Localized Algorithm for Aggregate Fairness in Wireless Sensor Networks

Shigang Chen
sgchen@cise.ufl.edu

Zhan Zhang
zzhan@cise.ufl.edu

Department of Computer & Information Science & Engineering
University of Florida
Gainesville, FL 32611

ABSTRACT

For data-collection applications in sensor networks, it is important to ensure all data sources have equal (or weighted) access to network bandwidth so that the base stations receive a complete picture about the monitored area. We point out the fairness problem in the current design of sensor networks, which may cause extremely biased bandwidth allocations. It is a challenge to design a fully distributed fairness solution due to the lack of global knowledge about the distribution of data sources and their routing paths. This paper proposes a new *aggregate fairness* model and a localized algorithm (called *AFA*) that implements the model. AFA is designed to work with any routing protocol. In particular, it allows the packets from a data source to follow an arbitrary set of forwarding paths to the base stations. This flexibility makes it considerably harder to allocate bandwidth fairly among different data sources. AFA solves the problem with only localized operations at the sensors. It is easy to implement, which is an attractive property for sensor networks. Moreover, the algorithm automatically adjusts a sensor's forwarding rate to avoid packet drops due to downstream congestion, which helps improve energy efficiency. We perform extensive simulations, demonstrating that the proposed algorithm can effectively improve end-to-end fairness.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms

Keywords

Sensor networks, aggregate fairness, congestion avoidance, distributed aggregate fairness algorithm (AFA)

1. INTRODUCTION

A wireless sensor network consists of battery-powered sensing devices that transmit their observation data to a set of base stations. In most cases, sensors are stationary after deployment. Due to limited transmission range, sensors far away from the base stations deliver their data through multihop communication.

Congestion control is of great importance in sensor networks [1, 2, 3, 4]. Although some applications may only query aggregate information such as the max/min/mean/ medium/avg of certain measurements [5, 6], other applications are interested in macroscopic imaging of certain features of the monitored field, which carries much more information than simple aggregation. Such location-specific information is often crucial in habitat monitoring, seismic structure response, ecosystem evaluation, and natural disaster monitoring. Many tradeoffs can be made in the network. Compression or partial-aggregation techniques trade information imprecision for less traffic volume. Periodic sleeping trades bandwidth reduction for energy saving [7]. However, there is a limiting factor. When a wired network scales up, more cables can be added to increase the network capacity. When a sensor network scales up with more sensors deployed in a larger area, the traffic volume increases but the channel capacity around the bottlenecks cannot be increased easily, given the low-cost, low-energy requirements. Therefore, we cannot assume the aggregate traffic volume from all data sources is always below the delivery capacity of the network. Congestion control must play a significant role in a sensor network architecture. It is important for *elastic* applications that are designed to work with varied reporting rates. For example, a monitoring application may prefer a reporting rate of 20 samples per unit of time, but a smaller rate is still acceptable if the desirable rate cannot be achieved due to congestion. When that happens, uniform (or weighted) reduction of reporting rates from all competing sensors is superior to the case where some (close-by) sensors send data at full rate while others are starved.

Consider a sensor network that typically operates under light load but may suddenly be active in response to certain important events such as fire outbreak or earthquake. The sudden surge of data from hundreds or even thousands of sensors must be delivered to a small number of base stations, which may cause congestion, especially near the base stations [2]. To handle this situation, we must solve two important problems. The first problem, *rate reduction*, is to reduce the number of packets produced by data sources to a level that can be transported by the network. The second problem, *fairness control*, is to ensure that all data sources have equal or weighted access to the end-to-end network bandwidth. Much effort on congestion control in traditional networks went to solve the second problem [8, 9, 10, 11], under the assumptions of fixed link capacities, single-path routing, and/or timely, reliable, end-to-end feedback, which make the solutions not immediately applicable in sensor networks. Recently there are several pioneering studies on congestion control in sensor networks [2, 3, 4]. However, these works either do not consider the fairness issue or have very restrictive assumptions on the routing structure, which limits their scope of applicability. As we will elaborate shortly, no prior work provides a practical distributed solution to the fairness problem in a data-collection network where packets from a data source are routed individually and may follow numerous different paths to the base stations.

In this paper, we propose a new *aggregate fairness* model, which defines end-to-end fairness in an aggregate manner, suitable for multipath multihop wireless sensor networks. A distributed aggregate fairness algorithm (AFA) is then proposed to implement the model. AFA does not maintain any per-flow information or global state. Each sensor performs localized operations, yet the collective outcome ensures that data sources sharing the same downstream bottleneck have equal access to network bandwidth. In addition, AFA automatically adjusts each sensor's forwarding rate to avoid packet drops due to downstream congestion. The simulation results demonstrate that the proposed algorithm effectively improves end-to-end fairness.

The rest of the paper is organized as follows. Section 2 describes the fairness problem and discusses the limitations of the prior works. Section 3 proposes a new aggregate fairness model. Section 4 presents a distributed algorithm for updating aggregate flow weights. Section 5 proposes a distributed algorithm that achieves aggregate fairness. Section 6 presents simulation results. Section 7 draws the conclusion.

2. FAIRNESS — AN IMPORTANT PROBLEM IN WIRELESS SENSOR NETWORKS

We describe the problem of (weighted) fairness in sensor networks. We discuss two major forms of congestion and how the MAC-layer local fairness exacerbates the congestion problem. We then explain the challenge of end-to-end fairness and outline the limitations of the existing solutions.

2.1 Fairness and Weighted Fairness

Data collection from a field (or structure) is one of the predominant applications of sensor networks [12, 13, 14, 15, 16, 17]. A basic requirement is that the sink should have uninterrupted access to data from all parts of the field. Although there are high-end sensors, many commonly-used ones (such as mica and mica2) have only 10-40 kBaud radio [18]. In addition, long sleep periods may be introduced for saving power in order to fulfill a lifetime requirement, which further reduces the available bandwidth. In case of congestion, it is well known that a sensor close to a base station tends to achieve much higher throughput than a sensor far away. If information around the base station is not more important than the information far in the field, the bias towards close-by sensors is undesirable. Ensuring fair access to network bandwidth is critical to keeping the reporting channel open for distant sensors, such that the base stations receive the complete view of the monitored area.

Some sensors may have more important data and thus require higher packet rates than others. In this case, weighted fairness instead of equality fairness is more suitable. It makes bandwidth allocation biased towards important data sources with higher weights.

2.2 Radio Collision, Buffer Congestion, and "Harmful" Local Fairness

Radio collision and buffer overflow are two main forms of congestion in a wireless sensor network. Solutions against collision include CSMA, TDMA, CDMA, and other multiple access schemes. Using a larger contention window can reduce the collision frequency in CSMA. Several CSMA-based MAC protocols are proposed for sensor networks, such as B-MAC [19] and S-MAC [7]. But reducing the level of radio collision does not mean solving the congestion problem. Wireless medium access brings a new congestion scenario that is not present in a wired network. The local fairness achieved by MAC-layer protocols such as CSMA directly contributes to buffer overflow. Consider the topology in Figure 1, where data sources y, z, and w send packets to a sink via x. In all figures, white nodes represent data sources (sensors actively producing data), black nodes represent forwarding sensors, and grey nodes represent sinks. If y, z, w, and x each obtain a fair share of the channel capacity, x will receive three packets for every packet it sends out, which is confirmed by the ns2 simulation results plotted in Figure 1. The packet queue at x will build up and eventually overflow. Therefore, it is not sufficient for the data sources to slow down to a level that does not cause serious radio collision. They must slow down further such that x is able to obtain a larger portion of bandwidth that matches the combined rate of y, z and w.

2.3 Challenges of End-to-End Fairness

The sequence of packets produced by a data source constitutes a *flow*. Many papers studied MAC-layer fairness among *one-hop flows* within a neighborhood [20, 21, 22]; we study network-layer fairness among *end-to-end flows*. The packets of a flow may be routed along different paths to a sink, which consists of one or multiple base stations. When all routing paths from the data source to the sink are congested, the source has to reduce the rate at which it produces packets, because the excess packets will be dropped anyway. The end-to-end fairness problem arises. What are the maximum rates at which individual sensors can produce data without causing network congestion and unfairness among the peers?

Congestion and fairness are two related but different problems. Resolving congestion does not guarantee fairness. In the topology of Figure 2, suppose a congestion control mechanism such as backpressure [23, 2] limits the forwarding rate of a sensor x to six packets per second in order to avoid downstream congestion. The combined incoming rate to x should be equal to the forwarding rate over the long run. Therefore, the upstream neighbors, y and z, can each send x three packets per second on average. Further upstream, u, v, and w can only send one packet per second. Consequently, y is able to generate packets at three times the average rate of u, v, and w. This analysis is confirmed by the ns2 simulation results shown in the figure. Therefore, for the purpose of fairness, it is insufficient that the combined rate of y and z does not exceed the forwarding rate of x. We must carefully distribute the available bandwidth between them, such that each upstream data source receives a fair share.

A real sensor network has far more complex topology than the one in Figure 2. The packets of a flow are routed independently and an intermediate sensor may forward a packet to an arbitrary neighbor that is closer to the sink. Therefore, a flow may follow numerous forwarding paths to reach the sink, as illustrated in Figure 3, where we assume two base stations are externally connected and any packet may be routed to either base station. With multipath routing, the paths of all flows may form a complex DAG (directed acyclic graph) instead of a tree. Maintaining a DAG routing structure from the sources to a sink is not necessarily harder than main-

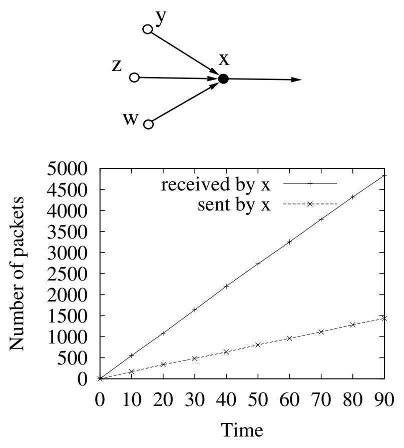

Figure 1: Local fairness directly contributes to buffer congestion in a sensor network.

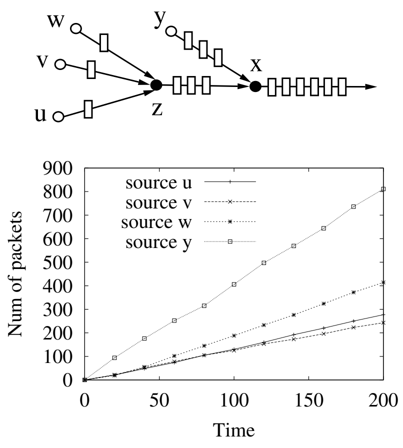

Figure 2: Local fairness among upstream neighbors causes end-to-end unfairness.

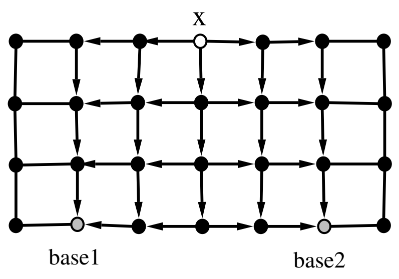

Figure 3: The sequence of packets produced by x constitutes a flow, which may be routed on numerous paths to the sink (the two base stations).

taining a tree. Geographic routing [24, 25, 26, 27] naturally creates a DAG, where a progress threshold may be used to select the candidates for the next hop [28]. Beacon broadcasts from the base stations can establish a DAG with reversed broadcast paths. A DAG is more resilient than a tree, which will be broken if any node or link fails. Consequently, even though BLess, Surge and mh6 [18] build a routing tree, they have an underlying DAG where every node has multiple candidate downstream neighbors, from which one is chosen to forward packets.

Achieving fairness in a multipath routing network is considerably harder than in a tree-routing network. The number of possible paths for a flow can be exponential in the distance from the source to the sink. At a forwarding node, the bandwidth should no longer be evenly divided among the passing flows because some flows may receive bandwidth from other paths while other flows may not.

2.4 Limitations of Existing Approaches

CODA [2] provides a comprehensive discussion on congestion control and proposes an open-loop *hop-by-hop backpressure mechanism* and a close-loop *multi-source regulation mechanism.* Fusion [4] combines three congestion mitigation techniques, *hop-by-hop flow control*, *rate limiting*, and *prioritized MAC.* These schemes do not consider the fairness issue, except for the rate limiting technique [4] that requires a tree routing structure to work correctly. Another tree-based fairness scheme is proposed in [3]. Each sensor learns the number of upstream data sources in the subtree rooted at itself. It measures the maximum downstream forwarding rate and computes per-source fair rate, which is propagated upstream such that the data sources do not send packets beyond this rate.

Other related works are surveyed below. Fairness at the MAC layer is studied in [20, 29, 30, 21, 22]. The TAP fairness in multi-rate wireless backhaul networks is investigated in [31]. In ESRT [32], the sink enforces a common reporting rate for all data sources. To relieve congestion, the reporting rate has to be set conservatively according to the worst hotspot in the network. Proportionally-fair congestion control in FDMA/CDMA networks is studied in [33], under the assumptions that each flow has a single routing path and per-flow state information is maintained at intermediate nodes, which is undesirable for resource-constrained sensor networks. The maxmin fairness among one-hop flows is studied in [34]. Maximizing the minimal flow rate in a routing tree is studied in [35]. The rate-control mechanism in [36] achieves fairness only for single-path routing. Under the assumption that each flow has one path, Kleinberg, Rabani, and Tardos [37] propose a centralized heuristic algorithm for route selection in order to achieve the best maxmin fairness.

In summary, some prior works [2, 28] consider multipath routing, but do not address fairness. Other works [34, 35, 4, 3] assume single-path routing, which simplifies the fairness problem. This paper investigates the general case where no specific restrictions are placed on routing. The proposed fairness solution will work with any routing protocol.

3. AGGREGATE FAIRNESS

In this section, we first discuss the basic idea that we will follow to achieve fairness. We then define the network model and notations. Finally, we give an aggregate fairness model.

3.1 Basic Idea

The basic idea is for a forwarding node to estimate the number of flows coming from each neighbor and allocate bandwidth proportional to that number. For example, in Figure 2, x should allocate 75% of its bandwidth to z and 25% to y.

Because packets of a flow may be routed on many paths (Figure 3), when the flow passes a link, chances are only a fraction of the flow does so. The sum of the flow fractions passing a link (or node), which may not be an integer, is called *aggregate flow number* or *aggregate number* for brevity. We want to emphasize that the aggregate number of a link is not proportional to the number of packets passing through the link. Consider flow X that generates 10 packets per second and flow Y that generates 2 packets per second. If a link receives 2 packets from flow X and 1 packet from flow Y per second, the aggregate number of the link is $\frac{2}{10} + \frac{1}{2} = 0.7$. On the other hand, if the link receives 1 packet from flow X and 2 packets from flow Y per second, the aggregate number is $\frac{1}{10} + \frac{2}{2} = 1.1$.

No forwarding sensor is allowed to maintain per-flow state. Therefore, a sensor does not know the exact fraction of a flow that it carries. Yet the sensor must somehow learn the sum of the flow fractions passing through it. Moreover, some data sources may be more important than others. To characterize such difference, weights are assigned to the data sources. A flow from a data source with a larger weight is expected to acquire a larger share of network bandwidth. The sum of weighted flow fractions passing a link (or node) is called *aggregate flow weight*, which will be precisely defined shortly. If we have a way to estimate the aggregate flow weights of all links (nodes), we can enforce rate limits on the links proportional to the aggregate weights, which will achieve fairness. We will support this argument by proof and simulations.

3.2 Network Model and Notations

We study data-collection sensor networks where the set of data sources is static or changes gradually. Data packets are sent from sensors to base stations. Although not all sensors are data sources, all of them will participate in relaying packets towards the base stations. Assume that the base stations are connected via an external network to a data collection center. It makes no difference which particular base station a packet is delivered to. Sensors are statically located after deployment. We do not consider mobile sensors that form a dynamic ad-hoc network.

The media contention protocol is CSMA/CA. There is a wireless communication "link" from one sensor to another if the latter can correctly receive the former's signal. According to the protocol of CSMA/CA, only symmetric links are used for sending data even though there may exist asymmetric links. We assume all transceivers operate at a single transmission rate, which is reasonable due to the cheap design requirement for inexpensive one-time sensors that are used in large quantities.

Let N be the set of sensors. Let D_i be downstream neighbors of i, which are the next hops on the routing paths from i to the base stations. $\forall j \in D_i$, (i, j) is called a downstream link of i. Let U_i be upstream neighbors, which uses i as a next hop on their routing paths to the base stations. $\forall k \in U_i$, (k, i) is called an upstream link of i. If i is a downstream neighbor of k, then k must be an upstream neighbor of i. D_i and U_i are determined by a routing protocol. For example, if geographic routing is used [24, 25, 26, 27], D_i consists of all neighbors that are closer to the nearest base station (by a certain threshold [28]). It is not true that increasing the size of D_i will always improve throughput. When the nodes in D_i forward packets, they compete with each other for media access, which can cause collisions. A compromise is to limit the size of D_i and pick those candidate downstream neighbors that are far from each other. Let E be the set of all routing links, i.e., $E = \{(k, i) \mid i \in N, k \in U_i\}$.

A flow s is the sequence of data packets generated from a data source s in N. The data rate of flow s is denoted as $d(s)$, and the weight is denoted as $w(s)$, which indicates the importance of this flow. A larger weight means the flow is more important and deserves a proportionally higher rate. Let $r_i(s)$ be the rate at which the packets of flow s pass through sensor i. As a special case,

$$r_s(s) = d(s) \tag{1}$$

Let $r_{k,i}(s)$ be the rate at which the packets of flow s pass through link (k, i). The total rates at which the packets of all flows pass through sensor i and link (k, i) are denoted as r_i and $r_{k,i}$, respectively.

$$r_i = \sum_{s \in N} r_i(s)$$
$$r_{k,i} = \sum_{s \in N} r_{k,i}(s)$$

A rate assignment $\{r_i(s),\ \forall i, s \in N;\ r_{k,i}(s),\ \forall s \in N, \forall (k, i) \in E\}$ is feasible if the following constraints are satisfied. The first two are flow conservation constraints, and the third is the capacity constraint.

1. $r_i(s) = \sum_{k \in U_i} r_{k,i}(s), \forall i, s \in N, i \neq s$,
2. $r_i(s) = \sum_{j \in D_i} r_{i,j}(s), \forall i, s \in N$, and
3. $r_i \leq b_i$, $\forall i \in N$, where b_i is the maximum rate at which i can forward packets to its downstream neighbors.

3.3 Aggregate Fairness

The weighted fraction of a flow s that passes sensor i is defined as

$$f_i(s) = \frac{r_i(s) \times w(s)}{d(s)} \tag{2}$$

The weighted fraction of a flow s that passes link (k, i) is defined as

$$f_{k,i}(s) = \frac{r_{k,i}(s) \times w(s)}{d(s)} \tag{3}$$

Aggregate Flow Weight: The sum of the weighted fractions of all flows that pass sensor i is called the aggregate flow weight of i. It is denoted as F_i.

$$F_i = \sum_{s \in N} f_i(s) = \sum_{s \in N} \frac{r_i(s) \times w(s)}{d(s)} \tag{4}$$

The sum of the weighted fractions of all flows that pass link (k, i) is called the aggregate flow weight of (k, i). It is denoted as $F_{k,i}$.

$$F_{k,i} = \sum_{s \in N} f_{k,i}(s) = \sum_{s \in N, s \neq i} f_{k,i}(s) = \sum_{s \in N, s \neq i} \frac{r_{k,i}(s) \times w(s)}{d(s)} \tag{5}$$

We assume acyclic routing, which means that packets generated from i will not be routed back to i, i.e., $r_{k,i}(i) = 0$.

Sensor i maintains local variables to store the values of F_i, $F_{k,i}$, $\forall k \in U_i$, and $F_{i,j}$, $\forall j \in D_i$. $r_i(s)$ and $r_{k,i}(s)$, $\forall s \in N$, are per-flow information and cannot be stored at sensor i. But the sensor can locally measure the values of r_i and $r_{i,j}$, $\forall j \in D_i$, based on which the values of F_i and $F_{i,j}$ must be calculated. The distributed, iterative algorithm for such computation is given in the next section. It should be stressed that the values of flow state, such as $f_i(s)$, $f_{k,i}(s)$, $r_i(s)$, and $r_{k,i}(s)$, are not stored or measured.

Weighted Mean Rate: The weighted mean rate at sensor i is defined as

$$R_i = \frac{r_i}{F_i} \tag{6}$$

The weighted mean rate at link (k, i) is defined as

$$R_{k,i} = \frac{r_{k,i}}{F_{k,i}} \tag{7}$$

If the weights of all flows are one, then $F_{k,i}$ will be the sum of the flow fractions passing (k, i), and $R_{k,i}$ will be the average flow rate among all flow fractions passing (k, i).

When a sensor i is not congested, it forwards all packets received from upstream. However, when it is congested, it must inform upstream neighbors to reduce their rates, and we want to equalize the weighted mean rates, $R_{k,i}, \forall k \in U_i$, for all upstream data sources.

Aggregate Fairness: We define a new fairness model. A feasible rate assignment is said to achieve the *aggregate fairness* if it satisfies the following condition: at every congested sensor i, the weighted mean rate $R_{k,i}$ of any upstream link (k, i) cannot be increased without decreasing the weighted mean rate $R_{k',i}$ of another upstream link (k', i), for which $R_{k',i} \leq R_{k,i}$.

The above condition requires i to equalize the weighted mean rates of all upstream links whenever possible. In other words, it requires i to assign bandwidth to the upstream links in proportion to their aggregate flow weights.

This is a local condition that can be enforced by local operations. Once a sensor knows the aggregate flow weights of all upstream links, it will work with the upstream neighbors to make their forwarding rates proportional to those weights. Below we address the problem of computing aggregate flow weights.

4. DISTRIBUTED COMPUTATION OF AGGREGATE FLOW WEIGHTS

We prove several properties for any feasible rate assignment. Based on the properties, we describe a distributed algorithm to compute the aggregate flow weights.

PROPERTY 1. *For any feasible rate assignment, the aggregate flow weight of a sensor is equal to the sum of the aggregate flow weights of the upstream links and the weight of the locally generated flow.*

Proof: Consider an arbitrary sensor i. By (1), $r_i(i) = d(i)$. By (4),

$$\begin{aligned} F_i &= \sum_{s \in N} \frac{r_i(s) \times w(s)}{d(s)} \\ &= (\sum_{s \in N, s \neq i} \frac{r_i(s) \times w(s)}{d(s)}) + \frac{r_i(i) \times w(i)}{d(i)} \\ &= (\sum_{s \in N, s \neq i} \frac{r_i(s) \times w(s)}{d(s)}) + w(i) \end{aligned}$$

Applying the first constraint of a feasible rate assignment, we have

$$\begin{aligned} F_i &= (\sum_{s \in N, s \neq i} \frac{\sum_{k \in U_i} r_{k,i}(s) \times w(s)}{d(s)}) + w(i) \\ &= \sum_{s \in N, s \neq i} \sum_{k \in U_i} f_{k,i}(s) + w(i) \\ &= \sum_{k \in U_i} F_{k,i} + w(i) \end{aligned} \tag{8}$$

□

PROPERTY 2. *For any feasible rate assignment, the aggregate flow weight of a sensor i is equal to the sum of the aggregate flow weights of its downstream links. Furthermore, the aggregate flow weight $F_{i,j}$ of each downstream link (i, j) is proportional to the data rate $r_{i,j}$ passing the link. Namely, $F_{i,j} \propto r_{i,j}, \ j \in D_i$.*

Proof: In a data collection network, all flows share a common set of destinations (base stations). No matter which sources they come from, packets routed through i are randomly divided among the downstream links with probabilities proportional to the link rates $r_{i,j}, \ j \in D_i$. When i forwards a packet, it sends the packet to a downstream node j with a probability of $\frac{r_{i,j}}{r_i}$. For any flow s that passes i, the fraction of packets sent to j is the same, $\frac{r_{i,j}}{r_i}$. That is,

$$\begin{aligned} \frac{r_{i,j}(s)}{r_i(s)} &= \frac{r_{i,j}}{r_i} \\ r_{i,j}(s) &= r_i(s) \times \frac{r_{i,j}}{r_i} \end{aligned} \tag{9}$$

$F_{i,j}$ can be calculated as follows.

$$\begin{aligned} F_{i,j} &= \sum_{s \in N} \frac{r_{i,j}(s) \times w(s)}{d(s)} \\ &= \sum_{s \in N} \frac{r_i(s) \times w(s)}{d(s)} \times \frac{r_{i,j}}{r_i} \\ &= F_i \frac{r_{i,j}}{r_i} \end{aligned} \tag{10}$$

Therefore, $F_{i,j}$ is proportional to $r_{i,j}$. It is easy to verify that $\sum_{j \in D_i} F_{i,j} = F_i$. □

PROPERTY 3. *For any feasible rate assignment, it holds that $R_{i,j} = R_i$, $\forall i \in N, j \in D_i$.*

Proof: By (7), $R_{i,j} = \frac{r_{i,j}}{F_{i,j}} = \frac{\sum_{s \in N} r_{i,j}(s)}{F_{i,j}}$. By (9), we have

$$R_{i,j} = \frac{\sum_{s \in N} r_i(s) \times r_{i,j}}{F_{i,j} \times r_i} = \frac{r_{i,j}}{F_{i,j}}$$

By (10), we have

$$R_{i,j} = \frac{r_{i,j} \times r_i}{r_{i,j} \times F_i} = R_i \tag{11}$$

□

Based on the above properties, we design a distributed algorithm that iteratively computes the aggregate flow weights of all nodes and all links. Let $\bar{F}_i$, $\bar{F}_{k,i}$, and $\bar{F}_{i,j}$ be the variables at sensor i storing the values of F_i, $F_{k,i}$, and $F_{i,j}$, respectively. Initially sensor i sets $\bar{F}_i = 0$, $\bar{F}_{k,i} = 0$, $\forall k \in U_i$, and $\bar{F}_{i,j} = 0$, $\forall j \in D_i$. Periodically i computes $\bar{F}_i$ based on (8) and then $\bar{F}_{i,j}, \ j \in D_i$, based on (10). After that, i sends $\bar{F}_{i,j}$ to j such that j knows the aggregate flow weight of its upstream link (i, j), which is needed by the next iteration of computation. Similarly, i learns $\bar{F}_{k,i}$ from its upstream neighbor k. Below we show that, after a certain number of iterations of computing (8)-(10) and exchanging the link weights between neighbors, $\bar{F}_i$ and $\bar{F}_{i,j}$ will stabilize to the correct values. The links in E form an acyclic routing graph from all data sources to the base stations. The furthest data sources that do not have upstream nodes will learn their aggregate flow weights after the first iteration by computing (8). And then by (10) they will calculate the aggregate flow weights of their downstream links, which are the upstream links for the next layer of sensors that are one hop closer to the base stations. These next-layer sensors will learn their aggregate flow weights after the following iteration. The process repeats until the aggregate flow weights of all nodes/links are known. The maximum number of iterations before all variables stabilize is bounded by the length of the longest routing path.

5. ACHIEVING AGGREGATE FAIRNESS

We first describe a congestion avoidance mechanism, based on which we design a distributed algorithm that achieves the aggregate fairness.

5.1 Congestion Avoidance

We present a congestion avoidance mechanism to solve the buffer congestion problem described in Section 2. The basic idea is to make sure that a sensor k sends a packet to a sensor i only when i has the buffer space to hold the packet. Assume all data packets have the same size and the buffer space is slotted with each slot holding one packet. The residual buffer of i changes when i receives a packet from a sensor in U_i or forward a packet to a sensor in D_i. To keep the upstream neighbors updated with i's buffer state, whenever i sends out a packet (RTS/CTS/DATA/ACK), it piggybacks its current buffer state in the frame header, for example, using one bit to indicate whether the buffer is full. When an upstream neighbor k receives or overhears a packet from i, it caches the buffer state of i. Note that, even if RTS/CTS are not used, the bit piggybacked in DATA/ACK is sufficient to keep the neighbors updated.

When k has a packet to forward, if there exists a sensor $i \in D_k$ whose buffer is not full, k forwards the packet to i. Otherwise, k has to hold the packet until it overhears a packet from a sensor in D_k, piggybacking a non-full buffer state.

Sensor k may not overhear packets from i due to temporary radio interference or sleeping. In this case, its knowledge about i's buffer may be stale. It may falsely think i's buffer is full and block itself forever. One solution is to piggyback the one-bit buffer state in the neighbor discovery messages that are exchanged periodically between neighbors. Therefore, the buffer-state information will be resynchronized between k and i as long as they remain neighbors of each other. An alternative solution is for k to attempt transmitting if it does not overhear i's buffer state for a period of time.

The above congestion avoidance mechanism prevents packet drops due to buffer overflow because the sender of a packet makes transmission only when the receiver has buffer to hold the packet. Therefore, over the long term, the rate at which a node fills up its buffer (i.e., the combined incoming rate from all upstream links) will be equal to the rate at which the node empties the buffer (i.e., the combined outgoing rate to all downstream links). This approach eliminates the complicated rate-based signaling that is required by many existing congestion control approaches [2, 4]. It naturally adapts the sending rates of upstream sensors to the forwarding rates of the downstream sensors by buffer-based backpressure. Suppose the data sources initially send as fast as they can. When the buffer at an intermediate sensor i is filled, with the above congestion avoidance mechanism, the forwarding rates of its upstream neighbors are forced to slow down, in accordance to i's forwarding rate. This may cause the upstream neighbors' buffers to fill up. Once their buffers are full, the further upstream sensors are forced to slow down. This process repeats towards the furthest sensors and eventually the whole network adapts toward the maximum congestion-free throughput.

5.2 Distributed Aggregate Fairness Algorithm (AFA)

We now present the distributed aggregate fairness algorithm (AFA). When a sensor i receives more than it can forward, its data queue will build up. If the buffer fills up, according to the above congestion avoidance mechanism, the upstream neighbors will have to frequently wait for buffer release, which means they are sending at reduced rates. Now the question is how to determine the amount of rate reduction each upstream neighbor should take. We know it is not fair for them to reduce equally (Section 2). Based on the definition of aggregate fairness, we should make the weighted mean rates of all upstream links equal whenever possible. This means the actual rate from an upstream link should be proportional to the link's aggregate flow weight.

When sensor i is congested,[1] it computes a rate limit for each upstream neighbor k as follows

$$l_{k,i} = \frac{F_{k,i}}{F_i} r_i \qquad (12)$$

where the values of $F_{k,i}$ and F_i are updated periodically by the distributed algorithm in the previous section and r_i is the current throughput of i, which is measured locally. It then advertises $l_{k,i}$ to k, possibly by piggybacking in an ACK packet to k. After receiving $l_{k,i}$, k uses a token-bucket algorithm to enforce the rate limit. Let c be a counter, which is initialized to zero. Let t be a time variable, which is initialized to the system-clock value. Let b be the bucket size. A packet at k can be scheduled for transmission to i only if it passes the following test.

```
Rate_Limit_Test()
(1) $c \leftarrow \min\{c +$ (current clock value $- t) \times l_{k,i}, b\}$
(2) $t \leftarrow$ current clock value
(3) **if** ($c \geq$ packet size)
(4)       schedule the packet for transmission
(5)       $c \leftarrow c -$ packet size
(6) **else**
(7)       wait for ( packet size $-c)/l_{k,i}$ before
          trying the test again
```

The actual rate of an upstream link will be bounded by the rate limit. If i itself is a data source, it will assign a local rate limit as follows

$$l_i = \frac{w(i)}{F_i} r_i \qquad (13)$$

i uses the token-bucket algorithm to make sure that the locally-generated data rate is bounded by the limit.

After an upstream neighbor enforces a rate limit, it may become congested because it now sends less. If its buffer is kept full, it will enforce rate limits in a similar way on its upstream neighbors. This process repeats towards the data sources. Eventually all affected data sources will adjust their rates to the fair bandwidth shares defined by the rate limits that are progressively pushed from the bottlenecks to the sources. Collectively, these rate limits reflect the network load on the congested paths and the number of competing flows on different paths, as well as the weights of those flows. Note that *only* congested sensors enforce rate limits on upstream links, and the rate limits are updated periodically.

After a congested sensor i enforces rate limits on upstream links, the congestion condition may change either because some upstream data sources cease to produce data or because some downstream sensors gain additional bandwidth (due to the dynamics of environmental interference). When i is able to forward more than it receives, its packet queue will clear up. If the buffer space keeps going back to empty whenever some packets are received, i will artificially increase the rate limits to use up the available bandwidth. In our simulations, when the length of the packet queue at i is below a threshold, the upstream rate limit is set to $1.1 \times l_{k,i}$, where $l_{k,i}$ is computed by (12), and the local rate limit is set to $1.1 \times l_i$,

[1]Congestion is signaled when the sensor's buffer is full. Note that radio collision is dealt with by the exponential backoff of CSMA/CA.

where l_i is computed by (13). When the buffer space at i becomes almost empty, the upstream rate limit is set to $2 \times l_{k,i}$ and the local rate limit is set to $2 \times l_i$. As the rate limits are increased, r_i (the throughput of i) may become larger, which in turn further increases the rate limits when (12) and (13) are computed next time. However, the increase of rate limits must stop when the congestion condition returns to i or when the congestion conditions of all upstream sensors are removed. The former case happens when the rate limits are raised too high and exceed the maximum rate that i is able to send downstream.[2] i learns the return of congestion by observing the buffer's fullness, and when that happens, it sets the rate limits to be $l_{k,i}$ and l_i, instead of over-setting them. The latter case is identified by the buffer emptiness of upstream neighbors. When that happens, i stops enforcing rate limits on upstream links.

THEOREM 1. *After AFA stabilizes the data rates on all routing links, the resulting rate assignment on the whole network achieves the aggregate fairness.*

Proof: Consider an arbitrary congested sensor i. Let U'_i be the set of upstream neighbors whose forwarding rates are constrained by the rate limits imposed by i.

$$\forall k \in U'_i,\ r_{k,i} = l_{k,i}$$

If $l_{k,i}$ is increased, $r_{k,i}$ will also be increased. $\forall k \in U'_i$, $R_{k,i} = \frac{r_{k,i}}{F_{k,i}} = \frac{l_{k,i}}{F_{k,i}} = \frac{r_i}{F_i}$, by (12).

Let U''_i be the set of upstream neighbors whose forwarding rates are not constrained by the rate limits imposed by i. It means that congestion further upstream restricts the rates even more.

$$\forall k \in U''_i,\ r_{k,i} \leq l_{k,i}$$

If $l_{k,i}$ is increased, $r_{k,i}$ will not be increased. $\forall k \in U''_i$, $R_{k,i} = \frac{r_{k,i}}{F_{k,i}} \leq \frac{l_{k,i}}{F_{k,i}} = \frac{r_i}{F_i}$.

Only the rate from an upstream neighbor $k \in U'_i$ can be increased by relaxing the rate limit. $R_{k,i} = \frac{r_i}{F_i}$, which is the largest weighted mean rate among all upstream neighbors. Now if we increase $r_{k,i}$, then $r_{k',i}$ for another upstream neighbor k' must be decreased, in order for the combined rate from all upstream neighbors to remain the same. Consequently $R_{k',i}$ ($\leq R_{k,i}$) must be decreased, which satisfies the condition for aggregate fairness. □

5.3 Optimizations

Various optimizations can be made to improve the performance of AFA. We give a few examples below. Suppose each node i periodically advertises its weighted mean rate R_i to the neighbors. It also learns the weighted mean rates of its neighbors.

- When i has a packet to forward, if two or more downstream neighbors do not violate the rate limits and their buffers are not full, i always forwards the packet to the neighbor with the largest weighted mean rate.
- When i is congested and computes a rate limit $l_{k,i}$ for an upstream neighbor k by (12), if R_k is smaller than R_i, i will increase the rate limit $l_{k,i}$ by a factor of R_i/R_k, which allows k to send more traffic to i in an effort to equalize R_k and R_i.
- If i has the smallest weighted mean rate in the neighborhood, it reduces its minimum congestion window to acquire a large portion of the channel capacity. On the contrary, if it has the largest weighted mean rate, it increases its minimum congestion window to give up some bandwidth. Different optimization schemes can be designed based on this idea. The one used in our simulations is described as follow. We define the range of window variations to be

$$range = \max\{(R_{max} - R_{min})/\Omega, 1\}$$

where R_{max} is the largest weighted mean rate in the neighborhood, R_{min} is the smallest weighted mean rate in the neighborhood, and Ω is the largest allowed rate of any data source, which is a system parameter. The minimum congestion window at sensor i is adjusted by the following formula.

$$factor = 1 - \frac{range}{2} + \frac{R_i - R_{min}}{R_{max} - R_{min}} range$$
$$min\_cong\_win = def\_win \times factor$$

where def_win is the default minimum congestion window.

[2]Other than filling i's buffer, no harm will be done because the congestion avoidance mechanism will prevent packet drops by restricting upstream neighbors from overflowing i's buffer.

6. SIMULATION

We have performed extensive simulations to evaluate the performance of AFA. We compare AFA with prior works in terms of end-to-end fairness, energy efficiency, and packet drops. We also study how well AFA achieves weighted fairness and how fast it converges.

The simulation parameters are described as follows. 500 sensors are randomly placed in a 1000×1000 area. The transmission range of the sensors is 100. The transmission rate is 512 kilobits per unit of time. The minimum congestion window is 40 units of time. 8 base stations are evenly spaced along one edge of the deployment area. There are 100 data sources randomly selected from the 500 sensors. A source generates up to 25 data packets per unit of time. However, the actual rate will be lower if the routing paths are congested. Each data packet is 30 bytes long. CSMA/CA with exponential backoff is fully implemented to resolve radio collision between contending nodes. Each control packet (RTS/CTS/ACK) is 3 bytes long. The buffer at each sensor can hold 30 data packets. The following four schemes are implemented.

- *No Congestion/Fairness Control*: Neither do the data sources adjust their packet generation rates, nor the intermediate sensors adapt their forwarding rates.
- *Backpressure*: This is CODA's hop-by-hop congestion control mechanism [2]. If a sensor x is congested (based on channel utilization and buffer level), it periodically sends backpressure messages to its neighbors, which reduce their forwarding rates to x by 50%. If an upstream neighbor is a data source, the neighbor reduces the rate at which it generates new data by the same percentage.
- Fairness Routing Tree (FRT): It is based on the work of [3], which addresses end-to-end fairness but assumes single-path routing. Each data source has a single routing path to the closest base station. To each base station, the routing paths form a routing tree rooted at the station. A sensor in the tree learns the number of upstream sources within the subtree rooted at itself. Based on the available bandwidth and other information, the sensor calculates the average rate an upstream source can send. If the sensor is congested, it allocates the available bandwidth to upstream neighbors proportional to the number of flows traversing that neighbor.
- Aggregate Fairness Algorithm (AFA): It performs both congestion avoidance and distributed rate limit based on aggregate flow weights, as proposed in this paper.

Rate limits are used in Backpressure, FRT, and AFA. After a rate limit is installed, it can be increased if the downstream traffic condition is improved. If the number of packets in the buffer of a previously congested sensor drops below 10, the forwarding rates of

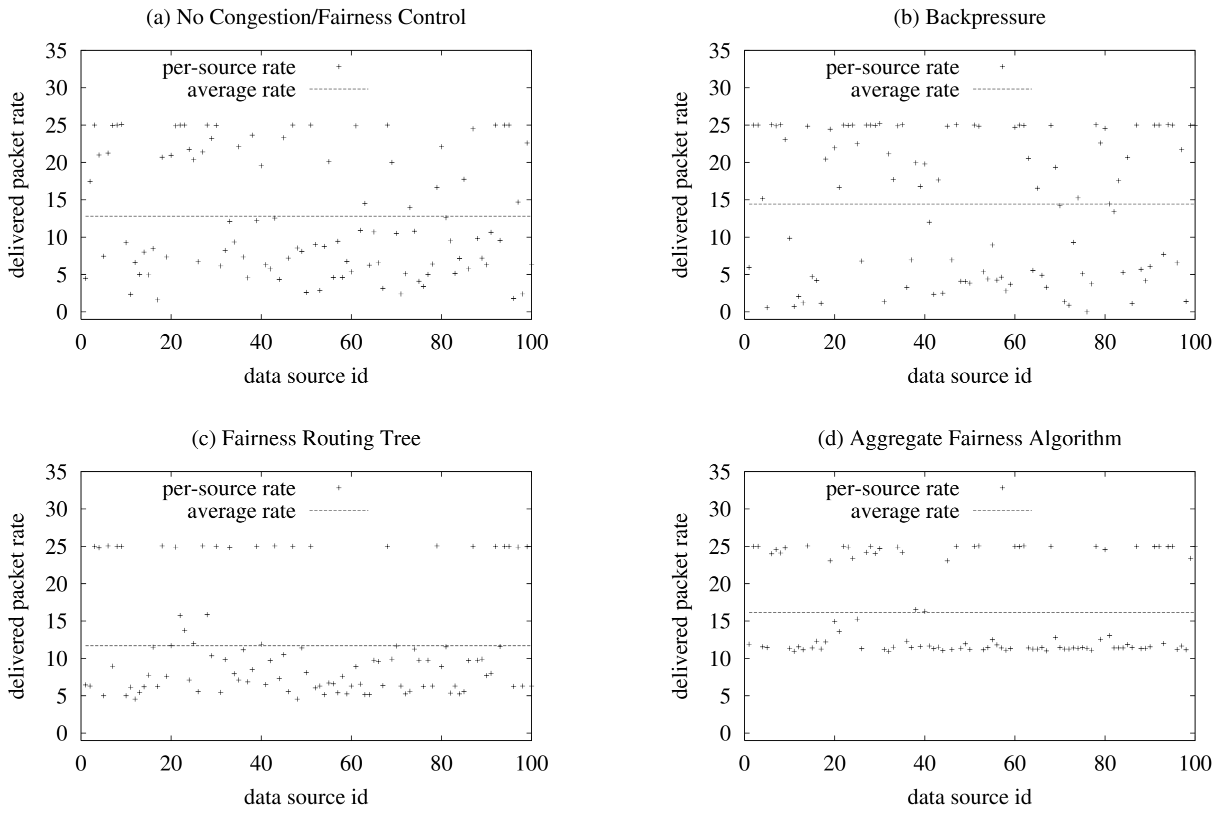

Figure 4: Packet rates of 100 data sources under different schemes

the upstream neighbors will be increased by 10% periodically until the buffer has more than 10 packets; if the sensor itself is a data source, its data generation rate will also be increased by 10%. If the buffer of a previously congested sensor is almost empty (fewer than 5 packets), the forwarding rates of the upstream neighbors will be doubled; if the sensor itself is a data source, its data generation rate will also be doubled.

No Control, Backpressure and AFA use multi-path routing. Each sensor has a set of downstream links that move one hop closer to the closest base station(s). In order to limit the level of media contention, when there are more than two candidate downstream neighbors, we only use two that are furthest away from each other. When a sensor forwards a packet, for No Control, it randomly picks a downstream link; for Backpressure, it randomly picks a downstream link that does not violate the rate limit if one exists; for AFA, the optimizations in Section 5.3 are applied. FRT uses tree-based routing.

6.1 Fairness Comparison

The first set of simulations demonstrates that AFA is able to achieve much better end-to-end fairness than other schemes. The performance metric is called *delivered packet rate* (or *packet rate* for brevity), which is the average number of packets that are successfully delivered from a data source to the base station(s) per unit of time. Figure 4 (a) shows the delivered packet rates of 100 data sources when no congestion/fairness control is applied. The rates are widely distributed between zero to 25 packets per unit of time. Figure 4 (b) shows that Backpressure, as a congestion control scheme, does not achieve good fairness among different data sources. When a sensor is congested, it reduces the forwarding rates of upstream neighbors by the same percentage even through the numbers of flows passing through them can be very different. Figure 4 (c) shows that FRT significantly improves fairness. Some sources can send at full rate (25 packets per unit of time) because their flows do not pass the congested points in the tree. The problem of single-path routing is that it cannot fully exploit the capacity of the network. When each flow is allowed to use multiple routing paths, if one path is congested, the packets can be redirected to other paths, which not only increases the rate of this data source but removes one flow from the congested path, helping improve the rates of other sources. Figure 4 (d) shows that AFA considerably improves fairness over FRT. The data sources fall in two groups. One group consists of data sources whose routing paths are not congested; their rates are about 25 packets per unit of time. The other group consists of sources whose routing paths are congested; their rates are mostly in the range of 11 to 12.5 packets per unit of time. The average rate of AFA is 38.3% higher than that of FRT. AFA not only produces a larger average rate per data source, but also redistributes the bandwidth from high-rate sources to low-rate sources. The reason is that, with multiple routing paths per source in AFA, the paths of a low-rate source have a better chance to come across the paths of a high-rate source, which allows the former to acquire the bandwidth of the latter.

6.2 Energy Efficiency and Packet Drops

The second set of simulations compares the energy efficiencies of the four schemes. After the rates stabilize, the energy efficiency of a scheme is defined as

$$\frac{T}{BH}$$

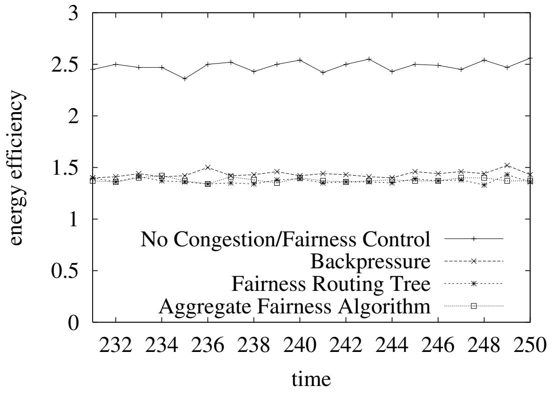

Figure 5: Backpressure, FRT, and AFA achieve much better energy efficiency than No Control.

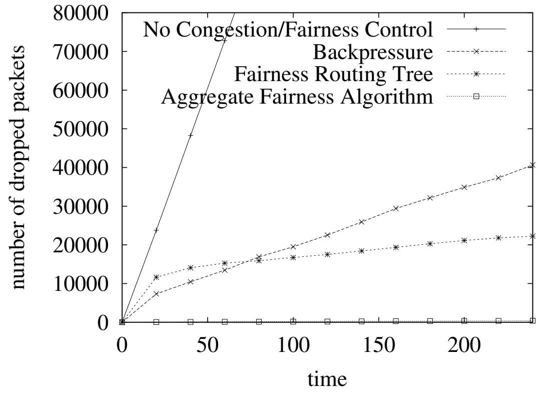

Figure 6: AFA seldom drops any packets due to its congestion avoidance mechanism.

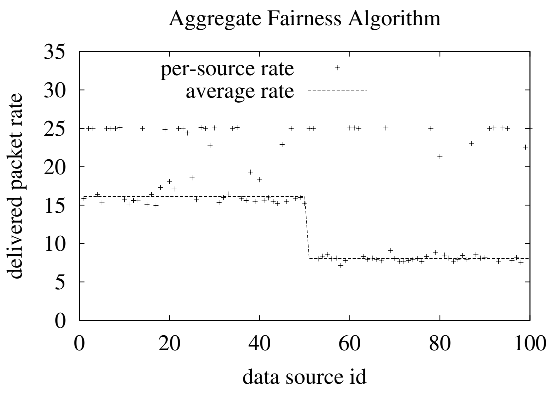

Figure 7: Data sources 1-50 have weight two. Data sources 51-100 have weight one.

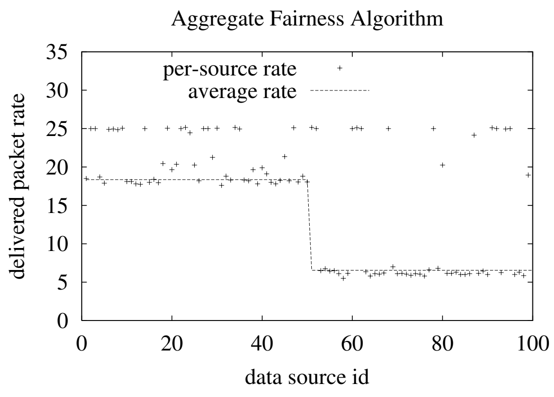

Figure 8: Data sources 1-50 have weight three. Data sources 51-100 have weight one.

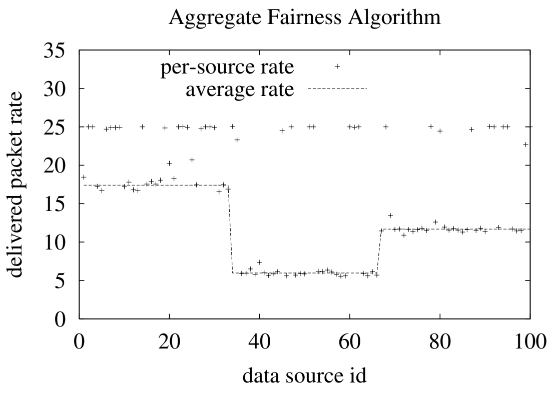

Figure 9: Data sources 1-33 have weight three. Data sources 34-66 have weight one. Data sources 67-100 have weight two.

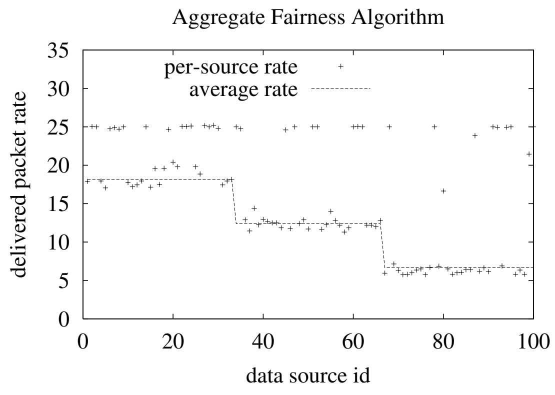

Figure 10: Data sources 1-33 have weight three. Data sources 34-66 have weight two. Data sources 67-100 have weight one.

where T is the number of bytes transmitted in the whole network during a period of time, B is the number of data bytes delivered to the base stations during the same time, and H is the average number of hops a delivered packet travels. Intuitively, it measures how many bytes must be transmitted in order to move one byte of data one hop towards a base station. A smaller value means better efficiency. This measurement includes the actual transmission of data, as well as the overhead of control packets, the energy waste due to collision, and the energy waste due to packet drops (as a result of buffer overflow).

Figure 5 compares the energy efficiencies of the four schemes during 230-250 units of time into the simulations. We let the simulations run for a sufficiently long time (230 units) such that the packet rates in all schemes are stabilized before we measure energy efficiency. Backpressure, FRT, and AFA achieve much better energy efficiency than No Control. They reduce the amount of data packets pumped from the sources into the network when congestion occurs. The excess packets would not be able to get through the network anyway but would instead be counterproductive by causing severe collisions and packet drops.

Figure 6 shows the accumulated number of dropped packets over time. Without any congestion/fairness control, the packet drop problem is severe for the reasons explained in Section 2.2. Backpressure has more packet drops than FRT because it is more aggressive by using multiple routing paths to carry additional packets, which also cause more frequent congestions. AFA seldom drops any packets due to its congestion avoidance mechanism.

6.3 Achieving Weighted Fairness by AFA

The third set of simulations demonstrates that data sources can acquire different rates by using different weights. In this set of simulations, when we calculate the average rates, we do not include those data sources whose routing paths are not seriously congested, namely, those sources whose rates are above 20 packets per unit of time.

In Figure 7, data sources 1-50 have weight two; their average packet rate is 16.13. Sources 51-100 have weight one; their average packet rate is 8.05. The results confirm that, on average, AFA allocates bandwidth to the flows proportional to their weights. Note that some sources with weight one have larger rates than some sources with weight two. The reason is that these weight-one sources do not share routing paths with the weight-two sources. When two nodes share a common congested routing path, the bandwidth is allocated based on weights. However, if a source with weight one has an uncongested routing path while another source with weight two does not, the former will have higher rate than the latter.

In Figure 8, data sources 1-50 have weight three; their average packet rate is 18.35. Sources 51-100 have weight one; their average packet rate is 6.55.

Figure 9 has three different weights. Data sources 1-33 have weight three; their average packet rate is 17.41. Data sources 34-66 have weight one; their average packet rate is 5.97. Data sources 67-100 have weight two; their average packet rate is 11.69. The average rates are roughly proportional to the weights.

Figure 10 also has three different weights. The weight assignment is made different from that in the previous figure to bring out the contrast. Data sources 1-33 have weight three; their average packet rate is 18.18. Data sources 34-66 have weight two; their average packet rate is 12.40. Data sources 67-100 have weight one; their average packet rate is 6.67. The average rates are again roughly proportional to the weights.

Now let's consider the average rate among all 100 data sources. In Figure 7, it is 16.19. In Figure 8, it is 16.23. In Figure 9, it is 15.87. In Figure 10, it is 16.13. The average rate among all sources is kept about the same in these simulations, which means different weight schemes do not change the overall system throughput.

6.4 Convergence

The fourth set of simulations compares the convergence speed of AFA with that of FRT. We have performed extensively simulations, which show that in general AFA and FRT have comparable convergence speeds. They both converge reasonably fast. Multipath routing does not seem to slow down the convergence of AFA. Figure 11 and Figure 12 show the packet rates under FRT and AFA, respectively, after 30 units of time into the simulations. The rates are clustered but have not been stabilized. After 60 units of time into the simulations, the rates are fully converged as shown in Figure 13 and Figure 14. A distributed algorithm is suitable for a dynamic network only when the algorithm converges much faster than the network changes. Each algorithm has its scope of applicability. Although AFA converges reasonably fast, it seems not suitable for highly-dynamic sensor networks with a large number of short-lived flows that come and go.

7. CONCLUSION

This paper studies the end-to-end fairness problem in data-collection sensor networks. We show that the fairness problem becomes considerably harder when each packet flow is forwarded on multiple routing paths. We formally define a new aggregate fairness model, prove its properties, and propose a distributed algorithm that implements the model. The new fairness algorithm can work with any routing protocol. The simulation results confirm the effectiveness of the algorithm in achieving (weighted) fairness among competing data flows.

8. REFERENCES

[1] S. Tilak, M. B. Abu-Ghazaleh, and W. Heinzelman, "Infrastructure Tradeoffs for Sensor Networks," *Proc. of 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, September 2002.

[2] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," *Proc. of SenSys'03*, November 2003.

[3] C. T. Ee and R. Bajcsy, "Congestion Control and Fairness for Many-to-One Routing in Sensor Networks," *Proc. of ACM SenSys 2004*, November 2004.

[4] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks," *Proc. of ACM SenSys 2004*, November 2004.

[5] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," *Proc. of ACM Symposium on Operating System Design and Implementation (OSDI),*, December 2002.

[6] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The Design of an Acquisitional Query Processor for Sensor Networks," *in Proc. of ACM SIGMOD International Conference on Management of Data*, 2003.

[7] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. of IEEE INFOCOM'02*, June 2002.

[8] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional

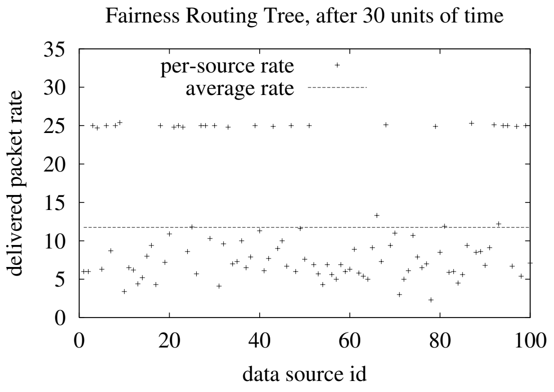

Figure 11: Packet rates achieved by FRT after 30 units of time

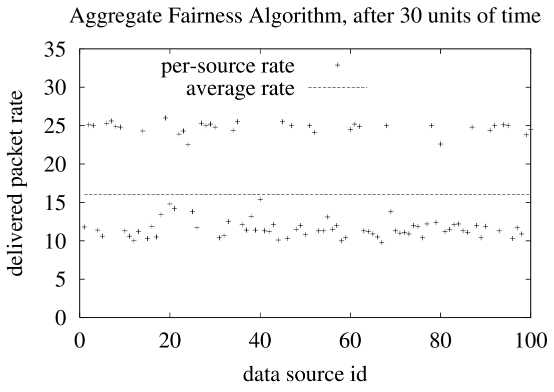

Figure 12: Packet rates achieved by AFA after 30 units of time

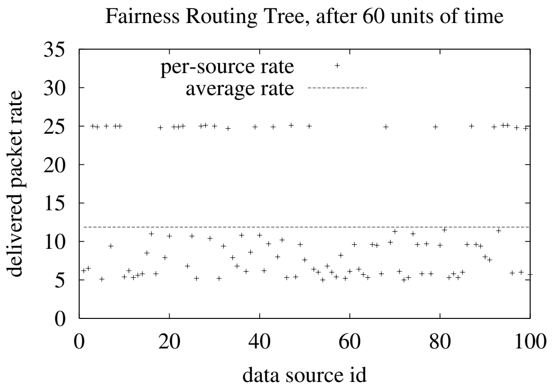

Figure 13: Packet rates achieved by FRT after 60 units of time

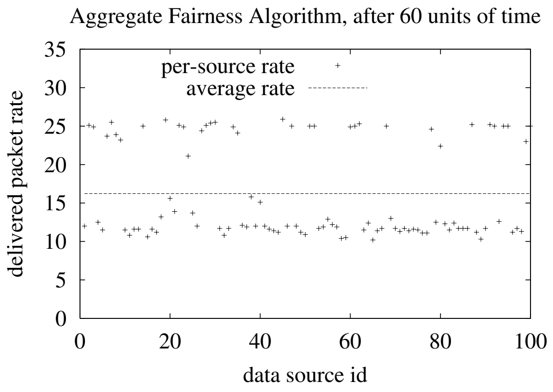

Figure 14: Packet rates achieved by AFA after 60 units of time

fairness and stability," *Journal of the Operational Research*, vol. 49, 1998.

[9] J. Mo and J. Walrand, "Fair End-to-End Window-based Congestion Control," *IEEE/ACM Transactions on Networking*, October 2000.

[10] L. Massoulie and J. Roberts, "Bandwidth Sharing: Objectives and Algorithms," *IEEE Transactions on Networking*, vol. 10, no. 3, June 2002.

[11] S. H. Low, "A Duality Model of TCP and Queue Management Algorithms," *IEEE/ACM Transactions on Networking*, October 2003.

[12] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat Monitoring: Application Drive for Wireless Communications Technology," *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[13] L. Schwiebert, S. Gupta, and J. Weinmann, "Research Challenges in Wireless Networks of Biomedical Sensors," *Mobile Computing and Networking*, pp. 151–165, 2001.

[14] E. Biagioni and K. Bridges, "The Applications of Remote Sensor Technology to Assist the Recovery of Rare and Endangered Species," *International Journal of High Performance Computing Applications, Special Issue on Distributed Sensor Networks*, April 2003.

[15] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, September 2002.

[16] M. D. Yarvis, W. S. Conner, L. Krishnamurthy, A. Mainwaring, J. Chhabra, and B. Elliott, "Real-World Experiences with an Interactive Ad Hoc Sensor Network," *Proc. of International Conference on Parallel Processing Workshops*, August 2002.

[17] Center for Embedded Networked Sensing, "Terrestrial Ecology Observing Systems," *http://www.cens.ucla.edu/portal/terrestrial_ecology_observing_systems.html*.

[18] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The Emergence of Networking Abstractions and Techniques in TinyOS," *Proc. of First USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2004.

[19] J. Polastre, "Sensor Network Media Access Design," *http://www.cs.berkeley.edu/∼culler/cs294-f03/finalpapers/bmac.pdf*, 2003.

[20] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, "Achieving MAC Layer Fairness in Wireless Packet Networks," *Proc. of MobiCom'00*, August 2000.

[21] X. L. Huang and B. Bensaou, "On Max-Min Fairness and Scheduling in Wireless Ad-hoc Networks: Analytical Framework and Implementation," *Proc. of MobiHoc'01, Long Beach, California*, October 2001.

[22] H. Luo, J. Cheng, and S. Lu, "Self-Coordinating Localized Fair Queueing in Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 1, 2004.

[23] L. Tassiulas, "Adaptive Back-pressure Congestion Control based on Local Information," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, February 1995.

[24] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," *Proc. of 3rd Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM'99)*, August 1999.

[25] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. of ACM MobiCom'00*, August 2000.

[26] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing without Location Information," *Proc. of ACM Mobicom'03*, April 2003.

[27] S. Chen, G. Fan, and J. Cui, "Avoid Void in Geographic Routing for Data Aggregation in Sensor Networks," *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), Special Issue on Wireless Sensor Networks*, 2006.

[28] T. He, J. A.Stankovic, C. Lu, and T. F. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," *Proc. of International Conference on Distributed Computing Systems (ICDCS'03)*, May 2003.

[29] H. Luo, S. Lu, and V. Bharghavan, "A New Model for Packet Scheduling in Multihop Wireless Networks," *Proc. of MobiCom'00*, August 2000.

[30] B. Bensaou, Y. Wang, and C. C. Ko, "Fair medium access in 802.11 based wireless ad-hoc networks," *Proc. of MobiHoc'00, Boston, Massachusetts*, July 2000.

[31] V. Gambiroza, B. Sadeghi, and E. W. Knightly, "End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks," *Proc. of Mobicom'04, Philadelphia, PA, USA*, September-October 2004.

[32] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks," *Proc. of MobiHoc'03*, June 2003.

[33] Y. Yi and S. Shakkottai, "Hop-by-Hop Congestion Control over a Wireless Multi-hop Network," *Proc. of IEEE INFOCOM'04, Hong Kong, China*, March 2004.

[34] L. Tassiulas and S. Sarkar, "Maxmin Fair Scheduling in Wireless Networks," *Proc. of IEEE INFOCOM'02*, June 2002.

[35] A. Sridharan and B. Krishnamachari, "Max-Min Fair Collision-Free Scheduling for Wireless Sensor Networks," *Proc. of Workshop on Multihop Wireless Networks (MWN'04)*, April 2004.

[36] A. Woo and D. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," *Proc. of MobiCom'01*, July 2001.

[37] J. M. Kleinberg, Y. Rabani, and E. Tardos, "Fairness in Routing and Load Balancing," *IEEE Symposium on Foundations of Computer Science*, 1999.