# Flash Flooding: Exploiting the Capture Effect for Rapid Flooding in Wireless Sensor Networks

Jiakang Lu and Kamin Whitehouse
Department of Computer Science, University of Virginia
{jklu,whitehouse}@cs.virginia.edu

*Abstract*—**We present the *Flash* flooding protocol for rapid network flooding in wireless sensor networks. Traditional flooding protocols can be very slow because of *neighborhood contention*: nodes cannot propagate the flood until neighboring nodes have finished their transmissions. The Flash flooding protocol avoids this problem by allowing concurrent transmissions among neighboring nodes. It relies on the *capture effect* to ensure that each node receives the flood from at least one of its neighbors, and introduces new techniques to either recover from or prevent too many concurrent transmissions. We evaluate the Flash flooding protocol on both a 48-node wireless sensor network testbed and in a trace-based simulator. Our results indicate that the Flash flooding protocol can reduce latency by as much as 80%, achieving flooding latencies near the theoretical lower bound without sacrificing coverage, reliability or power consumption.**

## I. Introduction

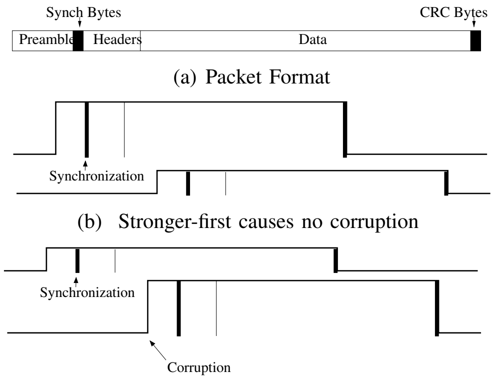
A *network flood* is a protocol that relays a message from a source node to all other nodes in the network. Network floods are common and important operations that are at the heart of most wireless sensor network (WSN) algorithms. They are the starting point for routing tree creation [9], [37], clock synchronization [29], code and data dissemination [22], [28], [36], node localization [34], and group formation [27]. Network floods form the foundation upon which essentially all other global network operations are built.

However, network floods are costly in terms of both latency and bandwidth. In most cases, the flood must be propagated by all nodes in order to ensure complete and reliable coverage of the network and therefore a key cause of flooding latency is *neighborhood contention*: nodes at the flood's frontier cannot propagate the flood immediately because they must wait for nodes in the flood's interior. Our experiments show that neighborhood contention typically makes flooding six times slower than the theoretical lower bound. This problem is exacerbated in low-duty cycle networks where protocols like LPL [5], [32] and X-MAC [2] send extremely long packets: in networks with one-second long packets, a node may have to wait several seconds for its parent and neighbors to finish transmitting before it can propagate the flood. Long delays in turn reduce overall flooding throughput: a flooding protocol that requires 15 seconds to complete can only relay 4 messages per minute to the network, while a protocol that completes in 1 second can relay 60 messages per minute.

In this paper, we explore several ways to improve flooding latency by exploiting the *capture effect*, which is the ability of some radios to correctly receive one of several concurrently transmitted messages, even if the received strengths of the two messages are almost the same [21]. We exploit this in a network flooding scenario by allowing nodes to concurrently propagate a flooding message. Since all nodes transmit the same flooding message, they can do so concurrently as long as every node receives the message from at least one of them. Our hypothesis is that this approach will decrease flooding latency by eliminating neighborhood contention, without sacrificing coverage, reliability or power consumption.

Several recent studies have shown that too many concurrent transmissions result in packet loss despite the capture effect [41]. Therefore, a fine balance must be achieved: being too aggressive with concurrency will result in collisions and packet loss while being too conservative will result in latency due to neighborhood contention. We develop techniques to automatically control the number of concurrent transmissions and to restart floods that have halted due to collisions and packet loss. We call the combination of these techniques the *Flash* flooding protocol. We evaluate Flash by running it on a 48-node wireless testbed called VineLab that uses the ChipCon CC2420 radio [4]. We also evaluate at larger scale and at higher density with a simulator that uses thousands of empirical data traces that we collected on VineLab. Our results indicate that Flash can reduce latency in high-duty cycle networks by 75% and in low-duty cycle networks by 80% while still achieving 100% coverage. These results approach the theoretical lower bound on flooding latency.

Rapid flooding can have immediate impact on many WSN applications. For example, rapid floods will allow the Marionette macroprogramming system to send more commands to the network each second and improve the execution speed [36]. Rapid floods will enable the Clairvoyant debugger to more quickly freeze all nodes in order to inspect global distributed state [39]. Finally, Flash will be a critical component for *rapid wakeup* in event detection networks, a canonical application of WSNs. Low-duty cycle WSNs may be deployed across Southern California to monitor for wildfires, or across entire cities to monitor for traces of chemical or biological weapons, and an event detection in such networks should trigger the entire network to quickly wake up and respond to the situation. The Flash flooding protocol will dramatically reduce the response time of such networks. Our experiments focus on WSNs where network flooding is extremely common but our techniques and results should generalize to all mesh networks.



(a) Packet Format

(b) Stronger-first causes no corruption

(c) Stronger-last causes corruption

Figure 1. Stronger-first collisions do not cause packet loss; however, stronger-last collisions do.

## II. Background and Related Work

The *capture* effect, also called *co-channel interference tolerance*, is the ability of some radios to receive a signal from one transmitter despite interference from another transmitter, even if the relative strengths of the two signals are almost the same [21]. If a packet radio exhibits the capture effect, it automatically receives one packet in what we call *stronger-first* collisions, where the stronger packet arrives first. In this case, the weaker packet arrives while the stronger packet is already being received, as shown in Figure 1(b), and does not cause interference. Therefore, the stronger packet is received but the weaker packet is lost. This phenomenon has been demonstrated in many networks, including 802.11 networks [20]. In a *stronger-last* collision, the stronger packet corrupts the end of the weaker packet and both messages are lost, as shown in Figure 1(c).

The ability to receive a packet during a collision can provide a number of important benefits for wireless networking, including higher throughput, lower latency and improved spatial reuse. For example, capture has been shown to increase throughput and decrease delay in Aloha [1], [30] and 802.11 networks [3], [16], [17], [40] with no additional effort. Some physical and link layer protocols have explicitly exploited the capture effect in order to improve performance such as Capture Division Packetized Access (CDPA) [8] and Power Control during Transmission (PCT) [19]. Similarly, the capture effect has been used as a mechanism for providing different *quality of service* to different clients by allowing them to control transmission power [31]. Others have shown that physical layer approaches can be used to detect collisions and to recover packets from stronger-last collisions [35]. *Conflict maps* have been used to control concurrent point-to-point transmissions [15]. In this paper, we exploit the capture effect to improve the latency and throughput of network flooding.

The Flash flooding protocol has two variants that target both always-on and low-power networks, respectively. In low-power networks, the nodes typically duty cycle between active state and sleep state in order to conserve energy. This allows nodes to sense periodically while still achieving lifetimes of months or years. One common way to maintain communication between low-duty cycle nodes is called low-power listening (LPL) where each packet is preceded by a preamble that is as long as a node's sleep interval [5], [32]. All nodes are guaranteed to wake up and check the channel at least once when this preamble is being transmitted, at which point they remain in the active state until the preamble is completed and the message is received. X-MAC has recently improved on this approach by replacing the preamble with a repeating pattern of the data packet [2]. This allows the receiver to return to sleep mode immediately after it receives any copy of the packet. In a network flood, LPL and X-MAC produce high latency because each node must wait for its parent's transmission to finish before repeating the packet. Neighborhood contention exacerbates these delays because a node waits for its neighbors and even its parent's neighbors to finish transmitting. Despite its high latency, a flood using LPL or X-MAC is currently the only existing technique that can be used for network-wide flooding in low-duty cycle CSMA networks. The techniques we present in this paper substantially improve on the latency of an LPL flood.

Many protocols such as SPIN [12], Trickle [23] and Deluge [13] have been proposed for flooding wireless sensor networks, but all of these focus on either reliability or message overhead; none of them explicitly optimize for latency. Many previous papers have also studied low-latency and real-time communication such as SPEED [11] and others [33], but these papers propose point-to-point, multicast, or data collection protocols; none of them propose low latency flooding protocols. Other approaches have studied low-latency and real-time communication in low-duty cycle networks, a process which we call *rapid wakeup*. Most use some form of *phase synchronization* such as D-MAC [25]. Similar approaches have been developed to minimize latency for floods, end-to-end paths, and ring topologies [18], [24], [26]. However, these approaches require the synchronization of sleep phases *with respect to some source or destination*. They do not provide low-latency communication from arbitrary locations in the network. Furthermore, they place constraints on the sleep cycles and phase of nodes in the network which are often modified by the application for other purposes such as sensing. Our approaches in this paper do not require phase synchronization and are application independent.

## III. The Flash Approach

The *Flash* flooding protocol exploits the capture effect to reduce flooding latency by allowing nodes to propagate the flood simultaneously, thereby eliminating neighborhood contention. We explore this approach in two fundamental network scenarios in wireless sensor networks: (i) high-duty cycle networks where all nodes are always on and (ii) low-duty cycle networks where nodes sleep most of the time and wake up only periodically to see if a message is being sent. We make no assumptions about the duty cycle phase of each node with respect to its neighbors and the entire network is asynchronous.

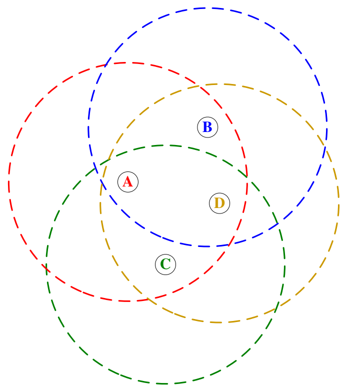


Figure 2. Example scenario in which node A is the source node.

We present Flash in three parts. Section III-A describes the most basic implementation called Flash-I, which is observed to suffer from coverage and reliability problems. Section III-B describes a version called Flash-II that addresses these problems. Section III-C describes a second variant called Flash-III that improves flooding throughput in low-duty cycle networks. We present these schemes in the context of the network shown in Figure 2. In this diagram, node $A$ is the source node and has nodes $B$, $C$ and $D$ in its transmission range. Node $D$ is a neighbor of both $B$ and $C$, but nodes $B$ and $C$ are not neighbors to each other.

### A. Flash-I: Complete Concurrency

Flash-I is identical to a standard flooding protocol except that nodes do not use media access control before propagating the flood. Thus, nodes in a Flash-I flood repeat the message as soon as they receive it, even if their neighbors are still transmitting. They do not perform clear channel assessment (CCA) as would a node in a CSMA network, nor do they wait their turn as would a node in a TDMA network. The result is that multiple nodes in the same neighborhood will be transmitting simultaneously. Flash-I can be applied in both high-duty cycle and low-duty cycle networks by transmitting either individual packets or X-MAC packet trains, respectively.

Figure 3 illustrates the dynamics of a Flash-I flood in a low-duty cycle version of the network shown in Figure 2. As soon as nodes $B$ and $C$ wake up, they each receive one of the messages in the source node $A$'s packet train and immediately begin transmitting their own packet trains without any CCA or MAC delay. Therefore, by the time $C$ begins transmitting there are three packet trains being transmit simultaneously. This high level of concurrency may cause packet collisions despite the capture effect, and in Figure 3 node $D$ does not receive the flood message because when it wakes up because there is a collision between the packets being sent by nodes $A$ and $B$. Thus, Flash-I has the potential of achieving the theoretical lower bound on latency but is not guaranteed to achieve high network coverage. As discussed in Section V, our experiments show that Flash-I floods can stop prematurely due to collisions, without covering the entire network.

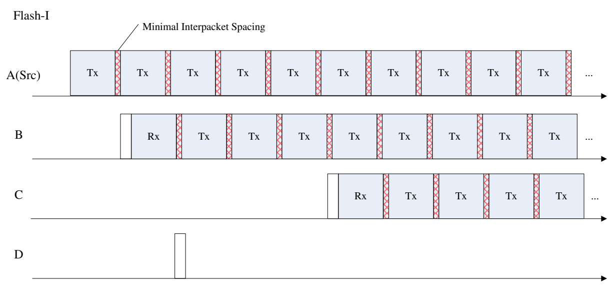


Figure 3. Flash-I flooding example: nodes B and C wake up and send immediately, simultaneously with A. Node D does not wake up due to a collision between A and B where the capture effect did not help.

### B. Flash-II: Maintained Concurrency

In Flash-II, each node sends a three-part flooding sequence. First, it immediately sends a packet transmission with no CCA or MAC delay, just like Flash-I. Then, it performs CCA and uses a MAC delay to wait for all neighbors to finish transmitting. Finally, it sends a second message, identical to the first. The purpose of the second packet is to reach any nodes that missed the first wave of packets due to concurrency and collisions. The CCA/MAC delay preceding the second message ensures that it will not interfere with the first wave of packets and that it will not itself be lost in a collision. The net effect is that the second packet is highly likely to reach any node or nodes that missed the first wave of packets, effectively *jump starting* the flood if it ended prematurely due to collisions. Flash-II can be applied straightforwardly to both high-duty cycle and low-duty cycle networks by using either single-packet transmissions or packet trains, respectively.

Flash-II has the potential to achieve the theoretical lower bound on flooding latency, just like Flash-I, while at the same time guaranteeing at least the same coverage and reliability as a traditional CSMA flood or X-MAC flood. If the first wave of packets stops prematurely due to collisions, Flash-II will introduce a single CCA/MAC delay before restarting the flood with a second message. Thus, in scenarios where Flash-I would not achieve full network coverage, Flash-II would achieve latencies slightly higher than the theoretical lower bound. Unlike Flash-I, Flash-II does not improve flooding throughput because the second transmissions take as long to complete as a traditional CSMA-like network flood.

### C. Flash-III: Controlled Concurrency

Flash-I's failure to achieve full network coverage highlights the fine balance that must be achieved to exploit the capture effect in a flood: being too aggressive with concurrency will result in collisions and packet loss, while being too conservative will result in latency due to neighborhood contention. This is particularly important for low-duty cycle networks where collisions are more likely because the long packet times increase the level of concurrency. To address this problem, Flash-III applies a new technique to *sense* the amount of transmission concurrency in a network. First, we introduce a small *inter-packet spacing* (IPS) between packets in the

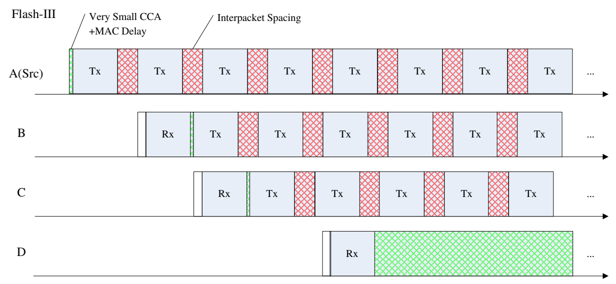


Figure 4. Flash-III flooding example: nodes B and C are able to transmit because the CCA period is smaller than the interpacket spacing. However, as the level of concurrency increases, node D is not able to transmit.

packet train. Second, we introduce a very small CCA before the packet train is sent, where $CCA < IPS$.

The key idea behind Flash-III is that a nodes is more likely to *pass* the CCA check during low levels of concurrency because its CCA interval is more likely to coincide with the IPS intervals of the transmitting nodes. Therefore, it will also begin transmitting and will increase the level of concurrency. On the other hand, a node is unlikely to pass the CCA check during high levels of concurrency because its CCA interval is more likely to coincide with at least one packet. This is illustrated in Figure 4, where the flood begins when the source node $A$ transmits a packet train. During this packet train, all of its neighbors $B$, $C$ and $D$ wake up and hear a flooding message. The CCA of nodes $B$ and $C$ coincide with the IPS intervals and these nodes transmit immediately, but the CCA of node $D$ overlaps with the packets of $C$ and so it will wait until $C$ has finished transmitting before propagating the flood. Node $D$ could have transmitted if the other nodes had larger IPS intervals. Thus, Flash-III can control the maximum degree of concurrency through a parameter $\alpha$, defined to be the ratio of the CCA and the IPS: $\alpha = \frac{CCA}{IPS}$. When $\alpha$ is small, nodes are more likely to transmit and a high degree of concurrency will be achieved, and vice versa. Thus, Flash-III is essentially the same as X-MAC when $\alpha \geq 1$, and the same as low-duty cycle Flash-I when $\alpha = 0$.

The value of $\alpha$ can be chosen to manage the trade-off between flooding latency and reliability. In a sensitivity analysis, we found that a value of $\alpha = 0.1$ to be quite robust, although details about these experiments are not included due to space limitations. The value of the IPS and CCA intervals can also be set to any value. In our experiments, we set the IPS equal to the packet length to ensure that a packet from one node will only overlap with at most one packet in the packet train of another node. We chose the CCA interval to be large enough not to produce too many false positives or false negatives. A longer IPS affects power consumption, as discussed in Section VI-B.

Flash-III has the potential to approach the theoretical lower bound on flooding latency but will always be slightly slower due to the small CCA before each packet train. Flash-III does not require a second transmission after the first, so any decrease in latency translates directly into an increase in throughput.

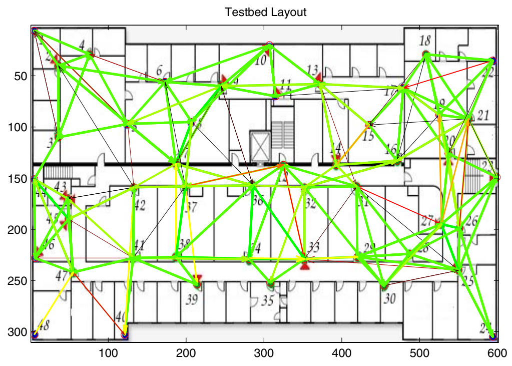


Figure 5. The VineLab topology used in our experiments. Line thickness and brightness indicates link quality at TX Power -10dBm.

## IV. Evaluation

We investigate the Flash protocol with two types of evaluation. First, we empirically evaluate the algorithms on a 48-node testbed. Second, we study the performance of the schemes at multiple scales and densities with our capture-aware simulator. As a baseline for comparison, we introduce an algorithm that provides the theoretical lower bound on the latency of any flooding scheme in an asynchronous network by assuming that all nodes can propagate the flood immediately without ever losing packets due to collision. For high-duty cycle networks, this will produce latencies equivalent to the hop count. For low-duty cycle, this will produce latencies slightly larger than hop count because latency also depends on the sleep phase of neighboring nodes. To maintain fairness when comparing latency, we only compare the algorithms that consistently provide full network coverage. Thus, Flash-I latencies are not included in the comparisons.

### A. Testbed Evaluation

The Flash protocol is deeply dependent on the physical characteristics of the radio and physical environment in which it is deployed. Therefore, a study of Flash *requires* empirical validation and experimentation in a real network environment.

*1) VineLab Testbed:* To explore the capture effect in a realistic environment, we utilize the VineLab wireless sensor network testbed deployed in the Computer Science building at the University of Virginia. The testbed consists of 48 Tmote-sky motes designed by Sentilla. These motes have a Texas Instruments MSP430 low-power microcontroller and a ChipCon CC2420 IEEE 802.15.4 compliant low-power radio. The nodes are deployed in an office environment with faculty, students and staff working in close proximity to the nodes. Many of the walls in the building are reinforced with steel mesh, which can impede transmissions in the 2.4GHz band. In order to further increase variability in the network, some nodes are deployed as low as one meter from the ground, while others are located just below the three meter drop ceiling.

The topology of the network, along with connectivity information, is depicted in Figure 5. This connectivity map was

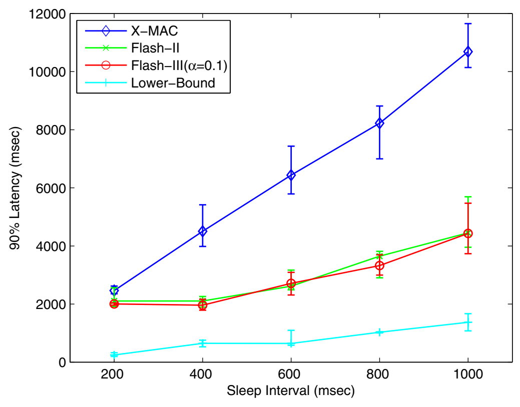


Figure 6. The time required to reach 90% of nodes in the network for various sleep intervals, using the VineLab wireless testbed. The Flash protocols are 60% faster than X-MAC flooding in this scenario.

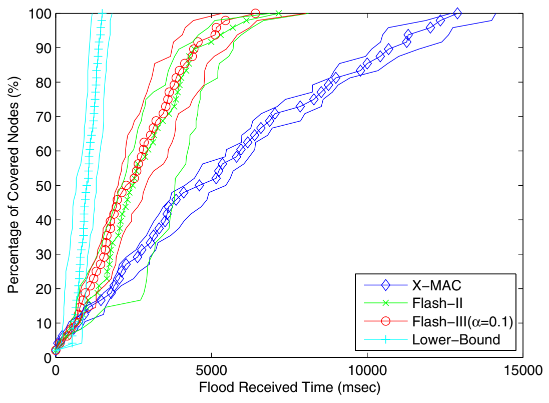


Figure 7. The cumulative distribution of nodes that receive the flood over time, with a sleep interval of 1000 milliseconds. All schemes slow down near the end of the flood indicating edge effects of the testbed.

calculated based on hundreds of readings between nodes at a transmission power of -10dBm. This is the minimal transmission power level that provides a reasonable hop count across the network while maintaining a fully-connected network. The thickness and brightness of the lines are directly proportional to measured link quality.

*2) Testbed Experiments:* We empirically evaluate the flood latency of the lower-bound algorithm, X-MAC, Flash-II and Flash-III($\alpha$=0.1) using the VineLab testbed. The $\alpha$ value of Flash-III was based on the tests showing that full network coverage is guaranteed. For all runs of the experiment, we use node 20 as the source node and a fixed transmission power of -10dBm. This means that our network is approximately 8-hops large with an average node degree of 7.3. To collect the actual flood reception times of each node, we utilize a hybrid of several existing time synchronization protocols which have been demonstrated to achieve between 1-4 microsecond accuracy [6], [7]. We tested each scheme with 5 different sleep intervals: 200 msec, 400 msec, 600 msec, 800 msec, 1000 msec, and assigned random sleep phases to each node. Each experiment is repeated ten times.

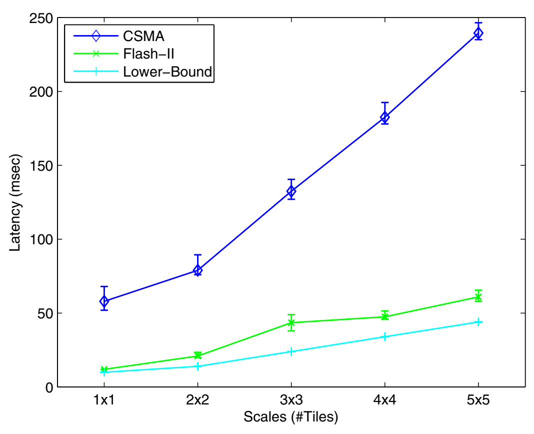
The experimental results are shown in Figure 6, where the lines indicate the median values and the error bars represent the upper and lower quartiles over ten runs. The y-axis indicates the latency with which the first 90% of the network is woken up because we observed that the last 10% of the network can be very slow to receive the flood due to edge effects, as shown by the cumulative distributions in Figure 7, where the curves represent the median, upper, and lower quartiles over the ten trials. Each scheme shows a linear trend for the first 90% of nodes and a slow tail for the last 10%, and we therefore decided to compare the predominant trends rather than the edge effects.

Our results show that latency increases with the sleep interval. As the lower bound results show, no asynchronous flooding scheme can flood the network faster than 1.5 seconds at a sleep interval of 1000 milliseconds. X-MAC flooding completes in 10.8 seconds, while Flash-II and Flash-III complete in 4.3 seconds. Thus, Flash-II and Flash-III are about 60% faster than X-MAC in this flooding scenario. In the testbed experiments, Flash-II and Flash-III show similar performance because the network density is fixed on the physical testbed. With the concurrency sensing mechanism, Flash-III should outperform Flash-II in low-duty cycle networks with higher densities, as shown later in the simulation evaluation.

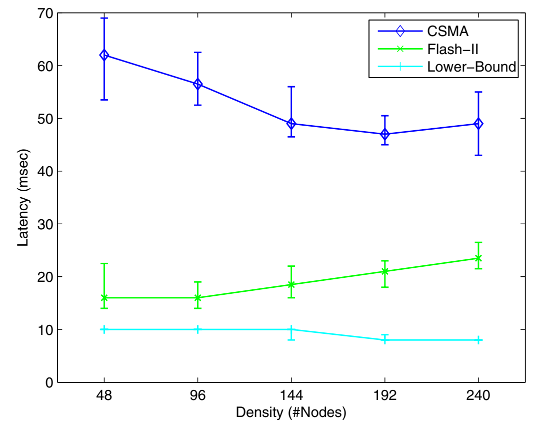
## B. Simulation Evaluation

Our testbed experimental results indicate that network scale and density are likely to play a significant role in practical performance of Flash. Therefore, it is important to evaluate Flash at large scales and high densities. However, controlled experiments on randomly generated topologies are extremely difficult to perform on a physical testbed and testing at large scales or extremely high densities is impossible due to space and resource constraints. To address this problem, we have developed a new *capture-aware* simulation framework that represents the capture dynamics in our testbed by employing empirical traces and statistics. This simulator allows us to evaluate the schemes at scales and densities that are not feasible on the physical testbed.

*1) Simulation Framework:* We define *transmitter set* to be a combination of concurrent transmitters in a single receiver's neighborhood. With a transmission power -10dBm, our VineLab testbed has an average node degree of 7.3 and 19,956 transmitter sets. We empirically measured which nodes receive a packet for each of these transmitter sets and stored the results in what we call a *capture map*. Our capture-aware simulator employs this capture map to accurately reproduce the capture dynamics of the testbed in simulation. Basically, we use the capture map as a *lookup table*. At every point in simulation and for each node, we check which of the node's neighbors are currently transmitting. We take this set of transmitters and look them up in the capture map to see which nodes can receive. If the node in question is in the receiver set, we simulate that the node received a packet. Otherwise, we

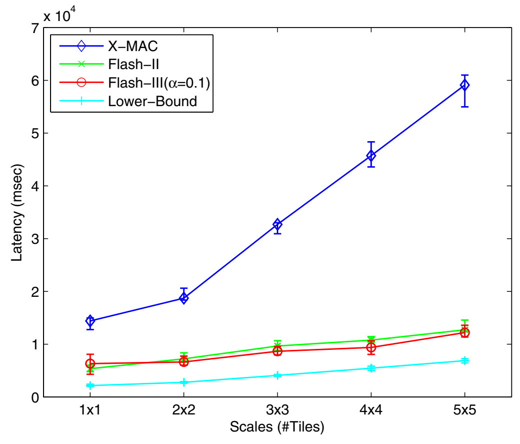


(a) Latency vs. Network Size

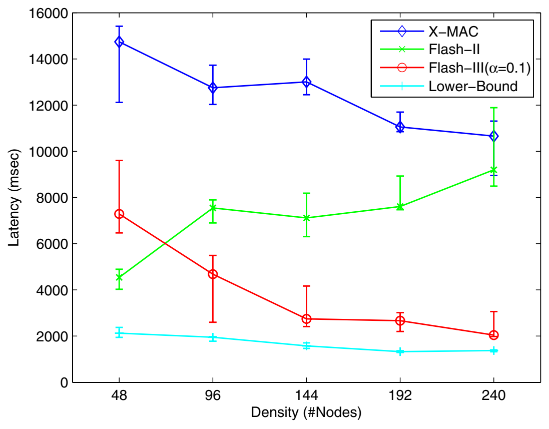


(b) Latency vs. Network Density

Figure 8. Flash-II outperforms CSMA and is close to the lower bound as scale increases, but gets worse as densities increases. Meanwhile, the latency of CSMA flooding and the theoretical lower bound decrease with density.



(a) Latency vs. Network Size



(b) Latency vs. Network Density

Figure 9. Flash-II performance quickly degrades as density increases, while Flash-III($\alpha = 0.1$) approaches the theoretical lower bound. Flash-III is able to keep transmission concurrency at a constant level even as density increases.

simulate that the node experienced corruption and packet loss. To extend the simulation to larger scales, we *tile* our testbed by placing multiple copies next to each other. For example, a 4×4 tiling with 768 nodes is composed of 16 copies of the testbed. To extend the simulator to higher densities, we randomly add nodes to our original 48-node testbed.

The techniques used to stitch multiple testbeds together or to add nodes to the testbed in simulation are based on statistical models of the capture map. Details are not presented in this paper due to space limitations, but we do validate the fidelity of our simulation by comparing simulation results with the testbed results in Figure 6 and Figure 7. We use the same parameters in simulation and the empirical experiments, and compare the curves of the cumulative distribution functions in Figure 7 by applying a paired Student's t-test to the empirical and simulated results. This compares the *progression* of the flood in the empirical experiments with those in simulation, instead of just comparing the final flooding latency. The results show that the flood progressions are identical, with p-values less than 0.01, using ten rounds of experiments over both scale and density simulation.

*2) Simulation Experiments:* We use the capture-aware simulator to evaluate Flash at multiple scales and densities and in both low-duty and high-duty cycle networks. These simulation experiments are complementary to our testbed results. With each scale or density, we run the algorithm twenty times in the simulator. The MAC backoff times in simulation are the same as the standard CC2420 radio stack in TinyOS.

*a) Flooding with High Duty Cycle:* Figure 8(a) shows the performance of Flash-II compared with CSMA flooding and the theoretical lower bound. Flash-III is not included in these experiments because it is only defined for low-duty cycle networks. On a 5×5 tiling of our testbed, CSMA flooding completes in about 240 milliseconds, while Flash-II completes in under 60 milliseconds. The theoretical lower bound is just under 50 milliseconds. Thus, Flash-II is nearly 75% faster than CSMA flooding and is only about 10% slower than the theoretical lower bound.

Figure 8(b) shows the same three schemes evaluated at multiple network densities. All these networks are the same

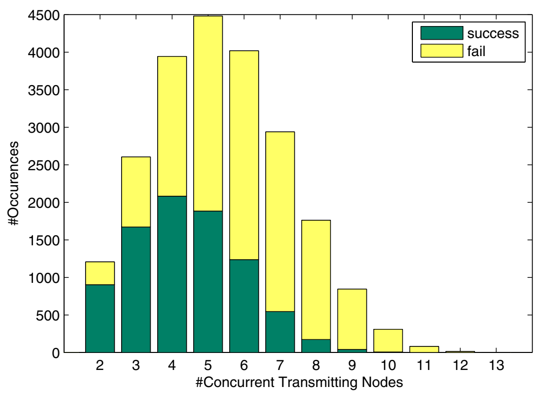


Figure 10. This figure shows the number of times concurrent transmissions from a transmitter set of a particular size are or are not received by neighboring nodes due to capture.

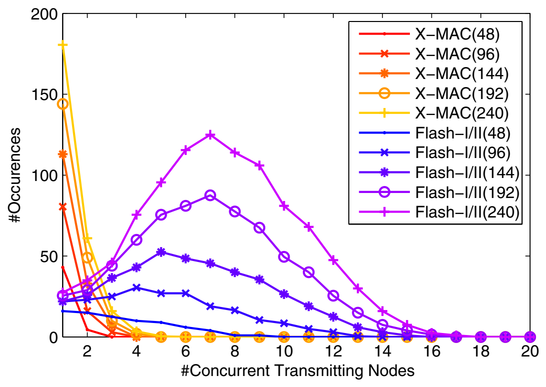


Figure 11. As density increases, the average number of nodes involved in each collision shifts to the right for both X-MAC and Flash-II. Multiplying these distributions by that in Figure 10 explains the performance decrease of Flash-II as density increases, and the performance increase of X-MAC.

size as our 48 node testbed, but have more nodes added to the topology at random locations. The results in this graph show that the latency of Flash-II increases with density. This is because higher density networks produce more concurrency and therefore more collisions. This result indicates that the advantage of Flash-II over CSMA flooding will diminish as density increases. However, even in a network with 240 nodes where the average node degree is as high as 35, Flash-II still outperforms CSMA flooding by almost 50%. Thus, for low latency floods, it will be rare for CSMA flooding to be more appropriate than Flash-II. It is interesting to note that latencies for both CSMA flooding and the theoretical lower bound actually *decrease* with higher density. This is because higher density networks are more likely to have nodes at the very edge of a node's radio range, and so transmissions tend to travel farther on average. Furthermore, higher density networks are more likely to have nodes at the very frontier of the flood, where fewer nodes are transmitting and neighborhood contention is the lowest.

*b) Flooding with Low Duty Cycle:* Figure 9(a) shows the performance of Flash-II and Flash-III in low-duty cycle networks with a sleep interval of 1000 milliseconds, as compared with X-MAC and the theoretical lower bound. The results show similar trends to the high-duty cycle networks: with fixed network density, the Flash protocol outperforms X-MAC across multiple network sizes and approaches the lower-bound algorithm in low-duty cycle networks.

Figure 9(b) shows the latency of all the four schemes at multiple densities. Both Flash-II and Flash-III are significantly faster than X-MAC at low densities and not far from the theoretical lower bound. However, the latency of Flash-II increases with density, suggesting that the long packet trains increase concurrency to the point that packet loss during collisions begin to outweigh the benefits of avoiding neighborhood contention. At high density, Flash-II performs almost six times worse than the theoretical lower bound (although almost always better than X-MAC flooding), while Flash-III($\alpha = 0.1$) approaches the theoretical lower bound. At high density, Flash-III produces average latencies of about 2 seconds, which is about 80% faster than X-MAC flooding and close to the theoretical lower bound.

## V. Analysis and Discussion

Our experimental results reveal several trends regarding Flash: (i) Flash-I does not consistently cover the entire network, (ii) the latency of Flash-II and Flash-III increases linearly with scale when density is held constant and (iii) the latency of Flash-II increases with density while the latency of Flash-III decreases with density. All of these trends can be explained in terms of the balance between too much and too little transmission concurrency.

To explain this balance, we extract Figure 10 from the capture map collected on the VineLab testbed. This figure shows, for simultaneous transmissions from transmitter sets of various sizes, how many of the nodes with connectivity to those transmitters did or did not receive a packet. This histogram shows that the success *ratio* of capture at the receiver decreases as more nodes transmit simultaneously. However, the absolute quantity of nodes that receive the message increases as the size of the transmitter set increases to size four, and there is over a 50% chance that a receiver can successfully hear the message if the transmitter set has five nodes or less. Based on this observation, we expect the flood latencies to decrease at this density as concurrency levels increase to about four simultaneous transmissions and to increase as concurrency increases beyond five.

We then analyze the level of concurrency at different network densities of both X-MAC flooding and Flash-II flooding whose first wave is equivalent to Flash-I, as shown in Figure 11. At low density (48), most of the nodes in a Flash-I/II flood are exposed to five or fewer concurrent transmissions. As the density increases to 240, we see two different effects: (i) the number of concurrent transmissions increases and (ii) the mode shifts to the right. Thus, both the number of concurrent transmissions and the degree of concurrency increase as density increases. In networks with

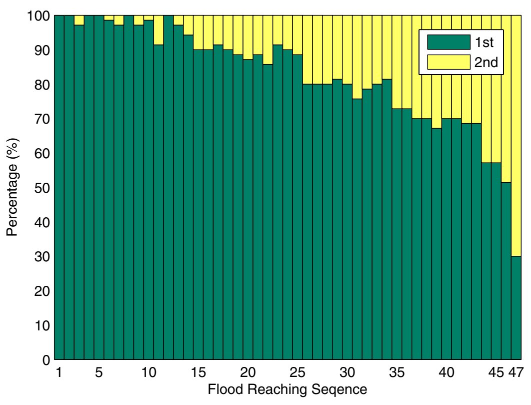


Figure 12. The fraction of nodes that receive the packet in the first wave of Flash-II packets decreases as the flood progresses.

144 nodes or more, over half of all concurrent transmissions have more than five nodes which, as we saw in Figure 10, means that more than half the flood messages are being lost in collisions. This explains why the latency of Flash-II increases with network density. Also, it indicates the reason why Flash-I does not reliably cover the entire network. Figure 12 shows the percentage of times that the $n$th node to be reached by a Flash-II flood heard the first or second packet transmission from its parent. Since the first train in Flash-II is actually the same as Flash-I, these results imply that Flash-I can cover most of the network, though full coverage is almost never achieved.

Finally, Figure 10 also helps explain why the latency of CSMA, X-MAC and Flash-III floods decreases as density increases. The figure shows that the level of concurrency increases in an X-MAC flood as density increases. This is because carrier sensing is more likely to fail to prevent simultaneous collisions as more nodes contend for the channel. However, the concurrency levels are always lower than four, so any increase should be beneficial. The level of concurrency in Flash-III is determined by the value $\alpha$, and presumably higher densities would be harmful to Flash-III when tested with higher values of $\alpha$ where concurrency levels reach the threshold of four.

## VI. Energy Analysis

The energy consumption of Flash variants can be broken down to two parts: the cost of transmitting the flooding protocol and the impact of the flooding protocol on duty cycle. Generally speaking, the cost of transmission is most important when floods are used frequently, while the impact on duty cycle is most important in long-lifetime networks where floods are sent very infrequently. We analyze each of these costs independently.

### A. Flood Transmission

Flash-I sends exactly the same number of messages as a CSMA or X-MAC flood and does not incur any additional power consumption due to transmission of the flood message. On the other hand, Flash-II nodes send the flood message twice, which doubles energy consumption due to transmission. Flash-III actually consumes less energy to propagate a flood than CSMA or X-MAC because the longer IPS used by Flash-III means that packet trains of a fixed length have fewer packets. The percentage of fewer packets transmitted is calculated as $1 - \frac{PL+MIPS}{PL+IPS}$, where $PL$ is the packet length and $MIPS$ is the minimum inter-packet spacing, which is a property of the radio hardware. We used an oscilloscope to measure these values on the CC2420 radio used in our experiments and found $PL = 1.4$ msec and $MIPS = 0.8$ msec. For the value $IPS = 1.4$ msec that we used in our experiments, as described in Section III-C, Flash-III transmits 21% fewer packets than X-MAC flooding.

The number of transmissions required to propagate a flood could conceivably be reduced by first finding a minimum connected dominating set (MCDS) and then limiting the flood propagation to only these nodes. This approach is orthogonal to Flash and could be applied to CSMA or X-MAC floods as well as Flash floods. We implemented the first algorithm provided by Guha and Khulle [10] in our testbed and simulator. The use of MCDS significantly improves the energy cost of Flash-II, consuming only 18% more energy than X-MAC flooding. However, we also observed that MCDS usually fails to achieve full network coverage on the testbed due to link dynamics; the very low level of redundancy makes the algorithm fragile and sensitive to link failure. Furthermore, our experimental results indicate that MCDS does not make much difference in terms of flooding latency and the benefit of MCDS only become apparent at very high network densities.

### B. Duty Cycling

In low-duty cycle networks, Flash-I and Flash-II send exactly the same packet trains that would be sent by X-MAC flooding. So the nodes do not need to change the way that they wake up to check the channel for incoming packets. On the other hand, Flash-III may use an increased IPS, which means that the nodes must wake up for a longer time to check for incoming packets. Specifically, they must wake up and sense the channel for at least $IPS + PL$ before going back to sleep. This will increase the duty cycle of the nodes and therefore decrease their lifetime. The amount of the increase can be calculated as $\frac{IPS-MIPS}{T}$, where $T$ is the duty cycle period. Thus, for very large periods a small increase in $IPS$ may not be significant, but for very small periods it will be. Typically, $T$ will be relatively large for networks where power is a major concern and for typical values of $T$ such as 60 seconds, Flash-III only consumes 0.001% more power than an X-MAC flood.

## VII. Conclusions

The Flash protocol enables multiple nodes to transmit concurrently in order to eliminate neighborhood contention and reduce the latency of network floods in wireless networks. Flash relies on the capture effect to ensure that packets are received during concurrent transmissions, exploiting the fact that all nodes in a flood transmit the same message so that each node can receive the message from *any* of its neighbors. A key issue in the design of Flash is to manage transmission concurrency: too much results in packet collisions while too little leads to neighborhood contention. Flash proposes two flooding-specific mechanisms to manage concurrency. The first, used in Flash-II for both high- and low-duty networks, is to transmit the flooding packet twice, with CCA and MAC delay in between. This allows the network to recover from floods that stop prematurely due to high concurrency. This protocol can approach the theoretical lower bound on flooding latency and sacrifices flooding throughput for reliability. The second mechanism, used in Flash-III for low-duty cycle networks only, is to use a large IPS and a very small CCA. The effect of this combination is to allow some but not too much concurrency and operate on the probability that the CCA of a node overlaps with the IPS of all nodes currently transmitting. This protocol approaches the theoretical lower bound on flooding latency and does not compromise reliability or power consumption. Flash is the first network flooding protocol for wireless networks that explicitly exploits the capture effect to optimize for latency. The implementation of Flash only involves changes to tens of lines of code with respect to the standard CC2420 radio stack of TinyOS [42], which will allow this protocol to have an immediate and practical impact.

## References

[1] Bhaskar Ramamurthi and Adel A.M. Saleh and David J. Goodman. Perfect-Capture ALOHA for Local Radio Communications. *IEEE Journal on Selected Areas of Communications*, 5, June 1987.

[2] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble mac protocol for duty-cycled wireless sensor networks. In *SenSys '06*.

[3] CC Chow and VCM Leung. Performance of IEEE 802.11 Medium Access Control Protocol over a Wireless Local Area Newtork With Ditributed Radio Bridges. In *IEEE WCNC '00*.

[4] ChipCon. CC2420 data sheet, rev 1.2. http://www.datasheetarchive.com/CC2420-datasheet.html.

[5] A. El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *IEEE ICC '02*.

[6] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *IPDPS '01*.

[7] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *OSDI '02*.

[8] Flaminio Borgonovo and Luigi Fratta and Michele Zorzi and Anthony Acampora. Capture Division Packet Access: a New Cellular Access Architecture for Future PCNs. *IEEE Communications Magazine*, September 1996.

[9] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: scalable point-to-point routing in wireless sensornets. In *NSDI'05*.

[10] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[11] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. Speed: a stateless protocol for real-time communication in sensor networks. In *ICDCS '03*.

[12] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99*.

[13] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04*.

[14] K. Jamieson, B. Hull, A. K. Miu, and H. Balakrishnan. Understanding the Real-World Performance of Carrier Sense. In *ACM SIGCOMM E-WIND Workshop '05*.

[15] M. Vutukuru, K. Jamieson, and H. Balakrishnan. Harnessing Exposed Terminals in Wireless Networks. In *NSDI '08*.

[16] JH Kim and JK Lee. Capture Effects of Wireless CSMA/CA protocols in Rayleigh and Shadow Fading Channels. *IEEE Transactions on Vehicular Technology*, 48, July 1999.

[17] G. Kai, H. Honglin, W. Guo, and Z. Jinkang. A capture-based mac protocol and its performance analysis. In *Vehicular Technology Conference '03*.

[18] A. Keshavarzian, H. Lee, and L. Venkatraman. Wakeup scheduling in wireless sensor networks. In *MobiHoc '06*.

[19] Kouichi Mutsuura and Hiromi Okada and Kazuhiro Ohtsuki and Yoshikazu Tezuka. A New Control Scheme With Capture Effect for Random Access Packet Communications. In *IEEE ICC '89*.

[20] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *WinTECH '07*.

[21] K. Leentvaar and J. Flint. The Capture Effect in FM Receivers. *IEEE Transactions on Communications*, 24(5):531–539, 1976.

[22] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks. In *ASPLOS X*, 2002.

[23] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *NSDI'04*.

[24] Y. Li, W. Ye, and J. Heidemann. Energy and latency control in low duty cycle MAC protocols. In *IEEE WCNC '05*.

[25] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in sensor networks. In *WMAN '04*.

[26] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay efficient sleep scheduling in wireless sensor networks. *IEEE INFOCOM '05*.

[27] L. Luo, T. F. Abdelzaher, T. He, and J. A. Stankovic. Envirosuite: An environmentally immersive programming framework for sensor networks. *Trans. on Embedded Computing Sys.*, 5(3):543–576, 2006.

[28] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *OSDI '02*.

[29] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *SenSys '04*.

[30] C. Namislo. Analysis of mobile radio slotted aloha networks. *IEEE Journal on Selected Areas in Communications*, 2(4):583–588, Jul 1984.

[31] A. Nyandoro, L. Libman, and M. Hassan. Service differentiation in wireless lans based on capture. In *Global Telecommunications Conference '05*.

[32] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04*.

[33] T. Watteyne, I. Augé-Blum, and S. Ubéda. Dual-mode real-time mac protocol for wireless sensor networks: a validation/simulation approach. In *InterSense '06*.

[34] K. Whitehouse and D. Culler. A robustness analysis of multi-hop ranging-based localization approximations. In *IPSN '06*.

[35] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler. The effects of ranging noise on multihop localization: an empirical study. In *IPSN '05*.

[36] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler. Marionette: using rpc for interactive development and debugging of wireless embedded networks. In *IPSN '06*.

[37] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03*.

[38] N. Sadagopan, B. Krishnamachari, and A. Helmy. The ACQUIRE mechanism for efficient querying in sensor networks. In *SNPA'03*.

[39] J. Yang, M. L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: a comprehensive source-level debugger for wireless sensor networks. In *SenSys '07*

[40] Zoran Hadzi-Velkov and Boris Spasenovski. Capture Effect in IEEE 802.11 Basic Service Area Under Influence of Rayleigh Fading and Near/Far Effect. *IEEE Internation Symposium on Personal Indoor Communication*, 2002.

[41] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. *IEEE SECON '04*.

[42] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. TinyOS: An Operating System for Sensor Networks. *Ambient Intelligence*