The Firecracker Protocol

Philip Levis and David Culler
{pal,culler}@eecs.berkeley.edu

EECS Department
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT

We propose the Firecracker protocol for data dissemination in wireless sensor networks. Firecracker uses a combination of routing and broadcasts to rapidly deliver a piece of data to every node in a network. To start dissemination, the data source sends data to distant points in the network. Once the data reaches its destinations, broadcast-based dissemination begins along the paths, like a string of firecrackers.

By using an initial routing phase, Firecracker can disseminate at a faster rate than scalable broadcasts while sending fewer packets. The selection of points to route to has a large effect on performance, indicating possible requirements for any-to-any routing protocols in wireless sensor networks.

1. INTRODUCTION

Reliably disseminating a piece of data to every node is a fundamental primitive for wireless sensor networks. Example uses include disseminating a new program to re-task the network, a pattern for in-network event detection, a communication schedule for radio duty cycling, or configuration constants to tune operation. For event-driven or rapid response systems, such as a medical sensor network, dissemination speed can be critical. However, dissemination must also be energy efficient, to maintain system lifetime.

Energy-efficient flooding is generally slow, to prevent broadcast storms, where large numbers of responses lead to collisions and high packet loss. To minimize the amount of energy spent to deliver data, nodes must either reserve the channel or rebroadcast carefully enough that collisions are unlikely. The former approach requires additional energy in the form of control messages, which introduce latency. The latter requires using suppression timers, which imposes latency on each hop.

In contrast, networks can route data very quickly. As each packet has a single destination, forwarding nodes can retransmit without worrying about suppression or local density. However, routing is along a path to a specific destination. To disseminate to every node using routing, the data source would have to name every other node in the network and send to each of them. Routing is faster than broadcasts, but reaching every node is less energy efficient.

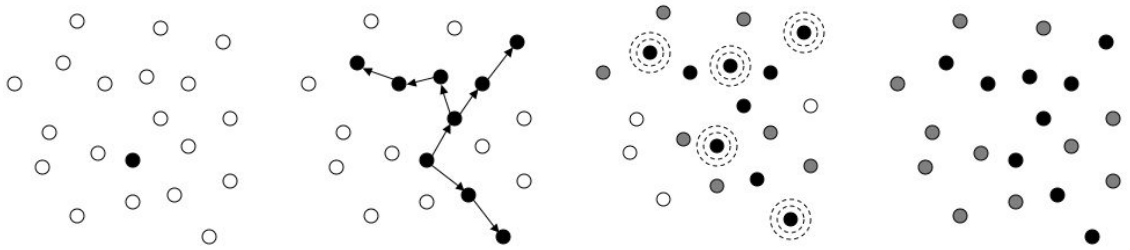
In this paper, we propose the Firecracker protocol. Firecracker is designed for small pieces of data that need to disseminate rapidly, such as tiny virtual programs or configuration constants. Using a combination of routing and broadcasts, Firecracker can achieve dissemination rates close to routing while maintaining the energy efficiency of broadcasts. When a node disseminates data using Firecracker, it routes the data to several distant nodes of the network. Once the data arrives, the destinations start a broadcast protocol. The data payload launches out into the network, then blossoms out at its target, hence the name: Firecracker.

Additionally, each node along the route can opportunistically cache data it forwards, and nodes along the route can overhear it. Instead of merely from a few points, the network can start disseminating data from each of the arcs the firecracker took. In these circumstances, the most efficient end-to-end route is not necessarily the best one. In sparse networks, the benefit of broadcasts over unicast in terms of data transferred per unit of transmit energy is not great: taking a long, circuitous route through the network can propagate data more quickly than a purely broadcast based dissemination. This raises questions of what sort of routing protocol is best suited to use in Firecracker dissemination.

In Section 2 we elucidate the relationship between routing and broadcasts in wireless sensor networks, considering energy, stability, and rate. In Section 3 we describe Firecracker, outlining an example implementation. In Section 4 we evaluate how well Firecracker performs in comparison to other approaches. We discuss related work in Section 5 and conclude in Section 6.

2. SENSOR NETWORKS

Sensor networks are embedded deployments of large numbers of small, wireless computing devices. The combination of embedment and application requirements necessitates lasting for long periods of time on very limited energy resources. Communication is usually the predominant energy cost; how often a node must be awake to communicate is a good estimate of energy consumption.



(a) Origin Node (b) Routing Data (c) Broadcasting (d) Completion

Figure 1: Example Firecracker protocol behavior. *First, the origin node routes the data to a set of distant nodes, with forwarding nodes opportunistically caching the data. These nodes are shown in black. These nodes then start forwarding the data with a broadcast dissemination protocol, reaching the remainder of the network, shown in gray.*

Disseminating a piece of data throughout a network is a common requirement for many applications. As sensor networks deployed for long periods in uncontrolled environments, changes to operations are inevitable, whether in response to changing needs or environmental events.

Broadcasting code too quickly can easily overload the network, causing the broadcast storm problem [10]. Current mote hardware can support approximately forty packets per second: each mote broadcasting once a second can become difficult for moderate densities. Suppression techniques can help in this regard, but loss in the network leads to unintended redundancy[7]. Without knowing how many other nodes may reply, nodes must broadcast carefully.

In contrast, networks can route data quickly. As there is only one retransmitter (the next hop), transmissions can be fast. However, routed data only traverses its end-to-end path, so routing data to every node is difficult. Not only does it require redundant traffic, wasting energy, but individually addressing each node in these lossy, transient networks may not be possible. Additionally, verifying transmission to each destination can be prohibitively expensive.

For example, current sensor network routing protocols running on the mica2 hardware platform [1], can route across approximately ten hops in a second, after considering retransmissions, media access, and contention. In contrast, the Trickle broadcast protocol [7] transmits at most once a second, to minimize packet losses due to collisions. Routing can transmit quickly. Broadcasts can scalably reach every node in a network. The remainder of this paper presents a way to combine these techniques, to achieve rapid, scalable, complete propagation.

3. FIRECRACKER

The purpose of the Firecracker protocol is to disseminate a piece of data to every node in a wireless sensor network. The protocol has two phases. When a node decides to disseminate data, it first routes the data to several other nodes in the network. Nodes along these routes store the data as if they had received it as well. Once a node receives the data, it uses a broadcast-based local dissemination protocol such as Trickle to spread the data. Figure 1 visually depicts the operation of the protocol.

To maximize energy efficiency, Firecracker seeks to minimize network contention and conflict. Broadcast based dissemination begins a few seconds after routing. This minimizes the seeding time and maximizes the number of nodes that receive broadcast messages. Because successful routing is rapid compared to broadcasting, Firecracker accomplishes this by putting a small timeout (e.g., three seconds) between receiving data through routing and starting the broadcast stage.

Firecracker composes into three parts: the broadcast protocol, the routing protocol and the selection of seed points. We visit the requirements for each in turn.

Broadcast Protocol: The basic mode of Firecracker is to run a broadcast-based dissemination protocol. The protocol must both propagate data to nodes that do not have the data as well as detect when to propagate. Temporary network disconnections should not prevent reception. The protocol should minimize the cost of detection but propagate rapidly.

Routing Protocol: The routing protocol must allow nodes to address arbitrary nodes in the network: traditional sensor network collection trees are insufficient. [13] Because the purpose of the routing phase is to spread data to distant points in the network, a naming scheme that allows nodes to choose such points is helpful. Nodes along the route should be able to snoop on routed traffic to cache the data as is passes by. Minimum hop paths are not as important as reliability and non-redundancy. Taking a long, winding path through different regions could quickly install the data on all the routing nodes and those that overhead the traffic.

Seed Selection: Selecting good seed nodes is critical to Firecracker's improving performance over a solely broad-

cast approach. The further the seed points from the origin, the faster data can propagate. In this situation, the distance is not physical, but logical: distance in the network by hops. These seeds should also be distant from one another, or traffic along their routes will be redundant; this distance requirement influences the number of seed nodes, as too many will lead to redundant traffic while too few may not cover the network well.

Routing data increases the propagation rate at the cost of increased transmissions. Using a broadcast approach allows data to reach a node through many possible paths, while routing defines the path to take. As routing algorithms generally minimize transmissions by communicating along low loss and symmetric paths, they cannot take advantage of auspicious receptions or long unidirectional links. However, as the network routes the data, nodes can snoop on traffic, taking advantage of these occurrences. As the base state of Firecracker is running an epidemic broadcast protocol, we analyze Firecracker's cost and benefit over a purely broadcast approach.

4. EVALUATION

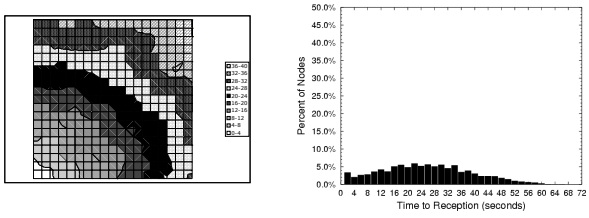
To evaluate Firecracker's cost/rate trade-offs, we incorporated it into the Maté virtual machine [6]. In the Maté programming model, a user writes high level scripts and compiles them to tiny (tens of bytes) VM bytecode programs. The user then sends the programs to a base station node, which starts propagating the code. As each Maté VM is customized for a specific application or deployment, complex programs within the application domain can be represented in a few packets.

4.1 Known Distant Destinations

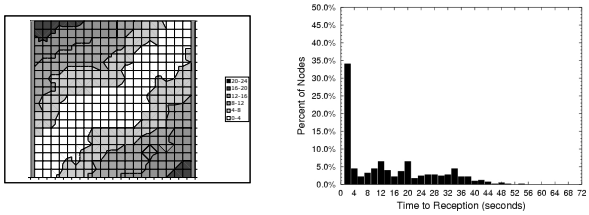
To evaluate how Firecracker performs with respect to a basic dissemination algorithm, we built a Maté VM that allows nodes to route their code to other nodes using a geographic grid-based routing protocol. Maté nodes periodically exchange summaries of the programs they have using the Trickle [7] algorithm.

Using TOSSIM, a TinyOS simulator, we organized nodes in a twenty by twenty grids, following the methodology used in our Trickle work. In these simulations, we sampled network loss rates using distributions generated from empirical data. Node spacing was ten feet: the grids were one hundred and ninety feet on a side. In this topology, the network is approximately sixteen hops across from corner to corner.

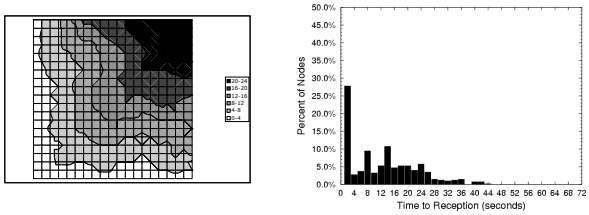
Just after the network had booted (but before Trickle started), we introduced a program at one corner that routed code to specified nodes in the network. Figure 2 shows the results. The program originated at the bottom left corner. Figure 2(a) shows propagation time using only broadcast-based dissemination. Figure 2(c) shows seed-



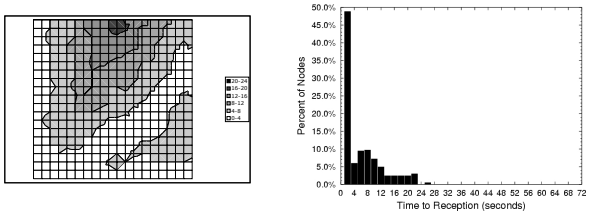
(a) Raw Trickle



(b) Route to Opposite Corner



(c) Route to Near Corners

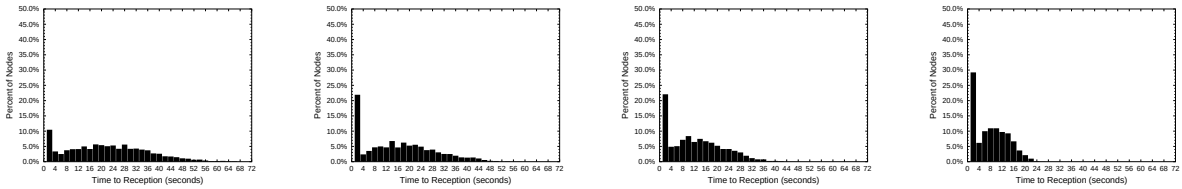


(d) Route to All Corners

Seeding	Mean (s)	Last (s)	Sends (packets)
Raw Trickle	25	60	20835
Opposite Corner	14	53	18275
Near Corners	12	43	19544
All Corners	5	25	6665

(e) Summary

Figure 2: Topographic Plots and Histograms of Propagation Time for Four Sample Executions. ***The plots on the left show time to data reception for nodes on the grid. The histograms on the right show the percentage of nodes that were programmed in a given two-second interval. The table shows average and worst-case time to reception in seconds as well as the number of packets transmitted.***



(a) One Seed (b) Three Seeds (c) Three from Center (d) Three Distant from Corner

Seeding	Mean	Last	Sends
One from Corner	22	55	21635
Three from Corner	16	51	19789
Three from Center	13	40	19251
Three Distant from Corner	8	24	6424

(e) Summary

Figure 3: Histograms of Propagation Time with Random Destination Selection. *Figures show one and three seeds from a corner, three seeds from the center, and three seeds from a corner that are assuredly distant. The table shows the average and worst-case time to reception in seconds as well as the number of packets transmitted across all experiments.*

ing at the two adjacent corners, Figure 2(b) shows to the opposite corner, and Figure 2(d) to all three corners. The network ran for twenty minutes.

Figure 2(e) shows summarized results for these experiments, including the mean time to reception for the nodes in the network, the time to the last reception, and the number of packets transmitted. Increasing the number and distances of seed points improves the reprogramming time. In addition, as routing to all corners shows, it can also reduce the number of network transmissions, for two reasons. First, routed data does not go through an advertisement-request metadata exchange. Second, increased synchronization improves Trickle's efficiency, up to a factor of two, and rapid routing of data effectively synchronizes nodes along the path (they reset timers when they receive the data).

4.2 Random Selection

Section 4.1 showed that by picking a small number of distant points in the network, a system can propagate data much more quickly and more efficiently than when only using a broadcast dissemination protocol. However, nodes may not be able to easily determine the most distant points. Being able to take advantage of Firecracker without requiring topology information increases its usefulness.

For example, instead of routing to a predefined set of nodes, the Firecracker root could route to a random set of nodes. In our experimental methodology, this means picking a random node address, and using grid routing to that point. One easily use logical or non-geographic routing to accomplish the same thing. This methodology assumes that the probability of picking a node as the end point is uniform over all the nodes. In networks with non-uniform density, this is advantageous: it's more likely to route to dense areas, increasing the number of nodes that receive the data quickly.

Figure 3 shows mote programming time histograms using Firecracker with random end point selection. As end points are random, each histogram is the aggregate result of twenty experiments. In the first three histograms (Figure 3(a) to Figure 3(c)), the destination address was one of the four hundred nodes in the network.

Figure 3(a) shows results for data originating from a corner being routed to a single end point. Figure 3(b) shows results for data originating from a corner being routed to three end points. Although better than raw Trickle (Figure 2(a)), and roughly similar to picking a subset of corners (Figures 2(b), 2(c)), three random end points is not nearly as efficient as picking all corners (Figure 2(d)). Seeding from the center of the network, instead of a corner (Figure 3(c)) does not improve performance significantly.

The issue that arises is that these random points may be close to the origin point, or only halfway across the network. The results in Figure 2 come from picking known *distant* points. For random point selection to work well, the random points must be distant. Being able to select such nodes without any knowledge of the actual network topology removes the cost of computing or transmitting that knowledge.

Given that nodes along the path receive the data, the easiest way to pick a distant seed point – without knowledge of the population of the coordinate space – is to pick

a point that is very, very far away, hopefully outside the coordinate space. In order to try to reach this point, the network will route the data to the edge, as far as it can go. In a routing protocol such as GPSR [5], for example, this would involve routing to a geographic location well outside the area of the network, while in GEM [9] it would involve routing to a very large depth value.

Figure 3(d) shows results for picking three random seed points outside the bounds of the network. Instead of constraining the destination address to the four hundred nodes in the network, the destination address was in the range $[0, 2048]$. These results are close to those obtained by picking the three corners: the time to the last reception is twenty-four seconds (instead of twenty-five), and approximately the same number of packets (sixty-five hundred) are sent. The mean reception time is not quite as close, it being eight seconds compared to the known corners' five, but it represents almost a three fold speedup over raw Trickle, with a third of the transmission cost.

5. RELATED WORK

Data propagation in wireless sensor networks is currently an area of significant research, motivated by binary code dissemination. The binary code dissemination problem has two characteristics that distinguish it from Firecracker's. First, the amount of data being transferred is very large: bandwidth, rather than per-hop latency, dominates rate. Second, the data is much larger than what can be stored in RAM. As access to non-volatile storage is expensive in terms of energy, this often leads to hierarchical [2] or windowing [12] approaches to minimize the amount of caching necessary on a receiver.

Overcast uses an underlying, single route network to distribute multicast data in Internet-class networks [4]. Overcast deals with constructed efficient distribution trees to selective end points, while Firecracker builds on top of an existing routing protocol. The distinct domains (Internet vs. sensor net) also lead to distinct issues in naming.

Many sensor network applications use basic collection trees [8]. However, as deployments grow in complexity, needing in-network storage [11] or novel addressing [3], more complex routing protocols will be needed. Like these approaches, Firecracker depends on any-to-any routing within the network. The prevailing approach so far has been to define logical coordinate spaces over a network topology [9]. For the purposes of Firecracker, a key requirement for these coordinate spaces is names indicating some notion of network distance.

6. CONCLUSION

By incorporating a routing phase into broadcast based dissemination, Firecracker can speed propagation significantly at improved transmission efficiency. In our examples, we used a simple ray-based approach; one could imagine larger networks using tree-based or fractal patterns, with branches being timed to minimize contention. Effective routing requires nodes to be able to easily address distant points in a network. Given dissemination's importance as a basic networking primitive in sensor networks, this suggests an interesting requirement for any-to-any routing protocols.

7. REFERENCES

[1] J. Hill and D. E. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, nov/dec 2002.

[2] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems*, 2004.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the International Conference on Mobile Computing and Networking*, Aug. 2000.

[4] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the 4th Operating Systems Design and Implementation (OSDI 2000)*, pages 197–212.

[5] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, Boston, MA, USA, 2000.

[6] P. Levis and D. Culler. Maté: a tiny virtual machine for sensor networks. In *Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*, Oct. 2002.

[7] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks. In *First USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI 2004)*.

[8] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, Sept. 2002.

[9] J. Newsome and D. Song. Gem: graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 76–88. ACM Press, 2003.

[10] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162. ACM Press, 1999.

[11] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *Proceedings of the first ACM international workshop on Wireless sensor networks and applications*, pages 78–87. ACM Press, 2002.

[12] T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. Technical Report CENS Technical Report 30, 2003.

[13] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 14–27. ACM Press, 2003.