CFlood: A Constrained Flooding Protocol for Real-Time Data Delivery in Wireless Sensor Networks ⋆

Bo Jiang[1], Binoy Ravindran[1], and Hyeonjoong Cho[2]

[1] Department of Electrical and Computer Engineering,
Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061, USA
{bjiang,binoy}@vt.edu
[2] Department of Computer and Information Science,
Korea University,
Seoul, Korea 136-701
raycho@korea.ac.kr

Abstract. Real-time performance is critical for many time-sensitive applications of wireless sensor networks. We present a constrained flooding protocol, called CFlood that enhances the deadline satisfaction ratio per unit energy consumption of time-sensitive packets in sensor networks. CFlood improves real-time performance by flooding, but effectively constrains energy consumption by controlling the scale of flooding, i.e., flooding only when necessary. If unicasting meets the distributed sub-deadline of a hop, CFlood aborts further flooding even after flooding has occurred in the current hop. Our simulation-based experimental studies show that CFlood achieves higher deadline satisfaction ratio per unit energy consumption than previous multipath forwarding protocols, especially in sparsely deployed or unreliable sensor network environments.

1 Introduction

Real-time performance is one of the most important QoS (Quality of Service) metrics for time-sensitive applications of wireless sensor networks (or WSN). For example, a target tracking system [1] may require sensors to collect and report target information to sink nodes before the target leaves the surveillance field. For improving real-time performance, we need to ensure that as many time-sensitive packets as possible, arrive at sink nodes within their deadlines. The delay that a packet may experience during transmission may be caused by many reasons, including those due to network congestion and node/link failures.

Multipath forwarding is a commonly used approach for enhancing various QoS metrics of WSN traffic [2]. With multiple paths, network congestion and node/link failures can be bypassed, and real-time performance can be improved. However, it is possible that network congestion or the connection status is not significant enough throughout the entire path from source to sink to warrant multipath forwarding for each hop.

⋆ This work was supported by research project grants from MKE/MND, South Korea.

Therefore sometimes, the redundant copies of data packets generated by multipath forwarding protocols, which often consume additional energy and bandwidth, are not necessary. Since sensor nodes are battery-powered and therefore energy must be efficiently consumed, energy-efficient forwarding protocols are critical toward enhancing the capability of delivering real-time packets within given end-to-end time constraints, while reducing the energy consumption.

Many previous research efforts [3, 4, 5, 6, 7, 8] have studied the efficiency of multipath forwarding protocols toward enhancing real-time performance while consuming optimized resources (we discuss related work in Section 5). Majority of these efforts have focused on reducing the number of flooding recipients at each hop so that the additional resource consumption can be minimized and QoS constraints such as real-time can also be satisfied. But even though the number of recipients is reduced, this approach introduces redundancy due to its probability-based recipient selection mechanism. In fact, for those hops in which the connection status is good enough so that unicasting does work, redundant multipath flooding is not necessary. The efficiency of flooding therefore can be further improved by controlling the redundancy.

In this paper, we present a constrained flooding protocol called CFlood that improves the flooding efficiency in sensor networks. The primary objective of CFlood is to enhance the deadline satisfaction ratio per unit energy consumption. We adopt the *deadline satisfaction ratio* (or *DSR*) [9] to characterize the real-time performance of a WSN, which is defined as the ratio of the number of real-time packets that arrive at sink nodes meeting their deadlines, to the total number of those transmitted.

CFlood uses flooding to enhance the deadline satisfaction ratio, and controls the scale of flooding to effectively reduce energy consumption. We design CFlood mainly in four components, including neighborhood table management, real-time guarantee verification, recipient selection and flooding control. CFlood maintains a neighborhood table on each node to save the information for routing and neighboring relations, and uses periodic HELLO message exchanges to estimate the per-hop delays and update the entries in the table. We also introduce a deadline partition scheme to distribute the end-to-end deadline to multiple hops. By comparing the estimated per-hop delays and the distributed sub-deadlines, the real-time guarantee verification component justifies whether a neighbor node can meet the deadline for a specific packet. Among the neighbors that can meet the deadline, CFlood selects a primary recipient and several secondary recipients according to the criteria on flooding-controllability, congestion avoidance, and computation simplicity. In addition, CFlood aborts further flooding from secondary recipients if unicasting to the primary recipient can meet the distributed sub-deadline. CFlood is designed as a hop-by-hop routing protocol with no global network information needed. Thus, it is scalable for large-scale sensor networks.

We conducted extensive simulation-based experimental studies to evaluate CFlood's performance. Our results reveal that CFlood achieves a higher deadline satisfaction ratio per unit energy consumption than previous multipath data delivery protocols, such as a multipath routing protocol MCMP [2] and a directional flooding protocol DFP [10], especially in sparse or unreliable network environments.

The paper makes the following contributions:

– We design a constrained flooding protocol that improves the flooding efficiency by enhancing the deadline satisfaction ratio per unit energy consumption. A flooding control mechanism is developed based on the cross-layer design, by adding a plug-in block to the MAC protocol.
– We compare the performance of CFlood against previous efforts through the simulation-based experimental studies. To the best of our knowledge, we are not aware of any other protocols that achieves a higher deadline satisfaction ratio per unit energy consumption than what CFlood yields.

The rest of the paper is organized as follows. In Section 2, we outline our system model. The design details of CFlood are presented in Section 3. Section 4 describes our simulation results. We compare and contrast past and related work against CFlood in Section 5, and conclude the paper in Section 6.

2 System Model

First we introduce the system model including the network, nodes, communication and the underlying MAC protocol.

Network. We assume a homogeneous WSN architecture. Thus, there are no cluster heads or relay nodes with different communication capabilities than that of other sensor nodes. We only consider many-to-one data transmission, i.e., each sensor node sends packets to only a single sink node.

Nodes. We assume that sensor nodes are static and are equipped with omni-directional antennas. Both the transmission range within which nodes can communicate, denoted as R, and the sensing range within which nodes can detect events, denoted as r, are fixed.

Communication. We adopt the protocol model in [11] for communication, where both transmission and interference depend only on the Euclidean distance between nodes.

MAC protocol. We assume that the underlying MAC protocol supports collision avoidance with the RTS/CTS (Request To Send and Clear To Send) exchange mechanism [12]. This mechanism is commonly used for handling the hidden terminal problem [13], which is a major cause of channel contention in wireless networks.

For convenience in discussion, we then summarize all the notations that we use in the paper in Table 1.

3 CFlood: A Constrained Flooding Protocol

We first describe CFlood's design intuition, and then discuss its functional components in detail.

3.1 Overview

CFlood is a decentralized flooding-based routing protocol. Routes are determined, i.e., recipients are chosen at each hop dynamically during data transmission. CFlood uses

Table 1: Notations

R	Transmission range	$NB(N_i)$	Neighbors of node N_i
r	Sensing range	$Parent(N_i)$	The parent of node N_i
L_h	Estimated per-hop delay	$Source(N_i)$	The source node from which N_i receives a packet
D_h	Distributed per-hop deadline	$Forward(N_i)$	Flooding recipients of node N_i
PR	Primary recipient	$Hop(N_i)$	The number of hops that N_i is away from the sink
SR	Secondary recipient	ρ	Node density
T_h	The average throughput of a node at hop h	C_h	The number of nodes that are h hops away from the sink node
SL_i	Slack time ratio	DSR	Deadline satisfaction ratio
δ	Real-time capacity per unit energy consumption	e	The average energy for transmitting a single time-sensitive packet

flooding to increase the deadline satisfaction ratio. But flooding may not be necessary at each hop. It is desirable to constrain the scale of flooding as much as possible to enhance energy efficiency.

We describe energy efficiency by measuring the average energy e consumed for transmitting a single time-sensitive packet, i.e., the ratio of the total energy consumption to the total number of time-sensitive packets generated. With a flooding protocol, a single packet may be copied multiple times. Thus, the total energy consumption consists of the energy for transmitting and receiving all the copies. We do not emphasize the unit of energy, since it depends on the specific hardware platform. We also define the metric *real-time capacity per unit energy consumption* as $\delta = \frac{DSR}{e}$, which measures what percentage of time-sensitive packets is delivered meeting their deadlines for unit energy consumption. Thus, higher δ is, the more efficient the flooding protocol is. The primary objective of CFlood therefore can be described as improving the real-time capacity per unit energy consumption δ as much as possible.

Our approaches used for controlling the scale of flooding and correspondingly increasing δ include: 1) reducing the flooding actions as much as possible, and using unicasting instead; and 2) reducing the number of recipients when flooding is necessary.

Our intuition in controlling the flooding scale of CFlood with the first approach is as follows. After a recipient at a given hop finds that the transmission of its next hop can meet the sub-deadline, it is unnecessary for other recipients to continue flooding for the next hop. Thus, this recipient can abort the subsequent flooding of other ones. For this hop, it seems as unicasting is used instead of flooding. This way, the end-to-end time constraint can be satisfied and the energy efficiency can be enhanced.

Even if flooding is necessary, we should reduce the number of recipients. We use several criteria for recipient selection, in which the end-to-end time constraint is the primary one. First, we introduce a deadline partition scheme to distribute the end-to-end deadline to multiple hops. Then, CFlood estimates the per-hop delays between nodes. If the estimated per-hop delay is longer than the distributed sub-deadline, it is highly unlikely that the route through the node can meet the end-to-end time constraint.

Otherwise, the node can be chosen as a recipient of the flooding. With this approach, the number of recipients can potentially be reduced with respect to meeting the end-to-end time constraint.

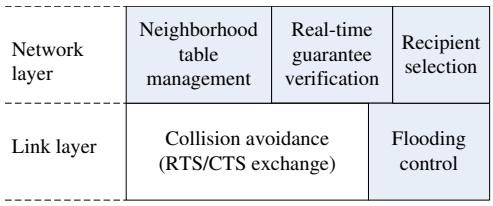


Fig. 1: Functional components of CFlood

Based on this intuition, we design CFlood with four functional components, which are shown in a network protocol stack in Figure 1. The components are described as follows:

- *Neighborhood table management.* CFlood maintains a neighborhood table on each node to save the routing information, the neighboring relations, and the estimated per-hop delays. These information fields are shared between neighbors through periodic HELLO message exchanges, and are used for making flooding decisions.
- *Real-time guarantee verification.* Among all the one-hop neighbors, we want to flood only to those nodes that can satisfy the time constraint. This component verifies whether a neighbor can meet the end-to-end time constraint, and therefore could be considered as a flooding recipient. This decision is made by comparing the estimated per-hop delay, denoted as L_h, and the distributed per-hop sub-deadline, denoted as D_h. If $L_h < D_h$, i.e., the transmission at this hop can be completed within the sub-deadline distributed to this hop, then this neighbor could be selected as a prospective recipient.
- *Recipient selection.* First, each node periodically computes a next-hop neighbor (or parent as in [14]), which has the highest probability of meeting the time constraint. This parent node is used as the primary recipient (or PR) of flooding. Then, each node also selects several secondary recipients (or SRs) based on the time constraint as well as the criteria on flooding-controllability, congestion avoidance and computation simplicity. When unicasting is determined to be sufficient for meeting a packet's time constraint, the PR aborts the SRs' next-hop flooding (this is done through the flooding control mechanism discussed next). Otherwise, the PR and all the SRs continue to flood the packet further.
- *Flooding control.* The working sequence of a MAC protocol, which supports collision avoidance with the RTS/CTS exchange mechanism, can be summarized as RTS-CTS-DATA or RTS-BACKOFF, depending on whether the RTS/CTS exchange succeeds. For controlling the scale of flooding, CFlood inserts an ABORT phase after CTS for the PR's flooding, and a WAIT phase before RTS for SRs' flooding. Thus, the working sequence of the PR will be modified as RTS-CTS-ABORT-DATA or RTS-BACKOFF, and that of the SRs will be modified as WAIT-ABORT,

WAIT-RTS-CTS-DATA or WAIT-RTS-BACKOFF. If a PR finds that the channel is clear after receiving a CTS, it broadcasts an ABORT message so that SRs can abort their subsequent flooding actions.

We now describe each of these components in detail.

3.2 Neighborhood Table Management

This component manages the neighborhood table, each entry of which corresponds to a neighbor node. An entry includes the following fields:

(NeighborID, ParentID, HopCount, SendDelay, TTL)

ParentID is the ID of this neighbor node's parent. HopCount is the number of hops by which the neighbor node is away from the sink node. SendDelay is the estimated delay for sending a packet to this neighbor node. TTL is short for Time To Live. The table management operations include adding, updating, expiring, and removing.

The entries in the neighborhood table are updated via HELLO message exchanges, which is a commonly used approach for sharing local knowledge among neighbors [3]. The mechanism has the advantage that it can adapt the network to possible topology changes (e.g., those caused by link failure, node failure). Each HELLO message includes the fields (SenderID, ParentID, HopCount, SendDelay), so that all the receivers may update the entry in their neighborhood tables for the node that sends this HELLO message. For example, at the beginning, when a network is deployed, each node in the network holds an empty neighborhood table. From a HELLO message received from a sink node, all the neighbors of this sink node will know that their HopCount is 1. By iteration, the HopCount will be increased by 1 at each hop from the sink node to all other nodes in the network. (We will discuss the selection of a node's parent and the estimation of SendDelay later in this section.)

3.3 Real-time Guarantee Verification

As previously discussed, CFlood compares the estimated per-hop delay L_h with the distributed per-hop sub-deadline D_h to determine whether or not a potential recipient can satisfy the time constraint. This component is responsible for estimating L_h, computing D_h, and conducting the comparison.

Per-hop delay estimation The delay experienced at a hop usually consists of the transmission delay, the propagation delay, and the receiving delay. The transmission delay is the time that a data packet experiences at the MAC and PHY layers of the sender. The propagation delay is the duration when a data signal together with its carrier travels in the air. The receiving delay is the time that a data packet experiences at the PHY and MAC layers of the receiver. Since the delay includes parts at both the sender and the receiver, the precise measurement will require time synchronization, which is generally energy inefficient [15]. We introduce a feasible mechanism without assuming time synchronization, although our estimation result may not be perfectly precise due to the asymmetry of wireless channels.

The problem of estimating the round-trip delay has been well studied in the past [16]. We simply apply the existing method into the HELLO message exchange mechanism. Suppose the neighbors of a node N are $\{N_i | N_i \in NB(N)\}$. Node N may append a round-trip delay estimation request for a specific neighbor node N_i in a randomly chosen HELLO message (e.g., one out of every twenty continuous HELLO messages). Neighbors other than N_i deal with this HELLO message as usual, while N_i is supposed to reply with a HELLO message immediately. Then a round-trip delay is obtained by node N, the half of which can be used as the estimated per-hop delay and is saved in the Send-Delay field of N_i's entry. This does not require time synchronization since the starting and ending time points of the round trip are sampled at the same node.

Per-hop deadline computation Most of the past works on the end-to-end deadline partition [17, 18] have adopted either uniform or exponential models. Uniform distribution allocates the total end-to-end deadline evenly to all the hops from the source to the sink, implicitly assuming that a packet suffers the same delay at each hop. The exponential model computes the per-hop sub-deadline as $D_h = \frac{D}{2^h}$, where h is the number of hops from the sink node and D represents the end-to-end deadline. These schemes are based on analytical models and do not consider the actual throughputs of the network. We introduce a deadline partition model by establishing a relationship between the per-hop sub-deadline and the number of nodes at each hop in an intuitive manner.

Let ρ denote the node density of the network. Now, the average number of nodes that are h hops away from the sink node, denoted as C_h, can be computed as $\rho(\pi(hR)^2 - \pi[(h-1)R]^2) = \rho\pi R^2(2h-1)$. Intuitively, at a specific hop in a convergecast network, lesser the number of nodes, greater will be the traffic that each node has to transport toward the sink node. Thus, longer will be the delay that a packet will suffer at the hop. Consequently, a longer sub-deadline will be needed for the hop. This relationship can be approximately modeled as $D_h \sim T_h \sim \frac{1}{C_h}$, where T_h is the average throughput of a node at hop h. When the node density ρ is fixed for a given implementation, we have $D_h \sim \frac{1}{2h-1}$. Thus, the end-to-end deadline D over an h-hop transmission can be distributed to each hop k as $D_k = \frac{\frac{1}{2k-1}}{\sum_{k=1}^{h}\frac{1}{2k-1}} \cdot D$. Especially the sub-deadline of the first hop from the source is:

$$D_h = \frac{\frac{1}{2h-1}}{\sum_{k=1}^{h}\frac{1}{2k-1}} \cdot D \tag{1}$$

Figure 2 shows the comparison among the uniform model, the exponential model, and the throughput-based model, for an example with a 200 ms end-to-end deadline over 20 hops. We can observe that compared with the uniform model, the throughput-based model is more adaptive for the many-to-one convergecast architecture of WSNs. In addition, compared with the exponential model for which the distributed per-hop deadline decreases quickly to zero, the throughput-based model supports a larger-scale network.

For a single node N, its hop count from the sink node may be different when considering different routes via different neighbor nodes. Therefore the result of (1) needs to be computed for each potential recipient. Suppose HopCount of a neighbor node N_i

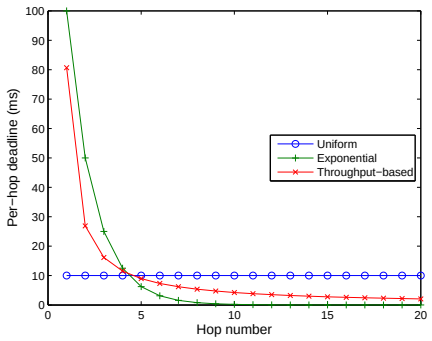


Fig. 2: Deadline Distribution

is h_i. Now, when N_i is used for relaying, the distributed per-hop sub-deadline at this hop is:

$$D_{h_i+1} = \frac{\frac{1}{2(h_i+1)-1}}{\sum_{k=1}^{h_i+1} \frac{1}{2k-1}} \cdot D \tag{2}$$

Real-time guarantee verification For a specific neighbor N_i, we establish a function $CanMeetDeadline(N_i)$, in which the per-hop deadline D_{h_i+1} is computed and compared with the per-hop delay L_{h_i+1} (i.e., the value of SendDelay in the neighborhood table). The function $CanMeetDeadline(N_i)$ returns true when $L_{h_i+1} < D_{h_i+1}$ and false otherwise. Only those neighbors that has a true return value will be considered as potential recipients.

3.4 Recipient Selection

The recipient selection component is responsible for selecting the flooding recipients (both a PR and multiple SRs) from the one-hop neighbors. With CFlood, each node computes a parent node as the PR periodically even when no data packets are passing through. The criteria used for SR selection include real-time guarantee, flooding-controllability, congestion avoidance, and simplicity of computation.

Primary recipient As one of CFlood's techniques to constrain the flooding scale is to substitute unicasting for flooding as much as possible, a parent needs to be prepared for each node as the next-hop neighbor of unicasting.

For a neighbor N_i, (2) shows the distributed per-hop deadline D_{h_i+1}. We can also estimate the per-hop delay L_{h_i+1} with the round-trip HELLO message exchange. Thus, we define a ratio $SL_i = 1 - \frac{L_{h_i+1}}{D_{h_i+1}}$ to describe the proportion of the slack time, and call it, the *slack time ratio*. The slack time ratio describes how likely N_i can meet a packet's sub-deadline for this hop. Based on our throughput-based deadline partition model, the slack time ratio also describes how likely a route via N_i can meet the end-to-end time

constraint. Therefore, we select a neighbor N_i with the maximum SL_i as the parent node, i.e., a neighbor with

$$\max_i(SL_i) \sim \min_i\{\frac{L_{h_i+1}}{D_{h_i+1}}\} = \frac{1}{D}\cdot\min_i\{L_{h_i+1}\cdot\frac{\sum_{k=1}^{h_i+1}\frac{1}{2k-1}}{\frac{1}{2(h_i+1)-1}}\}$$

Secondary recipients For selecting SRs, we progressively remove those neighbor nodes that cannot satisfy the following criteria:

1. *Real-time guarantee.* An SR should meet the time constraint (i.e., $CanMeetDeadline(N_i)$ returns true).

2. *Flooding-controllability.* The subsequent flooding of an SR should be able to be aborted by the PR.

3. *Congestion avoidance.* An SR should not introduce new congestion, since CFlood's major objective is to quickly bypass network congestion or connection failure. Thus we remove redundant SRs that share the parent with other SRs and have lower probabilities for meeting the time constraint. By strictly prohibiting two recipients from sharing a common parent, the network congestion could be avoided at least for the next hop.

4. *Simplicity of computation.* CFlood is designed to be as simple as possible due to the constrained computing capability of sensor nodes. We use a set of "common sense-based" operations to quickly reduce the problem size before applying the first three ones. These quick reduction operations include: 1) remove all the neighbors whose parent is also a neighbor of the flooding node, and thus also has a chance to receive the packet at this hop, i.e., $Parent(N_i) \in NB(N)$; 2) remove all the neighbors that send packets to the flooding node at the last hop, i.e., $N_i = Source(N)$; and 3) remove all the neighbors that have received packets at the last hop, i.e., $N_i \in Forward(Source(N))$.

Next, we describe the SR selection algorithm at a high-level of abstraction in Algorithm 1. The algorithm complexity is $O(n^2)$ for searching N_i in $Forward(Source(N))$, where n is the average number of one-hop neighbors of a node. The magnitude of n is small. For example, our simulation shows that in a network with node density 0.005 $node/m^2$ (i.e., each node covers an area of 200 m^2), n is only up to 5.

3.5 Flooding Control

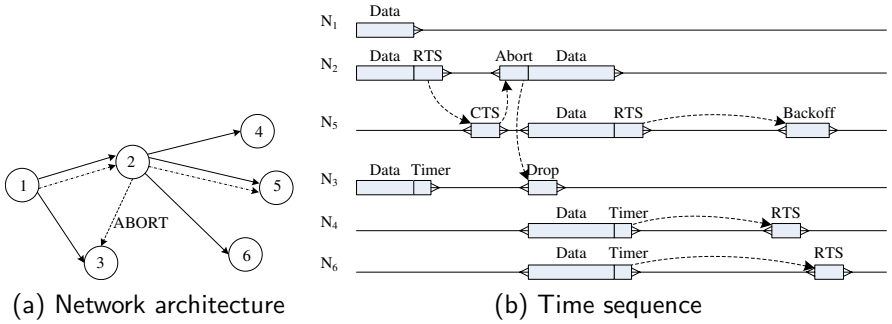
One of the most important contributions of this paper is the flooding control mechanism, i.e., to abort the subsequent flooding after the current flooding occurs. In detail, the PR and the SRs forward packets in different ways. As the flooding node of the next hop, the PR initiates an RTS/CTS exchange with its PR immediately after receiving a packet. If a CTS is received successfully, the PR broadcasts an ABORT message and then unicasts the data packet. Otherwise, the PR backs off for some period of time and again initiates the RTS/CTS exchange later. Unlike the PR, the SRs set an ABORT timer for each received data packet. If an ABORT message from a PR is received for a buffered data packet, the SRs drop that packet. Otherwise, if the timer runs out first, the SRs know that the PR's flooding for the next hop is delayed (i.e., backed off), and therefore they start to flood the packet to the next hop. In this way, when the network condition is good (e.g.,

Algorithm 1 Secondary recipient selection

1: Initialize the SR candidate set as $SR = \{N_i | N_i \in NB(N), Hop(N_i) < Hop(N)\}$;
2: **for all** ($N_i \in SR$) **do**
3: Examine N_i with the conditions of three quick reduction operations, i.e., remove N_i if ($Parent(N_i) \in NB(N)$) or ($N_i = Source(N)$) or ($N_i \in Forward(Source(N))$);
4: Remove N_i if it violates the time constraint, i.e., if ($CanMeetDeadline(N_i)$=**false**);
5: Remove N_i if it violates the flooding-controllability criterion, i.e., if it cannot hear from the PR ($N_i \notin NB(PR)$);
6: Remove N_i if it violates the congestion avoidance criterion by sharing a parent with the PR, i.e., if ($Parent(N_i) = Parent(PR)$);
7: **end for**
8: **if** ($SR == \phi$) **then**
9: **return** ϕ
10: **end if**
11: Sort the remaining SRs in a descending order of SL_i;
12: **for all** ($N_i \in SR$) **do**
13: Remove N_i if it violates the congestion avoidance criterion by sharing a parent with another SR with a higher SL_i, i.e., if ($\exists j < i, Parent(N_i) = Parent(N_j)$);
14: **end for**
15: **return** SR

the RTS/CTS exchange initiated by the PR succeeds without backoff), the flooding is reduced to unicast (because the SRs drop the packet on receiving the ABORT message).

Overhearing is also an approach for controlling flooding [19], e.g., the SRs abort the subsequent flooding upon overhearing the transmission of the PR. However, overhearing is not a good choice under the time constraint. Not until the SRs overhear the complete packet payload and send it up to the network layer, can they drop the corresponding packet saved in the buffer. Such a transmission through the network stack may introduce extra delays, especially when the data packet is long.



(a) Network architecture (b) Time sequence

Fig. 3: An example of CFlood's flooding control mechanism

Figure 3 shows an example of CFlood's flooding control mechanism. In the figure, the circle with the number i represents node N_i. In Figure 3a, the dash-dot arrows show the parent relation (e.g., $Parent(N_2) = N_5$), the solid lines show the actual data transmission (e.g., $Forward(N_1) = \{N_2, N_3\}$), and the dotted line represents the ABORT messages. Figure 3b shows the time sequence of the nodes. The dotted curves show the working mechanism of flooding control. When N_2 (as the PR of N_1) finds that the channel is clear on receiving the CTS reply, it broadcasts an ABORT message and then sends out the data packet. Node N_3 (as an SR of N_1) receives the ABORT message from N_2 before the timer runs out. Thus, it aborts the subsequent flooding and drops the packet. On the contrary, N_5 does not receive the CTS reply from its parent node. Thus, it has to backoff for sometime without broadcasting an ABORT message. As N_4 and N_6 do not receive the expected ABORT message before the timer runs out, they have to continue the flooding by initiating the RTS/CTS exchange. The expiration time for the ABORT timers on SRs can be either determined by specific application configurations, or computed as the minimum allowed slack time $\min_i\{D_{h_i+1} - L_{h_i+1}\}$ of that SR node at the next hop.

4 Experimental Evaluation

We evaluated CFlood using the simulation tools Qualnet 4.0 [20] and sQualnet [21], which is an extension to Qualnet for sensor networks. The simulation is based on the CSMA/CA MAC protocol implemented in sQualnet.

4.1 Simulation Environment

We deploy 20 to 150 sensor nodes uniformly in a square area of $200m \times 200m$, and assume Mica motes [22] as the hardware platform. We set $R = 60m$, $r = 30m$, and the data rate as 38.4 kbps. We leverage the statistics provided by sQualnet to estimate the energy consumption.

Each sensor node samples and reports an event (e.g., detection of a target) once per second. We configure the lengths of a data packet, a HELLO message, and an ABORT message as 150 bytes, 50 bytes, and 10 bytes, respectively. Usually 10 bytes (e.g. including the ID of the source node and the ID of this packet) are long enough for an ABORT message to identify a specific data packet.

We compared CFlood against three past competitor algorithms. Mint routing (or MR) is a single path delivery protocol [14], which serves as a lower bound on both real-time performance and energy consumption. MCMP [2], a multipath routing protocol, is one of the latest efforts on optimizing data delivery under both real-time and reliability constraints. DFP [10] (short for Directional Flooding Protocol) is a forwarding protocol that optimizes the delivery probability. We measure the performance metrics of interest of CFlood and these protocols under varying degrees of node density, link reliability, and end-to-end time constraint, the default values of which are 0.0015 $node/m^2$, 75%, and $100ms$, respectively.

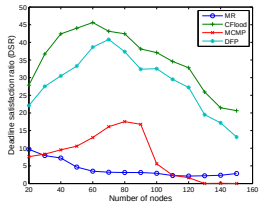


Fig. 4: Deadline satisfaction ratio *DSR* vs. node density

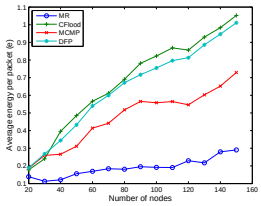


Fig. 5: Average energy consumption e vs. node density

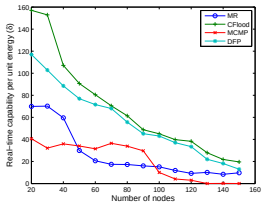


Fig. 6: Real-time capacity δ vs. node density

4.2 Simulation Results

Figure 4 shows the deadline satisfaction ratio of the four protocols under different node densities. Due to the competition of the two factors, the number of recipients and the congestion level, the curves present crests. We observe that CFlood yields the best *DSR*.

Figure 5 shows the average energy consumption per real-time packet under different node densities. We observe that MR, as the lower bound, consumes the least energy, and CFlood consumes the most. But the energy consumption of CFlood is very close to that of DFP.

Figure 6 shows the real-time capacity per unit energy consumption δ under different node densities. We observe that CFlood performs the best, especially when the node density is low. Detailed simulation results show that on average, CFlood is 197%, 346%, and 20% better than MR, MCMP and DFP, respectively.

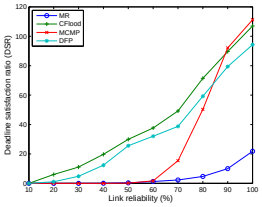


Fig. 7: Deadline satisfaction ratio *DSR* vs. link reliability

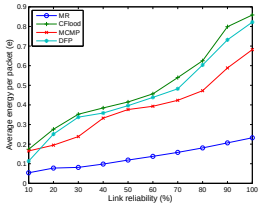


Fig. 8: Average energy consumption e vs. link reliability

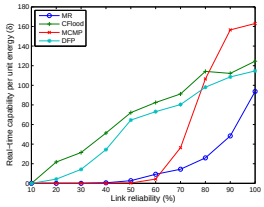


Fig. 9: Real-time capacity δ vs. link reliability

Figure 7 shows the deadline satisfaction ratio under various per-hop wireless link reliability. Among the four protocols, CFlood yields the best *DSR*, especially when the link reliability is low. As the link reliability increases, the *DSR*s of CFlood, MCMP, and DFP tend to be comparable, and MR remains as the lower bound. We observe that MCMP performs well especially when the network condition is good, while CFlood is more adaptive to unreliable network environments.

Figure 8 shows that CFlood consumes the most energy. However, when we consider the real-time capacity per unit energy consumption δ, CFlood outperforms the other three protocols when the link reliability is lower than 80%, as shown in Figure 9.

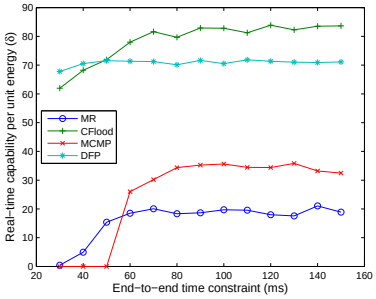


Fig. 10: Real-time capacity δ vs. end-to-end time constraint

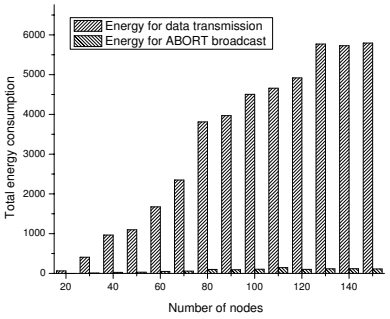


Fig. 11: Energy for data transmission and ABORT broadcast

Figure 10 shows that the real-time capacity per unit energy consumption δ of CFlood is higher than that of the other three protocols, as long as the end-to-end time constraint is not very tight.

CFlood's flooding control mechanism based on ABORT messages does introduce some overheads. However, the extra energy consumption is negligible. Figure 11 shows the contrast between the energy consumption for data transmission and that for the ABORT message exchange. We observe that the energy consumption for ABORT messages is only about 1% of the energy consumption for data transmission. This implies that CFlood's flooding control mechanism introduces little overhead.

Thus, our simulation results reveal that CFlood achieves better real-time capacity per unit energy consumption than past protocols, especially for sparse node deployment, unreliable wireless links, and loose end-to-end time constraints.

5 Related Work

Existing works on supporting real-time traffic in WSNs have focused on latency and timely delivery from various perspectives. For example, Abdelzaher *et. al.* discussed a WSN's real-time capacity from a macro perspective without making any detailed assumptions [17]. He *et. al.* present detailed delay bounds for each chain of target tracking applications in [1]. Other efforts considered real-time performance as a constraint that must be satisfied [23]. In contrast, we focus on improving the deadline satisfaction ratio per unit energy consumption.

In unreliable network environments, past data delivery protocols use different approaches to guarantee timeliness. Single path forwarding protocols transmit one copy of data along a predetermined single path, and depend on retransmissions to guarantee reliability [24]. In contrast, multipath data delivery protocols transmit a number of copies through multiple routes simultaneously. This will increase the reliability and the probability of delivering real-time data in a timely manner, but often at the expense of additional energy consumption.

Multipath routing protocols can be broadly classified into two categories, static routing and dynamic routing. Static multipath routing protocols setup multiple routes, which

are either disjoint [3] or braided [4], before sending out a data packet. The source nodes then either choose one of the routes or combine the resources of all the routes for a single flow. In contrast, dynamic multipath forwarding protocols decide the flooding recipients at each hop so that the data flow is split distributively.

Dynamic routing protocols can be further classified into multicast, broadcast (or flooding), and gossip. Multicast has been extensively studied for WSNs [5], most of which aim at multihop one-to-many communications. Broadcasting and flooding are usually used interchangeably, but there are also works that distinguish them with minor differences [25]. The HHB scheme introduced in [6] is a hop-by-hop broadcast protocol that leverages the broadcasting capability of wireless medium to guarantee the reliable delivery. The HHB protocol broadcasts at each hop with a specific probability to avoid degenerating into a flooding storm. Zhang *et. al.* present a constrained flooding protocol in [7], which exhibits good energy efficiency by constraining retransmissions. Gossip can be considered as a form of probabilistic flooding. In [8], Lu *et. al.* present a gossip algorithm called NBgossip, which forwards the linear combinations of the received messages.

Almost all of the multipath data delivery protocols introduce extra overheads even when unicasting is enough for satisfying QoS constraints such as timeliness. In contrast, CFlood uses flooding but constrains the energy consumption effectively by controlling the scale of flooding.

6 Conclusions

This paper presents a constrained flooding protocol, called CFlood, that enhances the real-time capability per unit energy consumption in WSNs. Besides the fundamental functions of a routing protocol such as neighborhood table management, we present a flooding control mechanism based on ABORT message exchanges, and a recipient selection method to reduce energy consumption. Our experimental evaluation based on Qualnet shows that CFlood outperforms past multipath routing/forwarding protocols, especially for sparsely deployed networks or unreliable wireless links. This result reveals that the cross-layer design of CFlood's flooding control, e.g. substituting unicasting for flooding as much as possible, effectively improves the flooding efficiency.

Directions for future work include:

1) improving CFlood's flooding and energy consumption control mechanisms to be more adaptive;

2) achieving reliability guarantees; and

3) extending CFlood to sensor networks with multiple sink nodes.

References

1. He, T., Vicaire, P., Yan, T., et al., L.L.: Achieving real-time target tracking using wireless sensor networks. ACM Transaction on Embedded Computing System (TECS) (2007)
2. Huang, X., Fang, Y.: Multiconstrained qos multipath routing in wireless sensor networks. Wirel. Netw. **14**(4) (2008) 465–478

3. Lu, Y.M., Wong, V.W.S.: An energy-efficient multipath routing protocol for wireless sensor networks: Research articles. Int. J. Commun. Syst. **20**(7) (2007) 747–766
4. De, S., Qiao, C., Wu, H.: Meshed multipath routing with selective forwarding: an efficient strategy in wireless sensor networks. Computer Networks **43**(4) (2003) 481–497
5. Silva, J.S., Camilo, T., Pinto, P., Rodrigues, R.R.A., Gaudncio, F., Boavida, F.: Multicast and ip multicast support in wireless sensor networks. In: Networks. (2008) 19–26
6. Deb, B., Bhatnagar, S., Nath, B.: Information assurance in sensor networks. In: WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications. (2003) 160–168
7. Zhang, Y., Fromherz, M.: Constrained flooding: A robust and efficient routing framework for wireless sensor networks. In: AINA: Proceedings of the 20th International Conference on Advanced Information Networking and Applications. (2006) 387–392
8. Lu, F., Chia, L.T., Tay, K.L., Chong, W.H.: Nbgossip: An energy-efficient gossip algorithm for wireless sensor networks. Journal of Computer Science and Technology **23**(3) (2008) 426–437
9. Li, P., Ravindran, B., Wang, J., Konowicz, G.: Choir: A real-time middleware architecture supporting benefit-based proactive resource allocation. Object-Oriented Real-Time Distributed Computing, IEEE International Symposium on **0** (2003) 292
10. Ko, Y.B., Choi, J.M., Kim, J.H.: A new directional flooding protocol for wireless sensor networks. In: Springer-Lecture Notes in Computer Science. Volume 3090. (2004) 93–102
11. Gupta, P., Kumar, P.: The capacity of wireless networks. Information Theory, IEEE Transactions on **46**(2) (2000) 388–404
12. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE (approved in 1999, reaffirmed in 2003, revised in 2007)
13. Ye, W., Heidemann, J.: Medium access control in wireless sensor networks. Technical Report ISI-TR-580, USC/Information Sciences Institute (October 2003)
14. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems. (2003) 14–27
15. Sivrikaya, F., Yener, B.: Time synchronization in sensor networks: a survey. Network, IEEE **18**(4) (July-Aug. 2004) 45–50
16. IETF: Rfc 2681 - a round-trip delay metric for ippm, http://www.ietf.org/rfc/rfc2681.txt?number=2681
17. Abdelzaher, T.F., Prabh, S., Kiran, R.: On real-time capacity limits of multihop wireless sensor networks. In: RTSS '04: Proceedings of the 25th IEEE International Real-Time Systems Symposium. (2004) 359–370
18. Liu, K., Abu-Ghazaleh, N., Kang, K.D.: Jits: just-in-time scheduling for real-time sensor data dissemination. Fourth Annual IEEE International Conference on Pervasive Computing and Communications (2006) 5–46
19. Basu, P., Redi, J.: Effect of overhearing transmissions on energy efficiency in dense sensor networks. Third International Symposium on Information Processing in Sensor Networks (IPSN), 2004. (2004) 196–204
20. Scalable-Networks: Qualnet network simulator, http://www.scalable-networks.com/
21. Vasu, B., Varshney, M., Rengaswamy, R., Marina, M., Dixit, A., Aghera, P., Srivastava, M., Bagrodia, R.: Squalnet: a scalable simulation framework for sensor networks. In: SenSys: Proceedings of the 3rd international conference on Embedded networked sensor systems, ACM (2005) 322–322
22. CrossBow: Mica data sheet, http://www.xbow.com
23. Lu, G., Sadagopan, N., Krishnamachari, B., Goel, A.: Delay efficient sleep scheduling in wireless sensor networks. In: IEEE Infocom. (2005)

24. Willig, A., Karl, H.: Data transport reliability in wireless sensor networks - a survey of issues and solutions. In: Praxis der Informationsverarbeitung und Kommunikation. (2005) 86–92
25. Pleisch, S., Balakrishnan, M., Birman, K., van Renesse, R.: Mistral: Efficient flooding in mobile ad-hoc networks. In: In Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), ACM Press (2006) 1–12