Separation of Sensor Control and Data in a Distributed Meteorological Sensing Network

Victoria Manfredi[†], Jim Kurose[†], Naceur Malouch[‡], Chun Zhang[*], Michael Zink[†]

Technical Report 2008-37
October 1, 2008

[†]Department of Computer Science
140 Governors Drive
University of Massachusetts
Amherst, Massachusetts 01003-4601
{vmanfred, kurose, zink}@cs.umass.edu

[‡] University Pierre et Marie Curie, LIP6
8, Rue du Capitaine Scott
75015 Paris - FRANCE
Naceur.Malouch@lip6.fr

[*] IBM T.J. Watson Research Center
Hawthorne, NY 10532 USA
czhang1@us.ibm.com

Abstract

Sensor networks are prone to congestion due to bursty and high-bandwidth data traffic, combined with many-to-one data routing to a sink. Delayed and dropped packets then degrade the performance of the sensing application. In this work, we investigate the value of separate handling of sensor control and data traffic, during times of congestion, in a networked sense-and-response system. We analyze a meteorological remote sensing application, running over a network of X-band radars. Our application measures the reflectivity (a measure of the number of scatterers in a unit volume of atmosphere known as a voxel) and tracks storms (i.e., regions of high reflectivity) using a Kalman filter. We find that separate handling of sensor control and data decreases the round-trip control-loop delay, and consequently increases either (i) the number of voxels V scanned with N_c reflectivity samples per voxel or (ii) the number of reflectivity samples N obtained per voxel scanned. In the former case, the utility increases linearly with the number of scanned voxels. In the latter case, since sensing accuracy improves only as a function of $\sqrt{N}$, the gain in accuracy for the reflectivity estimate per voxel as N increases is relatively *small* except when prioritizing sensor control increases N significantly (e.g., when sensor control packets suffer severe delays). Also, in this latter case, because accuracy degrades as a function of $\sqrt{N}$, the system can still perform well during times of congestion, but only when sensor control packets receive priority and remain unaffected by data overload. For the storm tracking application, we find that under severe congestion, performance losses when not prioritizing sensor control can actually accumulate over time.

1 Introduction

Sensor networks are prone to congestion due to bursty and high-bandwidth data traffic, combined with many-to-one data routing to a sink [1, 2, 3]. Delayed and dropped packets then degrade the performance of the sensing application. For a networked sense-and-response application, sensed data transmitted through the network may have considerable redundancy in both time and space, making performance somewhat insensitive to data packet loss and delay. Conversely, such an application is typically much more sensitive to loss or delay of sensor control packets, since these packets carry the application's sensor control commands generated in response to received data. Consequently, there are potential advantages to separating the forwarding of sensor control and data packets.

In this paper, we investigate the value of separate handling of sensor control and data traffic, during times of congestion, in a networked sense-and-response system. By sensor control we mean the commands specifying how the sensors should collect new data, based on previously sensed data. We ground our analysis in a networked meteorological remote sensing application [4, 5, 6]. Our application measures the reflectivity in the atmosphere and tracks storms (i.e., regions of high reflectivity) using a Kalman filter; reflectivity is a measure of the number of scatterers in a voxel (i.e., volume of atmosphere). In this application, data are transmitted from remote X-band radars over a shared wireline [4, 5] or wireless [6] network to a central site that periodically generates radar targeting (sensor control) commands based on features detected in the received data. To evaluate the utility of separation, we compare the performance of aggregate FIFO for both radar data and sensor control commands with that of priority forwarding of sensor control on the following application-specific performance metrics. First, we consider the number of voxels, V, that can be scanned given a constant number of reflectivity samples, N_c, obtained per voxel. Next, we consider the number of reflectivity samples, N, that can be obtained per voxel given a constant number of voxels, V_c, to scan; for this second metric, we also consider the corresponding gain in accuracy in the reflectivity estimate as N increases. Finally, we consider the root mean-squared error when tracking a storm, as in [7].

Our results from the meteorological application show that separate handling of sensor control and data decreases the round-trip control-loop delay, and consequently increases either *(i)* the number of voxels V scanned or *(ii)* the number of samples of reflectivity N obtained per voxel scanned. In the former case, the utility increases linearly with the number of scanned voxels. In the latter case, since sensing accuracy improves only as a function of $\sqrt{N}$, the gain in accuracy for the reflectivity estimate per voxel as N increases is relatively *small* except when prioritizing sensor control increases N significantly (e.g., when sensor control packets suffer severe delays). For the storm tracking application, we find that under severe congestion, performance losses when not prioritizing sensor control can actually accumulate over time.

In one sense, our results mirror those in [8] regarding differential traffic handling and network-based performance metrics: "that performance is generally satisfactory in a classical best effort network as long as link load is not too close to 100%," and that "there appears little scope for service differentiation beyond the two broad categories of 'good enough' and 'too bad.'" *However, unlike some other network applications, the sensing application can still perform well during times of severe congestion, when sensor control packets are given priority, precisely because sensing accuracy both improves and degrades slowly in the number of sensed values (reflectivity samples) obtained.*

The notion of separate handling of control and data packets in a network has a long history. While previous work [9, 10, 11, 12, 13] focuses on prioritizing network control packets, our focus here is on prioritizing sensor control packets. Whereas network control affects what data is transmitted and at what rate, sensor control additionally affects what data is actually sensed and thus available to be transmitted. For example, consider object tracking and suppose that the sensor controller incorrectly asks to sense data from one location in the environment when the object is at a different location. The data that should have been collected from sensing the different location cannot be collected retroactively, as the environmental conditions may have changed (i.e., the object may have moved) during the time that the incorrect location was sensed. Thus, in a networked sense-and-response system, the issue is not just that data may be received late, but that the opportunity to ever sense a particular set of data may be missed completely. Other work, in sensor networks,

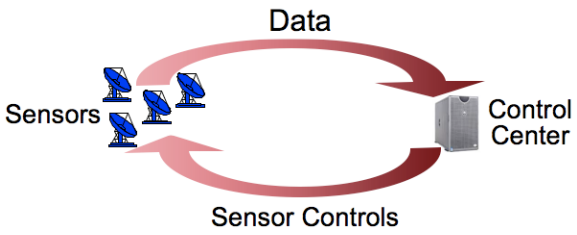
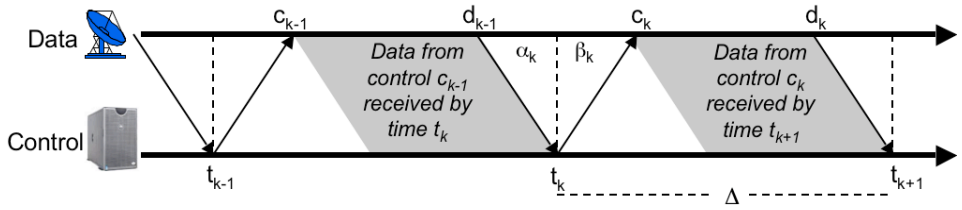
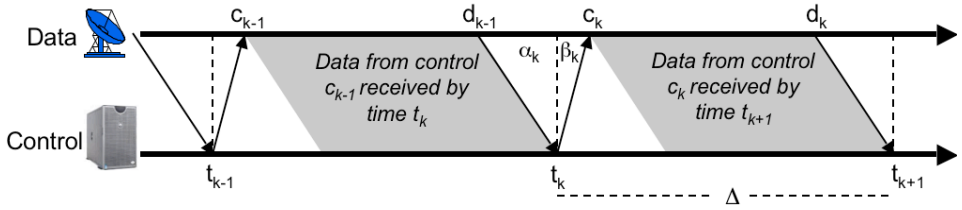


Figure 1: A networked sense-and-response system.



(a) $\alpha_k \approx \beta_k$



(b) $\alpha_k > \beta_k$

Figure 2: Timing of the sense-and-response control loop when (a) $\alpha_k \approx \beta_k$ and (b) $\alpha_k > \beta_k$.

has considered service differentiation [14, 15, 16], including during times of congestion [17], but does not specifically look at the effects of prioritizing sensor control nor consider sense-and-response systems.

The remainder of this paper is structured as follows. In Section II, we overview related work. In Section III, we discuss general characteristics of a networked sense-and-response system. In Section IV, we describe our meteorological sensing application. In Section V, we present simulation results comparing the performance of FIFO versus priority forwarding of sensor control packets. Section VI concludes this paper.

2 Related Work

The notion of separate handling of control and data packets in a network has a long history. The SS7 signaling system [9] that carries control packets in telephone networks is a packet-switched control network that is physically separate from the circuit-switched network carrying voice traffic. In ATM networks, Q2931 signaling packets for virtual circuit management are carried over connections that are logically separated from data traffic [10]. For IP networks, it is possible for operators to configure routers to provide prioritized service for "control" protocols such as BGP or SNMP. In wireless networks, [13] advocates for a separate control channel for controlling access to a shared medium. Proposals for priority handling of TCP acknowledgments [11, 12] can also be considered as providing a different level of service to control packets (ACKs) than data

packets. While this previous work focuses on prioritizing network control packets, our focus here is on prioritizing sensor control packets.

Other work, in sensor networks, has considered service differentiation for different classes of traffic. [14] assigns priority levels to packets, forwarding higher-priority packets more frequently over more paths to achieve higher probability of delivery. [15] allocates rates to flows based on the class of traffic being sent and the estimated load on the network. [16] considers bandwidth reservation for high-priority flows in wireless sensor networks. [17] proposes congestion-aware routing in sensor networks, providing differential service to high priority data traffic versus low-priority data traffic in congested areas of the network. None of these approaches, however, considers the effects of prioritizing only sensor control in a networked sense-and-response system.

Control theory considers the effects of a network within the control loop in the field of "Networked Control Systems" [18, 19]. As in networked sense-and-response systems, data and sensor control are sent over a network. Unlike in the sense-and-response system, however, the sensor control and data packets in a Networked Control System are constrained to be the feedback (sensor control) and measurements (data) of a classical control system. Consequently, the ratio of data to control is much smaller than that of a sense-and-response system such as the CASA network [4, 5]. Since any data packet (i.e., measurement) in a Networked Control System may now be as important as any sensor control packet (i.e., feedback), it is not necessarily beneficial to always give higher priority to sensor control. Instead, packets are scheduled to optimize expected performance using the control equations, for instance by incorporating the error incurred due to network delays directly into the control equations [20], or by optimally dropping selected data measurements during times of overload by analyzing the effect of the resulting missing measurements on the control equations [21]. Networked Control Systems can thus be considered a sub-class of the more general networked sense-and-response systems we consider in this work.

3 Networked Sense-and-Response Systems

A generic networked sense-and-response system is shown in Figure 1: data are streamed from sensors to a central site, while sensor commands flow from the central site back to the sensors. The central site closes the system's main control loop by ingesting data, computing statistics from the data, and selecting each sensor's future data collection strategy based on the statistics. As shown in Figure 2, a networked sense-and-response system periodically computes a new sensor control. At the start of the kth update interval, t_k, the control center issues a command to the sensor specifying how to collect data. After a delay of β_k, the command is received at the sensor. The sensor then begins transmitting back measured data, where α_k is the delay of data from the sensor to the control center before the kth control update interval. After time Δ, the sense-and-response cycle then repeats. As shown in Figure 2, the sensor control computed at time t_k is based on data that was sent by time $t_k - \alpha_k$ by the sensor. The sensor control is then applied at the sensor at time $c_k = t_k + \beta_k$. We assume that the duration, Δ, of each control update interval is fixed, but the length of Δ could also depend on time or another metric.

Suppose now that packets are delayed. Then the α_k and β_k delays will increase and the total amount of data from control c_k received by time t_{k+1} at the central control will decrease. Since the ratio of data to sensor control is large, it should be possible to provide significantly better performance to sensor control traffic (e.g., lower end-to-end delays and lower loss) with only a minimal degradation in the performance of the data traffic, as illustrated in Figure 2(b), in keeping with queueing theoretic conservation laws [22]. This then decreases the "round trip" delay for the control loop - the summed delay of sensed data from a sensor to the central site and the delay of sensor control packets back to the sensor. Lower round-trip delays enable the central site to examine more data before making a control decision, resulting in more data (as when our utility metric is the number of voxels sensed), or more accurate estimates of the sensed quantity of interest (such as reflectivity in our meteorological application). More accurate estimates should then give better application-level performance.

Can we quantify how much better the application performance can be when prioritizing control? Consider the effect of the length of the control update interval Δ. For FIFO scheduling, data are collected during a time interval of length $\Delta - \alpha_k - \beta_k$, while for priority scheduling, data are collected during a time interval of at most length $\Delta - \alpha_k$. Consequently, priority scheduling has a percentage gain of at most $\beta_k/(\Delta - \alpha_k - \beta_k)$ more time over FIFO. As Δ decreases, the percentage gains from priority scheduling thus increase. As Δ increases, although the total amount of data that is collected will increase, the gains from priority scheduling will decrease. Note that when Δ is sufficiently small that the ratio of sensor control to data approaches one, (e.g., as is possible in a Networked Control System, see [18, 19]), then we do not expect that always prioritizing control is beneficial.

Now consider the number of i.i.d. data samples $X_1, \ldots, X_n$ collected during time $t = \Delta - \alpha_k - \beta_k$. Increasing t linearly increases the number of samples n collected, e.g., such as when our performance metric of interest is the number of voxels. Suppose, however, that we use those samples to compute an unbiased estimator $W(\mathbf{X})$ of some parameter θ, such as reflectivity in the case of our meteorological application. The Cramer-Rao bound [23] says that the standard deviation of $W(\mathbf{X})$ from θ, $\mathrm{SD}_\theta(W(\mathbf{X}))$, can be lower bounded as follows,

$$\mathrm{SD}_\theta(W(\mathbf{X})) \quad \geq \quad \frac{1}{\sqrt{n\,\mathcal{I}}} \tag{1}$$

where $\mathcal{I}$ is the Fisher information, representing the information a sample contains about the estimator $W(\mathbf{X})$. As $\mathrm{SD}_\theta(W(\mathbf{X}))$ decreases only at the rate of $1/\sqrt{n}$, sensing accuracy improves slowly in the number of sensed data values obtained. We will see this $1/\sqrt{n}$ behaviour again in Section IV-B when computing the reflectivity standard deviation.

Now suppose that the system is overloaded and that packets may be dropped. Since prioritizing control limits the number of sensor control packets dropped, suppose that only data packets are dropped. As the number of data samples, n, decreases, the standard deviation of the estimate increases only at the rate of $1/\sqrt{n}$. Thus, sensing accuracy both improves *and* degrades slowly in the number of data samples. Consequently, the sensing application may still perform well in overload, if data packets, but not sensor control packets, are dropped.

The discussion in this section so far only considers potential gains from priority scheduling within a single control update interval. Whether possibly small per-interval gains (or losses) can accumulate over multiple intervals depends on how the sensed data is used. To investigate this we consider a meteorological tracking application, described in Section IV-C.

4 Meteorological Application

In this section, we describe the networked meteorological remote sensing application we use to analyze the effect on application performance when prioritizing sensor control. As in Figure 1, remote X-band radars transmit measured reflectivity values to a central site. The Meteorological Command and Control (MC&C) [24] component at this central site identifies meteorological features from the radar data, reports the features to end-users, and determines each radar's future scan strategy (i.e., the volume of the atmosphere to be scanned by each radar). A 4-node system has been developed and deployed in southwestern Oklahoma as part of the CASA project [25]. Like in Figure 2, the system operates on a Δ=30-second control update interval. We now describe our network and radar meteorology models, and the meteorological tracking application running over the network.

4.1 Network Model

In this section, we describe how we model the effect of prioritizing sensor control on the data delays α_k and sensor control delays β_k during congestion, as well as how we model packet drops. We first describe how we

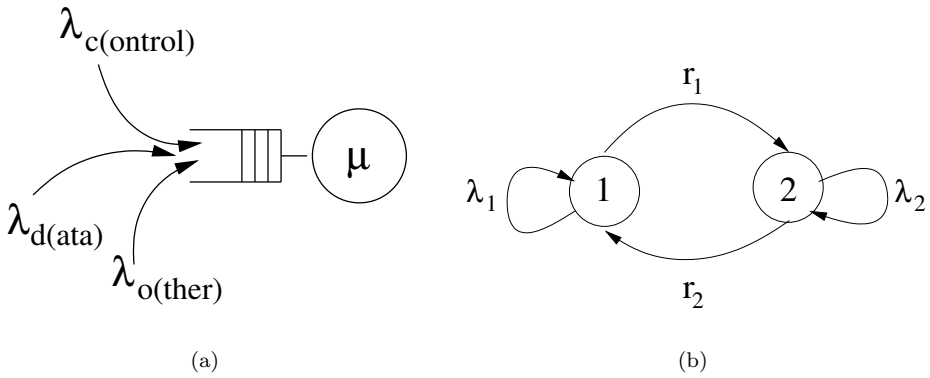


Figure 3: (a) Model of the bottleneck queue in the network. (b) The 2-state Markov modulated Poisson process used to model the "other" traffic.

model packet delays. We assume a wireless network where data is sent from radars (sources) to a control center (the sink), and sensor control commands are sent back to the radars from the control center. We analyze the packet delays incurred at the bottleneck link, assuming other delays may be ignored. In a wireless sensor network, the bottleneck link might be the last hop node before the sink. We consider two scheduling mechanisms: *(1)* aggregate FIFO service of sensor control and data packets and *(2)* nonpreemptive priority forwarding of sensor control packets. For instance, when using 802.11, priority forwarding over the wireless links could be done as in [26] by assigning queuing, waiting, and back-off times based on priority level.

Figure 3(a) shows our queue model of the bottleneck link. We group traffic contending for the bottleneck queue into three flows: sensor *control* traffic destined for some node r, *data* traffic generated by node r, and *other* traffic including data and sensor control traffic either generated by or destined for nodes other than node r. Although data and sensor control packets might always travel in opposite directions in a sensor network, due to the wireless links, data and sensor control packets will still compete against each other for access to the wireless link; consequently, we model the bottleneck link as a single queue. For wired links, data and sensor control packets may end up in the same outgoing queue when there are, e.g., (i) multiple control centers, (ii) multiple sensor applications using the same network, or (iii) asymmetric routing.

Since data and sensor control are generated at deterministic intervals in the CASA network, we assume that sensor control and data packets have deterministic arrivals with rates λ_c and λ_d respectively. "Other" packets arrive according to a two-state Markov-modulated poisson process, see Figure 3(b), where packets arrive on average at rate λ_o; in state 1 packets arrive at rate λ_1, in state 2 packets arrive at rate λ_2, and transitions from states 1 to 2 and from states 2 to 1 occur at rates r_1 and r_2 respectively. To vary the burstiness, as measured using the index of dispersion, see [27], we vary the values of λ_1 and λ_2 while keeping λ_o constant. Finally, we assume that the packet service time is exponentially distributed with rate μ. To compute the delays through the bottleneck queue, we use the ns-2 simulator [28], see Section V.

To model packet drops, e.g., due to overload, we compare the worst-case and best-case scenarios. For the worst-case scenario, we assume that all sensor control packets are dropped and that the default sensor control must be used. For the best case scenario, we assume that no sensor control packets are dropped and that the scan strategy specified by the sensor controls always collect data that is optimal for the application performance metrics defined in the next section.

4.2 Radar Meteorology Model

In this section, we describe the radar meteorology model we use to evaluate the effect of prioritizing sensor control on application performance. A radar operates by sending out pulses at a given rate, as it sweeps

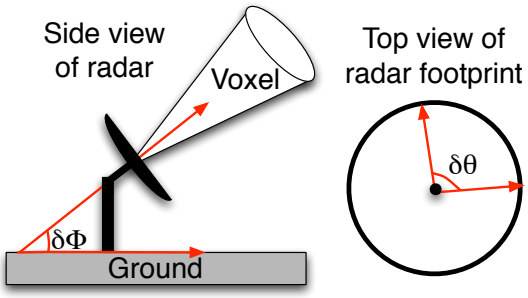


Figure 4: Radar definitions.

through the sector it is scanning. For a given time duration, the smaller the sector scanned, to some minimum sector size, the better the estimated reflectivity values, since the radar can send more pulses per volume of atmosphere, see [29]. Meteorological algorithms use reflectivity values to identify, e.g., storms and tornados. We now describe the application performance metrics of interest.

4.2.1 Number of Voxels Scanned

Suppose that the number of pulses, N_c, transmitted per voxel is fixed, where a voxel is a unit volume of atmosphere. Then the simplest metric of interest is the number of voxels, V, that can be scanned during time $\Delta - \alpha_k - \beta_k$, given by,

$$V = \frac{(\Delta - \alpha_k - \beta_k) f_p}{N_c} \tag{2}$$

where $f_p = 3$ kHz is the pulse repetition frequency. We ignore here how the voxels are distributed to form a sector scan. As $\Delta - \alpha_k - \beta_k$ increases, the number of voxels V that can be scanned, each with N_c pulses, also increases linearly.

4.2.2 Reflectivity Standard Deviation

We now relax the assumption that the number of pulses transmitted per voxel is constant. We focus here on the quality of the metrics estimated from the data. The number of pulses, N, transmitted per voxel given a constant number of voxels, V_c, and during a time interval of length $\Delta - \alpha_k - \beta_k$ is [29],

$$N = \frac{(\Delta - \alpha_k - \beta_k)\, f_p}{V_c} \tag{3}$$

where $f_p = 3$ kHz is the pulse repetition frequency, $V_c = \frac{\delta\theta\, \delta\phi}{\theta\, \phi}$, $\delta\theta$ is the size of the sector scanned in degrees, see Figure 4, $\theta = 1.8°$ is the antenna beamwidth, $\delta\phi = 12°$ is the elevation height, and $\phi = 2°$ is the elevation step (i.e., the increase in elevation after a horizontal scan). Figure 5 plots N for different $\alpha_k + \beta_k$ delays: as the sector size decreases, more pulses can be transmitted per voxel. Following Equation 3, Figure 5 shows that N decreases linearly as $\alpha_k + \beta_k$ increases.

Each pulse transmitted per voxel returns an estimate of the reflectivity for that voxel. Reflectivity is a measure of the number of scatterers in a volume of atmosphere. Averaging over more samples increases the confidence in the estimated reflectivity value. Given N samples for a voxel, the reflectivity standard

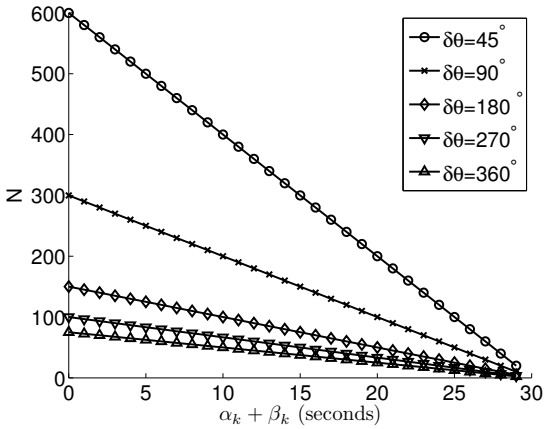


Figure 5: Number of pulses, N, per voxel for $\Delta = 30$ sec and varying $\alpha_k + \beta_k$.

deviation, $\hat{\sigma}_r$, for the voxel is [29]:

$$\hat{\sigma}_r \quad = \quad 1 + \sqrt{\frac{1}{N}\left(\left(1 + \frac{1}{S_n}\right)^2 + \left(\frac{1}{S_n}\right)^2\right)} \tag{4}$$

where S_n is the signal to noise ratio and has a typical value of 10dB. Computing $\sigma_r = 10\log_{10}(\hat{\sigma}_r)$, we obtain the reflectivity standard deviation in decibels (dB). While increasing $\Delta - \alpha_k - \beta_k$ linearly increases the number of samples N collected, the standard deviation of the estimated reflectivity value of the voxel decreases only at the rate of $1/\sqrt{N}$, exemplifying the Cramer-Rao result seen in Section IV.

4.2.3 Root mean-squared error (RMSE) when tracking a storm

Both the number of voxels scanned and the reflectivity standard deviation evaluate system performance only within a single control update interval, Δ. To capture whether per-interval gains accumulate across multiple intervals, we look to simulations tracking a storm (i.e., a region of high reflectivity). We first discuss how we convert the standard deviation of reflectivity to the standard deviation of the location of peak reflectivity (i.e., the location of the storm centroid). We then describe how the standard deviation of the location of the storm centroid affects the RMSE when tracking the centroid.

The reflectivity standard deviation, σ_r, depends, through N in Equation (3), on the scan sector size, $\delta\theta$, and the time spent scanning the sector, $\Delta - \alpha_k - \beta_k$. An increase in σ_r should translate into an increase in the standard deviation of the location of the peak reflectivity, σ_z. As there are many algorithms for detecting peak reflectivity, and the uncertainty associated with the location depends on the algorithm, we adopt a simple approach here and set the value of σ_z along a radial from the radar as,

$$\sigma_z \quad = \quad \frac{\sigma_r D_r}{30dBZ} \tag{5}$$

where D_r is the distance of the object from the radar and 30dBZ is a mid-range reflectivity value. This assumes that uncertainty in the reflectivity estimate translates into an equivalent amount of uncertainty in the location of peak reflectivity.

We use the standard deviation in the location of peak reflectivity in the covariance matrix of the Kalman filter used to track storms, as described in the next section. For meteorological algorithms, it is not sufficient to scan only the storm centroid. Instead, reflectivity data from the surrounding area (i.e., the entire storm cell) is also needed [4]. Hence our experiments will perform tracking based on the location of the storm centroid, but will also scan the surrounding area, corresponding to the expected storm radius.

4.3 Meteorological Tracking Application

In this section we describe a storm tracking application. Let $\mathbf{x}_k$ be the true location of the storm centroid at time k and let $\mathbf{y}_k$ be noisy measurements of the location. A Kalman filter [30] assumes that the true location at time k is a linear function of the true location at time $k-1$ plus Gaussian noise, and that the noisy measurements at time k are a linear function of the true location at time k plus Gaussian noise. I.e.,

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + N[\mathbf{0}, \mathbf{Q}_k] \quad (6)$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + N[\mathbf{0}, \mathbf{R}_k] \quad (7)$$

We now describe our Kalman filter model of the movement of a storm centroid. We use $\mathbf{x}_k = [x^1, x^2, x^3, x^4]$, where x^1 is the true x-location of the storm centroid, x^2 is the true y-location, x^3 is the true x-velocity, and x^4 is the true y-velocity. For the noisy measurements, we use $\mathbf{y}_k = [y^1, y^2]$, where y^1 is the measured x-location and y^2 is the measured y-location. Then,

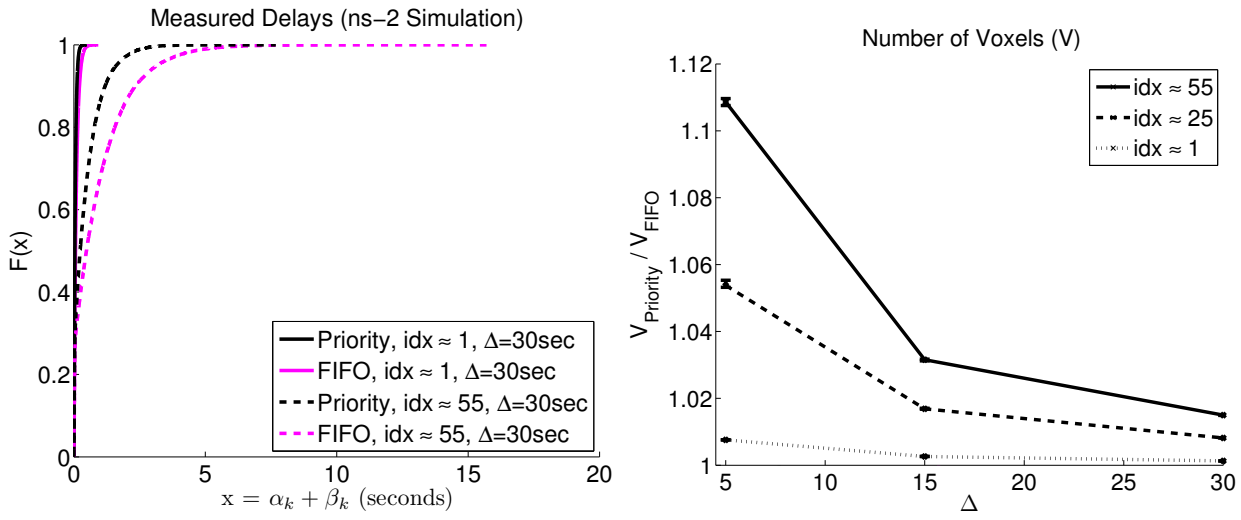
$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \ \mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & q^3 & 0 \\ 0 & 0 & 0 & q^4 \end{bmatrix}, \ \mathbf{R}_k = \begin{bmatrix} r^1 & 0 \\ 0 & r^2 \end{bmatrix}_k$$

We obtain the covariance matrix $\mathbf{Q}$ as follows. First, we assume that the latitude and longitude noises are uncorrelated and set the off-diagonal elements of $\mathbf{Q}$ to zero. We also assume that there is no noise in the latitude and longitude locations. We then compute the noise in the latitude and longitude velocities from 39 existing storm tracks from the National Severe Storms Laboratory courtesy of Kurt Hondl and the WDSS-II software [31]. Each track is a series of $(latitude, longitude)$ coordinates. We first compute the differences in latitude and longitude, and in time, between successive pairs of points. We then fit the differences using Gaussian distributions. Since the length of a latitude degree at 40° latitude equals 111.04 km and the length of a longitude degree at 40° latitude equals 85.39 km, we obtain, in units of km/hour, that the latitude velocity is $\sim N(9.1, 1268)$ and that the longitude velocity is $\sim N(16.7, 836)$. For example, the latitude velocity is on average 9.0 km/hr with one standard deviation of $\sqrt{1268} = 35.6$ km/hr. Working in seconds, we set $q^3 = 0.0001$ and $q^4 = 0.00006$. While the process noise $\mathbf{Q}$ is not a function of time k, the measurement noise $\mathbf{R}_k$ is, as it depends both on the radar scan strategy at time k, and on the delay given by $\alpha_k + \beta_k$. Thus, at time k, we compute σ_z as in Equation (5) and set $r^1 = r^2 = \sigma_z^2$.

We use the Kalman filter parameters just described in the tracking algorithm used in Section V. As the system is not directly observable, it can be difficult to exactly obtain the covariance matrices in practice. Consequently, to parameterize the Kalman filter used to generate the trajectory of the storm centroid, we use the same parameters as the Kalman filter used for tracking, but we perturb the covariance matrices as follows.

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & q^3 & 0 \\ 0 & 0 & 0 & q^4 \end{bmatrix}, \ \mathbf{R}_k = \begin{bmatrix} 5r^1 & 0 \\ 0 & 5r^2 \end{bmatrix}_k$$

We now describe the tracking application. Assume that the radar is located at the origin, that the radar radius is 40km (e.g., as with the CASA radars [24]), and that the radar stops tracking a storm when it exits the radar's footprint. We represent a storm as a circle with a 3km radius based on work by [32] on storm cells which gives 2.83km as "the radius from the cell center within which the intensity is greater than e^{-1} of the cell center intensity"; the initial location of the storm centroid is chosen randomly and subsequent movement is governed by Equation 6. To compute the measured location of the storm centroid, $\mathbf{y}_k$, we use Equation 7, using $\Delta - \alpha_k - \beta_k$ and the procedure described earlier to obtain the parameters for the covariance matrix $\mathbf{R}_k$. To compute the estimated true location $\hat{\mathbf{x}}_k$ from $\mathbf{y}_k$, and to compute the predicted true location $\hat{\mathbf{x}}_{k+1}^-$ and covariance matrix $\mathbf{P}_{k+1}^-$, we use the filtering equations, e.g., see [30]. To compute the area that



(a) (b)

Figure 6: (a) CDFs of the measured $\alpha_k + \beta_k$ delays. (b) Number of times more voxels V scanned under priority scheduling than under FIFO; 95% bootstrap confidence intervals over 10 simulation runs are shown.

contains $\hat{\mathbf{x}}^-_{k+1}$ with 99% confidence, we use its covariance matrix $\mathbf{P}^-_{k+1}$. The 99% confidence area is an ellipse centered at the point $(\hat{x}^1_{k+1}, \hat{x}^2_{k+1})$ whose semi-axes are given by the submatrix $\mathbf{P}^-_{k+1}[1,2;1,2]$. $\hat{x}^1_{k+1}$ and $\hat{x}^2_{k+1}$ are, respectively, the x- and y-locations of the storm centroid and are the first two components of the vector $\hat{\mathbf{x}}_{k+1}$. To account for the storm radius, we expand the confidence ellipse by 3km (since the ellipse gives the area in which the storm centroid will be found 99% of the time, but does not include the storm radius). We compute the radar's next scan sector to be the smallest scan angle that covers the expanded confidence ellipse. The radar then scans this sector for $\Delta - \alpha_{k+1} - \beta_{k+1}$ seconds during the next update interval; the radar scans 360° initially, whenever the true location lies outside of the scanned area, and when $\alpha_k + \beta_k \geq \Delta$.

5 Simulation Results

In this section we use the models described in Section IV to investigate the value of separate handling of sensor control and data traffic in our meteorological application. We first assume that packets may be delayed but are not dropped, and then consider the case where packets may additionally be dropped.

5.1 Delayed Packets

5.1.1 Simulation Set-up

To compute the control-loop delays we use the ns-2 simulator [28]. We set the queue size to be large enough that no packets are dropped. The data delay is the delay incurred by the last packet that is processed at the bottleneck node by the start of each update interval. The corresponding control delay is the delay of the associated sensor control packet for that update interval. Based on experimental results from the CASA radar testbed, we use $\frac{\lambda_c}{\lambda_c+\lambda_d} = 0.0005$ and $\Delta = 30$ sec, setting $\lambda_c = \frac{1}{30}$ pkts/sec and $\lambda_d = \frac{2000}{30}$ pkts/sec. We also use $\Delta = \{5, 15\}$ sec, setting $\lambda_c = \frac{1}{\Delta}$ pkts/sec while leaving λ_d unchanged; such Δ's are feasible for a

phased array radar [33]. For the "other" traffic, we set $\lambda_o = \frac{2000}{30}$ pkts/sec, with $\lambda_1 = p\lambda_o$ and $\lambda_2 = (1-p)\lambda_o$, for $p = \{0.5, 0.2, 0.05\}$. We set the transition rates r_1 and r_2 for the Markov modulated Poisson process to each be 1.0sec on average. Computing the index of dispersion (idx), see [27],

$$idx = 1 + \frac{2(\lambda_1 - \lambda_2)^2 r_1 r_2}{(r_1 + r_2)^2(\lambda_1 r_2 + \lambda_2 r_1)} \quad (8)$$

shows that our parameters consider $idx \approx \{1, 25, 55\}$. $idx = 1$ corresponds to a Poisson process while larger values correspond to increased traffic burstiness. Finally, since $\lambda_c + \lambda_d + \lambda_o \approx 133.37$pkts/sec we set $\mu = 148.5$pkts/sec achieving a load of about 0.90. Even with $\frac{(\lambda_c+\lambda_d+\lambda_o)}{\mu} < 1$, however, the bursty "other" traffic introduces temporary overload conditions. Using this network model, for each parameter setting, we perform 10 simulation runs, of 100,000 sec each. This gives, e.g., 20,000 update intervals per run for $\Delta = 5$ sec. For each run we obtain a time-varying series of $\alpha_k + \beta_k$ delays. Figure 6(a) shows the delay distributions for $\Delta = 30$ sec; we plot the data from all runs to obtain the CDF for each scheduling mechanism and idx pair. Although not shown, we also find that on average, the $\alpha_k + \beta_k$ delay for priority scheduling is about half that of FIFO, regardless of Δ. We expect that λ_o will increase the $\alpha_k + \beta_k$ delay. Recall from Section III, however, that prioritizing sensor control, has a percentage gain of at most $\beta_k/(\Delta - \alpha_k - \beta_k)$ more time over FIFO. Consequently the relative performance gain of priority over FIFO should be bounded regardless of λ_o.

5.1.2 Number of Voxels Scanned

Using Equation 6 and the delays generated from ns-2, Figure 6(b) shows the number of times more voxels scanned under priority scheduling than under FIFO. Figure 6(b) shows that as Δ decreases and burstiness increases, the benefits of prioritizing increase: for $\Delta = 5$ sec and $idx = 55$, priority scheduling scans about 1.15 times as many voxels as FIFO.

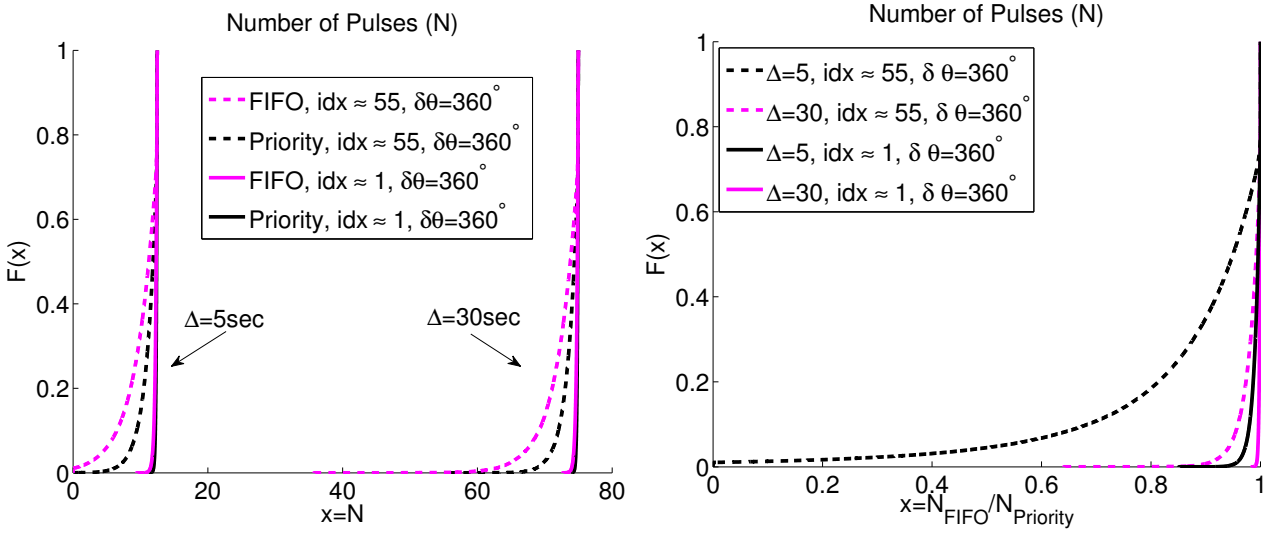
5.1.3 Reflectivity Standard Deviation

Using the delays obtained from NS-2, we first use Equation 3 to compute a corresponding time-varying series of Ns, given a fixed scan angle of $\delta\theta = 360°$. The empirical CDFs for N are shown in Figure 7(a), using the data from all 10 runs for each CDF: we see that FIFO and priority each achieve about 6 times as many pulses for $\Delta = 30$ sec as for $\Delta = 5$ sec and that the total number of pulses gained over FIFO from using priority is independent of Δ. Figure 7(b) plots the ratio of each FIFO CDF in Figure 7(a) with that of the corresponding priority CDF. Figure 7(b) shows that for $idx = 1$ or $\Delta = 30$ sec, FIFO achieves at least 90% as many pulses as priority, more than 95% of the time. Only for $idx = 55$ and $\Delta = 5$ sec, (i.e., very bursty traffic and a small update interval), does FIFO perform significantly worse (i.e., achieving about 80% as many pulses as priority about 80% of the time).

Now, using the time-varying series of the number of pulses N and Equation 4 we compute the corresponding series of reflectivity standard deviation σ_r values. Figure 7(c) plots the ratio of each priority σ_r CDF with that of the corresponding FIFO σ_r CDF, again using the data from all runs for each CDF. Due to the $1/\sqrt{N}$ behavior in Equation 4, Figure 7(c) shows that the gains in N from prioritizing sensor control are diminished: e.g., now for $idx = 55$ and $\Delta = 5$ sec, priority scheduling has at least 90% as much uncertainty as FIFO about 90% of the time. To summarize, Figures 6 and 7 show that while the $\alpha_k + \beta_k$ delay for priority scheduling is about half that of FIFO, these gains do not translate into equivalent percentage gains in N or σ_r, and that the gains are greater for smaller Δs and more overloaded links.

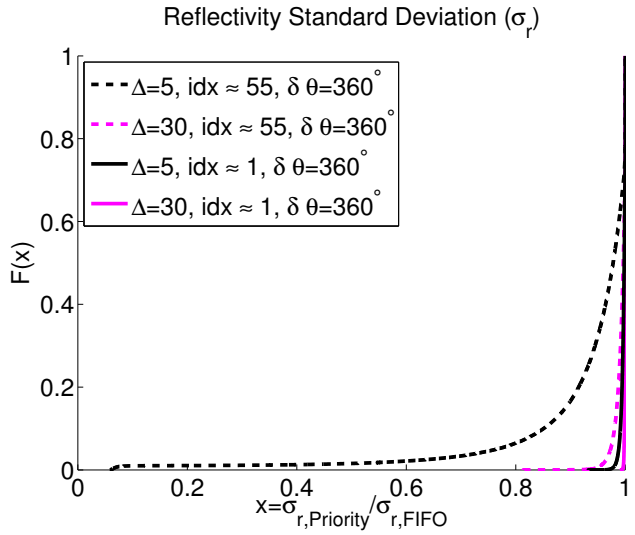
5.1.4 RMSE when tracking a storm

To perform this simulation we input the time-varying series of delays for each run, obtained from NS-2, directly into the tracking application in Section IV-D. Figure 8(a) shows a sample storm trajectory. At each



(a)

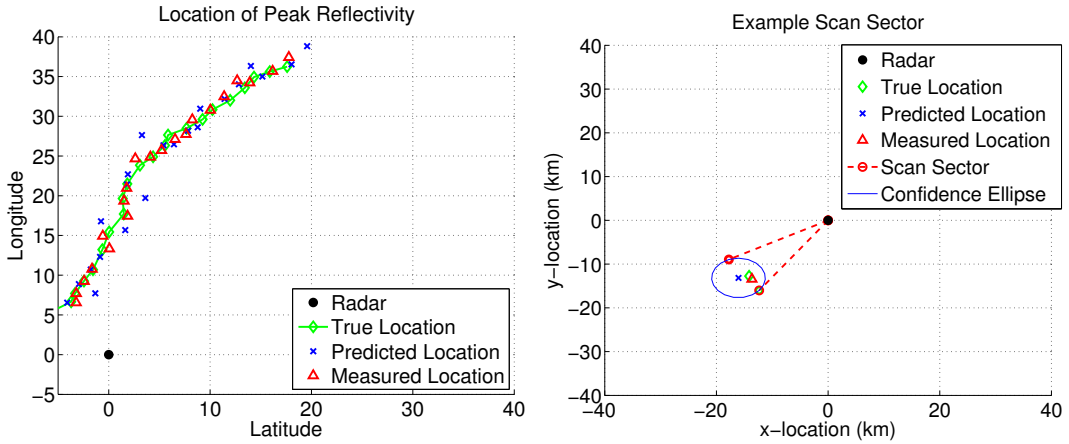
(b)



(c)

Figure 7: CDFs of (a) the number of pulses, N, (b) the normalized number of pulses, and (c) the normalized reflectivity standard deviation.

control update we compute the next scan sector, see Figure 8(b). Figure 9 shows the RMSE under FIFO relative to the RMSE under priority. The RMSE is computed over the differences between the true, $\mathbf{x}_k$, and estimated true, $\hat{\mathbf{x}}_k$, locations of the storm centroid. As Δ decreases and burstiness increases, Figure 9 shows that the benefits of prioritizing increase: for $\Delta = 5$ sec and $idx = 55$, FIFO has a median of about 1.06 times the RMSE of priority scheduling. We also see some outliers: e.g., for $\Delta = 5$ sec and $idx = 55$, when FIFO has about 4.17 times the RMSE of priority. For this outlier run, the average scan angle was about 56° while for the other 9 runs, the average scan angle ranged from 36° to 46°. Hence, once the scan angle (and consequently the measurement noise) is sufficiently large, the Kalman filter less effectively filters out



(a) (b)

Figure 8: (a) Example storm trajectory. (b) Scan sector computed by central control to cover 99% confidence ellipse. We show the true ($\mathbf{x}_k$), predicted ($\hat{\mathbf{x}}_k^-$), and observed ($\mathbf{y}_k$) locations of the storm centroid at time k.

the noise when estimating the true location. Thus, for tracking, it is possible for per-interval performance gains or losses to accumulate across multiple update intervals, unlike with the voxel and reflectivity standard deviation metrics.

5.2 Dropped Packets

5.2.1 Simulation Set-up

To model packet drops, we compare the worst-case (all sensor control dropped) and best-case (no sensor control dropped) scenarios. Suppose that the sensor control packets always tell the radar to scan 45° and two elevation angles within that sector (i.e., the smallest sector that would be scanned by the CASA radars, and correspondingly, the highest quality data that would be obtained). Consequently N_{45} samples per voxel would be collected in the specified 45° sector. Now suppose that as a result of overload, a fraction p_{loss} of packets are lost. For both FIFO and priority scheduling, a fraction of the data samples will be lost. Additionally for FIFO, however, since we assume the worst case, all sensor control packets will be lost, and the radars will always use the default strategy of scanning 360° and all six elevation angles (i.e., the largest volume of space that the CASA radars would scan), collecting N_{360} samples per voxel. Hence FIFO will have $N' = N_{360} \times (1 - p_{loss})$ data samples per scanned voxel reach the control center while priority will have $N' = N_{45} \times (1 - p_{loss})$ data samples per scanned voxel reach the control center. From [29] we use $N_{360} = 750$ and $N_{45} = N_{360} \times 3 \times 8 = 18000$.

5.2.2 Reflectivity Standard Deviation

Figure 10 plots the reflectivity standard deviation when N' samples reach the control center (i.e., there is loss), normalized by the reflectivity standard deviation when N_{45} samples reach the control center (i.e., there is no loss). We assume that the network delivers packets at its capacity and that traffic beyond network capacity is lost. Hence, $N' = N_{45}$ for both FIFO and priority when arrivals are less than capacity; when arrivals exceed capacity, $N' = N_{360} \times (1 - p_{loss})$ for FIFO and $N' = N_{45} \times (1 - p_{loss})$ for priority. Figure 10 shows that as the system goes into overload, σ_r degrades gracefully for priority scheduling, as the offered

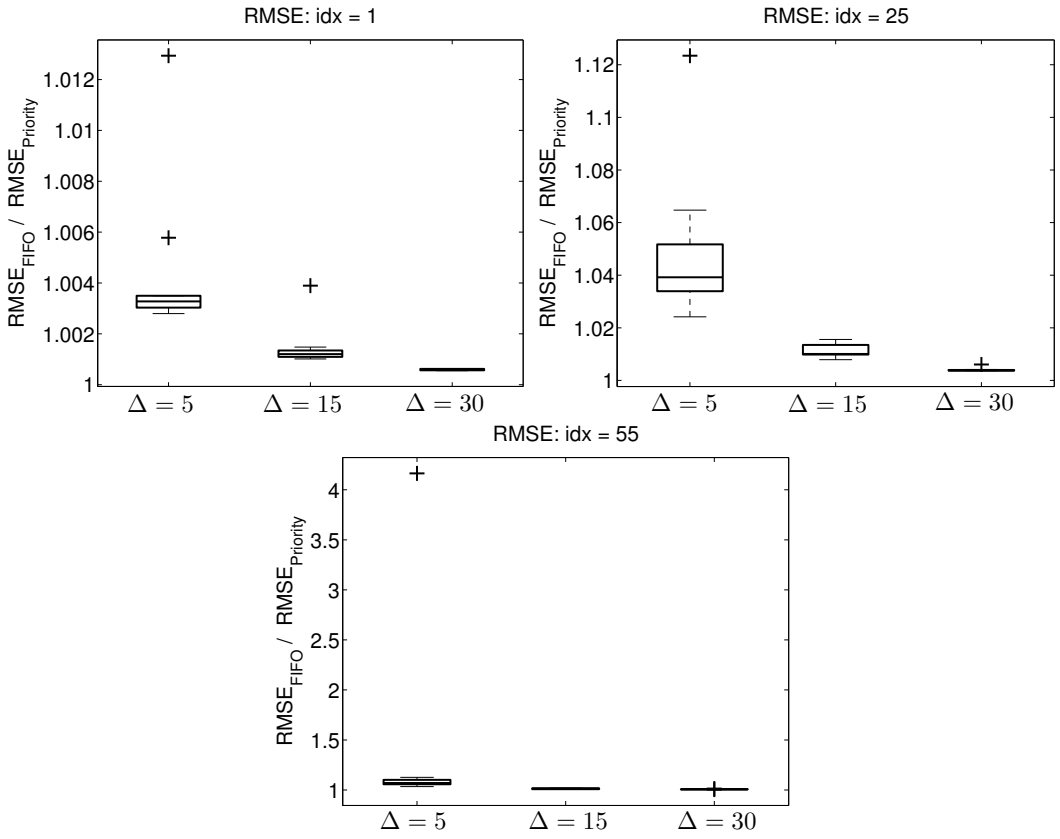


Figure 9: RMSE from tracking application. Boxplots are over 10 runs. Boxes show the median and first and third quartiles; +'s indicate outliers, i.e., data values more than 1.5 times greater (smaller) than the third (first) quartile.

load increases (i.e., the fraction of lost data samples increases). These results show that the sensing system is robust to network overload conditions, and suggest that in times of congestion, it is preferable for the end-to-end data transfer protocol to ignore lost data samples, rather than adopting an ARQ protocol for retransmission, that would then increase the data delays to RTT timescales.

6 Summary

In this paper, we have examined the value of separating control and data in a networked remote-sensing application. Our results show that separate handling of sensor control and data decreases the round-trip control-loop delay, and consequently increases either (i) the number of voxels V scanned or (ii) the number of samples of reflectivity N obtained per voxel scanned. In the former case, the utility increases linearly with the number of scanned voxels. In the latter case, since sensing accuracy improves only as a function of $\sqrt{N}$, the gain in accuracy for the reflectivity estimate per voxel as N increases is relatively *small* except when prioritizing sensor control increases N significantly (e.g., when sensor control packets suffer severe delays). Also, in this latter case, because accuracy degrades as a function of $\sqrt{N}$, the system can still perform well during times of congestion, but only when sensor control packets receive priority and remain unaffected by data overload. For the storm tracking application, we find that under severe congestion, performance losses when not prioritizing sensor control can actually accumulate over time.

This work can be extended in several directions. For our meteorological sensing application, the uncertainty in

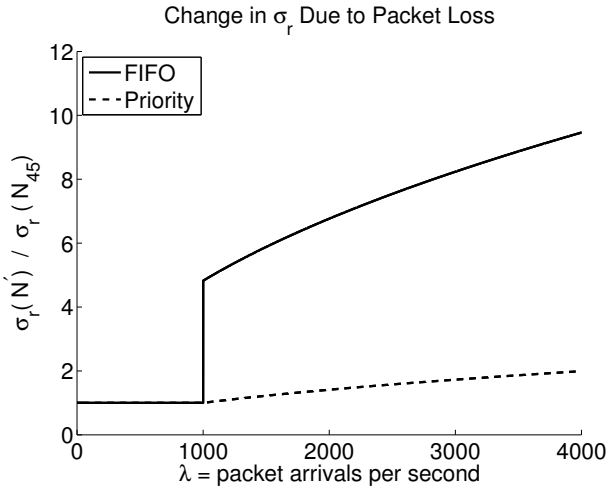


Figure 10: Packet loss under different arrival rates. Capacity is 1000 pkts/sec.

the sensed values and the performance of a tracking application are relatively insensitive to change in round-trip control-loop delay through the network. It would be interesting to identify other sensing applications where such behavior is, or is not, the case. We have assumed that each sensed value is equally valuable. In practice, sensed data from areas of interest, such as areas of high reflectivity in the meteorological application, are likely to be more important to a sensing application, e.g., see [34]. These data values could be handled at higher priority, while other data values can be transmitted at lower priority or discarded in times of congestion. The more general challenge is to define an overall architecture for pushing application-level performance considerations down into the lower layers of the system stack in an application-independent manner.

Acknowledgments

The authors thank Brian Donovan and David McLaughlin for enlightening discussions about radar meteorology, and Bruno Ribeiro for comments on a paper draft. This research was supported in part by the National Science Foundation under the Engineering Research Centers Program, award number EEC-0313747, and via an International Research in Engineering Education supplement to award number EEC-0313747. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

References

[1] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *SenSys*, 2004.

[2] C. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," in *SenSys*, 2004.

[3] C.-Y. Wan, S. Eisenman, A. Campbel, and J. Crowcroft, "Siphon: Overload traffic management using multi-radio virtual sinks in sensor networks," in *SenSys*, 2005.

[4] M. Zink, D. Westbrook, S. Abdallah, B. Horling, V. Lakamraju, E. Lyons, V. Manfredi, J. Kurose, and K. Hondl, "Meteorological command and control: An end-to-end architecture for a hazardous weather detection sensor network," in *ACM Mobisys Workshop on End-end Sense-and-Response Systems*, 2005.

[5] D. McLaughlin, V. Chandrasekar, K. Droegemeier, S. Frasier, J. Kurose, F. Junyent, B. Philips, S. Cruz-Pol, and J. Colom, "Distributed collaborative adaptive sensing (DCAS) for improved detection, understanding, and predicting of atmospheric hazards," in *Proc. American Meteorological Society Annual Meeting*, 2005.

[6] B. Donovan, A. Hopf, J. M. Trabal, B. J. Roberts, D. J. McLaughlin, and J. Kurose, "Off-the-grid radar networks for quantitative precipitation estimation," in *Proc. European Radar Conference*, 2006.

[7] V. Manfredi and J. Kurose, "Scan strategies for adaptive meteorological radars," in *Advances in Neural Information Processing Systems 21*, 2007.

[8] S. B. Fredj, T. Bonald, A. Proutiere, G. Regnie, and J. Roberts, "Statistical bandwidth sharing: A study of congestion at flow level," in *Proc. ACM Sigcomm*, August 2001.

[9] I. E. Consortium, "Signaling system 7 (ss7)," http://www.iec.org/online/tutorials/ss7/, Tech. Rep., 2006.

[10] U. Black, *ATM: Foundation for Broadband Networks*. Prentice Hall, 1995.

[11] L. Kalampoukas, A. Varma, and K. Ramakrishnan, "Improving TCP throughput over two-way asymmetric links: Analysis and solutions," in *Proc. ACM Sigmetrics*, 1998, pp. 78–89.

[12] H. Balakrishnan, V. Padmanaban, G. Fairhurst, and M. Sooritabandara, "TCP performance implications of network path asymmetry," Request for Comment, Tech. Rep. RFC 3449, Dec. 2002.

[13] P. Kyasanur, J. Padhye, and V. Bahl, "On the efficacy of separating control and data into different frequency bands," in *Proc. Broadnets*, 2005.

[14] S. Bhatnagar, B. Deb, and B. Nath, "Service differentiation in sensor networks," in *Fourth International Symposium on Wireless Personal Multimedia Communications*, 2001.

[15] K. Karenos, V. Kalogeraki, and S. V. Krishnamurthy, "A rate control framework for supporting multiple classes of traffic in sensor networks," in *26th IEEE International Real-Time Systems Symposium*, 2005.

[16] W. L. Tan, O. Yue, and W. C. Lau, "Performance evaluation of differentiated services mechanisms over wireless sensor networks," in *Vehicular Technology Conference*, 2006.

[17] R. Kumar, R. Crepaldi, H. Rowaihy, A. F. H. III, G. Cao, M. Zorzi, and T. F. L. Porta, "Mitigating performance degradation in congested sensor networks," *IEEE Transactions on Mobile Computing*, vol. 7:6, 2008.

[18] P. F. Hokayem and C. T. Abdallah, "Inherent issues in networked control systems: A survey," in *Proc. American Control Conference*, 2004.

[19] J. Nilsson, "Real-time control systems with delays," Ph.D. dissertation, Dept. of Automatic Control, Lund Institute of Technology, 1998.

[20] G. C. Walsh and H. Ye, "Scheduling of networked control systems," *IEEE Control Systems Magazine*, 2001.

[21] M. Lemmon, Q. Ling, and Y. Sun, "Overload management in sensor-actuator networks used for spatially-distributed control systems," in *SenSys*, 2003.

[22] L. Kleinrock, *Queueing Systems Volume II*. Wiley Interscience, 1976.

[23] G. Casella and R. Berger, *Statistical Inference*, 2002.

[24] M. Zink, E. Lyons, D. Westbrook, J. Kurose, and D. Pepyne, "Closed-loop architecture for distributed collaborative adaptive sensing of the atmosphere: Meteorological command and control," *International Journal of Sensor Networks*, to appear.

[25] Center for Collaborative Adaptive Sensing of the Atmosphere, "http://www.casa.umass.edu," 2006.

[26] X. Pallot and L. Miller, "Implementing message priority policies over and 802.11 based mobile ad hoc network," in *Milcom*, 2001.

[27] H. Heffes and D. Lucantoni, "A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance," *IEEE Journal on Selected Areas in Communication*, vol. 4:6, 1986.

[28] "The network simulator ns-2," in *http://www.isi.edu/nsnam*.

[29] B. Donovan and D. J. McLaughlin, "Improved radar sensitivity through limited sector scanning: The DCAS approach," in *Proc. AMS Radar Meteorology*, 2005.

[30] G. Welch and G. Bishop, "An introduction to the Kalman filter," U of North Carolina at Chapel Hill, Dept. of Computer Science, Tech. Rep. TR95-041, 1995.

[31] K. Hondl, "Capabilities and components of the warning decision and support system - integrated information (WDSS-II)," in *Proc. American Meteorological Society Annual Meeting*, 2003.

[32] I. Rodrigues-Iturbe and P. Eagleson, "Mathematical models of rainstorm events in space and time," *Water Resources Research*, vol. 23:1, pp. 181–190, 1987.

[33] M. Sanchez-Barbetty and R. Jackson, "Architecture for low cost electronically steered phased arrays," in *IEEE MTT International Microwave Symposium*, 2008.

[34] M. Li, T. Yan, D. Ganesan, E. Lyons, P. Shenoy, A. Venkataramani, and M. Zink, "Multi-user data sharing in radar sensor networks," in *SenSys*, 2007.