

# Resource-Efficient Dynamic Channel Reservations for Real-Time Streams in Wireless Multihop Networks

Jun Yi, Christian Poellabauer, Xiaobo Sharon Hu, Thidapat Chantem  
Computer Science and Engineering, University of Notre Dame  
{jyi, cpoellab, shu, tchantem}@nd.edu

Liqiang Zhang  
Computer and Information Sciences, Indiana University South Bend  
{liqzhang@iusb.edu}

## Abstract

Many wireless ad-hoc networks, such as sensor networks and networked embedded control systems, must support data dissemination with strict end-to-end latency constraints. In addition, such networks are often resource constrained, e.g., they should support energy management techniques such as coordinated sleep/wakeup mechanisms. Existing solutions to the real-time streaming problem are inefficient in resource consumption and have large coordination overheads (such as TDMA-based approaches), or they are unpredictable, i.e., they can only support soft real-time systems (such as contention-based approaches). This paper introduces DARTS, a protocol for dynamic resource allocations for real-time streams in static wireless multihop networks. With DARTS, each node dynamically negotiates (i) channel accesses for data transfers to its neighbors (for all real-time streams going through this node) and (ii) future negotiation points, thereby ensuring contention-free communication and negotiation. DARTS also performs end-to-end admission control along a stream's route to ensure that all deadlines will be met under ideal network conditions.

## 1 Introduction

Many wireless multi-hop networks carry streams of data with time-critical information (e.g., video streams in surveillance networks or sensor streams in monitoring and control applications), i.e., all packets in a stream must reach their destination before their end-to-end deadlines [15]. In addition, these networks often operate in resource-constrained environments, necessitating energy-conscious computations and communications. Although there exist numerous data dissemination approaches focusing on

conserving energy or improving the Quality-of-Service of communication, there is a dearth of research efforts focusing on meeting all packets' end-to-end deadlines for high-assurance time-critical applications. TDMA-based schemes have been proposed to provide energy-awareness and to satisfy timeliness requirements [4–6], but they are inflexible, bandwidth-inefficient, sensitive to time synchronization errors, and they incur large coordination overheads in multihop networks. Contention-based approaches that rely on prioritization [1, 7, 10] can only provide probabilistic real-time performance even if the channel conditions are perfect. Moreover, they require nodes to stay active to continuously sense or listen to the channel and are therefore not energy-efficient. Hybrid approaches [12, 14] usually adopt a duty-cycle approach, adapt a node's sleep/wake states based on traffic status, and contend for a slotted channel with certain preferences and prioritizations, but they usually aim for low energy consumption at the cost of increased latencies.

This paper proposes *DARTS (Dynamic Allocations for Real-Time Streams)*, a protocol combining TDMA-like contention-free channel accesses with the benefits of scheduling-based prioritized approaches. In DARTS, the TDMA approach of allocating fixed (and bandwidth-inefficient) slots is replaced by a dynamic reservation mechanism, while maintaining the contention-free channel access property of TDMA. Scheduling-based utilization analysis is used to ensure that there are sufficient transmission opportunities to guarantee that end-to-end deadlines are met under ideal channel conditions. As a consequence, DARTS more efficiently utilizes resources compared to TDMA-based solutions and more effectively provides end-to-end real-time traffic management compared to TDMA-based, contention-based, and hybrid approaches. In addition, our Linux-based implementation of DARTS indicates that it is able to achieve real-time and energy results comparable to

the ones obtained through simulation, even when DARTS is implemented on top of the 802.11 MAC protocol.

## 2 Related Work

An early work studying the issue of real-time communication in sensor networks is RAP [10], where a deadline- and distance-aware velocity monotonic scheduling (VMS) algorithm was proposed. In contrast, DARTS employs a reservation-based (instead of prioritization-based) technique to assure end-to-end timeliness. SPEED [7] uses a feedback control technique to provide soft real-time communication service and focuses on routing for soft real-time communication, whereas DARTS focuses on message scheduling to meet end-to-end deadlines. Both RAP and SPEED require nodes to continuously listen to the channel and do not take energy consumption into account, whereas in DARTS nodes only wake up during reserved channel access intervals to transmit/receive packets and to listen to channel access negotiations of their neighbors.

Exploiting channel diversity to support soft real-time data flows in multihop wireless ad-hoc networks was proposed in [2], where a prioritized MAC protocol called RT-Chains was presented. With RT-Chains, a route is established for every stream, where potentially interfering links use different channels, thus avoiding packet collisions. In contrast, DARTS addresses the problem of avoiding interferences on a single channel, but could be extended to multichannel scenarios in a manner similar to RT-Chains.

I-EDF [3] is a scheme that exploits the transmission structure to realize collision-free real-time scheduling, where a hexagonal cellular architecture partitions the networks into cells and EDF is used in each cell to schedule real-time data. RI-EDF [5] is a table-driven protocol for timeliness and energy-efficiency, where packets are transmitted following an EDF schedule. RI-EDF is a protocol without a central point of failure, however, it assumes that the network is fully connected. CR-LST [8] is a centralized mechanism to schedule messages by carefully exploiting spatial channel reuse for each per-hop transmission to avoid MAC-layer collisions. A leader node periodically announces transmission schedules to follower nodes in a multihop network. DARTS can be considered as a distributed version of CR-LST with an explicit admission control procedure that guarantees timely end-to-end communications.

Numerous energy-conserving protocols for wireless sensor networks have also been proposed, e.g., MAC-layer solutions include S-MAC [14], Z-MAC [12], and TRAMA [11]. All these protocols adopt a duty cycle approach, trading off throughput and latency for energy savings. They are not concerned with latency constraints, although they use various heuristics to avoid indefinitely prolonging delays. In contrast, DARTS takes both latency and

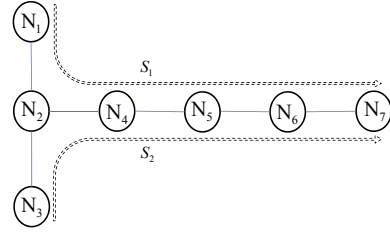


Figure 1. Topology used for discussion in this paper.

energy-efficiency as primary objectives. Moreover, for best-effort messages, DARTS behaves like an energy-efficient protocol, while still guaranteeing the timeliness of all real-time streams.

## 3 Preliminaries

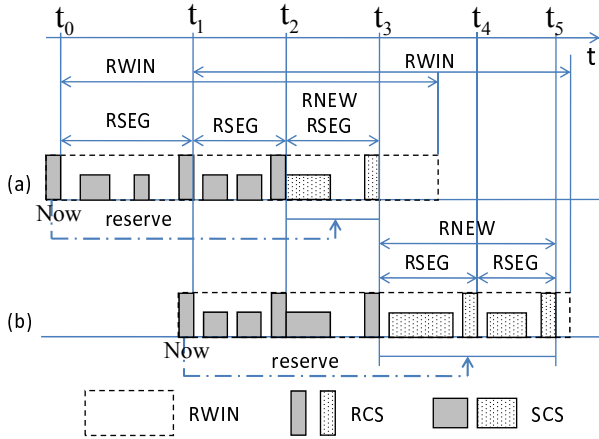
### 3.1 Traffic and Network Model

We consider a network of wireless nodes  $\{N_1, \dots, N_i, \dots, N_m\}$  generating and forwarding a set of periodic real-time streams  $\{S_1, \dots, S_n\}$ . Each stream  $S_v$  is characterized by a tuple  $S_v = (p_v, e_v, D_v, \mathcal{R}_v)$ , where  $p_v$  is the period of  $S_v$ ,  $D_v$  is the end-to-end deadline for each packet in  $S_v$ ,  $e_v$  is the worst-case datagram length (equal to the time necessary to deliver the largest possible datagram across links en route), and  $\mathcal{R}_v$  represents the route from the source to the sink, which is a sequence  $(N_{v_1}, \dots, N_{v_n})$  of nodes along the stream (with  $N_{v_1}$  and  $N_{v_n}$  being the source and the sink, respectively).

In this work, routing and scheduling are considered to be separate tasks, i.e., we assume that a route has been determined a priori and this paper is concerned with the end-to-end admission control and per-hop scheduling along that route. Figure 1 shows an example of two periodic streams going through a multihop network (we use this example for our discussions in the remainder of this paper).

Further, we assume a network with one channel and little to no node mobility. The interference model is bidirectional as, e.g., the 802.11 standard, i.e., when two nodes A and B communicate, all other nodes in the communication range of either A or B cannot receive or transmit. Distances between links or nodes are an important consideration in determining interference. In this paper, we express all distances as *link distances*. That is, two links are considered to be  $n$ -link ( $n \geq 0$ ) away if a node of one link can reach a node of the other link via at least  $n$  links. Similarly, a link is  $n$ -link away from a node if the node can reach a node of the link via at least  $n$  links. We call the set of links that are *within*  $n$ -link distance from





**Figure 3.** An illustration of CS reservations.

fore, to simplify discussion, we consider links to be active resources that negotiate reservations and perform stream transmissions (instead of nodes). In DARTS, a link utilizes an RCS to reserve future CSs without the risk of interference. The window between two consecutive RCSs of a link is referred to as *Reservation Segment (RSEG)*. A second window, called *Reservation Window (RWIN)*, is used to describe how far an RCS can look ahead to perform reservations. Consider the example in Figure 3(a). Here, a taller bar indicates an RCS and a shorter bar indicates an SCS. At the RCS marked with “Now”, a link already has two RSEGs filled with reservations (indicated by the filled bars between  $t_0$  and  $t_2$ ) and the goal is to add one or more RSEGs up to RWIN (indicated by shaded bars). The sum of all RSEGs that are negotiated in this RCS is referred to as RNEW (between  $t_2$  and  $t_3$  in this example). At the next RCS (Figure 3(b)), RNEW then consists of RSEG=  $[t_3, t_4]$  and RSEG=  $[t_4, t_5]$ . This process is repeated at each RCS.

Key to interference-free communication in DARTS is that all nodes that could potentially make conflicting reservations are made aware of new reservations before they start their negotiations. One-hop neighbors can simply stay awake during their neighbors’ RCSs to *overhear* their negotiations. Other nodes multiple hops away will learn about new reservations when they overhear their neighbors’ negotiations or when they perform reservations themselves (details are discussed in Section 4.2.2). DARTS must ensure that (i) no two potentially interfering links can make reservations within two overlapping RNEWs simultaneously, (ii) there is sufficient amount of time to propagate new reservations to all other links that may potentially reserve conflicting CSs, and (iii) any CS reserved by a link must be relayed to all its interfering links before the CS occurs. We achieve these by judiciously selecting the values for RWIN, RNEW, and RSEG (details are given in Section 4.2.1).

Finally, to ensure end-to-end timeliness of every admit-

ted stream, DARTS employs a real-time stream admission control mechanism. This mechanism guarantees that at least a certain amount of idle time within a specific time range (referred to as *local deadline*) will be set aside for the stream under consideration. Also, the accumulated local deadlines will be no greater than the end-to-end deadline required by the real-time stream. Since a new stream may impact the timeliness of all admitted streams, DARTS performs a *neighborhood admission control* on a link-by-link basis, which also results in the assignment of appropriate relative local deadlines (details of the admission scheme are provided in Section 4.3).

## 4.2 Dynamic Channel Reservations

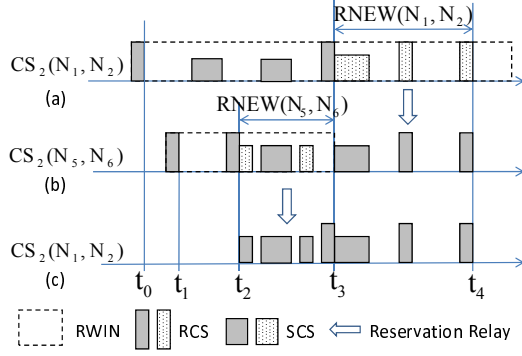
Before we provide the details on the reservation process, we first discuss the sources of interferences and how to choose RWIN, RNEW, and RSEG to guarantee contention-free CS reservations.

### 4.2.1 Interference Avoidance

Depending on whether an RCS or SCS is to be reserved, different sets of links may be considered as *interfering links*. When link  $(N_i, N_j)$  reserves an RCS, all links within a *2-link* distance can interfere with this reservation. That is, any link  $(N_k, N_l) \in L_2(N_i, N_j)$  can interfere with an RCS reservation of  $(N_i, N_j)$ . For example, in Figure 1, when RCS( $N_2, N_4$ ) is active, any CS over link  $(N_6, N_7)$  cannot be active at the same time, otherwise the transmissions would collide at node  $N_5$  and  $N_5$  cannot correctly overhear the reservation announcement of RCS( $N_2, N_4$ ). On the other hand, when link  $(N_i, N_j)$  reserves an SCS, only links  $(N_k, N_l) \in L_1(N_i, N_j)$  within a *1-link* distance can interfere with this reservation.

To track the channel status, each node  $N_i$  maintains the *communication session reservation status*  $CS_2(N_i)$ , which consists of known CS reservations within the *2-link* distance of  $N_i$ . By combining  $CS_2(N_i)$  and  $CS_2(N_j)$  of a link  $(N_i, N_j)$ , the link knows the communication session reservation status within its *2-link* distance. To avoid interferences, DARTS is designed such that a link’s RCS has a *consistent view* of all CSs during its current RNEW before it starts to make new CS reservations. Note that to obtain a consistent view, a link only needs to know all existing CS reservations that overlap with its current RNEW within its *2-link* distance.

To help maintain a consistent view for every link, *1-link* neighbors of negotiating nodes overhear new reservations and propagate (or *relay*) this information further to *2-link* neighbors when they perform their own negotiations. However, the timing of these relays is crucial to maintaining a consistent view and to avoiding conflicting reservations. This timing is achieved through judicious selection of



**Figure 4.** An illustration of worst-case reservation relay and RWIN assignment constraint.

RWIN for each link among all potentially interfering links. If the difference between RWINs of two interfering links is chosen too small, reservations made by one link during its RWIN may not have sufficient amount of time to propagate to other links in the 2-link distance. Consider two interfering links, e.g.,  $(N_1, N_2)$  and  $(N_5, N_6)$  in Figure 1 and Figure 4. Suppose  $RWIN(N_1, N_2) \geq RWIN(N_5, N_6)$  (the dashed boxes in Figure 4(a) and (b) indicate RWINs). During the RCS immediately before  $t_1$ ,  $(N_5, N_6)$  makes new reservations between  $(t_1, t_1 + RWIN(N_5, N_6)]$ . In order for  $(N_5, N_6)$  to have a consistent view during this RCS, it must have received the new CS reservations made by  $(N_1, N_2)$  at time  $t_0$  if  $RWIN(N_5, N_6)$  starting at  $t_1$  has any overlap with the newly reserved CSs. (Note that  $CS(N_1, N_2)$  in  $[t_0, t_3]$  should already be known to  $(N_5, N_6)$  at this point). Now assume that there is not sufficient amount of time between  $t_0$  and  $t_1$  for  $(N_5, N_6)$  to receive the new  $CS(N_1, N_2)$  made in  $[t_3, t_4]$ . Then, we must have

$$t_0 + RWIN(N_1, N_2) - RNEW(N_1, N_2) \geq t_1 + RWIN(N_5, N_6).$$

Otherwise, during  $RCS(N_5, N_6)$  immediately before  $t_1$ ,  $(N_5, N_6)$  may reserve CSs that interfere with  $CS(N_1, N_2)$  in  $[t_3, t_4]$ . This relationship can be rewritten as

$$RWIN(N_1, N_2) - RWIN(N_5, N_6) \geq (t_1 - t_0) + RNEW(N_1, N_2).$$

Note that  $(t_1 - t_0)$  is the time available for  $(N_5, N_6)$  to receive  $CS(N_1, N_2)$  made during the RCS immediately before  $t_0$ . To determine how much time it may take for  $CS(N_1, N_2)$  reserved during the RCS before  $t_0$  to be known by  $(N_5, N_6)$ , observe that  $CS(N_1, N_2)$  reserved during the RCS before  $t_0$  is “overheard” by  $N_4$  during the same RCS (since the RCS announces its reservations to its neighbors when it makes the reservations). In the worst case, after  $\max RSEG(N_4, N_5)$ , a new  $RCS(N_4, N_5)$  has occurred which relays  $CS(N_1, N_2)$  to  $N_5$  and  $N_6$ . Hence, in the worst case, it takes  $\max RSEG(N_4, N_5)$  for  $CS(N_1, N_2)$  reserved during the RCS before  $t_0$  to be known by  $(N_5, N_6)$ .

We generalize the above observations in Lemma 4.1, which states the lower bound on the differences between RWINs of interfering links.

**Lemma 4.1.** Given  $(N_i, N_j)$  and  $L_2(N_i, N_j)$ , for any  $(N_k, N_l) \in l_2(N_i, N_j)$  and suppose  $RWIN(N_i, N_j) > RWIN(N_k, N_l)$ ,  $RWIN(N_i, N_j)$  must satisfy

$$RWIN(N_i, N_j) \geq RWIN(N_k, N_l) + \min_{(N_m, N_n) \in l_1(N_i, N_j)} \{ \max RSEG(N_m, N_n) \} + \max RNEW(N_i, N_j) \quad (1)$$

*Proof.* Suppose at time  $t_0$  there exists an RCS of  $(N_i, N_j)$  and the unreserved time interval of  $(N_i, N_j)$  is  $[t_0 + RWIN(N_i, N_j) - RNEW(N_i, N_j), t_0 + RWIN(N_i, N_j)]$ . That is,  $[t_0, t_0 + RWIN(N_i, N_j) - RNEW(N_i, N_j)]$  contains CS reservations of  $(N_i, N_j)$  made by the RCSs of  $(N_i, N_j)$  before  $t_0$ . The CS reservations within RNEW will be received by link  $(N_k, N_l)$  via at least one of their shared neighbors, e.g.,  $(N_m, N_n) \in l_1(N_i, N_j)$ . The longest relay delay occurs when  $(N_m, N_n)$  hears the announcement of RNEW from  $(N_i, N_j)$  just after an RCS of  $(N_m, N_n)$ . This RNEW will be further relayed at the next RCS of  $(N_m, N_n)$ . The duration between the two RCSs of  $(N_m, N_n)$  will be the relay delay of the RNEW announcement. The longest interval between two consecutive RCSs is  $\max RSEG$  and assume the reception time of RNEW of  $(N_i, N_j)$  at  $(N_k, N_l)$  is  $t_1$ , then we have:

$$t_1 - t_0 \leq \min_{(N_m, N_n) \in l_1(N_i, N_j)} \{ \max RSEG(N_m, N_n) \} \quad (2)$$

To prevent  $(N_k, N_l)$  from making reservations during a time interval that overlaps with RNEW of  $(N_i, N_j)$ ,  $(N_k, N_l)$  must receive the RNEW before it has a chance to make such a conflicting reservations. That is, at time  $t_1$ , the end of RWIN of  $(N_k, N_l)$  must be no greater than the start of the RNEW of  $(N_i, N_j)$ . Formally, this is expressed as  $t_1 + RWIN(N_k, N_l) \leq t_0 + RWIN(N_i, N_j) - RNEW(N_i, N_j)$ , which can further be formulated as:

$$RWIN(N_i, N_j) - RWIN(N_k, N_l) \geq t_1 - t_0 + RNEW(N_i, N_j) \quad (3)$$

Substituting  $t_1 - t_0$  (Inequality 2) and RNEW with their respective maximum values in Inequality 3, we derive Lemma 4.1. Here,  $\max RNEW(N_i, N_j)$  is the largest RNEW  $(N_i, N_j)$  within which link  $(N_i, N_j)$  can make new CS reservations.  $\square$

Note that the RWINs of links within 1-link distance do not need to satisfy Equation 1, since the reservations made

by a link can be overheard by all its *1-link* neighbors immediately. Based on Lemma 4.1, DARTS can compute feasible values for links' RWINs to ensure consistent views by each link.

In order to support best-effort (BE) traffic, additional constraints on RWIN have to be considered. If a link wants to transmit BE traffic during an unreserved time interval, the link must know all CS reservations made by other links within its *2-link* distance. Therefore, every node must have an *instant* view of the status of all its interfering links at any time. To maintain an instant view, again, a link's CSs must be relayed to all nodes within its *2-link* distance before the CSs occur. For example, in Figure 4(b) and (c), RSEG( $t_2, t_3$ ) made by  $(N_5, N_6)$  must be relayed to  $(N_1, N_2)$  before  $t_2$ . The worst-case occurs when  $(N_5, N_6)$  make CS reservations within the longest reservation region  $\max \text{RNEW}(N_5, N_6)$ , then the CSs in this RNEW are received by  $(N_1, N_2)$  at the latest possible time (i.e., after the longest relay delay  $\max \text{DELAY}((N_5, N_6), (N_1, N_2))$ ). In general, maintaining an instant view requires a lower bound on RWINs, which is stated by Lemma 4.2.

**Lemma 4.2.** *For any  $(N_i, N_j)$ , it must satisfy*

$$\text{RWIN}(N_i, N_j) \geq \max_{(N_m, N_n) \in l_1(N_i, N_j)} \{ \max \text{RSEG}(N_m, N_n) \} + \max \text{RNEW}(N_i, N_j) \quad (4)$$

*Proof.* Suppose an RCS of  $(N_i, N_j)$  at time  $t_0$  makes new reservations within RNEW  $[t_0 + \text{RWIN}(N_i, N_j) - \text{RNEW}(N_i, N_j), t_0 + \text{RWIN}(N_i, N_j)]$ . The CS reservations within the RNEW will be received by any link  $(N_k, N_l)$  in  $l_2(N_i, N_j)$  via at least one neighbor  $(N_m, N_n)$  of  $(N_i, N_j)$ . The longest delay is  $\max_{(N_m, N_n) \in l_1(N_i, N_j)} \{ \max \text{RSEG}(N_m, N_n) \}$ . Assume that at time  $t_1$  every node in  $l_2(N_i, N_j)$  has received notification of RNEW. Since every CS reservation of a link must be received by every link in its *2-link* distance before the CS actually occurs (i.e., the start time of RNEW  $t_0 + \text{RWIN}(N_i, N_j) - \text{RNEW}(N_i, N_j)$  must be no less than  $t_1$ ), we have:

$$t_1 \leq t_0 + \text{RWIN}(N_i, N_j) - \text{RNEW}(N_i, N_j) \quad (5)$$

otherwise, at time  $t_1$ , some node will not have the reservation status at  $t_1$  since the RNEW has not been received. That is, we must also have:

$$t_1 - t_0 \geq \max_{(N_m, N_n) \in l_1(N_i, N_j)} \{ \max \text{RSEG}(N_m, N_n) \} \quad (6)$$

Combining Inequality 5 and 6, we derive Lemma 4.2.  $\square$

Equations 1 and 4 provide two constraints guiding the assignment of RWINs in the general case. Since this work is concerned with periodic packet transmissions of real-time

streams (and therefore periodic reservations), DARTS models a link  $(N_i, N_j)$  with a *reservation periodic server (RPS)* with a period  $p(N_i, N_j)$  equal to the deadline  $D(N_i, N_j)$ . An RPS then reserves an RCS once for each period before the RPS' deadline and at each RCS, the RPS reserves RNEW within its RWIN. As a result,  $\max \text{RSEG}(N_i, N_j)$  is equal to  $2p(N_i, N_j)$  when an RCS is reserved at the ready time of an RPS and the immediately following RCS is reserved at the deadline of the next RPS instance. Similarly,  $\max \text{RNEW}(N_i, N_j)$  is equal to  $2 \max \text{RSEG}(N_i, N_j) = 4p(N_i, N_j)$  when there is a longest possible unreserved RSEG at the end of the RWIN immediately after the RCS currently occurring and this current RCS and its immediately following RCS span over a longest possible RSEG as well. Then, at this next RCS, there is a  $2 \max \text{RSEG}(N_i, N_j)$  unreserved time interval within the RWIN.

A link's RWINs are determined in a distributed and dynamic fashion satisfying both constraints stated in Lemmas 4.1 and 4.2. To accomplish this, a simple neighborhood discovery mechanism (e.g., [11]) can be enhanced with a clustering or a (*2-link* distance) link-coloring mechanism (e.g., [3, 12]) to recursively assign RWINs and converge fast. Here, we provide a simple approach (Algorithm 1) that can be used by each node to determine the node's RWIN. We impose a static total ordering relation on links, e.g.,  $(N_k, N_l) > (N_i, N_j)$ , if and only if, the maximum identification ( $\max(\text{id}(N_k), \text{id}(N_l))$ ) of the pair of nodes of  $(N_k, N_l)$  is greater than that ( $\max(\text{id}(N_i), \text{id}(N_j))$ ) of  $(N_i, N_j)$ , or they have the same maximum identification ( $\max(\text{id}(N_k), \text{id}(N_l)) = \max(\text{id}(N_i), \text{id}(N_j))$ ) but the minimum identification of  $(N_k, N_l)$  is greater than the minimum identification of  $(N_i, N_j)$  ( $\min(\text{id}(N_k), \text{id}(N_l)) > \min(\text{id}(N_i), \text{id}(N_j))$ ). The algorithm forces a pair of links within a *2-link* distance to maintain the relationship expressed by Lemma 4.1. A link with larger ordering always holds a larger RWIN (lines 7- 18). Since there exists a unique total ordering, the RWIN updating procedure will settle and no more broadcasts of updated RWINs are generated ultimately, i.e., the algorithm will converge. The same algorithm is also used by a node joining a network.

For example, we assume that  $\text{id}(N_i) > \text{id}(N_j)$  if  $i > j$  in Figure 1 and that the periods of all RPSs are identical ( $p(N_i, N_j) = p$ ), then by Algorithm 1,  $\text{RWIN}(N_1, N_2)$  (as well as  $\text{RWIN}(N_2, N_3)$ ,  $\text{RWIN}(N_2, N_4)$ , and  $\text{RWIN}(N_4, N_5)$ ) can be set to the base RWIN  $6p$  using Equation 4 and  $\text{RWIN}(N_5, N_6)$  (as well as  $\text{RWIN}(N_6, N_7)$ ) can be set to  $12p$ .

---

**Algorithm 1** RWIN computation
 

---

```

1: initialization:
2: broadcast base RWIN determined by Equation 4 to neighbors

3: upon receiving RWIN( $N_i, N_j$ ) at ( $N_m, N_n$ ), where  $(N_i, N_j) \in l_1(N_m, N_n)$ :
4: forward (rebroadcast) the received  $\text{RWIN}(N_i, N_j)$ ,  $\max \text{RSEG}(N_m, N_n)$ , and  $\max \text{RNEW}(N_m, N_n)$ 

5: /* suppose a total ordering on links, e.g.,  $(N_k, N_l) > (N_i, N_j)$  iff  $\max(\text{id}(N_k), \text{id}(N_l)) > \max(\text{id}(N_i), \text{id}(N_j))$  or  $\max(\text{id}(N_k), \text{id}(N_l)) = \max(\text{id}(N_i), \text{id}(N_j)) \cap \min(\text{id}(N_k), \text{id}(N_l)) > \min(\text{id}(N_i), \text{id}(N_j))$  */
6: upon receiving RWIN( $N_i, N_j$ ),  $(N_i, N_j) \in l_2(N_k, N_l)$ , at  $(N_k, N_l)$  via ( $N_m, N_n$ )  $\in l_1(N_i, N_j)$ :
7: if  $(N_k, N_l) > (N_i, N_j)$  then
8:   compute  $\text{RWIN}'(N_k, N_l)$  with  $\text{RWIN}(N_i, N_j)$ ,  $\max \text{RSEG}(N_m, N_n)$ , and  $\max \text{RNEW}(N_m, N_n)$  by Equation 1
9:   if  $\text{RWIN}'(N_k, N_l) > \text{RWIN}(N_k, N_l)$  then
10:      $\text{RWIN}(N_k, N_l) = \text{RWIN}'(N_k, N_l)$ 
11:     broadcast the updated  $\text{RWIN}(N_k, N_l)$  to neighbors
12:   end if
13: else  $\{(N_i, N_j) > (N_k, N_l)\}$ 
14:   compute  $\text{RWIN}'(N_i, N_j)$  with  $\text{RWIN}(N_k, N_l)$ ,  $\max \text{RSEG}(N_m, N_n)$ , and  $\max \text{RNEW}(N_m, N_n)$  by Equation 1
15:   if  $\text{RWIN}'(N_i, N_j) > \text{RWIN}(N_i, N_j)$  then
16:     broadcast  $\text{RWIN}(N_k, N_l)$  to neighbors
17:   end if
18: end if

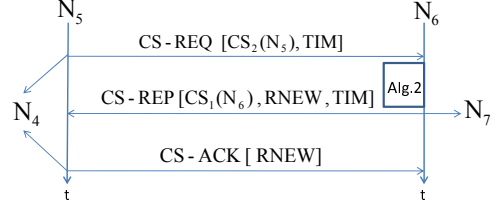
```

---

#### 4.2.2 Reservation Negotiation, Announcement, Monitoring, and Relay

DARTS models each stream delivery as the distributed communications of a chain of *stream periodic servers* (SPSs) per link and per stream en route from the source to the sink. The SPS for the delivery of stream  $S_v$  over link  $(N_i, N_j)$  is denoted as  $\text{SPS}_v(N_i, N_j)$  with the same period and CS duration as  $S_v$ . Each SPS is responsible for the reliable packet transmission of its associated stream in a contention-free SCS. RPSs can be treated as a special stream server and we call both RPSs and SPSs as PSs. We denote  $\text{PS}_v(N_i, N_j) \approx \text{PS}_u(N_i, N_j)$  if CSs of  $\text{PS}_v(N_i, N_j)$  potentially interfere with CSs of  $\text{PS}_u(N_i, N_j)$ .

We use  $\text{RCS}_n(N_i, N_j)$  (and  $\text{SCS}_n(N_i, N_j)$  and  $\text{CS}_n(N_i, N_j)$ , respectively) to refer to the RCSs (SCS and CS, respectively) reserved by links  $\in L_n(N_i, N_j)$ . Recall that every node  $N_i$  has a consistent view ( $\text{CS}_2(N_i)$ ) of the CS reservation status within its *2-link* distance before it starts to negotiate CS reservations within an RNEW. By combining the consistent views of the two nodes of a link  $(N_i, N_j)$ , a link has a consistent view  $\text{CS}_2(N_i, N_j)$ . DARTS guarantees that *every CS reservation within every new RNEW of a link will not conflict with the consistent*



**Figure 5.** Reservation relay, transfer, negotiation, announcement, and best-effort traffic indication within an RCS.

view of the link. Moreover, each node involved in an RCS also relays the received CS reservations from its *1-link* neighbors since the last RCS in which this node communicated with other *1-link* neighbors.

The negotiation and relay procedure consists of transmitting three messages (CS-REQ, CS-REP, CS-ACK) and a CS allocation routine (Algorithm 2). Consider link  $(N_5, N_6)$  in Figure 1. Suppose  $N_5$  initiates CS-REQ (see Figure 5) and relays the newly heard CS reservations in its *1-link* distance (i.e.,  $\text{CS}_1(N_5)$ ), which is announced by  $N_4$  or  $N_6$ . Moreover,  $N_5$  transfer to  $N_6$  the known CS reservations within the current RNEW of link  $(N_5, N_6)$ , i.e., the reservations  $(\text{CS}_2(N_5) - \text{CS}_1(N_5))$  limited to the time interval of the current RNEW, which are the reservations  $\text{CS}(N_1, N_2)$  and  $\text{CS}(N_3, N_2)$  relayed by  $N_4$ . All neighbors of  $N_5$  (i.e.,  $N_4$  and  $N_6$ ) will listen to this CS-REQ. Note that all relays for *1-link* reservations and all transfers for *2-link* reservations are incremental; and these combined with the overheard *0-link* reservations result in a consistent view of a link. Then,  $N_6$  executes Algorithm 2 and greedily reserves CSs. The reserved CSs are announced by  $N_6$  via CS-REP and  $N_5$  returns them via CS-ACK.  $N_6$  also relays newly heard CS reservations via CS-REP. However,  $N_6$  does not transfer any reservations  $(\text{CS}_2(N_6) - \text{CS}_1(N_6))$  in its *2-link* distance to  $N_5$ , since the CS reservations are made by executing Algorithm 2 at  $N_6$ . No reservations are made beyond RWIN, which keeps the state memory limited to RWIN.

Wrapping up the discussion of the reservation scheme, we state a set of rules governing the reservation procedures, which implement the conflict-free reservation described in Section 4.2.1 in a temporally continuous manner. Then we prove that such a scheme satisfies the condition of interference avoidance and therefore provides contention-free channel accesses.

- RWIN assignment rules and initialization:
  - W1:** Order links using a clustering or link-coloring mechanism (e.g., Algorithm 1), and then recursively produce RWINs of links using Equations 1 and 4.
- Reservation monitoring rules:
  - M1:** Every node  $N_i$  listens to RCSs within its *1-link*

---

**Algorithm 2** Communication session reservations at an  $RCS^t(N_i, N_j)$ 


---

- 1:  $CS_2(N_i, N_j) = CS_2(N_i) \cup CS_2(N_j)$
  - 2:  $T_{last}$  = the completion time of the last reserved RNEW
  - 3: RNEW =  $[T_{last}, t + RWIN(N_i, N_j)]$
  - 4: /\*traverse all periodic servers which have not been allocated reservations or possibly have reservations within RNEW\*/
  - 5: **for** each PS instance whose respective CS have not been allocated **do**
  - 6:     **if** PS is an RPS **then**
  - 7:         greedily allocate intervals equal to its SCS length within RNEW using a non-preemptive EDF policy and ensuring that neither a reserved RCS nor SCS within its 2-link distance overlaps with this RCS
  - 8:     **else** {PS is a SPS}
  - 9:         greedily allocate intervals equal to its RCS length within RNEW using a non-preemptive EDF policy and ensuring that no reserved RCS within its 2-link distance and no reserved SCS within its 1-link distance overlaps with this SCS
  - 10:    **end if**
  - 11:    mark allocated intervals as reserved in both  $CS_2(N_i, N_j)$  and RNEW
  - 12: **end for**
  - 13: trim off reservations after last reserved RCS in RNEW
  - 14:  $T_{last}$  = the end of last reserved RCS in RNEW
- 

distance. It records (using  $CS_2(N_i)$ ) all overheard and relayed reservations that are encapsulated within CS-REQ, CS-REP, or CS-ACK messages. It further sets its TIM if the TIM field of either CS-REQ or CS-REP indicates backlogged best-effort messages.

- Reservation rules:

**R1:** At each  $RCS(N_i, N_j)$ ,  $(N_i, N_j)$  allocates and announces reservations for all periodic servers (PSs including RPSs and SPSs) over  $(N_i, N_j)$  not conflicting with the consistent view  $CS_2(N_i, N_j)$  of  $(N_i, N_j)$  and relays newly heard CS reservation in its 1-link distance to its neighbors (following the procedure described in Figure 5 and Algorithm 2). The TIM field of a message to a neighbor is set if there are backlogged best-effort messages pending for that neighbor.

- Node sleep/wakeup rules:

**S1:** Node  $N_i$  must be awake and negotiate reservations with its peers during every reserved  $RCS(N_i, *)$ .

**S2:** Node  $N_i$  must be awake during any 1-link  $RCS(N_j, N_k)$ .

**S3:** Best-effort messages at  $N_i$  can be transmitted during unreserved time intervals (i.e., time intervals not overlapping with reserved RCSs within the 2-link distance and SCSs within the 1-link distance). If  $N_i$  has backlogged best-effort traffic for one of its neighbors and the TIM to this neighbor is set, then  $N_i$  contends (fairly [1] or using prioritization [7, 10]) for access to the medium.

**S4:** If  $N_i$  has best-effort traffic to receive, which is indicated by TIM fields in messages received from its

**Figure 6.** Timing parameters of  $S_1$  and  $S_2$

$S$	$e$	$p$	$N_4$		
$S_1$	4	45	60		
$S_2$	4	75	100		
RPS(*, *)	1	45	45		
$S$	$r(N_1, N_2)$	$r(N_2, N_4)$	$r(N_4, N_5)$	$r(N_5, N_6)$	$r(N_6, N_7)$
$S_1$	0	15	30	45	60
$S_2$	0	25	50	75	100

neighbors, it stays awake until all backlogged traffic has been received (e.g., a *more* field within a packet set to 0 as in [1] indicates the end of best-effort traffic).

**S5:** Node  $N_i$  goes to sleep otherwise.

**Theorem 4.3.** *DARTS guarantees contention-free accesses for real-time streams and best-effort messages in a stationary and bidirectional network.*

*Proof.* Using **W1** and Lemmas 4.2 and 4.1, non-interfering CSs can be reserved. Using **R1**, new RCSs will be reserved continuously. And, using **M1**, every link will record the newly reserved RCSs within its 2-link distance via over-hearing directly announced or relayed reservations. Then, using **S2**, these neighbors will wake up to listen to these reserved RCSs as well. Using **S1-S5**, best-effort traffic does not interfere with real-time traffic. Therefore, non-interfering CSs can be continuously reserved and every reservation will be contention-free.  $\square$

We use a small example to further illustrate this procedure. The example network and the stream parameters are shown in Figure 1 and Figure 6. Figure 7 presents a snapshot of the reservations of each node, where  $r(N_i, N_j)$  is the release time of a stream over  $(N_i, N_j)$ . Each line shows the reservation status  $CS_2(N_i)$  of  $N_i$ .

### 4.3 Stream Admission Control

The stream admission control decides whether a new stream is admissible and assigns local deadlines to the chain of PSs of the new stream. The deadline assignment scheme guarantees that 1) every SPS of the new stream will have an SCS reserved within its local deadline and 2) all existing SPSs and RPSs in the network will continue to meet their respective deadlines.

The stream admission control consists of three procedures (shown in Figure 8): 1) a request procedure from the source to the sink, 2) a local deadline assignment procedure at the sink, and 3) a response procedure from the sink to the source. The *neighborhood admission control* is performed

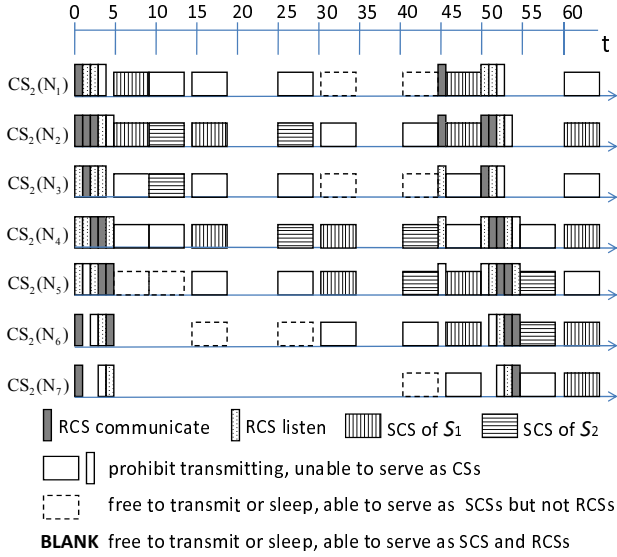


Figure 7. A snapshot of CS reservations.

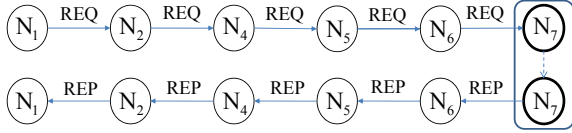


Figure 8. Stream admission control procedure.

on a link-by-link basis and it checks the timeliness of each PS affected by the joining stream  $\mathcal{S}_u$ .

To calculate the latest reservation time of an affected PS  $\mathcal{P}_{S_v}(N_i, N_j)$  relative to its release time using Algorithm 2, we reduce it to the calculation of a fixed-priority worst-case response time (Theorem 6.5 in [9]) based on time-demand analysis.  $\mathcal{P}_{S_v}(N_i, N_j)$  has the latest reservation time (i.e., worst-case response time) if 1) a critical instant among the set of PSs affected by  $\mathcal{P}_{S_v}(N_i, N_j)$  occurs and 2)  $\mathcal{P}_{S_v}(N_i, N_j)$  has the lowest priority. The time-demand function  $W_v^{(N_i, N_j)}(t)$  is then given by:

$$W_v^{(N_i, N_j)}(t) = \sum_{\mathcal{P}_{S_u}(N_k, N_l) \approx \mathcal{P}_{S_v}(N_i, N_j)} \left\lceil \frac{t}{p_u} \right\rceil (e_u + \delta) \quad (7)$$

where  $\delta$  is the minimum gap that can accommodate an effective communication. The latest reservation time  $W_v^{(N_i, N_j)}$  of  $\mathcal{P}_{S_v}(N_i, N_j)$  is the fixed point of Equation 7, i.e.,  $W_v^{(N_i, N_j)} = FP(W_v^{(N_i, N_j)}(t))$ . We also collect a workload heuristic  $\mathcal{U}_u(N_i, N_j)$  around each link en route to assist the local deadline assignment at the sink, which is given

by

$$\mathcal{U}_u(N_i, N_j) = \sum_{\mathcal{P}_{S_u}(N_k, N_l) \approx \mathcal{P}_{S_v}(N_i, N_j)} \frac{e_u}{p_u}. \quad (8)$$

Combining the aforementioned discussion, we give the neighborhood admission control algorithm (Algorithm 3) performed by each link during the request procedure. A

---

**Algorithm 3** Neighborhood admission control at  $(N_i, N_j)$

---

- 1:  $\mathcal{S}_u$  is a join stream
  - 2:  $\Delta_u$  is initialized to the end-to-end deadline  $D_u$  at the source node of  $\mathcal{S}_u$
  - 3:  $U = 0$
  - 4: **for** each  $\mathcal{P}_{S_v}(N_k, N_l) \approx \mathcal{P}_{S_u}(N_i, N_j)$  **do**
  - 5:      $W = FP(W_v^{(N_k, N_l)}(t))$  /\*compute the latest reservation time of  $\mathcal{P}_{S_v}(N_k, N_l)$  by Equation 7\*/
  - 6:     **if**  $W > D_v(N_k, N_l)$  **then**
  - 7:         reject  $\mathcal{S}_u$
  - 8:     **else**
  - 9:          $U = U + \frac{e_v}{p_v}$
  - 10:     **end if**
  - 11: **end for**
  - 12:  $\mathcal{U}_u(N_i, N_j) = avg(U)$  /\*workload heuristic\*/
  - 13:  $\Delta_u = \Delta_u - FP(W_u^{(N_i, N_j)}(t))$  /\*deadline slack\*/
  - 14: **if**  $\Delta_u \geq 0$  **then**
  - 15:     accept  $\mathcal{S}_u$  at  $(N_i, N_j)$
  - 16:     return the latest reservation time  $FP(W_u^{(N_i, N_j)}(t))$ ,  $\Delta_u$ , and the workload heuristic  $\mathcal{U}_u(N_i, N_j)$
  - 17: **else**
  - 18:     reject  $\mathcal{S}_u$
  - 19: **end if**
- 

sink performs the local deadline assignment by relaxing the time slack  $\Delta_u = D_u - \sum_{(N_i, N_j) \in \mathcal{R}_u} W_u^{(N_i, N_j)}$  of a new stream  $\mathcal{S}_u$  based on the neighborhood workload heuristic. That is, we use  $\mathcal{U}_u(N_i, N_j)$  to proportionally distribute the slack  $\Delta_u$  to each hop using:

$$D_u(N_i, N_j) = W_u^{(N_i, N_j)} + \frac{\mathcal{U}_u(N_i, N_j) \Delta_u}{\sum_{(N_k, N_l) \in \mathcal{R}_u} \mathcal{U}_u(N_k, N_l)}. \quad (9)$$

**Theorem 4.4.** *If the neighborhood admission control is successful at every link of a joining stream, the stream is admissible and the end-to-end deadline of the stream will be met.*

*Proof.* At each link, DARTS checks the latest reservation times of all PSs conflicting with at least one link  $\in \mathcal{R}_u$ . If every neighborhood admission control is successful, then all local deadlines of all these PSs are satisfied. Moreover, this guarantees that the sum of latest reservation times en route of an admitted stream  $\mathcal{S}_u$  is no greater than the end-to-end deadline. For a PS which does not conflict with any link  $\in \mathcal{R}_u$ , the workload within its 2-link distance remains unchanged. They will be unaffected by the new stream and

thus the admission control mechanism does not need to consider them. Therefore, the lemma is proven.  $\square$

#### 4.4 Discussion

An implementation of DARTS in a real wireless network must deal with several practical, yet important technical issues that we have not touched yet. These issues arise mainly from transmission and node failures, joining nodes, broadcast/multicast support, and time synchronization.

First, since in DARTS all nodes within a link's  $l$ -link distance will listen to the link's RCSs, this provides an opportunity to integrate existing time synchronization mechanisms into RCSs. DARTS requires time accuracy only within RWINs, which are typically very small, thus infrequent synchronization and coarse-grained clocks will suffice. Simple timestamp mechanisms (e.g., as used in [10] and [11]) can be used for synchronization.

Second, DARTS can be enhanced to support reliable multicast and broadcast in a real-time and energy-efficient manner simply by allowing a node to negotiate the same communication session multiple times over different links.

Third, nodes in a DARTS-based network must receive or overhear RCSs to operate correctly. In practical networks, nodes may miss RCSs; but even if an RCS is missed, a node may hear the missed information in the next RCS of another neighbor if these nodes share multiple neighbors (as is not uncommon in multi-hop networks). In addition, nodes can exploit unused time intervals over their links for re-transmissions of failed communications.

Finally, a new node joining an ad-hoc network must listen to existing members for at least the maximum RWIN of all links within its  $l$ -link distance to collect sufficient information for participating in future negotiations (similarly, a failed node can re-insert itself with this approach). To allow the new node to initiate its own reservations, DARTS' negotiation and relay procedure is modified to provide the node with opportunities to alert its new neighbors of its presence. This can be achieved by reserving a brief time interval after the CS-ACK in the three-way negotiation process inside an RCS, where a new node can transmit a short *request signal*. Nodes receiving this signal will stay awake until the link's next RCS in order to listen for reservation requests from the new node during unreserved time intervals (note that multiple nodes can join at the same time and they can compete for the channel using RTS/CTS sequences). Once the first RCS has been obtained, the new node can negotiate future RCSs like any other existing node.

### 5 Overhead Analysis

In this section, we provide an analysis of the worst-case communication and energy overheads of DARTS. The over-

	CS-REQ	CS-REP	CS-ACK
Worst-case	$4( h_2  +  h_1 )\mathcal{F}$	$4( h_1  + 1)\mathcal{F}$	$4\mathcal{F}$
Average-case	$( h_2  +  h_1 )\mathcal{F}$	$( h_1  + 1)\mathcal{F}$	$\mathcal{F}$

**Table 2.** Total number of CS reservations announced, relayed, and transferred at an RCS.

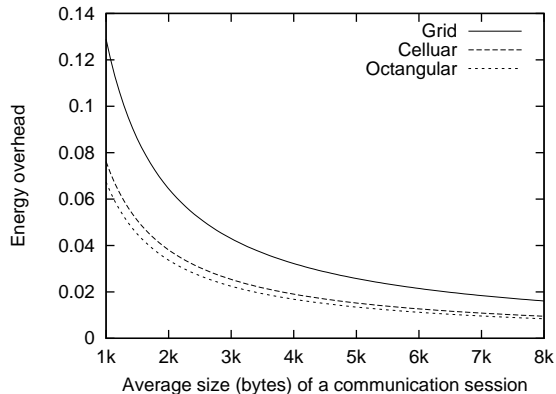
heads are most significant when the network is at the maximum theoretical throughput (at which point the communication and energy overheads also reach their maximums). The analytical results shows that DARTS has negligible energy and communication overheads.

We assume a link's bandwidth of  $\mathcal{B}$  bytes/second. The average reservation period of RCS is  $\mathcal{P}$  seconds and the average size of an SCS is  $\mathcal{X}$  bytes.  $\mathcal{K}$  bytes are used to encode each CS (time interval, type, and link identification) within an RCS. Note that a node only encodes link identifications for CS reservations exactly  $l$ -link away. We denote the number of links that are  $n$ -links away and within  $n$ -link distance from a node as  $|h_n|$  and  $|H_n|$ , respectively (hence,  $|H_n| = \sum_{0 \leq i \leq n} |h_i|$ ). The number of links that are  $n$ -links away and within  $n$ -link distance from a link are identified as  $|l_n|$  and  $|L_n|$ , respectively.

First, we calculate the maximum number  $\mathcal{F}$  of CSs per  $\mathcal{P}$  a set of interfering links can reserve. Any two CSs within a  $l$ -link distance cannot be active simultaneously, so the total bandwidth utilization within any  $l$ -link distance of a link is at most to 1, i.e.,  $\frac{\mathcal{F}\mathcal{X}}{\mathcal{B}\mathcal{P}}|L_1| \leq 1$ , which produces:

$$\mathcal{F} \leq \frac{\mathcal{B}\mathcal{P}}{|L_1|\mathcal{X}}. \quad (10)$$

In Section 4.2.1, we stated that  $\text{maxRNEW}$  and  $\text{maxRSEG}$  of a link are  $4\mathcal{P}$  and  $2\mathcal{P}$ , respectively, requiring  $4\mathcal{F}\mathcal{K}$  and  $2\mathcal{F}\mathcal{K}$  bytes to encode, respectively. During any time interval equal to or less than  $\text{maxRSEG}$ , a link can make new reservations that stretch over a time interval of length  $\text{maxRNEW}$ . Therefore, a node can hear at most  $4\mathcal{F}|h_1|$  new CS reservations since its last RCS occurrence. According to Section 4.2.2, in the worst case scenario,  $N_i$  informs  $N_j$  (using a CS-REQ message) of the CS reservations over links  $l_2(N_i)$  within the link's  $\text{maxRNEW}$ , which contains at most  $4\mathcal{F}|h_2|$  CSs. Also, the CS-REQ further relays newly heard reservations over links  $l_1(N_i)$  since  $N_i$ 's last RCS, which contains at most  $4\mathcal{F}|h_1|$  CSs. Via the CS-REP,  $N_j$  announces at most  $4\mathcal{F}$  new CS reservations over  $(N_i, N_j)$  and relays at most  $4\mathcal{F}|h_1|$  newly heard CS reservations over links  $l_1(N_j)$ . The CS-ACK re-announces the received  $4\mathcal{F}$  CSs. We summarize the above analysis in Table 2. By substituting  $\mathcal{F}$  with Equation 10 in Table 2 and adding up the three components, we obtain the upper bound



**Figure 9.** Worst-case energy overhead of DARTS in a fully-loaded network.

of an RCS duration per period  $\mathcal{P}$ :

$$|\text{RCS}| \leq \frac{(4|h_2| + 8|h_1| + 8)\mathcal{K}\mathcal{P}}{|L_1|\mathcal{X}}, \quad (11)$$

$$E[|\text{RCS}|] = \frac{(|h_2| + 2|h_1| + 2)\mathcal{K}\mathcal{P}}{|L_1|\mathcal{X}}. \quad (12)$$

The bandwidth overhead,  $\mathcal{W}$ , is equal to the ratio of the sum of RCS durations within the 2-link distance of a link per period to the value of the period. It is bounded by

$$\mathcal{W} \leq \frac{(4|h_2| + 8|h_1| + 8)\mathcal{K}|L_2|}{|L_1|\mathcal{X}}. \quad (13)$$

The extra energy consumption of a node is due to the node listening to its neighbors' RCSs or negotiating with its neighbors during RCSs. The two nodes of a link will listen or communicate in  $|L_1|$  RCSs per period  $\mathcal{P}$ . While an RCS is completed, all involved nodes will go to sleep, therefore the maximum energy overhead per node,  $\mathcal{E}$ , is bounded by

$$\mathcal{E} \leq \frac{(0.5|h_2| + |h_1| + 1)\mathcal{K}}{\mathcal{X}}. \quad (14)$$

Now we illustrate the worst-case communication and energy overheads through example scenarios. We investigate three different network topologies: grid, cellular, and octangular, with  $(|l_0|, |l_1|, |l_2|, |h_1|, |h_2|) = (7, 16, 24, 12, 18)$ ,  $(5, 8, 14, 6, 12)$ , and  $(5, 8, 11, 6, 9)$ , respectively. In all cases,  $\mathcal{K} = 6$  bytes. We vary the average size of a CS and study its impact on the overhead. Since the bandwidth overhead is proportional to the energy overhead, we only draw the worst-case energy overhead in Figure 9. Even in the worst-case, DARTS causes only small energy (and bandwidth) overheads. Moreover, the overhead does not increase as the network scales up since the reservation delays and announcements are localized within neighborhoods.

## 6 Experiments

We implemented DARTS in a simulation environment and as extension to a Linux kernel and this section provides results with respect to real-time and energy performance. Results for best-effort traffic show a similar performance as presented in [13] and we therefore omit these results for brevity and focus on real-time traffic behavior.

The simulation setup is as follows. The network is modeled as a  $10 \times 10$  square with a node at each intersection point  $F[i, j]$ ,  $1 \leq i, j \leq 10$ . The *basic workload unit* consists of 40 identical real-time streams, each of which starts at a boundary node ( $F[i, j]$  with  $i$  or  $j$  equal to 1 or 10) going straight (vertically or horizontally) to the boundary node of the opposite side (via 9 hops). The period and end-to-end deadline of each stream are each 10 time units. The maximum duration of an SCS is 0.01 time units. The energy consumptions for the sleep, idle listening, receiving, and sending modes of the network cards are modeled as 0:1:1.05:1.4 relationship as in [14] and [12]. A node consumes the largest possible amount of energy when it continuously transmits and we normalize the measured energy consumptions to the largest possible energy consumption. Further, the transmission error rate without interference is zero.

In the following graphs, we continuously increase the network workload in multiples of the *basic workload unit* (as described above) and we compare the real-time performance and energy of DARTS with STDMA [6], VMS [10], and 802.11 DCF [1]. STDMA (as proposed in [6]) is a traffic-aware spatial-reuse TDMA mechanism, where the slot assignments of a link are done offline and proportional to the traffic across the link. The slot size of STDMA is set to 0.01 units (the same as the SCS duration in DARTS). We vary the number of slots in a frame to capture the range of optimal frame lengths (between 40 and 64 slots per frame for our workload pattern). A node in STDMA sleeps when it does not transmit or receive data. For VMS, we simulate the prioritized queuing and the contention window of VMS to the best of our knowledge. We simulate both VMS and 802.11 DCF with RTS/CTS enabled and packets of the same stream are transmitted continuously within the same RTS/CTS session, i.e., in the form of RTS/CTS/DATA/DATA/.../ACK. The reservation period of every link in DARTS is 20 time units and the length of an RCS is 0.01 units. To evaluate the performance of the distributed dynamic reservation mechanism for both light loads and overload scenarios, DARTS' admission control mechanism has been disabled. Note that when a stream that should have been rejected is accepted, the  $\Delta$  parameter becomes negative and therefore the local deadline assignment becomes a proportional contraction, rather than a proportional relaxation.

Figure 10 shows the ratio of deadlines met over an increasing number of basic workload units. At light workloads, STDMA, VMA, and DARTS are able to meet all deadlines, while 802.11 DCF meets most of the deadlines. Once the workload increases beyond 2 units, the throughput of real-time streams in DARTS becomes saturated and the ratio of deadlines met begins to decrease. However, in comparison, the number of missed deadlines increases rapidly with 802.11 DCF and VMS due to increased contentions and collisions. Compared to VMS, DARTS is able to meet 30-50% more deadlines. At light workloads, STDMA performs comparable to VMS, but at higher workloads, STDMA outperforms VMS due to its traffic adaptive and contention-free nature. However, DARTS outperforms STDMA due to STDMA's fixed slot arrangement for each frame.

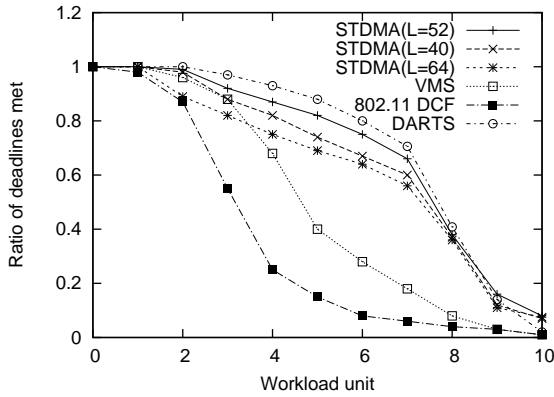


Figure 10. Ratio of deadlines met with varying workload.

Figure 11 shows the energy consumption normalized to the peak energy consumption of a node. The energy consumption of nodes using VMS and 802.11 DCF is very large since nodes must continuously listen to their neighbors. STDMA's energy performance is the lowest, with DARTS' small additional energy consumption being due to nodes staying awake for their neighbors' RCSs. Both STDMA's and DARTS' energy consumption is only 1-2.5% of the peak energy consumption, even for highly loaded networks.

We further implemented a prototype of DARTS as an extension to the Linux kernel. To achieve satisfactory performance of the implementation, the protocol has been implemented immediately above the wireless driver (e.g., using Linux' functions such as `hard_start_xmit` and `netif_rx`) and the Linux timer has been replaced with a high resolution timer (i.e., `hrTimer` with an accuracy of  $< 10\mu s$ ). While there are noticeable performance tradeoffs, our experiments with the Linux implementation indicate that it is feasible to implement a protocol such as DARTS on top of an existing

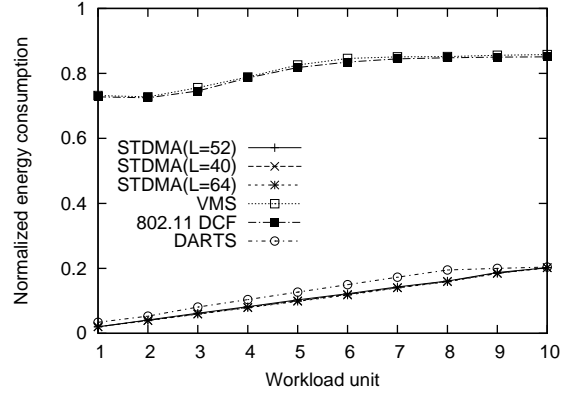


Figure 11. Energy consumption with varying workload.

MAC layer such as 802.11 DCF.

The experimental results for real-time performance (Figure 12) and energy performance (Figure 13) show a similar behavior as the simulations, even though the experimental network setup is small (compared to the simulation setup) and the interference constraints are not complex. Also, we

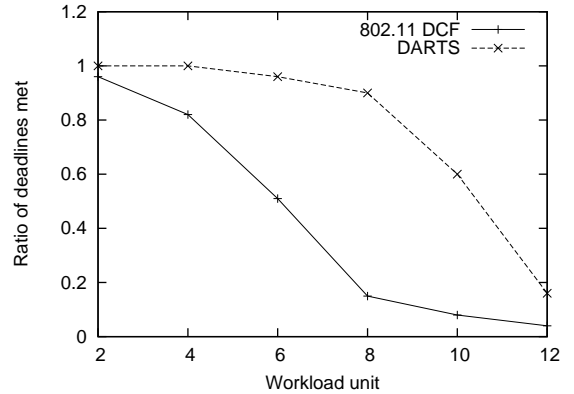
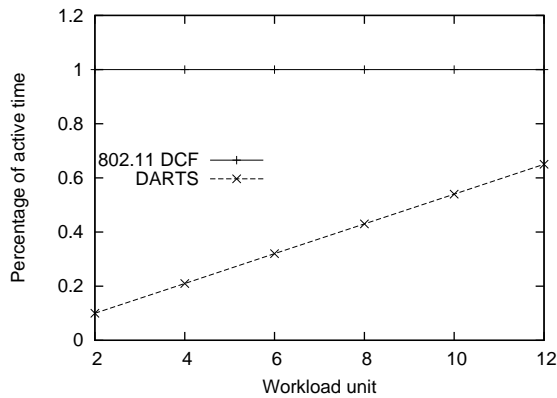


Figure 12. Ratio of deadlines met with varying workload.

further assume that wireless cards are always active in the 802.11 protocol. We measure the active time of DARTS using the communication session times (i.e., when a node listens to the medium and communicates with its neighbors). In medium and high workload scenarios, the average active time per node is high, similar to the results shown via simulation.

## 7 Conclusions and Future Work

Providing timeliness guarantees for real-time streams in energy-constrained multihop networks is a challenging task.



**Figure 13.** Energy consumption with varying workload.

This paper proposes DARTS, a protocol that implements a dynamic reservation technique that ensures contention-free communication among nodes in a wireless ad-hoc network. Furthermore, DARTS' admission control ensures that only those streams are admitted, where DARTS can guarantee satisfactory end-to-end real-time performance. However, experimentation indicates that DARTS performs well, both in terms of timeliness and energy, even in overload scenarios. In summary, DARTS obtains the energy performance of techniques such as STDMA without the inflexibility of frame-based protocols.

Our future work will study adaptive approaches to the local deadline assignment problem, i.e., a node in DARTS will be able to re-negotiate its stream's deadline to allow it to accept more real-time streams. Further, DARTS assumes that a route has been established before admission control takes place, however, we believe that DARTS' performance can be improved when route establishment and admission control are performed together.

## References

- [1] IEEE 802.11 WG Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE 802.11 Standard*, 1999.
- [2] B. D. Bui, R. Pellizzoni, M. Caccamo, C. F. Cheah, and A. Tzakis. Soft real-time chains for multi-hop wireless ad-hoc networks. In *RTAS'07*, pages 69–80, 2007.
- [3] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *RTSS'02*, pages 39–48, 2002.
- [4] I. Chlamtac and S. S. Pinter. Distributed nodes organization algorithm for channel access in a multihop dynamic radio network. *IEEE Trans. Comput.*, 36(6):728–737, 1987.
- [5] T. L. Crenshaw, S. Hoke, A. Tirumala, and M. Caccamo. Robust implicit EDF: A wireless MAC protocol for collaborative real-time systems. *Trans. on Embedded Computing Sys.*, 6(4):28, 2007.
- [6] J. Grönkvist. Assignment methods for spatial reuse TDMA. In *MobiHoc'00*, pages 119–124, 2000.
- [7] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *ICDCS'03*, page 46.
- [8] H. Li, P. Shenoy, and K. Ramamritham. Scheduling messages with deadlines in multi-hop real-time sensor networks. In *RTAS'05*, pages 415–425, 2005.
- [9] J. W. S. Liu. *Real-Time Systems*. Prentice Hall, Upper Saddle River, NJ, 2001.
- [10] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *RTAS'02*, pages 55–66, 2002.
- [11] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *SenSys'03*, pages 181–192, 2003.
- [12] I. Rhee, A. Warriar, M. Aia, and J. Min. Z-MAC: a hybrid MAC for wireless sensor networks. In *SenSys'05*, pages 90–101, 2005.
- [13] Y. C. Tseng, C. S. Hsu, and T. Y. Hsieh. Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. In *INFOCOM'02*, pages 200–209, 2002.
- [14] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(3):493–506, 2004.
- [15] J. Yi, C. Poellabauer, S. X. Hu, J. Simmer, and L. Zhang. Energy-conscious co-scheduling of tasks and packets in wireless real-time environments. In *RTAS'09*, pages 265–274, 2009.