are obtained using Wolfram Mathematica 6 [18] and the results are presented in Figures 4 to 6. For reasons previously explained, these

results present a comparative (rather than a quantitative) assessment of the performance of the disciplines under similar conditions.

3.4.1. Queue with low traffic: $\rho < 1$

Two useful observations can be drawn from Figure 4. First, the values of the expected waiting time and variance of waiting times stabilize to constant values as the queue size increases, independently of the discipline. Because the service rate is greater than the arrival rate, there is (practically) no drop of packets, thus no difference between Taildrop and Frontdrop performance.

Second, the variance of waiting times of packets eventually served are higher under LIFO than FIFO. With LIFO, certain packets will stay longer in the queue because of new packet arrivals, yet they are eventually served because the traffic intensity makes it unlikely that the queue becomes full ($\rho < 1$).

3.4.2. Queue with even traffic: $\rho = 1$

Figure 5 shows that when $\rho = 1$, unlike the case of $\rho < 1$, the expected waiting time of successful packets increase linearly with the buffer size for all four disciplines, albeit with different slopes. In the context of FIFO, the linear increase results from packets gradually moving to the head of the queue to be serviced: the longer the queue, the longer the time needed to reach the head position.

We also observe that in the case of Taildrop, the average waiting time does not depend on FIFO–LIFO. To understand why, let's consider $\overline{W}$ as the average waiting time of all packets, and $\overline{W_s}$, $\overline{W_d}$ as the average waiting time until service and drop, respectively. Then, regardless of the discipline, we have $\overline{W} = (1-\pi_N)\overline{W_s}+\pi_N\overline{W_d}$. In case of Taildrop, packets are dropped upon arrival and encounter no waiting at all. Thus $\overline{W_d} = 0$ and $\overline{W_s} = \frac{\overline{W}}{1-\pi_N}$ regardless of whether we use FIFO or LIFO and independently of the traffic intensity.

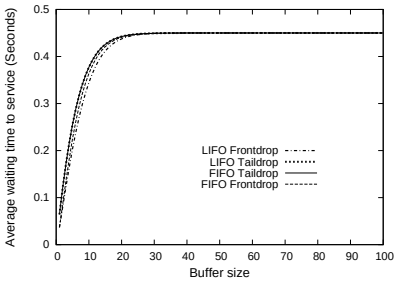
The issue is different for the variances, with LIFO leading to very high variances of waiting times compared to FIFO (Figure 5(b)). This high variance can be explained by the fact that LIFO has a higher probability of almost no waiting, but also a higher probability of a long wait. Because the packets arrival at the queue equals the rate of departure from the queue, little difference is observed between FIFO-TD and FIFO-FD. The asymptotes of the variance with FIFO are similar for both Taildrop and Frontdrop.

3.4.3. Queue with high traffic: $\rho > 1$

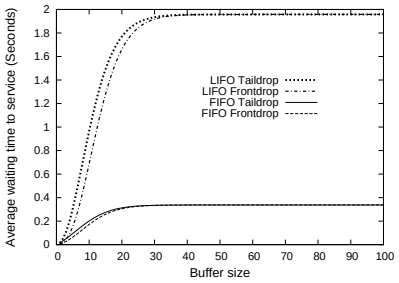
Figure 6 shows the differences in performance of the four disciplines when $\rho > 1$. The expected waiting time of packets eventually served is proportional to the size of the queue under FIFO-TD, FIFO-FD, and LIFO-TD. Figure 6(a) confirms the previous analysis showing identical expected waiting times results for FIFO-TD and LIFO-TD. Comparatively, FIFO-FD has lower values of expected waiting times than FIFO-TD. More specifically, for $\rho > 1$ and large buffer sizes, the expected waiting time with FIFO-TD is roughly ρ times the expected waiting time of FIFO-FD. Indeed, with $\rho > 1$ and a buffer size N, the waiting time under FIFO-TD is about $\frac{N}{\mu}$ time units, while the waiting time under FIFO-FD is about $\frac{N}{\lambda}$ time units. In both cases this is the waiting time from entering into a full system until reaching the head of the queue. More explicitly, for $\rho > 1$ (i.e. $\lambda > \mu$) and N large, in FIFO-FD, "all" packets arrive while there are "almost" N packets in the queue. It then takes an amount of time roughly $\frac{N}{\lambda}$ until the packet reaches position 1, after which it is dropped with probability roughly $\frac{\lambda-\mu}{\lambda}$ and served with probability roughly $\frac{\mu}{\lambda}$. In this case, for N large, $\overline{W_d} \sim \overline{W_s} \sim \overline{W} \sim \frac{N}{\lambda}$.

In contrast to the other disciplines, under LIFO-FD, the expected waiting time values stabilize as the queue size increases. Indeed, for $\rho > 1$ and N large, an incoming packet in LIFO-FD either gets served quite quickly, or gets buried in the queue and eventually gets dropped. In this situation the expected waiting time of packets eventually served goes to a constant if N becomes large. Similar asymptotic results for large N hold for much more general queuing systems.

Figure 6(b) shows the variances of waiting times, focusing on FIFO and LIFO-FD. The variance of expected waiting times have linearly increasing asymptotes under FIFO-FD and FIFO-TD while the asymptote is constant for LIFO-FD. With Frontdrop and $\rho > 1$ and a large buffer, a packet will be discarded without being served with probability $\frac{\mu}{\lambda+\mu}$. The discarded packets are also the oldest in the queue, which explains the smaller increase of variance of waiting times for FIFO-FD and LIFO-FD.

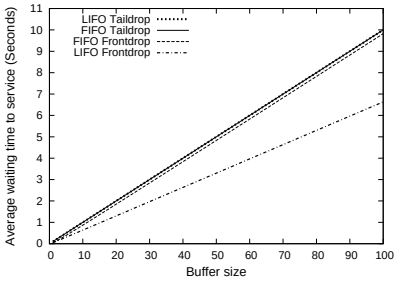


(a) Expected waiting time

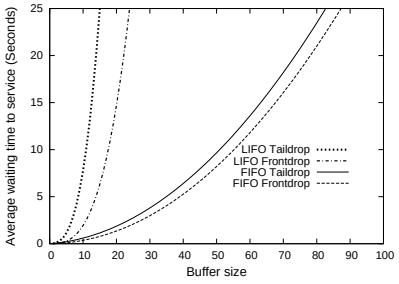


(b) Variance of waiting times

Figure 4. Expected waiting times and variance of waiting times for packets eventually served with a traffic intensity $\rho = 0.75$

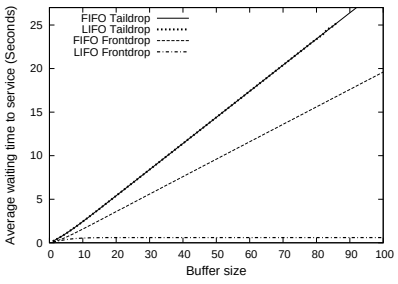


(a) Expected waiting time

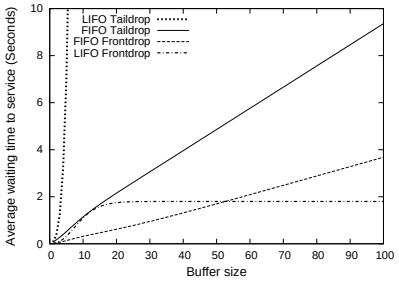


(b) Variance of waiting times

Figure 5. Expected waiting times and variance of waiting times for packets eventually served with a traffic intensity $\rho = 1.0$



(a) Expected waiting time



(b) Variance of waiting times

Figure 6. Expected waiting times and variance of waiting times for packets eventually served with a traffic intensity $\rho = 1.5$

3.4.4. Summary of Results

These results suggest that the Frontdrop policy enables FIFO and LIFO to have smaller expected waiting times as well as smaller variance of waiting times under congestion when compared to Taildrop. In particular, the ability of LIFO-FD to stabilize both the expected waiting times and the variance of waiting times of packets served indicates that it may be a suitable queuing discipline under congestion.

The results also suggest that FIFO requires the buffer size to be carefully chosen to make it work under light load ($\rho < 1$) as well as heavy load ($\rho > 1$). The buffer size should not be too small (to prevent loss in light traffic) or too large (to prevent huge delays in heavy traffic). LIFO is much less sensitive. In particular, LIFO-FD works as long as the buffer is large enough.

LIFO-TD would need a separate mechanism to discard old packets (packets no longer of interest to anybody) to still work when $\rho > 1$ even with large buffers. LIFO-FD has this mechanism built-in implicitly.

4. Network Simulations

This section presents the evaluation through simulations of data transfers in ad hoc networks under the four queuing disciplines. The goals of these simulations are to (1) validate the results for single queue obtained in the previous section, (2) understand whether the theoretical conclusions for single queue (i.e., LIFO-FD works best for high traffic situations, and FIFO works best for low traffic situations) can extend to multi-hop static ad hoc networks, (3) quantify the benefits of different queuing disciplines in terms of end-to-end UDP delay and jitter (and implicitly quantify the benefits for multimedia streaming in ad hoc networks), (4) identify any potential negative effects of queuing disciplines on fairness and/or TCP, and (5) understand the impact of node mobility on the end-to-end performances (i.e. whether the benefits observed in static ad hoc networks apply to mobile ad hoc networks as well).

The simulations are conducted using the Network Simulator NS-2.30 [19]. Table 1 summarizes the simulations settings. The wireless nodes are configured to transmit data at a rate of 2Mbps. We conduct simulations for both UDP and TCP. In the UDP simulations, the data sources inject UDP traffic at a Constant Bit Rate (CBR) with variable sending rates. In the TCP simulations, the sources use FTP and TCP-Reno to transfer files of 2MB-5MB size.

Table 1
Simulation Setup

Parameter	**Value**
Simulation area	1200m x 600m
Number of nodes	16, 250
Number of TCP/UDP sources	1-10
Transmission range	250m
Simulation time	100s-1000s
CBR rate	1 - 20 packet/second
MAC protocol	IEEE 802.11 DCF
Data packet size	1000 bytes

4.1. Metrics

The network performance under all four queuing disciplines is evaluated by varying the queue size, the CBR data rates, and the numbers of concurrent flows. The metrics used to assess the performance are the following:

- **Average end-to-end delay.** This metric is defined as the average delay incurred in the transmission of all data packets delivered successfully. This does not include the delay due to resequencing at the destination (i.e. waiting for packet with lower sequence number or lower timestamp). The average end-to-end delay is directly influenced by the queuing approach.
- **End-to-end jitter.** This metric is defined as the variation in the end-to-end delay values of successive data packets successfully received at destination. Lost packets are ignored in this evaluation (because their arrival times are undefined). The lower the jitter, the better the performance of multimedia streaming applications. The jitter also quantifies the effect of the out-of-order delivery of LIFO discipline compared to FIFO to answer questions such as: will LIFO impact negatively the TCP performance?
- **Throughput.** In this paper, we define this metric as the number of data packets successfully

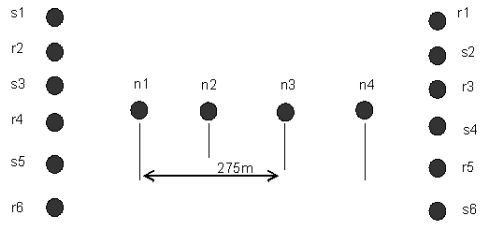


Figure 7. Topology used in the static ad hoc network simulation study

delivered at destinations per time unit. The throughput shows the effect of the queuing discipline on the useful data transferred end-to-end.

4.2. Static Network Simulations

4.2.1. Topology

The static ad hoc network topology is shown in Figure 7. The topology contains 16 wireless nodes and data is transferred between pairs of nodes (s_i,r_i). We selected this topology to ensure in a simple and controlled way that the set of nodes n_i become a bottleneck during the transmissions due to both shared medium contention and congestion. Besides increasing the data sending rates, the addition of transmitting pairs (s_i,r_i) increases the data traffic forwarded through nodes n_i, making them bottlenecks. The distance between the n_i ensures that each packet forwarded from source to destination will hop through all four n_i nodes (i.e., the order n_1, n_2, n_3 and n_4, or vice-versa, is always maintained). This topology allows us to examine the queues of nodes n_i to understand single queue behavior, while the end-to-end measurements will give the performance of the network.

In all the simulations, the value of the queue (buffer) size is set to the same value for all nodes. At the network level, we use greedy geographical routing to forward data between communicating pairs. Each node periodically broadcasts "hello" packets to maintain the neighborhood tables required by geographical routing.

4.2.2. Single buffer

We first wanted to confirm the results of the analytical model described in Section 3. For this purpose, we consider the individual n_i queues and observe the time spent by packets at each queue. As expected, the nodes n_1 and n_4 contribute for most of the end-to-end delay. This is because the source flows use these nodes as the entry points toward the destination nodes, leading to high traffic contention around n_1 and n_4.

Figure 8 presents the average buffer waiting times at host n_1 for the four disciplines, under different traffic intensities. The data sending rate is used as an indicator of the traffic intensity, with 5pkt/s corresponding to light traffic (i.e. $\rho < 1$) and 20pkt/s corresponding to heavy traffic (i.e. $\rho > 1$). Identifying a data rate corresponding to $\rho = 1$ is less trivial in wireless network and we use the data sending rate of 10pkt/s as a middle traffic load between light and heavy traffic.

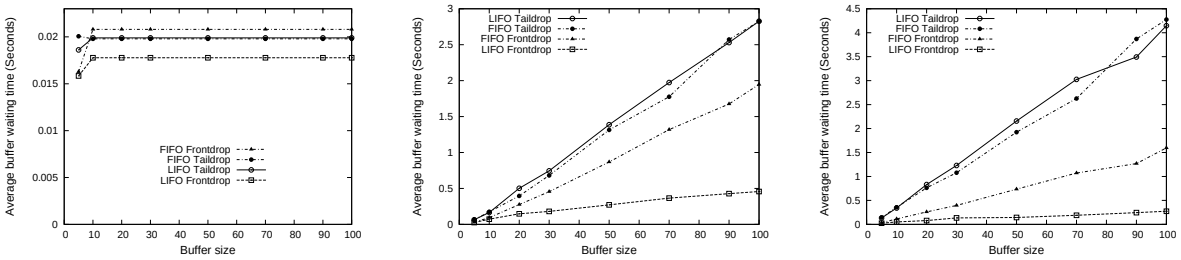
For light data traffic (figure 8(a)), the performance of the four disciplines are comparable. Under heavy load (figure 8(c)), however, LIFO-TD performs best as it enables packets to spend less than 300ms waiting time at the queue, independent of the queue size. These results are comparable to those of the analytical model (Figure 6), and the explanation presented in Section 3 remains the same.

4.2.3. End-to-end delay

Figure 9 presents the measured average end-to-end delay, obtained with 6 simultaneous UDP CBR flows, as function of the queue (buffer) size. We observe that the drop policy impacts the end-to-end delay sooner than the service discipline. For queue sizes of 20 or less, the end-to-end delay values are about identical for LIFO and FIFO. The reason is that the time required by a packet to reach the head of the queue is low for relatively small size queues. Thus, the impact of servicing the next packet from the head or the tail of the queue is reduced in small size queues.

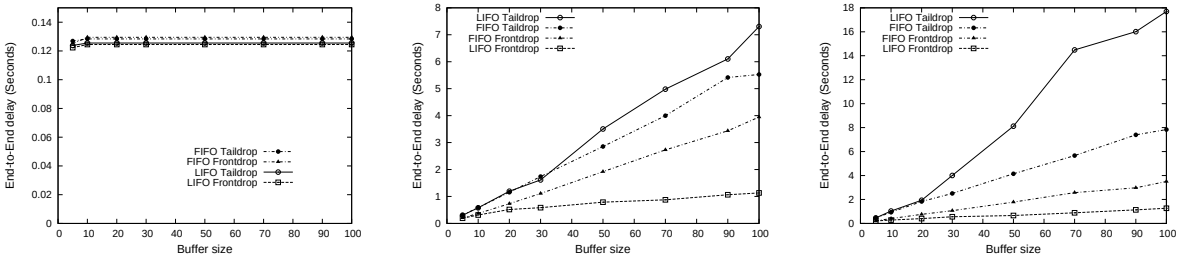
Figure 9 also shows that the end-to-end delay values are comparable under light data traffic (5pkt/s). Under high data traffic (20pkt/s), LIFO-FD provides the lowest measured delays. This result confirms that the measured end-to-end performance results match with the results of a single queue.

We were also interested in understanding the impact of UDP CBR sending rate for a single queue



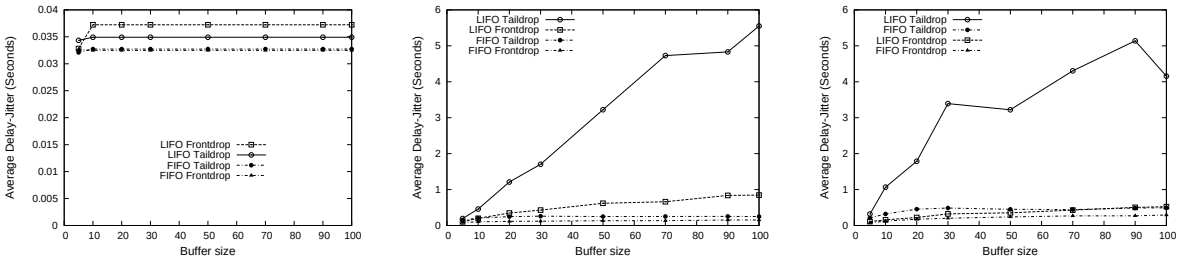
(a) UDP CBR sending rate = 5pkt/s (b) UDP CBR sending rate = 10pkt/s (c) UDP CBR sending rate = 20pkt/s

Figure 8. Average buffer waiting time vs. buffer size at node n_1, UDP CBR data traffic with different sending rates



(a) UDP CBR sending rate = 5pkt/s (b) UDP CBR sending rate = 10pkt/s (c) UDP CBR sending rate = 20pkt/s

Figure 9. Average end-to-end delay vs. buffer size, UDP CBR data traffic with different sending rates



(a) UDP CBR sending rate = 5pkt/s (b) UDP CBR sending rate = 10pkt/s (c) UDP CBR sending rate = 20pkt/s

Figure 10. End-to-end jitter with UDP CBR traffic, UDP CBR data traffic with different sending rates

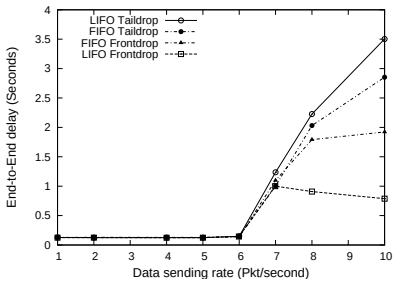


Figure 11. Average end-to-end delay vs. UDP CBR sending rate, buffer size = 50 slots

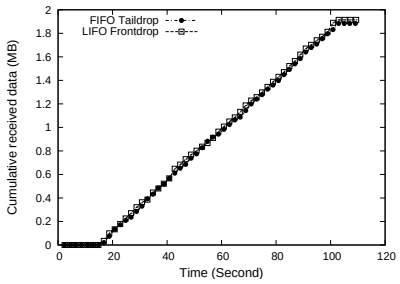


Figure 12. Total instantaneous UDP data received, UDP CBR sending rate = 10pkt/s, buffer size = 50 slots

size (figure 11). Before analyzing this figure, it is important to address the selection of the queue size (50 slots) used for this set of simulations. From the analytical results (Section 3.3), we know that the FIFO disciplines are particularly impacted by the queue size. How to determine the correct queue size is a current research topic [20,21] both for simulation and industrial purposes. We used 50 slots because it is the most common in simulations (default value in NS2 [19]) and it is the value recommended for routers used in some real-life networks [22].

Figure 11 shows that the Frontdrop policy leads to shorter average end-to-end delay for both FIFO and LIFO. For small values of the sending rates (up to 6pkt/s for this topology), all the queuing disciplines have similar results. However, for rates of 7pkt/s and up, the queues become full and the results differ considerably. LIFO-FD has the shortest average end-to-end delay followed by the FIFO-FD. These results are similar to Figure 4(a) (for up to 6pkt/s) and Figure 6(a) for 7pkt/s and up. The reasons explained for those figures apply here as well.

4.2.4. End-to-end jitter

So far, we have seen that LIFO-FD performs best in terms of end-to-end delay under high traffic load, while the four disciplines showed comparable results under light traffic load. We now examine the end-to-end jitter for the same transmissions. Considering that LIFO is expected to have more out-of-order packets at destinations, the question is whether or not its end-to-end jitter has acceptable values that do not counterbalance the gains in terms of end-to-end average delay.

To answer this question, we measure the end-to-end jitter between successive data packets at the destination nodes. In the event of a missing packet (e.g., dropped), we ignore it and consider the next successive packets to reach the destination. The end-to-end jitter of packet $i+1$ is obtained as $jitter(i+1) = |(recv(i+1) - send(i+1)) - (recv(i) - send(i))|$, where $send(i)$ is the time packet i is sent at the source of the flow and $recv(i)$ is the time packet i is received at its intended destination.

Figure 10 shows that FIFO-FD performs best in terms of end-to-end jitter, across all the CBR traffic rates. FIFO-FD takes advantage of the Frontdrop policy and additionally serves the packets in order, causing fewer end-to-end out of order deliveries. LIFO-FD and FIFO-TD present similar average end-to-end jitter for higher data traffic rates. This result in conjunction with the one that showed significantly lower end-to-end delay for LIFO-FD demonstrate that LIFO-FD should be the selected discipline when networks experience high traffic periods.

4.2.5. UDP throughput

The preceding experiments showed that LIFO-FD provided the smallest end-to-end delay and similar jitter compared to FIFO-TD. These improvements do not adversely affect UDP data throughput as shown in Figure 12. The amounts of data transferred are similar when using FIFO-FD or LIFO-FD. This is expected

because in both disciplines a direct consequence of a host overload is the continuous presence of a packet to serve at that queue. Consequently, while LIFO-FD does provide significant reduction in the end-to-end delay, it does not adversely affect the amount of transferred data with UDP.

4.2.6. TCP throughput and fairness

While LIFO-FD proved to work best for UDP transfers overall, the following experiments will compare the TCP throughput under LIFO-FD and FIFO-TD (i.e., the commonly used queuing discipline). Specifically, we want to see if LIFO-FD leads to too many duplicate acknowledgments and consequently to window collapse due to out-of-order deliveries. The maximum TCP window size in NS2 corresponds to 20000 bytes in these simulations, and the buffer size at a host is set to 50 slots.

Figure 13 shows how the total instantaneous TCP throughput (over all flows) varies over time for the two queuing disciplines. The instantaneous throughput is the measured throughput for every 2 seconds interval. Although the instantaneous throughput differs between the two disciplines on specific intervals, the results show that the transfers complete almost at the same time. Obviously, these results show that the potential problem of TCP window collapse does not happen in our simulations for LIFO-FD. In none of the experiments did network congestion happen (i.e., dropped packets due to full queues) because the TCP congestion control algorithm kicked in earlier due to long delays caused by contention or lost packets caused by channel errors. This also explains the relatively low throughput of about 220Kbps for both disciplines. Such throughput results are not surprising in multi-hop ad hoc networks, as documented in [23].

The question we address next is whether or not LIFO-FD provides the same level of fairness for TCP flows as FIFO-TD. We quantify the level of fairness by observing whether some flows are starved during transmissions. The literature [24] has proposed ways to improve the fairness of 802.11 DCF networks, mainly assuming the FIFO-TD policy. Our goal here is not to propose ways to improve the fairness of 802.11 ad hoc networks, but to assess whether under similar circumstances, a queuing discipline results in flow starvation.

Figure 15 shows that no particular flow is starved under LIFO-FD, and the completion times are similar to those of FIFO-TD (figure 14) when multiple flows are transferred (figure 15(b) and figure 14(b)). Thus, LIFO-FD does not disproportionally alter how TCP flows are transmitted, when compared to FIFO-TD.

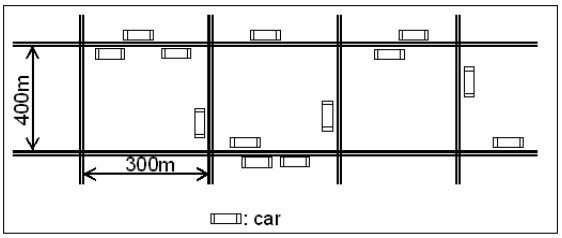


Figure 16. Topology used in the mobile ad hoc network simulation study

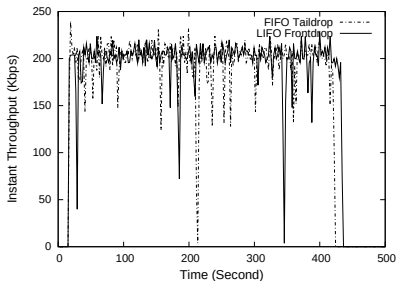
4.3. Mobile Network Simulations

4.3.1. Simulation Setup

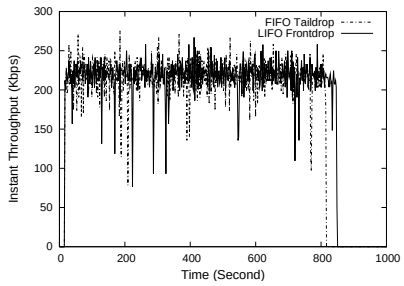
Since the expected live streaming applications may often run over mobile ad hoc networks with a larger number of nodes than in the previous experiments, the second simulation scenario uses a 250-node vehicular ad hoc network on a 1500m x 1500m area extracted from the TIGER/Line database of the US Census Bureau [25]. The region simulated is an area of Fellsmere, FL with center point coordinates latitude 27.784728° North and longitude 80.604385° West. The roads form a grid layout as shown in Figure 16. For each road, we assign bi-directional traffic with two lanes in each direction.

The vehicles movements are generated using our microscopic mobility generator based on the car-following and lane-changing models proposed by Gipps [26,27]. When a vehicle enters a road segment, we determine what its action at the end of the segment should be (i.e., left turn, right turn, u-turn, or straight ahead) based on the average traffic flows of the roads crossing the end intersection. We discard the first 2000 sec of output to obtain more accurate movements of nodes. The simulations run for 400 sec.

The routing protocol used is RBVT-R [4]. RBVT-R is a reactive source routing protocol which creates stable paths consisting of road segments that have enough vehicles on them to avoid broken routes when

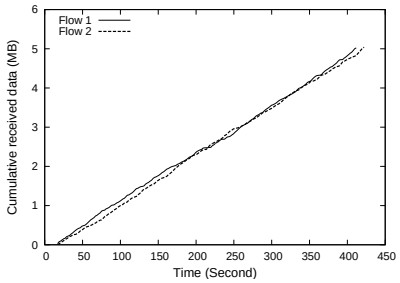


(a) 2 flows, 5MB each

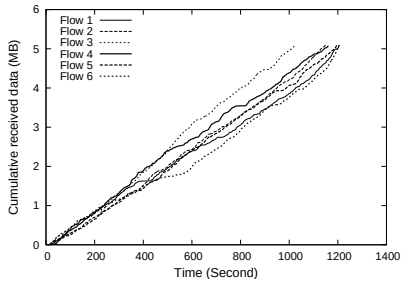


(b) 10 flows, 2MB each

Figure 13. Total instantaneous TCP throughput, Buffer size = 50 slots

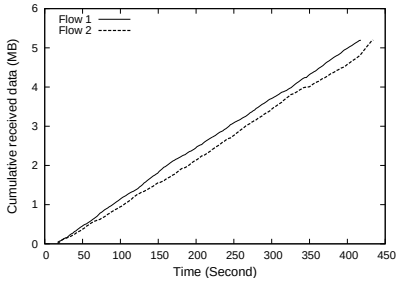


(a) 2 flows, 5MB each

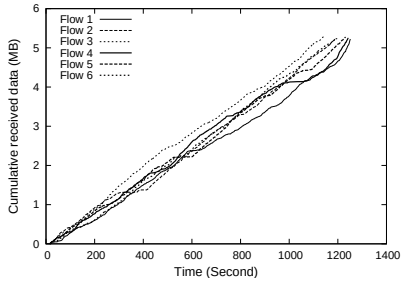


(b) 6 flows, 5MB each

Figure 14. TCP fairness: per flow cumulative received data, FIFO-TD, Buffer size = 50 slots

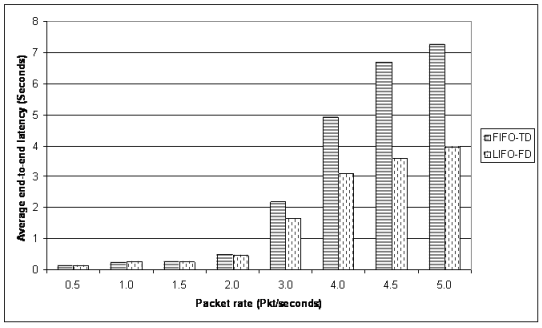


(a) 2 flows, 5MB each

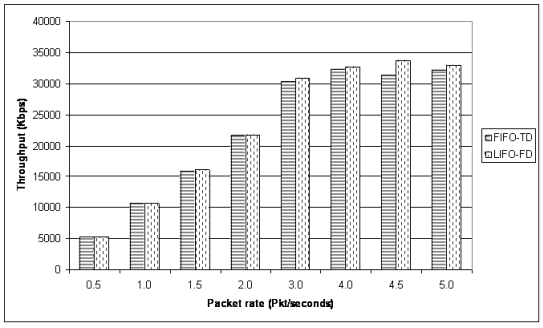


(b) 6 flows, 5MB each

Figure 15. TCP fairness: per flow cumulative received data, LIFO-FD, Buffer size = 50 slots



(a) Average end-to-end delay



(b) Average throughput

Figure 17. Average end-to-end delay and average throughput comparison: 802.11 LIFO-FD queuing vs. 802.11 FIFO-TD queuing. The routing protocol is RBVT-R, and the network size is 250 nodes. 30 nodes exchange UDP CBR data traffic.

geographical forwarding is employed.

4.3.2. Simulation Results

Figure 17 demonstrates not only that LIFO-FD results in lower delay (as much as 45%) than FIFO-TD but also that it does not adversely affect the delivery ratio in mobile ad hoc networks. When we compare the static network results, Section 4.2, to those obtained in the mobile network, we see that the static network provides slightly better results than the mobile network. The reasons for this can be attributed, in part, to the average number of hops which was higher in the mobile network simulations, 7 hops compared to 5 hops used for the static network. Additionally, the node movements sometimes lead to periods of broken connectivity between certain source and destination pairs. In those instances, the routing algorithm must repair or recreate a new communication path, which increases the overall end-to-end transmission delay.

5. Conclusions and Future Work

This article studied the relative performance of four simple queuing disciplines, FIFO-TD, FIFO-FD, LIFO-TD, and LIFO-FD, to understand if another discipline can perform better than the commonly-used FIFO-TD in ad hoc networks. Our focus was on improving the performance of real-time multimedia applications, which are already used in real-life scenarios and have the potential to become the main driving force behind ad hoc networks. More exactly, the goal was to find which queuing discipline works best for UDP traffic in terms of latency and jitter, without compromising other parameters such as throughput, or protocols such as TCP.

The main conclusion of this article is that LIFO-FD performs best for high traffic periods, and FIFO-TD performs best for low traffic periods. While LIFO is generally perceived as unfair, our theoretical analysis and simulation results demonstrated that this unfairness is tied mainly to LIFO-TD. In our experiments, LIFO-FD achieves similar TCP fairness and throughput with FIFO-TD.

As future work, we plan to design an adaptive queuing mechanism that dynamically switches between LIFO-FD and FIFO-TD function of the locally monitored traffic load. An important issue to be studied is how the performance will be affected by independent switching decisions that could result in some nodes on the path using one queuing discipline and other nodes using the other discipline. We believe that this mechanism could improve significantly the performance of real-time multimedia applications without incurring any network overhead. Additionally, its expected computational overhead is low as it does not perform per-flow management. Finally, this mechanism could be seamlessly integrated with existing higher layer solutions for streaming.

A. Appendix

In this section, we derive the expressions of $\sigma_{k,N}(.),\ \ \phi_{k,N}(.)$ as well as $\zeta_{k,k,S}(s)$. The notation used is the same as introduced in Section 3.

A.1. Deriving Expressions of $\sigma_{k,N}(.),\ \ \phi_{k,N}(.)$:

A host queue is modeled as a Markov Chain $X(t)$ on the state space $\{-1, 0, \cdots, N\}$, where $X(t)$ is the state at time t. For all disciplines considered, if a packet arrives while $X(t^-) = (k)$ and $k < N$ then the state changes to $X(t^+) = (k+1)$, all packets already in the system keep their position, and the new packet goes into position $k+1$.

Given $X(0^+) = k$, $1 \le k \le N$, what is the probability that after time 0 state (0) will be reached before state $(N+1)$ is reached? and what is the probability that state $(N+1)$ will be reached before state (0) is reached? We will use the shorthand that "Success" stands for "the state (0) is reached before the state $(N+1)$ is reached", and "Failure" stands for "the state $(N+1)$ is reached before the state (0) is reached".

We define

$$\sigma_{k,N}(s) = E[e^{-sT_{k,N}}\chi(success)],$$
$$\phi_{k,N}(s) = E[e^{-sT_{k,N}}\chi(failure)]. \quad \text{(A.1)}$$

The Difference Equation we solve here was considered in (at least) Feller II [28] page 454 etc, but with boundary conditions different from (A.2) below. It may have been considered over the years in other places in Applied Probability, possibly with other or similar Boundary Conditions as in (A.2).

Conditional First Passage Time Distributions for Birth–Death Processes have been studied extensively. Most interest has been in Birth–Death processes with State–Dependent arrival rates λ_n and departure rates $\mu - n$. We have not found the simple case we are studying in this Appendix.

We have $\sigma_{k,N}(0) = \sigma_{k,N}$, $\phi_{k,N}(0) = \phi_{k,N}$, and $\sigma^*_{k,N}(s) = \frac{\sigma_{k,N}(s)}{\sigma_{k,N}}$ is the Laplace Transform of the time until the waiting room is empty for the first time, given we start in state (k) and given that the waiting room goes empty before the first drop occurs. Similarly $\phi^*_{k,N}(s) = \frac{\phi_{k,N}(s)}{\phi_{k,N}}$ is the Laplace Transform of the time until the first drop occurs, given we start in state (k) and given a drop occurs before the first time the waiting room goes empty.

We have the boundary conditions

$$\sigma_{0,N}(s) \equiv 1,\ \sigma_{N+1,N}(s) \equiv 0,$$
$$\phi_{0,N}(s) \equiv 0,\ \ \phi_{N+1,N}(s) \equiv 1. \quad \text{(A.2)}$$

Considering the boundary conditions in A.2, we have

$$\sigma_{k,N}(s) = \frac{\lambda+\mu}{\lambda+\mu+s}\cdot \cdot\left(\frac{\mu}{\lambda+\mu}\sigma_{k-1,N}(s) + \frac{\lambda}{\lambda+\mu}\sigma_{k+1,N}(s)\right) \quad \text{(A.3)}$$

$$\phi_{k,N}(s) = \frac{\lambda+\mu}{\lambda+\mu+s}\cdot \left(\frac{\mu}{\lambda+\mu}\phi_{k-1,N}(s) + \frac{\lambda}{\lambda+\mu}\phi_{k+1,N}(s)\right). \quad \text{(A.4)}$$

Solving

$$(\lambda+\mu+s)x = \mu + \lambda x^2 \quad \text{(A.5)}$$

gives

$$x_1(s) = \frac{(\lambda+\mu+s) + \sqrt{(\lambda-\mu)^2 + 2(\lambda+\mu)s + s^2}}{2\lambda},$$

$$x_2(s) = \quad \text{(A.6)}$$
$$\frac{(\lambda+\mu+s) - \sqrt{(\lambda-\mu)^2 + 2(\lambda+\mu)s + s^2}}{2\lambda}.$$

We now have

$$\sigma_{k,N}(s) = C_{1,S,N}(s)(x_1(s))^k + C_{2,S,N}(s)(x_2(s))^k,$$
$$\phi_{k,N}(s) = C_{1,F,N}(s)(x_1(s))^k + C_{2,F,N}(s)(x_2(s))^k, \quad \text{(A.7)}$$

where S and F stand for "success" and "failure". $C_{1,S,N}(.), C_{2,S,N}(.), C_{1,F,N}(.), C_{2,F,N}(.)$ are computed from the boundary conditions (A.2):

$$C_{1,S,N}(s) + C_{2,S,N}(s) = 1,$$
$$C_{1,S,N}(s)(x_1(s))^{N+1} + C_{2,S,N}(s)(x_2(s))^{N+1} = 0,$$
$$C_{1,F,N}(s) + C_{2,F,N}(s) = 0,$$
$$C_{1,F,N}(s)(x_1(s))^{N+1} + C_{2,F,N}(s)(x_2(s))^{N+1} = 1.$$

This gives:

$$C_{1,S,N}(s) = -\frac{(x_2(s))^{N+1}}{(x_1(s))^{N+1} - (x_2(s))^{N+1}},$$
$$C_{2,S,N}(s) = +\frac{(x_1(s))^{N+1}}{(x_1(s))^{N+1} - (x_2(s))^{N+1}},$$
$$C_{1,F,N}(s) = +\frac{1}{(x_1(s))^{N+1} - (x_2(s))^{N+1}},$$
$$C_{2,F,N}(s) = -\frac{1}{(x_1(s))^{N+1} - (x_2(s))^{N+1}}.$$

This finally gives

$$\sigma_{k,N}(s) = \frac{(x_1(s))^{N+1}(x_2(s))^k - (x_2(s))^{N+1}(x_1(s))^k}{(x_1(s))^{N+1} - (x_2(s))^{N+1}},$$

$$\phi_{k,N}(s) = \frac{(x_1(s))^k - (x_2(s))^k}{(x_1(s))^{N+1} - (x_2(s))^{N+1}}. \quad \text{(A.8)}$$

Hence

$$\sigma_{k,N}(0) = \frac{1-\rho^{N+1-k}}{1-\rho^{N+1}},\ \phi_{k,N}(0) = \frac{\rho^{N+1-k}-\rho^{N+1}}{1-\rho^{N+1}}.$$

If $\rho = 1$ this reduces to

$$\sigma_{k,N}(0) = \frac{N+1-k}{N+1},\ \ \phi_{k,N}(0) = \frac{k}{N+1}. \quad \text{(A.9)}$$

A.2. Deriving Expression of $\zeta_{k,k,S}(s)$:

We have

$$\zeta_{0,L,S}(s) \equiv 1,\ \ \zeta_{0,L,F}(s) \equiv 0, \quad \text{(A.10)}$$

$$\zeta_{1,N,S} = \frac{\lambda+\mu}{\lambda+\mu+s}.\frac{\mu}{\lambda+\mu} = \frac{\mu}{\lambda+\mu+s}. \quad \text{(A.11)}$$

Hence, $\zeta_{k,L,S}(.)$ is known as long as $k = 1$ and $L = N$.

For $1 = k \leq L < N$ we have

$$\zeta_{1,L,S}(s) = \\ = \frac{\lambda+\mu}{\lambda+\mu+s}\left(\frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu}\eta_{1,L+1,S}(s)\right) \\ = \frac{\mu}{\lambda+\mu+s} + \frac{\lambda\zeta_{1,L+1,S}(s)}{\lambda+\mu+s}.$$

By starting with (A.11) and then iterating (A.12) backward we compute $\zeta_{k,L,S}(.)$ for $k = 1$ for $L = N, N-1, \cdots, 1$.

Suppose we know $\zeta_{\kappa,L,S}(s)$ for all $\kappa < k$ and all L, with $k \geq 2$. We have

$$\zeta_{k,N,S}(s) = \frac{\lambda+\mu}{\lambda+\mu+s}. \\ \cdot\left(\frac{\mu}{\lambda+\mu}\zeta_{k-1,N-1,S}(s) + \frac{\lambda}{\lambda+\mu}\zeta_{k-1,N,S}(s)\right).$$

This allows computation of $\zeta_{k,N,S}(s)$. Next we have for $k \leq L < N$:

$$\zeta_{k,L,S}(s) = \frac{\lambda+\mu}{\lambda+\mu+s}. \\ \cdot\left(\frac{\mu}{\lambda+\mu}\zeta_{k-1,L-1,S}(s) + \frac{\lambda}{\lambda+\mu}\zeta_{k,L+1,S}(s)\right).$$

By iterating (A.12) backward ($L = N-1, N-2, \cdots, k$) we now compute $\zeta_{k,L,S}(s)$ for all $1 \leq k \leq L \leq N$.

REFERENCES

1. The National Oceanic and Atmospheric Administration (NOAA), NOAA unmanned aircraft system, http://uas.noaa.gov/. Last accessed July 2009.
2. US Northern Command, Military airborne imagery supported immediate hurricane recovery efforts, http://www.northcom.mil/news/2008/091908I.html. Last accessed July 2009.
3. Federal Communications Commission, FCC 03-024, FCC Report and Order, February 2004.
4. J. Nzouonta, N. Rajgure, G. Wang, C. Borcea, VANET routing on city roads using real-time vehicular traffic information, IEEE Transactions on Vehicular Technology 58 (7).
5. T. V. Lakshman, A. Neidhardt, T. Ott, The drop from front strategy in TCP and TCP over ATM, in: Proceedings IEEE International Conference on Computer Communications (INFOCOM), Los Angeles, CA, USA, 1996, pp. 1242–1250.
6. S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413.
7. S. Cen, P. C. Cosman, G. M. Voelker, End-to-end differentiation of congestion and wireless losses, IEEE/ACM Transactions on Networking 11 (5) (2003) 703–717.
8. M. Chen, A. Zakhor, Rate control for streaming video over wireless, IEEE Wireless Communications 12 (4) (2005) 59–65.
9. Y. Yi, S. Shakkottai, Hop-by-hop congestion control over a wireless multi-hop network, IEEE/ACM Transactions on Networking 15 (1) (2007) 133–144.
10. M. Handley, S. Floyd, J. Padhye, J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification (2003).
11. C. E. Perkins, E. M. Royer, Ad hoc on-demand distance vector routing, in: Proceedings 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, USA, 1999, pp. 90–100.
12. D. B. Johnson, D. A. Maltz, Dynamic source routing in ad hoc wireless networks, Mobile Computing 353 (5) (1996) 153–161.

13. C. Sarr, C. Chaudet, G. Chelius, I. Guerin-Lassous, Bandwidth estimation for IEEE 802.11-based ad hoc networks, IEEE Transactions on Mobile Computing 7 (8) (2008) 1228–1241.
14. E. Setton, T. Yoo, X. Zhu, A. Goldsmith, B. Girod, Cross-layer design of ad hoc networks for real-time video streaming, IEEE Wireless Communications 12 (4) (2005) 59–65.
15. E. Setton, J. Noh, B. Girod, Congestion-distorsion optimized peer-to-peer video streaming, in: Proceedings IEEE International Conference on Image Processing (ICIP), Atlanta, GA, USA, 2006, pp. 721–724.
16. A. Majumdar, D. G. Sachs, I. V. Kozintsev, K. Ramchandran, M. M. Y. g, Multicast and unicast real-time video streaming over wireless LANs, IEEE Transactions on Circuits and Systems for Video Technology 12 (6) (2002) 524–534.
17. E. Kohler, M. Handley, S. Floyd, Designing DCCP: Congestion control without reliability, in: Proceedings of ACM SIGCOMM, Pisa, Italy, 2006, pp. 27–38.
18. Wolfram Research, Mathematica 6, http://www.wolfram.com/products/mathematica/. Last accessed July 2009.
19. The network simulator: NS-2, http://www.isi.edu/nsnam/ns. Last accessed July 2009.
20. R. S. Prasad, C. Dovrolis, M. Thottan, Router buffer sizing revisited: the role of the output/input capacity ratio, in: Proceedings 2007 ACM CoNEXT Conference, New York, NY, USA, 2007, pp. 1–12.
21. R. Chertov, S. Fahmy, N. B. Shroff, A device-independent router model, in: Proceedings 27th IEEE Conference on Computer Communications (INFOCOM), Phoenix, AZ, USA, 2008, pp. 1642–1650.
22. Juniper Networks, Configuring the ATM1 queue length, https://www.juniper.net/techpubs/software/junos/junos92/swconfig-network-interfaces/configuring-the-atm1-queue-length.html. Last accessed July 2009.
23. G. Holland, N. Vaidya, Analysis of TCP performance over mobile ad hoc networks, Wireless Networks 8 (2) (2002) 275–288.
24. T. Razafindralambo, I. Guerin-Lassous, Increasing fairness and efficiency using the MadMac protocol in ad hoc networks, Elsevier Ad Hoc Networks 6 (3) (2008) 408–423.
25. U.S.Census Bureau - TIGER/Line, http://www.census.gov/geo/www/tiger/.
26. P. G. Gipps, A behavioural car-following model for computer simulation, Transportation Research Board 15 (1981) 105–111.
27. P. G. Gipps, A model for the structure of lane-changing decisions, Transportation Research Board 20B (5) (1986) 403–414.
28. W. Feller, An Introduction to Probability Theory and its Applications, Vol II, Second Corrected Printing, Wiley (November 1966).