Explicit and *Precise* Rate Control for Wireless Sensor Networks

Avinash Sridharan
Department of Electrical Engineering
University of Southern California
asridhar@usc.edu

Bhaskar Krishnamachari
Department of Electrical Engineering
University of Southern California
bkrishna@usc.edu

Abstract

The state of the art congestion control algorithms for wireless sensor networks respond to coarse-grained feedback regarding available capacity in the network with an additive increase multiplicative decrease mechanism to set source rates. Providing precise feedback is challenging in wireless networks because link capacities vary with traffic on interfering links. We address this challenge by applying a *receiver capacity model* that associates capacities with nodes instead of links, and use it to develop and implement the first explicit and precise distributed rate-based congestion control protocol for wireless sensor networks — the *wireless rate control protocol* (WRCP). Apart from congestion control, WRCP has been designed to achieve lexicographic max-min fairness. Through extensive experimental evaluation on the USC Tutornet wireless sensor network testbed, we show that WRCP offers substantial improvements over the state of the art in flow completion times as well as in end-to-end packet delays.

Categories and Subject Descriptors

C.2.m [**Computer-Communication Networks**]: Miscellaneous

General Terms

Algorithms Design Experimentation

Keywords

Congestion Control, Wireless Sensor Networks

1 Introduction

For low powered wireless sensor networks (WSN), the degradation of per-source sustainable rate is quite drastic with increase in network size. In our experiments on the USC Tutornet testbed [13], with a 40 byte packet, a 4-node network yields per-source rates as low as 16 packets per sec. A 20-node network under similar conditions results in a reduction of per-source rates to ~ 2 packets per sec, and in a 40-node network this rate reduces to about one packet every two seconds. Given the drastic reduction in per-source rates with increase in the number of sources, if sufficiently large number of flows (sources) are active the reduced capacity can be easily exceeded if the flows are operating without the knowledge of sustainable rates; which will happen in the absence of a rate control protocol. This observation reflects the importance of rate control protocols in these networks.

Given the need for rate control in wireless sensor networks, there have been several proposals for implementing rate control mechanisms in these networks (ARC [23], CODA [21], FUSION [10], IFRC [16], RCRT [15]). The core mechanism for rate control used in existing proposals are based on additive increase multiplicative decrease (AIMD) algorithms. An AIMD-based scheme has the advantage that the protocol is agnostic to the underlying link layer, requiring no prior knowledge of the available capacity. This allows for modular protocol design. Despite the benefits presented by the AIMD mechanism, a key drawback of AIMD-based rate control protocols is their long convergence time to the achievable rate, and long queue backlogs as the rates frequently exceed the available capacity (this is used as a signal from the network to indicate that it is time to cut back) [8]. This is illustrated in Figure 1, which presents the performance of IFRC [16], the state-of-the-art congestion control protocol in wireless sensor networks. These results are from a simple, single-hop, fully connected, 4-node experiment with 1 sink and 3 sources. It is observed that the rate allocation takes more than 300 seconds to converge, and queue sizes routinely reach 8-10 packets.

The long convergence times do not affect the goodput of flows when the flows are long *i.e.*, flows whose duration of activity is much longer than the convergence time and the set of flows in the network is a constant (a *static scenario*). However, we believe AIMD based rate control protocols will adversely affect the goodput of flows when the set of flows active in the system is continuously changing (a *dynamic scenario*). Note that this will occur whenever there exist short flows in the network. In this scenario, the per-flow available capacity is continuously changing (due to the rapidly changing active flow count). If the long convergence time of the AIMD-based protocol prevents it from adapting to these changes fast enough, it is inevitable that

SenSys'09, November 4–6, 2009, Berkeley, CA, USA.

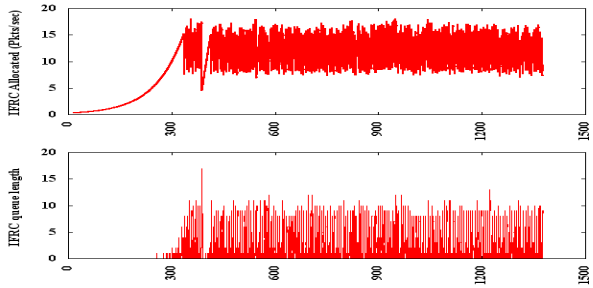


Figure 1. Rate allocation and queue backlog behavior for IFRC as observed at a particular node.

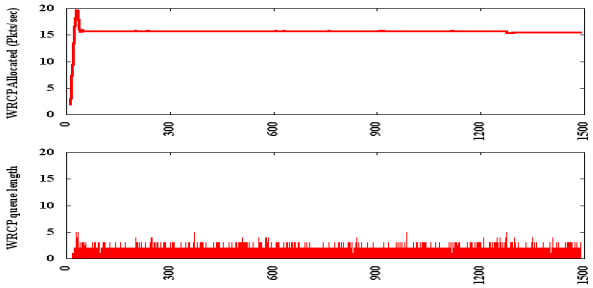


Figure 2. The behavior of allocated rate, queue back logs for WRCP for the same node, for which the same metrics have been presented for IFRC.

active flows will be allocated sub-optimal rates. This sub-optimality has significant ramifications in terms of energy consumption, and hence on network lifetime. The lower the goodput, the longer it will take for the flows to complete, forcing radios in the network to be awake for a longer duration, and hence consuming more energy. Such scenarios are particularly relevant to event-driven sensor networks, and those that deploy duty cycling to conserve energy.

In this work, we aim to verify and address the above problems faced by AIMD-based rate control protocols. We focus on designing a distributed rate control protocol for WSN, one that will perform well not only in static scenarios but in dynamic scenarios as well. We show that drastic improvements in the convergence time of a rate control protocol can be achieved if the protocol design is based on the knowledge of explicit capacity information, rather than on an AIMD mechanism. The key challenge in achieving this, of course, lies in overcoming the difficulty in computing the capacity, given that the bandwidth of each link is affected by interference from other links in its vicinity.

Our principal contribution in this work, for the specific case of a collection tree, is the design and implementation of a distributed rate control protocol, that we refer to as the *Wireless Rate Control Protocol* (WRCP). WRCP uses an approximation of the available capacity in order to provide explicit and precise feedback to sources. This approximation is obtained by exploiting performance knowledge of the underlying CSMA MAC protocol. The key idea in our approach is to associate a constant capacity with the nodes instead of the links. The gains of this approach, in terms of convergence times (few tens of seconds for WRCP as compared to hundreds of seconds for IFRC) and smaller queue back logs are highlighted in Figure 2. The fast convergence times translate to higher goodput, and hence faster flow completion times (which indirectly results in energy savings), and the reduced queue size improves end-to-end packet delays.

The rest of the paper is organized as follows. First, in section 2, we present a useful notion of capacity in a WSN operating over a CSMA protocol, which is referred to as *receiver capacity*. In section 3, we present the software architecture used to design a rate control stack in the TinyOS-2.x operating system. In section 4, we present the design and implementation of WRCP. This protocol has been designed to work specifically over a collection tree. It uses the receiver capacity model to provide explicit and precise rate control information to the sources, striving to achieve a lexicographic max-min fair allocation. In section 5, we present an analysis to estimate the parameter settings for WRCP that guarantees a stable operation for the protocol over any given topology. In section 6, we present our experimental setup for evaluating WRCP on TinyOS-2.X, running on Tmote Sky devices, using the IEEE 802.15.4-compliant CC2420 radios. In section 7, we present empirical evidence, justifying our analysis of parameter selection for WRCP. In section 8, we undertake a comparative evaluation with IFRC [16]. The results show substantial improvements in flow completion times and end-to-end packet delays. We place our contributions in light of prior work in section 9, and present concluding comments on future work in section 10.

2 Receiver Capacity

The primary requirement for designing an explicit and precise rate control algorithm (such as RCP [8], XCP [12], WCPCAP [17]) is a usable notion of achievable capacity. In traditional wired networks, the notion of capacity is associated with a link existing between any two nodes. All flows traversing the link are assumed to linearly share the constant capacity of the link. In wireless networks, however, the capacity of a link is not constant, but rather is affected by activity on interfering links in its vicinity. We therefore need to redefine the notion of capacity.

Each node can be perceived as having a receiver domain consisting of all transmitting nodes within range, including itself. The crux of our approach is to associate the concept of capacity with nodes instead of links; we refer to this as *receiver capacity*. Although, in general, the region of achievable rates in a given receiver domain is not linear we approximate it with a linear rate region by making the receiver capacity a constant. Thus, our assumption is that the receiver capacity depends only upon the number of neighboring nodes and not their transmission rates. This approximation allows us to model the relationship between the receiver capacity and the rates of flows traversing the corresponding receiver's domain as a simple linear equation.

Using the notion of receiver capacity, we can determine constraints on rate allocation to flows over a sink-based tree (collection tree). Let N_i be the set of all neighbors of i (consisting of i itself, and all nodes within its communication range); C_i the set denoting the subtree rooted at i (including

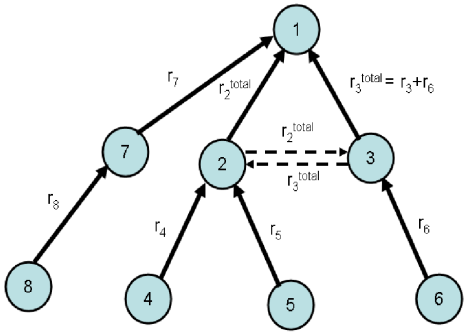


Figure 3. An illustrative example of the receiver capacity model

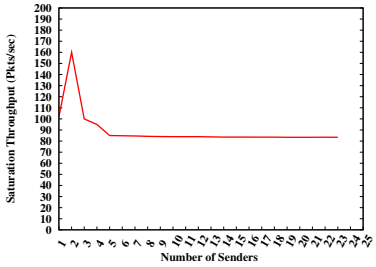


Figure 4. Saturation throughput for multiple senders for the CC2420 CSMA MAC on TinyOS-2.0.2.2, with a packet size of $\sim 40 bytes$.

itself); r_i the rate at which data generated at source node i is being transmitted; and B_i the value of node i's receiver capacity. The receiver capacity constraint at a node i is then given as follows:

$$\sum_{j \in N_i} \sum_{k \in C_j} r_k \leq B_i \quad (1)$$

We explain this with an example. Figure 3 shows an 8 node topology. The solid lines indicate a parent-child relationship in the tree. The dashed lines are interference links. Rates indicated on interference links [1] quantify the amount of interference generated by a neighboring node when it is transmitting data to its parent. Thus, when node 2 sends its data to node 1 at some rate, node 2 not only consumes the corresponding amount of capacity at node 1 but also at node 3; the rate label on interference link $2 \rightarrow 3$ is the same as that on link $2 \rightarrow 1$.

Based on our model, the constraint on the rates at node 2 would be as follows:

$$r_2^{tot} + r_3^{tot} + r_4 + r_5 \leq B_2 \quad (2)$$

where B_2 is the receiver capacity of node 2 and r_4 and r_5 are the source rates of nodes 4 and 5. r_2^{tot} and r_3^{tot} are the output rates at node 2, node 3 and are given by $r_2^{tot} = r_2 + r_4 + r_5$, and $r_3^{tot} = r_3 + r_6$.

The half-duplex nature of the radios results in the term r_4 and r_5 appearing twice in equation (2). Constraints similar to equation (2) can be obtained for every node in the example topology presented in Figure 3, using equation (1).

In order to make the receiver capacity model applicable to a real system, we need a good estimate of receiver capacities B_i. Since most sensor networks today use a randomized CSMA MAC as the de facto data link layer, we need to adapt the receiver capacity model to work over a CSMA MAC. In the next section we will show how the value of B_i can be estimated for the specific case of the CC2420 CSMA MAC.

2.1 Receiver capacity and the CC2420 CSMA MAC

Our implementation is performed on the Tmote sky mote [1], which uses the CC2420 radios, running TinyOS-2.0.2.2. Figure 4 presents the empirically measured saturation throughput for this platform as the number of senders in-range is varied. The saturation throughput [4] of a CSMA MAC is defined as the throughput observed by the receiver when all senders are backlogged and are within each others interference range. In order to associate a value to the receiver capacity, we equate the capacity of a receiver with the saturation throughput of the CSMA MAC. The saturation throughput is used to represent the 1-hop capacity of the CC2420 CSMA MAC.

The reason we believe that the saturation throughput of the CC2420 CSMA MAC is an acceptable estimate for the receiver capacity is as follows: It has been shown in [20], that as long as we choose an estimate of $B_i \leq \frac{L}{3}$, where L is the link rate presented by the physical layer, the rates presented by the receiver capacity model can be globally TDMA scheduled. The link rate presented by the CC2420 radio is 240 $kbps = 30$ $kBps$. For a packet size of ~ 40 bytes, this amounts to a link rate of 750 packets per second. If we were to use an ideal TDMA MAC, as shown in [20], setting each of the B_i to one-third of the link rate ($B_i = 250$ packets per second) will guarantee that the rate vector presented by the receiver capacity model can be TDMA scheduled. Since, the saturation throughput (~ 90 packets per second for greater than 3 neighbors) is much smaller than this required limit, and assuming the loss due to collisions is small making the CC2420 CSMA MAC behave as an inefficient TDMA MAC, the rate vectors obtained by setting the receiver capacity constraint to the 1-hop capacity of the CC2420 CSMA MAC should be achievable. The viability of the receiver capacity model for the CC2420 CSMA MAC will be further justified by our empirical results for WRCP, which we present in sections 7 and 8.

We wish to reiterate that the objective of this model is not to represent the exact rate region for WSN; instead it provides a tractable approximation that we show is good in practice. Also, we believe that the receiver capacity model can be easily used with other 802.15.4 radios, as well as with other CSMA-based radios with similar small packet-lengths, though this remains to be verified experimentally.

3 Software Architecture of a Rate Control Stack in TinyOS-2.x

In Figure 5, we present the software architecture that will be used to implement a rate control stack over TinyOS-2.x.

[1] For now, we assume links are perfectly bidirectional. In the WRCP protocol we will relax this assumption and handle lossy/asymmetric links.

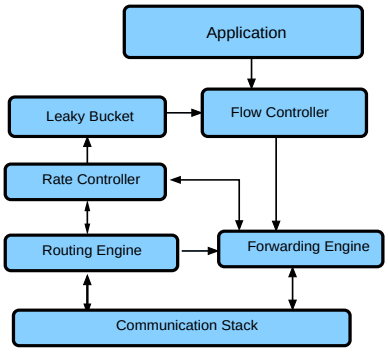


Figure 5. Software Architecture for WRCP.

TinyOS 2.x already provides a framework for building collection trees in the form of the *Collection Tree Protocol* (CTP) (TEP 123 [2]). Since the objective of this work is to design a rate control protocol that aims at achieving lexicographic max-min fairness among sources over a collection tree, the architecture is designed to integrate the rate control protocol with CTP. In Figure 5, the "Routing Engine" and the "Forwarding Engine" blocks are implemented by CTP.

For our implementation, a modification was made to the forwarding engine of CTP since the default forwarding engine of the collection tree protocol does not implement a FIFO queue. It implements a form of priority queuing which restricts the number of packets originating from an application in the node itself to one, giving higher priority to flows from its descendants. Since our algorithm explicitly assumes that the forwarding engine treats all flows equally, we implemented a variant of the forwarding engine that implements a FIFO queue.

In TinyOS-2.x, the communication block of the software architecture is usually implemented using a CSMA MAC (CC2420 CSMA for the Tmote sky platform). Given the scarce resources in these networks, we wanted to optimize the bandwidth used by the control traffic. Thus, we added fields to the existing CSMA MAC header, to piggy-back the rate control information along with data packets. By piggybacking the rate control information with the data packet, neighbors of a node can obtain control information by simply operating in promiscuous mode.

In our software architecture, the core functionality of the rate control protocol is implemented in the "Rate Controller", "Flow Controller" and "Leaky Bucket" blocks. The Rate Controller ascertains the sustainable rate for flows originating at this node, and sets the token generation rate in the leaky bucket [3]. The Flow Controller then uses tokens from the leaky bucket to admit packets from the application into the system. The Rate Controller interfaces with the Forwarding Engine in order to insert rate control information into MAC header of outgoing packets, in order to exchange this information with its neighbors.

4 The Wireless Rate Control Protocol

While describing the receiver capacity model in Section 2, we assumed an idealized setting with constant-bit-rate static flows from backlogged sources and lossless links. A real world sensor network on the other hand would have asynchronous communication, lossy links and dynamic flows which might result from the on-demand nature of the sensing application. To implement the algorithm in a practical setting, we need to relax these assumptions. To this end we have designed the **Wireless Rate Control Protocol** (WRCP) which incorporates a number of mechanisms to handle these real-world concerns.

The objective of WRCP is to achieve *lexicographic max-min* fairness [3] over a collection tree. A lexicographic max-min fair rate vector $\overrightarrow{r^*}$ is a rate vector, such that if the elements of $\overrightarrow{r^*}$ are arranged in ascending order, for any other feasible rate vector $\overrightarrow{\ddot{r}}$ whose elements are also arranged in ascending order, $\overrightarrow{r^*}$ will always be lexicographically greater than $\overrightarrow{\ddot{r}}$, *i.e.* there exists no j, such that $r^*_j < \ddot{r}_j$, given that $\forall\ i,\ 0 < i < j,\ r^*_i = \ddot{r}_i$. In WRCP, we achieve lexicographic max-min fairness by using a simple idea, that every receiver, at every time step, divides the available capacity at the receiver equally amongst all flows that are consuming capacity at the receiver. In an idealized sensor network setting it is feasible to show that an algorithm that incorporates such a bandwidth allocation policy will achieve a max-min fair rate vector using the proof techniques presented in [3]. We however omit this proof due to space constraints.

4.1 The WRCP Algorithm

The WRCP algorithm is a distributed algorithm, that runs on every node in the network. This algorithm is run by each node at a periodic interval of T seconds; in order to estimate the allowable rate for flows originating at the given node, and for flows traversing the neighborhood of the node. The algorithm, running at a node i at the n^{th} time-step, consists of the following three key steps:

- Calculating γ_i, the *per-flow available capacity*.
- Calculating γ_i^{min}, the *minimum per-flow available capacity*.
- Using γ_i^{min} to calculate the *allowable* source rate r_i for flows originating at node i, as well as for flows consuming the receiver capacity of node i.

At the end of a T second interval, in an idealized receiver capacity model with perfect links the *per-flow available capacity* at a node i can be calculated as follows:

$$\gamma_i((n+1)T) = \frac{B_i(nT) - \sum_{j \in N_i(nT)} r_j^{tot}(nT)}{\sum_{j \in N_i(nT)} F_j(nT)} \qquad (3)$$

Where $B_i(nT)$ is the receiver capacity of node i, r_i^{tot} is the rate at which a neighbor j is forwarding packets (both packets generated at node j, and packets routed through node j), F_i is the total number of flows in node i's neighborhood, and N_i is the set of neighbors of node i (including its one hop children). Equation (3) captures the essence of the simple idea described above, that of allowing WRCP to achieve max-min fairness by distributing the available capacity equally amongst the contending flows in a neighborhood. The numerator in equation (3), is simply the remaining capacity at node i, and the denominator is the total number of flows that are consuming capacity at i.

Though equation (3) captures the essence of the WRCP algorithm, we cannot use this equation as is in practice. Since the link quality from each neighbor will be less than perfect, in reality the consumption of capacity by a neighbor will be less than that predicted by equation (3) making the estimate of the *per-flow available capacity* overly conservative. Further, due to overlap with 802.11 channels the capacity estimation will need to take into account consumption of capacity due to external interference as well; hence, to make the estimate more practical we weigh the transmission rate of a neighbor, and the number of flows being forwarded by each neighbor by the packet reception rate p_{ji} from the neighbor j to the node i; we also introduce a new term called the *rate of external interference* (r_i^{ext}) to account for the capacity lost due to external interference. A more realistic estimate of $\gamma_i((n+1)T)$ can therefore be given by:

$$\gamma_i((n+1)T) = \frac{B_i(nT) - r_i^{ext}(nT) - \sum_{j \in N_i(nT)} p_{ji} r_j^{tot}(nT)}{\sum_{j \in N_i(nT)} p_{ji} F_j(nT)} \quad (4)$$

Once a node i has calculated its *per-flow available capacity* ($\gamma_i((n+1)T)$), it needs to calculate the *minimum per-flow available capacity* ($\gamma_i^{min}((n+1)T)$) in its neighborhood. It does this by comparing its per-flow available capacity with the per-flow available capacity of its neighbors, which the neighbors explicitly inform by tagging this information in the MAC header of data packets. All flows that are consuming capacity at this node can increment their rates by at most this amount, $\gamma_i^{min}((n+1)T)$. The need for calculating $\gamma_i^{min}((n+1)T)$ is that the *per-flow available capacity* ($\gamma_i((n+1)T)$) might be greater than that calculated by its neighbor j whose receiver capacity node i might be consuming; thus, if the node i allows flows that it's forwarding (originating from itself and its children) to increment their rates by the amount $\gamma_i((n+1)T)$ instead of $\gamma_i^{min}((n+1)T)$, it might exceed the receiver capacity of node j resulting in congestion.

Given $\gamma_i^{min}((n+1)T)$ node i updates the current source rate $r_i((n+1)T)$ of any source-application that is operating at this node, or any flow that is consuming the receiver capacity of this node. The way the rate update is performed depends on the following three cases; if $\gamma_i^{min}((n+1)T)$ is positive; if $\gamma_i^{min}((n+1)T)$ is negative and node i is a "*bottleneck node*", or if $\gamma_i^{min}((n+1)T)$ is negative and node i is *not* a "*bottleneck node*". A "*bottleneck node*" is a node j in the neighborhood of i such that $\gamma_j((n+1)T) = \gamma_i^{min}((n+1)T)$. For the first two cases, if $\gamma_i^{min}((n+1)T)$ positive, or if the node itself is the "*bottleneck node*", the source rate is updated using the following equation:

$$r_i((n+1)T) = r_i(nT) + \alpha \times \gamma_i^{min}((n+1)T) \quad (5)$$

where α is a constant set to a value less than 1. The need for α in equation (5) arises due to the lack of accurate time synchronization, which along with multi-hop communication can lead to inherent delays in the dissemination of rate control information, specifically the available capacity, across the network. This can lead to nodes constantly exceeding capacity, getting delayed feedback on congestion and reducing their rates by the excess capacity, exhibiting an oscillatory behavior. These oscillations are dampened by setting α in the rate update equation to a value smaller than 1; ensuring that nodes acquire available capacity slowly, allowing for convergence. In section 5 we show how to calculate the value of α for WRCP.

For the last case when $\gamma_i^{min}((n+1)T)$ is negative, and the negative $\gamma_i^{min}((n+1)T)$ has been learnt from another node, implying that node i is *not* the "*bottleneck node*" the current application-source rate r_i remains unchanged if it is less than the "allowed" *per-flow source rate* advertised by the bottleneck node; if, however, the source rate r_i is greater than that advertised, it is reset to the *per-flow source rate* advertised by the bottleneck node.

As can be seen from equation (4), to function properly the algorithm requires to estimate various parameter such as the update interval T, the total number of active flows $\mathcal{F}_i$ consuming capacity at node i, the sender link quality p_{ji}, and the amount of external interference (r_i^{ext}) in order to calculate the per-flow available capacity γ_i, and the minimum per-flow available capacity γ_i^{min}. In the remainder of this section, we describe different mechanisms implemented as part of the WRCP protocol in order to calculate these parameters, which are used as an input to the WRCP algorithm.

4.2 Rate update interval (T)

WRCP relies on a T second timer to present nodes with an interval to calculate their rate updates. In order to maintain a fair rate allocation, and system stability, it is essential that T be large enough to guarantee that rate control information has propagated to all nodes in the network within this update interval T. The value of T depends on the depth, and quality of links for a given collection tree. The larger the depth of the tree, the longer it will take to propagate the control information to the leaves of the collection tree; also, the poorer the quality of links, the larger the number of retries required to propagate the control information across a single link, resulting in a higher value of T.

Traditionally transport protocols on the Internet, such as TCP, XCP [12] and RCP [8], rely on the end-to-end reliability built into the protocols to obtain an RTT estimate for each source in the network. They then use this RTT estimate to determine the rate update interval. WRCP, similar to existing rate control protocols ([16, 15]) for sensor networks, however, does not have an end-to-end reliability mechanism. Hence WRCP needs to explicitly implement a mechanism to estimate this update interval for a given topology.

The mechanism implemented is as follows: the root maintains an exponential moving average of T, referred to as T_{avg}. Every T_{avg} seconds, on receiving a data packet the root generates a control packet (in a collection tree the root consumes all data packets and hence has to explicitly generate a control packet); it associates a control sequence number with this packet. The control sequence number is added to MAC header before broadcasting the packet. The root increments the control sequence number by one, if and only if it has received an acknowledgement from all its 1-hop children. A node sends acknowledgement to a specific control sequence

number as follows: if a node is a leaf node, it acknowledges every control sequence number it overhears by setting a control sequence acknowledge field in the MAC header of all its outgoing data packets. A parent, if it sees the control sequence acknowledgement field set on receiving a packet from its child, will set the control sequence acknowledgement field in the MAC header of its data packets if it has received an acknowledgement from all its 1-hop children. In this manner control sequence number acknowledgement gets aggregated at root of each sub-tree, and flows up the collection tree. The root will receive a control sequence acknowledgement for its current control sequence number, when all its 1-hop children received an acknowledgement from their respective sub-trees. On receiving acknowledgement for its current control sequence number it updates the exponential moving average T_{avg} as follows:

$$T_{avg} = \beta T_{avg} + (1-\beta) T_{inst} \tag{6}$$

Where T_{inst} is the estimate of T, based on the acknowledgement received for the current control sequence number. It will then propagate this estimate T_{avg} throughout the network, to keep rate updates in sync for all nodes.

4.3 Estimating Receiver Capacity (B_i)

As mentioned earlier, we approximate the receiver capacity by the saturation throughput, which is a function of the number of senders in the receiver's listening range. The saturation throughput is pre-calculated and stored as a lookup table. Figure 4 shows that the saturation throughput will almost remain a constant as long as the number of senders is greater than 4. It is important to note that while the current implementation of WRCP is specific to the CC2420 CSMA MAC, it can be adapted to any other CSMA MAC by empirically calculating saturation throughput values with different number of sender for the desired CSMA MAC, and creating a lookup table of these saturation throughput values to be used by WRCP.

4.4 Estimating Active Flow Counts ($\mathcal{F}_i$)

To calculate the available per-flow capacity (equation (4)), WRCP requires the number of active flows at a receiver. In a dynamic environment, the number of active neighbors and hence the number of active flows, in a given neighborhood is not constant. To handle the ephemeral flow count, an *active flow state* tag is associated with each neighbor entry. Aging this entry in the absence of packets from the neighbor helps give a good estimate of active flows in the network. The number of flows in a neighborhood determine the per-flow available capacity at a receiver. A conservative estimate can be obtained by simply looking up the total number of active sources that each neighbor is forwarding, without regard for the link quality with the neighbor. We consider this approach conservative since this approach assumes that all flows irrespective of their link quality will consume the same amount of capacity at the receiver. However, recent empirical results have shown that capture effects are quite dominant in these networks [19]. These results suggest that nodes with stronger links will cause more interference (or consume more capacity) than nodes with weaker links. Therefore, to have a more realistic estimate of the actual number of flows that are consuming capacity at a receiver i, we implement a heuristic which weighs the number of flows from a sender j to a receiver i by its link quality $p_{ji} \in [0,1]$. The active flow count at a node i is thus given by the following equation:

$$\mathcal{F}_i = \sum_{j \in N_i} p_{ji} F_j(nT) \tag{7}$$

Where F_j is the number of flows being forwarded by a node j.

4.5 Estimating Transmission Rates (r_i^{tot})

Another term required in the calculation of the available per-flow capacity (equation(4)) at a node i, is the current transmission rate r^{tot} of each neighbor. To cater for non-CBR traffic, we maintain an exponential weighted moving average of transmission rates as follows:

$$r_i^{tot} = (1-\beta) r_i^{tot} + \beta \frac{Pkts\ Transmitted}{1\ sec} \tag{8}$$

Pkts Transmitted are the total number of packets[2] sent out in 1 second, including retransmissions. Thus, the exponential moving average of the transmission rate is computed every second. Incorporating retransmissions into the calculations implicitly incorporates the affects of link losses into per flow available capacity calculations, since retransmissions due to link losses will result in a higher transmission rate forcing the receiver to advertise a smaller available capacity. An important point to be noted in equation (4) is that, as with the estimation of the active flow counts (section 4.4), the transmission rate used by a node j is also weighed by the link quality from j to i. The argument for implementing this heuristic of weighing the transmission rate by the link quality while estimating the remaining available capacity is the same as that presented in section 4.4, for calculating the active flow count.

4.6 Estimating Sender Link Quality (p_{ji})

WRCP needs the link quality between a sender and the receiver in order to estimate the per-flow available capacity (equation(4)). WRCP requires link quality estimate only in a single direction (from the sender to the receiver) simplifying the link estimation. Since every node is operating in promiscuous mode, the forwarding engine of node i maintains a variable rcv_{ji}, which count the total number of packets received from a sender j, for the last 10 packets that the sender had transmitted. Once it observes that the sender has sent out 10 packets, (which the receiver realizes with the help of a transmission counter that the sender sends along as part of the MAC header of data packets) the receiver updates the moving average estimate from a particular sender j as follows:

$$p_{ji} = \beta p_{ji} + (1-\beta) \frac{rcv_{ji}}{10} \tag{9}$$

After updating the link quality p_{ji}, the receiver resets the receiver counter to $rcv_{ji} = 1$.

[2]These packets include those originated at this node, as well as those being forwarder by this node.

4.7 Estimating External Interference (r_i^{ext})

IEEE 802.15.4, the de-facto standard PHY standard used in sensor networks, suffers from severe external interference by 802.11 networks due to spectrum overlap. WRCP predicts the amount of external interference by observing the queue size at a node. We believe the queue size represents a good estimate of the external interference, since the only case when WRCP rate predictions can go wrong is in the presence of external interference (since the receiver capacity model does not take external interference into account). To estimate the amount of external interference to be used in WRCP's rate calculations we therefore use the following mechanism; every node i maintains an exponential moving average of its forwarding queue size U_i. The external interference experienced by a node i is then given by the following equation:

$$r_i^{ext} = (1-\beta)r_i^{ext} + \beta\frac{U_i}{1\ sec} \tag{10}$$

As is evident, the above moving average is updated every 1 second. The external interference, along with the transmission rates of the neighbors (as well as the nodes own transmission rate) are used to calculate the available per-flow capacity, described in the next section.

4.8 Rate Bootstrapping for Flow Joins

If WRCP were to use equation (5) when a flow i joins the network, flow i might never get to increment its rate (or it might receive unfair rate allocation) if the network has been operational for a while, resulting in complete consumption of the network capacity. In such a scenario the new flow will see $\gamma_i^{min}((n+1)T) = 0$, not allowing the rate r_i to increment. In order to allow WRCP to adapt to flow dynamics we use a *bootstrapping* mechanism in which a new flow i enters the network in a phase called the *bootstrap* phase. In the *bootstrap* phase, a flow i joining the network uses equation 5 if $\gamma_i^{min} > 0$, else it uses the following rate update equation:

$$r_i((n+1)T) = 2 \times r_i(nT) \tag{11}$$

The *bootstrap* phase allows the new flow to increment its rate even if the remaining capacity has become negative. If the remaining network capacity is negative, this will force existing flows j to reduce their rates. The *bootstrap* phase for a flow i ends when its rate exceeds the per flow rate of the bottleneck node, while the remaining available capacity is still negative, i.e. when $\gamma_i^{min}((n+1)T) < 0$, and $r_i((nT)) > r_k$, where k is the *bottleneck* node. The end of the *bootstrap* phase indicates that the new flow i, has forced the existing flows j to reduce their rates making flow i's rate equal to its bottleneck flow rate.

5 Parameter Selection

The performance of WRCP, in terms of its stability, depends primarily on the parameters α and T. As has been described in section 4.2, the parameter T is determined dynamically by WRCP based on the topology and the routing tree. The mechanism used to determine the parameter T also ensures that the rate update interval for all sources in the network is homogeneous. Thus, the only tunable parameter required to guarantee the stability of WRCP is α. In this section we present an analysis to determine bounds on the parameter α that will guarantee a stable operation for WRCP.

The rate update equation used by WRCP is given by equation (5). If B_i is the receiver capacity at a bottleneck node i, the term γ_i^{min} can be given by :

$$\gamma_i^{min}((n+1)T) = \frac{\left(B_i - \sum_{j\in C_i} r_j(nT)\Gamma_j - \sum_{g\in N_i}\sum_{k\in C_g} r_k(nT))\Gamma_g\right)}{\mathcal{F}_i}$$

Here Γ_j is the expected number of transmissions required to successfully send a packet from a node j to its parent. Effectively, the second term in equation (5) has been replaced in the above equation, by a term that represents the remaining available capacity at bottleneck node i.

The CSMA MAC uses a stop and wait policy to ensure reliability at the data link layer. The term Γ_i, the number of packet transmissions required to ensure that a packet successfully transmitted from a node i to its parent, can be calculated using the following recursive equation:

$$\Gamma_i = (1+\Gamma_i)(1-p_i^f) + (2+\Gamma_i)p_i^f(1-p_i^r) + 2p_i^f p_i^r$$

Where p_i^f is the probability of successfully transmitting a packet from a node i to its parent, and p_i^r is the probability of successfully receiving an ACK from the parent. Solving the recursive equation, we can represent Γ_i in terms of the link quality p_i^f and p_i^r as follows:

$$\Gamma_i = \frac{1+p_i^f}{p_i^f p_i^r}$$

We now present some assumptions which help simplify our analysis. We replace the term Γ_j for each j, by Γ_{avg} where Γ_{avg} is the average packet transmissions between parent and a child for the entire network. For this analysis we assume nodes have perfect synchronization, and accurate information allowing all flows at the bottleneck node i to have the same per flow rate at any given instant of time. Thus, $r_k(nT) = r_i(nT)$, for each flow k that consumes capacity at node i. Also the number of active flows is assumed to be a constant in the network making $B_i(nT) = B_i$. Equation (5) can be rewritten as:

$$r_i((n+1)T) = r_i(nT) + \alpha \times (\frac{B_i}{\mathcal{F}_i} - r_i(nT)\Gamma_{avg}) \tag{12}$$

Based on equation (12) we state the following Lemma:

LEMMA 1. *A rate control protocol using equation (12) will converge if :*

$$0 < \alpha < \frac{2}{\Gamma_{avg}}$$

PROOF. Assuming that all sources start with minimum constant rate $r_i(0)$, due to the recursive nature of equation (12), we can rewrite equation (12) in terms of $r_i(0)$, B_i, p_{avg}, α and n as follows;

$$r_i(nT) = r_i(0)(1-\alpha\Gamma_{avg})^n + \alpha\frac{B_i}{\mathcal{F}_i}\left(\sum_{k=0}^{n-1}(1-\alpha\Gamma_{avg})^k\right) \tag{13}$$

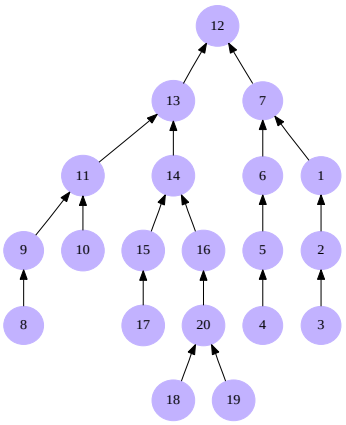


Figure 6. A 20 node topology.

Thus,

$$r_i(nT) = r_i(0)(1-\alpha\Gamma_{avg})^n + \frac{B_i}{\mathcal{F}_i\Gamma_{avg}}(1-(1-\alpha\Gamma_{avg})^n)$$

For $r_i(nT)$ to converge as $n \to \infty$, $0 < \alpha\Gamma_{avg} < 2$. Thus, for WRCP to converge it is essential that $0 < \alpha < \frac{2}{\Gamma_{avg}}$. □

6 Experimental Setup

Our implementation is performed on the Tmote sky motes, which have IEEE 802.15.4-based CC2420 radios, running TinyOS-2.0.2.2. The experiments are conducted over 20-node (Figure 6) and 40-node (Figure 7) topologies on the USC TutorNet testbed [13]. Experiments were conducted over a period of few months to ascertain any change in the performance of the protocols due to link quality variations in the topologies. The average link quality ranged from 30%-75%. It was observed that the link quality variations for the topologies over this large time frame were negligible, lending validity to the results. The experiments were conducted on channel 26 of 802.15.4 standards. The experiments were conducted on this specific channel to have an external interference free environment, allowing us to present reproducible results. In order to show that the protocol exhibits good performance on channels suffering from external interference as well, in Section 8.6 we present WRCP performance results in the presence of external interference. In all experiments, sources are generating CBR traffic.

7 Stability Analysis

In this section we present empirical evidence to validate the analysis presented in section 5, which is used in estimating the parameter settings for WRCP. In section 5 we had shown that for a given topology as long as $\alpha < \frac{2}{\Gamma_{avg}}$, where Γ_{avg} is the average number of transmissions between a node and its parent, WRCP will remain stable. In order to empirically justify this statement we ran WRCP on the two topologies shown in figures 6 and 7. For each of the topologies we varied the value of α from 0.05 to 1.0 and observed different metrics of performance for WRCP. The observed metrics were the variance of the allocated rate and the average end-to-end packet delay observed amongst all packets received

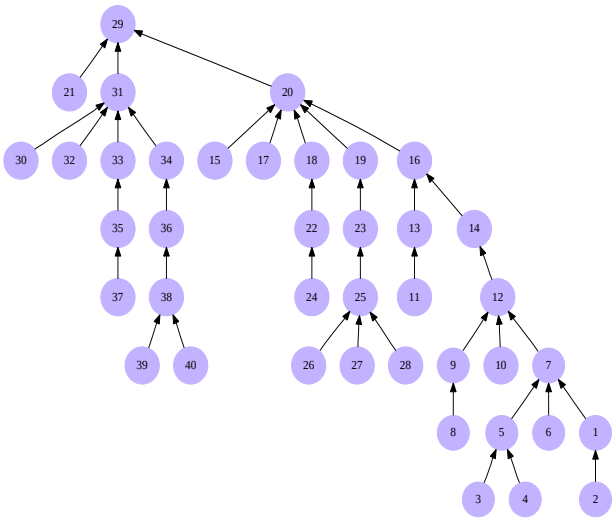


Figure 7. A 40 node topology.

Topology	Γ_{avg}	$\frac{2}{\Gamma_{avg}}$
20-node, Power = 5	4.2	0.476
20-node, Power = 10	5.61	0.3565
40-node, Power = 5	7.35	0.2721
40-node, Power = 10	12.05	0.1659

Table 1. The change in theoretical bound of α, with change in topology.

at the base station. For values of α for which WRCP is stable, the variance of the allocated rate should be small. For each of the topologies, these experiments were performed at two different power levels, at a power level of 5 and a power level of 10. For each of the two topologies Table 1 presents the estimated values of Γ_{avg}, and the bound on α, measured at different power levels.

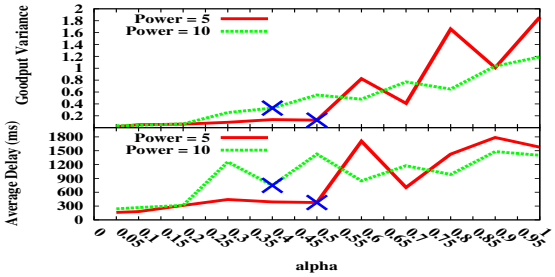


Figure 8. Evaluating behavior of WRCP with α for the 20-node topology.

As can be seen from the figures 8, and 9 the variance in the allocated rate rises quite sharply once α becomes greater then the corresponding value of $\frac{2}{\Gamma_{avg}}$, presented in table 1. Increase in the variance indicates that as $\alpha \to \frac{2}{\Gamma_{avg}}$ the system takes a longer time to converge, and once $\alpha > \frac{2}{\Gamma_{avg}}$ the variance becomes large, implying oscillations. This behavior is observed for the delay metric as well. The delay increases as

$\alpha \rightarrow \frac{2}{\Gamma_{avg}}$, and for values of $\alpha > \frac{2}{\Gamma_{avg}}$ the delay is higher than the delay when $\alpha < \frac{2}{\Gamma_{avg}}$. The increase in delay, for large values of α, is primarily due to the delayed feedback in the system. The delayed feedback results in nodes having stale information for their rate updates, resulting in erroneous increments and decrements. These erroneous increments regularly force the system to operate beyond the sustainable region, resulting in large queues and hence longer delays.

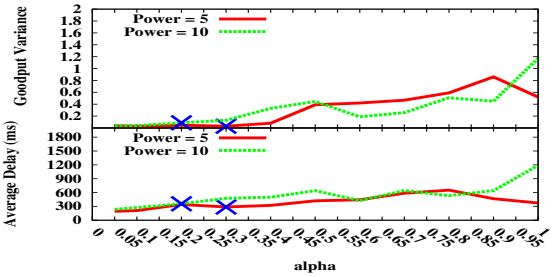


Figure 9. Evaluating behavior of WRCP with α for the 40-node topology.

The empirical evidence presented here validates our estimates for α, and proves that as long as $\alpha < \frac{2}{\Gamma_{avg}}$ the system remains stable.

8 Comparative Evaluation

In this section we present a comparative evaluation of WRCP with the Interference Aware Rate Control protocol (IFRC) [16], the state-of-the-art AIMD mechanism for rate control over a collection tree. The comparison of WRCP with IFRC highlights the advantages of having an explicit capacity based rate control protocol, as compared to one based on an AIMD mechanism, especially in a dynamic flow scenarios.

8.1 IFRC

Rate allocation in IFRC is split into two phases. When a source joins the network it starts in the *slow-start* phase. The slow start phase is similar to the slow-start in TCP. In this phase a source starts with a small initial allocated rate (r_{init}) and increases its allocated rate by a constant amount ϕ every $1/r_i$ seconds, where r_i is the current allocated rate of the source. Thus, in the slow start phase at every step the source increments its rate by $\phi \times r_i$ leading to an exponential increase. The slow-start phase continues till the source detects its first congestion event (average queues exceed a certain threshold). At this point the source enters the *additive increase phase*. In the additive increase phase the source starts with an initial value of $rThresh = \frac{r_i(t_{cong})}{2}$, where $r_i(t_{cong})$ is the source rate at the last congestion event. In the additive increase phase a source increments its rate by $\frac{\delta}{r_i}$ every $\frac{1}{r_i}$ seconds, leading to a constant increment of δ at every step. The details of each of these mechanisms and the methodology for parameter selection is given in [16].

As will be seen in our evaluation, for IFRC the speed of convergence, in terms of allocating the achievable max-min rate, to each source in the slow-start as well as the additive increase phase, depends on the initial values (r_{init} for slow-start and $r_i(t_{cong})$ for additive increase) and the maximum achievable rate.

IFRC was originally implemented over TinyOS-1.x. On performing experiments with the 1.x stack we observed a considerable difference between the throughput achieved by IFRC on 1.x [16] and WRCP in 2.0.2.2. The gap was due to the performance difference between the communication stack of 1.x, that had to be modified for enabling software ACK's required by IFRC, and the communication stack of 2.0.2.2. In order to have a fair comparison between WRCP and IFRC, we decided to port IFRC to TinyOS-2.x. In order to validate the porting of IFRC from TinyOS-1.x to 2.0.2.2, the behavior of the allocated rates observed in TinyOS-2.0.2.2 was compared to the ones presented in [16] and found to be the same, and performance of IFRC over TinyOS-2.x was found to be better than the performance of IFRC over TinyOS-1.x.

For all experimental results presented in this section, the size of the payload was 14 bytes. WRCP adds 16 bytes, whereas IFRC adds 26 bytes of overhead to each packet. Since both protocols exchange control information over the data path using a promiscuous mode of operation, WRCP exhibits lower protocol overhead.

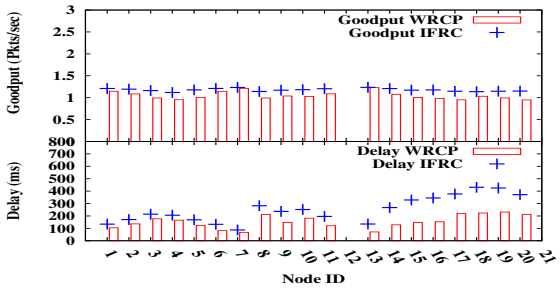


Figure 10. Goodput and end-to-end packet delays for 20-node *static* case.

For the purposes of comparison we have set the IFRC parameters $r_{init} = 0.1$ $pkts/sec$, $\phi = 0.0125$, $\varepsilon = 0.02$. The upper queue threshold was set to 20 packets. These parameters were calculated as per the analysis presented in [16] for a 40 node topology, since this is the maximum size network we are dealing with in our experiments. For WRCP we set $\alpha = 0.1$, as per the analysis presented in section 5, and the empirical evidence presented in section 7.

8.2 Evaluation Metrics and Hypotheses

We list the key metrics considered in the experimental evaluation, along with hypotheses/expectations regarding WRCP's performance on these metrics, given the design goals:

- **Goodput:** We have set MAC layer retransmissions to infinite, thus implementing hop-by-hop reliability; we therefore expect goodput of sources to match the allocated rates.
- **Packets delivered:** Since goodput is a long-term time average metric, it is not clear how we can quantify this metric for short flows which have small active lifetimes. Thus, in scenarios where there exist a combination of

short and long flows instead of comparing goodput, we compare the number of packets delivered by flows. Since, WRCP strives to achieve lexicographic max-min fairness, and has been designed to be responsive to network dynamics as compared to IFRC (in terms of the speed of informing sources of the available capcity) by using explicit capacity information; the expected behavior is that in highly dynamic scenarios (having a mix of short and long flows) flows using WRCP will deliver *lexicograpically* higher number of packets than IFRC.

- **End-to-End Packet Delays:** Since WRCP uses an approximation of the achievable network capacity, it should be able to operate the system within the capacity region, maintaining small queue backlogs and outperforming IFRC in terms of queueing delay. The End-to-End delay of a packet is measured by logging at each node, for the forwarded packet, the total delay (queueing + transmission) incurred at that node; adding the delays incurred at each hop during post processing of experimental traces allows us to calculate the end-to-end delay incurred by the packet.

8.3 Comparative Evaluation Methodology

In order to have comparable results from WRCP and IFRC, we ran IFRC and WRCP over the same topologies (Figures 6 and 7).

Initially we consider a scenario where all flows start at the same time, and all flows remain active for the entire duration of the experiment. We refer to this scenario as the static scenario. Since IFRC and WRCP both strive to achieve lexicographic max-min fairness, this scenario acts as a validation for the WRCP design and implementation. This scenario also highlights the advantage of using a rate control protocol (WRCP) that always makes the system operate with the rate region, in terms of the end-to-end packet delays.

We then consider dynamic scenarios where flows originate in the network at different times (hence the distinction with the static scenario). In this scenario flows are intermittent. Certain flows remain on for the complete duration of the experiment, while a few flows turn on only for a portion of the experiment. The dynamic scenario captures instances when short flows exist in the network. This scenario will present the advantage that a explicit rate control protocol exhibits over an AIMD protocol in terms of higher goodput, and hence better flow completion times. As mentioned earlier, it is important to note that faster flow completion times imply better energy utilization, since they result in shorter network uptime, conserving energy.

8.4 Static Scenario

In these experiments, all nodes except the root node (node 12 for the 20 node topology, and node 29 for the 40 node topology) are sources. All flows start at the beginning of the experiment. Once a flow starts, it remains active for the duration of the experiment which lasted approximately 900 seconds (15 minutes).

Figure 11 presents the rate allocation behavior of WRCP and IFRC on the 40-node topology (we omit the presentation of the rate allocation behavior for the 20-node topology due to the similarity of the behavior to the 40-node topology, and

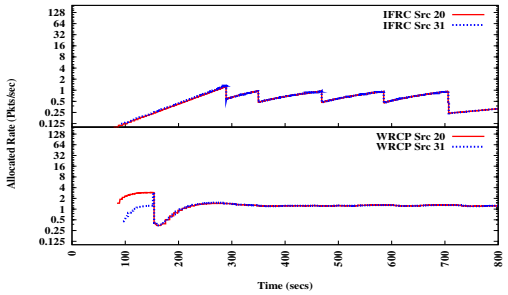


Figure 11. Rate allocation for 40-node *static* case.

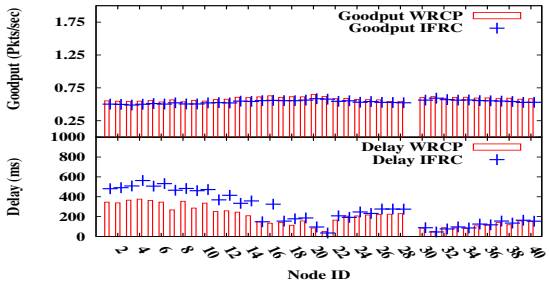
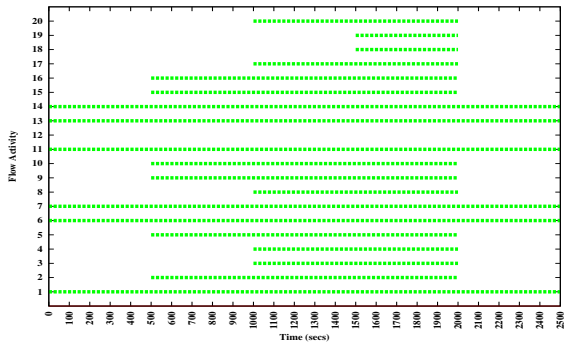


Figure 12. Goodput and delay performance for 40-node *static* case.

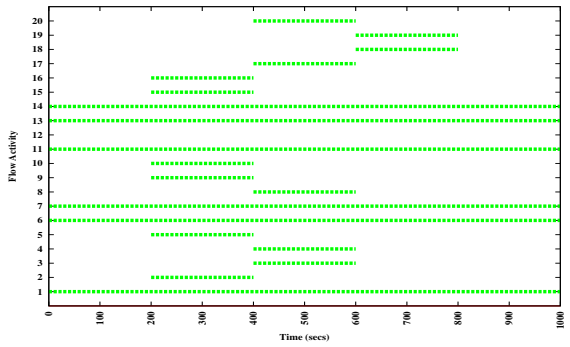
due to space constraints). For both topologies, it can be seen that the goodput of WRCP is better than or equal to that presented by IFRC (Figures 10 and 12). Given that IFRC has the same objective as WRCP to present a lexicographic max-min fair vector, WRCP should present the same or a lexicographically greater vector than IFRC. Thus, these results highlight the functional correctness of WRCP. The gains of WRCP in this setting are reflected in the end-to-end packet delay performance also presented in figures 10 and 12). Since WRCP uses explicit capacity information it allows the system to operate within the rate region. IFRC on the other hand needs to constantly exceed the capacity region in order to estimate the capacity. The constant probing of IFRC results in the protocol exhibiting higher queue sizes than WRCP, resulting in larger end-to-end packet delays.

8.5 Dynamic Scenario

In this section we deal with a more practical setting where the work load, in terms of the number of flows active in the network, varies over time. This represents a more dynamic environment. This setting will highlight the gains of using an explicit capacity based rate control protocol, over an AIMD-based protocol. The gains are primarily in terms of shorter flow completion times, which will in turn manifest themselves into energy savings. For each of the two topologies under consideration, we chose two different types of dynamic work loads to capture the different test scenarios. The two types of work loads, for each of the topologies is shown in figures 13 and 14. For each case the x-axis of the figures represent the duration of the experiment, and the horizontal bars when a source is active. In these experiments, when a source is active it continuously generates packets,



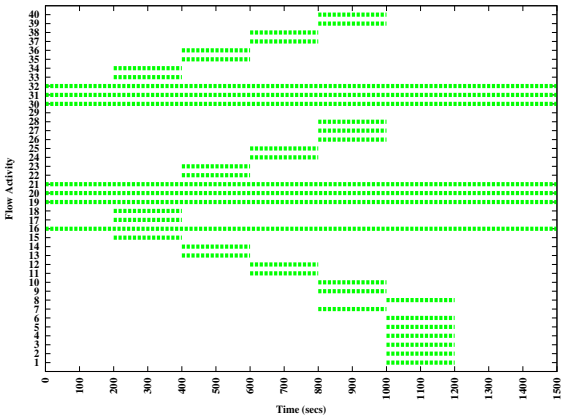
(a) Case 1



(b) Case 2

Figure 13. Flow activity for the 20-node topology.

(a) Case 1



(b) Case 2

Figure 14. Flow activity for the 40-node topology.

and tries to transmit these packets at the maximum rate allowed by the rate control protocol. Sources that are active for the entire duration of the experiment are representative of long flows, and sources that turn on and off intermittently during the experiment are representative of short flows. We believe this distribution of activation intervals allows us to emulate a mixed traffic of long and short flows.

8.5.1 Case 1

For case 1, both protocols, over both topologies, are able to adapt well to flow joins in the network (due to space constraints we present rate allocation behavior only for the 40-node topology in Figure 16) . Both protocols cut down source rates aggressively to avoid congestion collapse. The key difference in the protocol performance comes when flows depart the network. If a large number of flows are active in the network, the per-flow rate is quite small (1 pkt/sec for 19 active flows, and ~ 0.5 pkts/sec for 39 active flows). At this juncture if a majority of flows depart, suddenly a large amount of capacity becomes available for consumption. Such condition occurs at 2000 second for the 20 node topology, and at 2500 second for the 40 node topology. The problem with IFRC under this condition is that since its rate of additive increase depends inversely on the r_{thresh} values, the rate increments are small, and its takes a long time for IFRC to consume this freed up capacity. WRCP on the other has explicit knowledge of the available capacity, its increments being independent of the current source rates and dependent purely on the available capacity. WRCP thus easily outperforms IFRC in consuming this freed up capacity. This is reflected in the packets delivered by the long flows for both the 20-node, and the 40-node topologies (Figures 15 and 17).

A direct impact of the increase in higher number of packets delivered in the same time frame, is a much smaller flow completion time. For the 20 node topology, for *e.g.*, the sources 7 and 13 are able to send out 9000 packets in 2600 seconds using WRCP, as compared to only 6000 packets under IFRC. For the 40 node topology, the sources 20 and 31 are able to send out 10000 packets in 3600 seconds for WRCP, as compared to only 7000 packets under IFRC. As mentioned earlier, shorter flow completion times will require short network up-time, and hence will result energy savings. The end-to-end packet delay performance for IFRC and WRCP (Figures 15 and 17) for this dynamic case also reflects on the ability of WRCP to operate within the capacity region.

8.5.2 Case 2

Unlike case 1, in case 2 the duration of the short flows is comparatively shorter ($\sim$ 200 secs). The gains of having an explicit capacity rate control protocol, for improving flow completion times of short flows, are clearly evident in this scenario. The packets delivered by short flows using WRCP in both topologies (figures 18 and 19) is much higher than the packets delivered by short flows using IFRC. The long

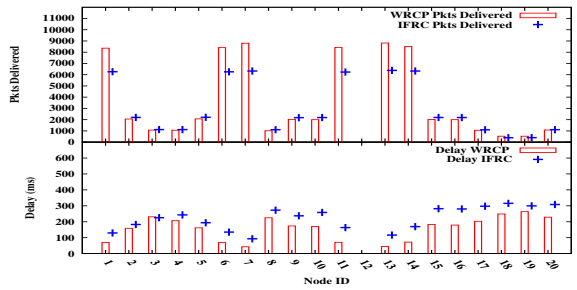


Figure 15. Packets delivered and end-to-end delay for dynamic scenario *(case 1)* on the 20-node topology.

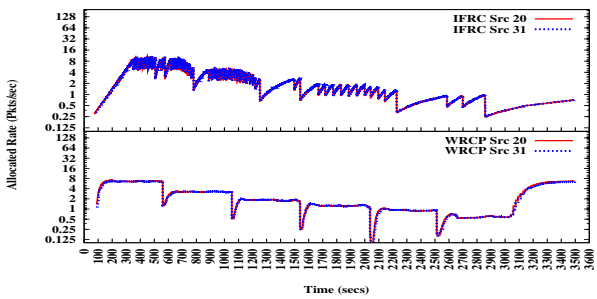


Figure 16. Rate allocation 40-node dynamic *(case 1)*.

flows in WRCP get a lower allocated rate, since WRCP is more fair and allows a higher rate for the short flows. IFRC on the other hand gives very high rates to long flows and starves the short flows. In short WRCP gives a higher *lexicographic* max-min fair solution than IFRC. As mentioned earlier the increase in number of delivered packets will also result in comparatively shorter flow completion times for the short flows.

To get a perspective on the gains exhibited by WRCP over IFRC, in terms of the flow completion times, *e.g.*, for the 20-node topology sources 2 and 16 are able deliver 450 packets in 200 seconds using WRCP, compared to only 50 packets using IFRC; for the 40-node topology, sources 17 and 33 are able to deliver $\sim$ 500 packets in 200 seconds compared to 50 packets using IFRC. The delay performance of WRCP is far superior to IFRC for the 20 as well as the 40 node topologies.

The two cases for the dynamic scenario exhibit the gains

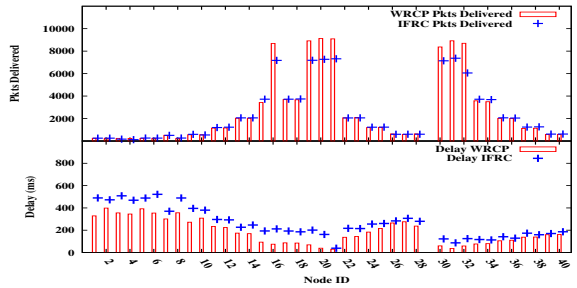


Figure 17. Packets delivered and end-to-end delay for dynamic scenario *(case 1)* on the 40-node topology.

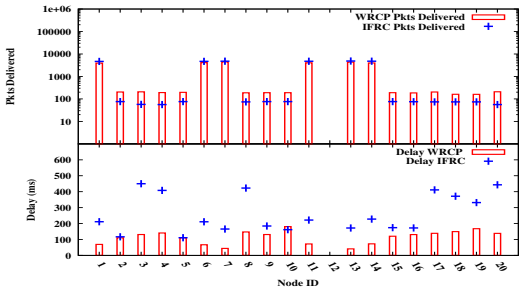


Figure 18. Packets delivered and end-to-end delay for dynamic scenario*(case 2)* on the 20-node topology. The y-axis for the packets delivered is in *log* scale.

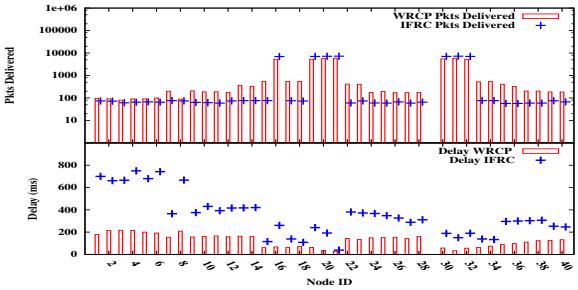


Figure 19. Packets delivered and delay for dynamic scenario*(case 2)* on the 40-node topology. The y-axis for the packets delivered is in *log* scale.

that WRCP presents for short flows as well as long flows in terms of flow completion times and delays.

8.6 WRCP performance in the presence of external interference

In section 4.7, we described how WRCP uses the forwarding queues at a node to predict the amount of external interference. We validate this design choice in this section.

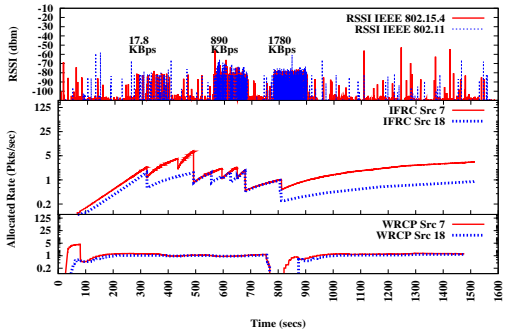


Figure 20. Rate allocation behavior with controlled external interference from an 802.11 source operating at channel 14 (2.482 GHz).

Figure 20 shows the rate allocation behavior for WRCP and IFRC on the 20-node topology in the presence of 802.11 interference. Recall that we are performing these experiments on channel 26 of 802.15.4. Only channel 14 of 802.11 can cause interference in channel 26 of 802.15.4. For these set of experiments we therefore operate an 802.11 radio,

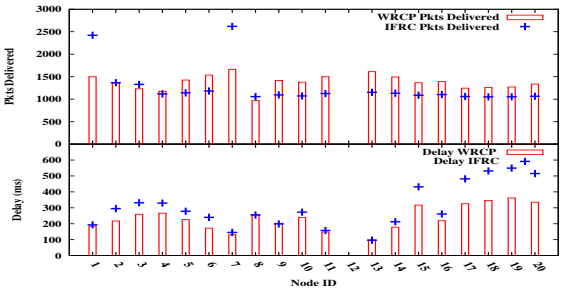


Figure 21. Packets delivered and end-to-end packet delay with external interference for 20-node topology.

close to node 1, on channel 14, transmitting UDP packets of size 890 bytes, in order to emulate external interference. It is interesting to note that it is not only the power level of the interference, but also the rate at which this interference is generated that affects the achievable rates of a sensor network. This can be seen between 250-400 seconds, 550-700 seconds and 750-900 seconds. During these intervals the power of the external interference (802.11) was much higher than the power at which the 802.15.4 radios; however, the rate at which this external interference was being generated was varied (17.8 KBps for 250-400 seconds, 890 KBps 550-700 seconds, 1780 KBps for 750-900 seconds). As can be seen in the rate allocation curve, at a lower rate (17.8 KBps) the external interference hardly affects the achievable rate of a sensor network, but as the rates start increasing the achievable rate starts going down; with the sensor network being completely shut down when the external interference starts operating at a high rate of 1780 KBps.

Both IFRC and WRCP adapt their rates based on the amount of external interference. IFRC relies on coarse grained binary feedback asking nodes to simply cut their rates by half when it observes congestion. In the presence of external interference there is a high probability of these control signals being lost, resulting in nodes decrementing their rates by different amounts leading to asynchronous and unfair rate allocation between nodes. This can be seen in the rate allocation curve of Figure 20. The affect of this lack of synchronization can be seen in the number of packets delivered (Figure 21), resulting in WRCP presenting a lexicographically higher delivery rate than IFRC. Further, Figure 21 shows that even in this scenario with external interference WRCP outperforms IFRC in terms o end-to-end packet delay due to the use of explicit capacity information, which results in much smaller queue sizes when using WRCP as compared to IFRC.

9 Related Work

Given the constraints on resources in wireless sensor network, it has been shown that congestion control algorithms are essential for the operational effectiveness of these networks [15]. Given the importance of this problem, there have been a series of proposals aiming to mitigate the affects of congestion in a WSN. We summarize some key papers briefly below: ARC [23] proposes an AIMD rate control strategy whereby the nodes increase rates proportional to the size of their sub tree and performs multiplicative decrease on sensing packet drops. ESRT [18] is a sink-centric approach that measures available capacity, and allows for rate increments and decrements, by observing the ability of sources achieve a certain event detection reliability target. CODA [21] congestion control mechanism [21] provides for both open-loop hop-by-hop back-pressure and closed-loop multi-source regulation whereby sources vary their rate based on feedback from the sink. FUSION [10] uses a token based regulation scheme that allows for additive increase of source rates. It detects congestion using queue lengths and mitigates congestion by a combination of hop by hop back pressure and an adaptive MAC back-off scheme. In the work by Ee and Bajcsy [9], each node determines its own average aggregate outgoing rate as the inverse of its service time for packets and shares this rate equally amongst the nodes served in the subtree. This scheme does not always achieve a max-min fair solution as information is not shared explicitly with neighbors. IFRC [16] is a state of the art distributed approach that mitigates congestion by sharing congestion information explicitly with the set of all potential interferes of a node, and uses AIMD to react to the feedback. However, its design focuses primarily on achieving steady state fairness rather than rapid convergence or low delay. RCRT [15] is a recent centralized scheme where the sink employs an AIMD mechanism to calculate achievable rates and explicitly informs the sources as to the rate as which they can transmit. Though RCRT is more recent protocol than IFRC, we choose to compare WRCP against IFRC; since, like IFRC RCRT is also an AIMD-based protocol and will suffer from the same drawbacks that IFRC does when comparing against WRCP, making the comparison redundant.

While all these schemes are rate-based, router-centric (with the exception of sink-centric ESRT and RCRT), and most of them use explicit feedback, they differ greatly from WRCP. The common theme in most of these proposals is that they use AIMD based mechanisms to perform rate control, while WRCP takes a different approach by using explicit and precise feedback regarding the available capacity, in order to provide rapid convergence and low end-to-end delays.

The idea of using explicit/precise feedback regarding available capacity to perform congestion control has been explored in the wired context. There exist prior works in the ATM network literature where mechanisms have been proposed for providing explicit and precise congestion feedback using the resource management (RM) cells for ABR (available bit rate) traffic ([5], [11], and [14]). In the Internet context, recent protocols such as XCP [12] and RCP [8] have highlighted the gains of using precise feedback using network capacity information, as compared to traditional AIMD approaches followed by TCP and its many variants. XCP [12] is a window based protocol that presents improved stability in high delay bandwidth networks, and RCP is a rate based protocol that improves the flow completion times for short flows. In a multi-hop wireless setting, WCPCAP [17] is a distributed rate control protocol that can achieve max-min fairness using explicit capacity information. The key difference between WCPCAP and WRCP is that WCPCAP relies on a model that is very specific to an 802.11 multi-hop

network. It is not clear how this model can be ported to a sensor network setting. WRCP on the other uses a very simple model, that we show works well in the context of a CSMA MAC for sensor networks. Further, WCPCAP does not cater for external interference, or present validation for its parameter settings, whereas as has been demonstrated WRCP works well in the presence of external interference, and the parameter settings for WRCP are well understood. WRCP is similar in spirit to RCP in its design and goals, since it is a rate based protocol and attempts to shorten the flow completion times by explicitly allocating rates based on available capacity in the network. The key difference between RCP (as well as XCP) and WRCP is that in the wired context, to keep the system scalable, the core challenge is to perform a router centric explicit and precise congestion notification without maintaining any flow state information. In the wireless sensor network context, flow states can be maintained (due to the potentially small number of flows), but the main challenge is how to estimate the available capacity.

Our design of WRCP has been enabled by the use of the *receiver capacity model*. It quantifies the capacity presented by a receiver to flows that are incident on the receiver, and presents constraints on the receivers that defines the bandwidth sharing that takes place between flows incident on a receiver. The model is particularly useful in our setting, since it caters to a CSMA based wireless sensor network. There are other interference models in the literature. Among the most commonly used models are graph based models, such as the clique capacity model [7], and link based models such as the SINR model [6] and the ALOHA model [22]. We believe these models, which have been largely used in theoretical studies, are not well suited to CSMA-based networks. For the clique-capacity model it is hard to determine all the possible cliques in a distributed manner; the SINR model is more akin to modeling MAC's with variable bit rate radios; the ALOHA model is very specific to the ALOHA MAC.

10 Conclusions

We have presented the design and implementation of the **Wireless Rate Control Protocol**, which is the first protocol to use explicit feedback based on precise capacity information to achieve a max-min fair rate allocation over a collection tree. Through a comparative evaluation with IFRC [16] we have shown the advantages of using explicit capacity information in designing rate control algorithms when compared to AIMD approaches, in terms of flow completion times, goodput and end-to-end packet delays.

11 Acknowledgements

This work was supported in part by NSF grants CNS-0627028, CNS-0347621, CNS-0325875, and the USC Annenberg fellowship. Also, we thank our shepherd Andreas Terzis whose valuable inputs and detailed comments helped us greatly improve the quality of our presentation.

12 References

[1] *http://www.moteiv.com*.

[2] *http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html*.

[3] Bertsekas and Galagher. Data networks. *Prentic Hall*.

[4] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE JSAC*, 18:535–547, March 2000.

[5] F. Bonomi and KW Fendick. The Rate-based Flow Control Framework for the Available Bit Rate ATM service. *Network, IEEE*, 9(2):25–39, 1995.

[6] M. Chiang. Balancing Transport and Physical Layers in Wireless Multihop Networks: Jointly optimal congestion control and power control. *IEEE JSAC*, 23(1):104–116, 2005.

[7] C. Curescu and S. Nadjm-Tehrani. Price/utility-based Optimized Resource Allocation in Wireless Ad-hoc Networks. *IEEE SECON 2005*.

[8] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor Sharing Flows in the Internet. *IWQoS*, 2005.

[9] C. T. Ee and R. Bajcsy. Congestion Control and Fairness for Many-to-One Routing in Sensor Networks. *ACM Sensys*, 2004.

[10] B. Hull, K. Jamieson, and H. Balakrishnan. Techniques for Mitigating Congestion in Sensor Networks. *ACM Sensys*, 2004.

[11] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore. The ERICA switch algorithm for ABR traffic management in ATM networks. *IEEE/ACM Transactions on Networking (TON)*, 8(1):87–98, 2000.

[12] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. *ACM SIGCOMM*, 2002.

[13] Embedded Networks Laboratory. *http://testbed.usc.edu*, 2007.

[14] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, and H. Miyahara. Rate-based congestion control for ATM networks. *ACM SIGCOMM Computer Communication Review*, 25(2):60–72, 1995.

[15] J. Paek and R. Govindan. RCRT: Rate-controlled Reliable Transport for Wireless Sensor Networks. *ACM Sensys*, 2007.

[16] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-Aware Fair Rate Control in Wireless Sensor Networks. *ACM SIGCOMM*, 2006.

[17] S. Rangwala, A. Jindal, K.Y. Jang, K. Psounis, and R. Govindan. Understanding congestion control in multi-hop wireless mesh networks. In *ACM MobiCom*, pages 291–302, 2008.

[18] Y. Sankarasubramaniam, O.B. Akan, and I.F. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. *ACM MobiHoc*, 3:177–188, 2003.

[19] D. Son, B. Krishnamachari, and J. Heidemann. Experimental Analysis of Concurrent Packet Transmissions in Low-PowerWireless Networks. *ACM Sensys*, 2006.

[20] A. Sridharan and B. Krishnamachari. TDMA scheduling feasibility of the Receiver Capacity Model. *WiOpt*, 2009.

[21] C.Y. Wan, S.B. Eisenman, and A.T. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks. *ACM Sensys*, 2003.

[22] X. Wang and K. Kar. Cross-layer Rate Control in Multi-hop Wireless Networks with Random Access. *IEEE JSAC*, February 2005.

[23] A. Woo and D.E. Culler. A Transmission Control scheme for Media Access in Sensor Networks. *ACM MobiCom*, pages 221–235, 2001.