Journal of Computing and Information Technology - CIT 21, 2013, 2, 97–107
doi:10.2498/cit.1002195

The Impact of Packet Fragmentation and Reassembly in Resource Constrained Wireless Networks

James Pope and Robert Simon

Department of Computer Science, George Mason University, Fairfax, Virginia, United States

Low-power and lossy (LLN) networks, including Wireless Sensor Networks, provide an important building block for Internet of Things (IoT) technology. The resource constraints associated with LLN devices necessitate the redesign of a number of basic Internet protocols. This paper evaluates the effect of packet fragmentation on the performance of protocols redesigned for LLN systems. We focus on the important class of tree-based LLNs that exhibit both one-to-many and many-to-one communication patterns. In particular, we measure the impact of fragmentation on these communication protocols. Our results demonstrate that excessive packet fragmentation has a tremendously negative impact on communication within a tree-based LLN system. This work provides a guideline for IoT engineers in techniques for avoiding these damaging effects.

Keywords: Internet of Things communication, Internet protocols over embedded systems, low-power and lossy networks

1. Introduction

The Internet of Things (IoT) refers to a collection of technologies designed to interconnect physical devices with the Internet [1]. A system or application is considered part of the IoT if it can run the IP protocol stack. One particularly important type of IoT system is characterized by topologies populated by resource-constrained nodes. This system class, including wireless sensor networks, are referred to as *Low Power and Lossy* (LLN) networks [2, 3, 4]. We consider an LLN to be *tree-based* if it has a base-station or gateway that coordinates the activities of the devices in the network [5]. Tree-based LLNs have two types of communication patterns. The first is one-to-many, where the base station sends control or management packets to each of the devices in the system. This mode requires system-wide broadcasting [6]. The second is many-to-one, where each device transmits its packets up the tree to the gateway. This mode is sometimes called *convergecasting* [5]. Transforming tree-based LLNs to work within the IoT requires the redesign and reimplementation of a number of basic Internet protocols. Our paper examines the performance impact of these redesigned protocols on tree-based LLN systems.

In particular, we focus on the effect that packet fragmentation has on protocols redesigned for LLN applications. Packet fragmentation occurs when a device tries to transmit a packet that is larger than the maximum transmission unit size, or MTU, allowed by the link layer. When this occurs, the device can either drop the packet or fragment the packet into several smaller packets so that each will be no larger than the MTU. Using standard IP protocols, if the packet is fragmented and not dropped, then the original packet can be reassembled when all of the packets are received. The advantage of fragmentation is that it offers a way around the limitations imposed by a small MTU. There are, however, several potential disadvantages. Fragmentation and reassembly protocols incur additional complexity within the system. Further, since multiple packets now must be transmitted for each original packet, the likelihood of packet loss increases. The impact of packet loss may be particularly severe for wireless IoT systems, where resources are heavily constrained[7].

Our goal is to precisely quantify the differences in metrics such as Packet Delivery Ratio (PDR) and network overhead between situations where fragmentation occurs and where it does not occur. We study this impact by deploying several types of applications in Contiki, a popular and widely used open source operating system for LLNs and IoT applications [8]. This testbed enabled us to understand the dynamics and impact of packet fragmentation on both network management and application level performance.

For the one-to-many application, we designed and implemented *Radiate*, a semi-reliable broadcast protocol. Radiate has several different operating modalities, trading off reliability with message complexity and packet overhead. For the many-to-one application we modeled a basic, non-aggregating data collection architecture, where each node periodically transmits a value up the tree to the base station. We varied packet rates at both the base station and interior nodes to study how fragmentation impacted the PDR performance.

The major contribution of this paper is to show that, using a general purpose LLN operating system, packet fragmentation causes a dramatic decrease in packet delivery rates for both the one-to-many and the many-to-one communication scenarios. This decrease in delivery rates occurs well before the network starts to get congested or reaches its transmission capacity. For the one-to-many broadcast communication pattern, our results show the need, depending on the required message update rate, to change the reliability mechanism. For the many-to-one data gathering communication pattern, our results show that either different network fragmentation and reassembly support is required or the application must alltogether forbid fragmentation.

The paper proceeds as follows: Section 2 presents background and related work. Section 3 describes Radiate, our multi-modal semi-reliable broadcast protocol. Section 4 discusses our evaluation methodology and presents the results of our experiments. Finally, Section 5 offers some observations and conclusions.

2. Background and Related Work

We first present some background on our targeted network architecture, and then discuss related work.

2.1. Tree-based Resource Constrained Networks

Our work centers on tree-based wireless networks using low power communication architectures such as those supported by the IEEE 802.15.4 specification[9]. This technology is designed to support applications such as the Smart Grid, data center power control, industrial networks or building and home automation [10, 11, 12, 13]. These LLN systems are populated by resource constrained devices, have unreliable communication links and low data rates, and, as explained in the Introduction, are referred to as low-power lossy networks (LLNs)[14].

Interest in tree-based LLNs is partially motivated by the architecture of the Internet-of-Things. IoT systems must support two-way and end-to-end IP enabled communication. Traditional IP protocols failed to address the operating characteristics within a LLN environment. This forced a redesign of several basic protocols [3, 15]. For instance, the 6LoWPAN protocol was designed to allow the use of the IP layer from the Internet protocol stack to be run directly on LLN devices [16].

The 6LoWPAN protocol is designed to be independent of the data link layer, but is often assumed to be running over an 802.15.4 link. In order to effectively run in this environment, the protocol allows a number of options for header compression. It also supports packet fragmentation and reassembly. Packet fragmentation is the process of breaking packets into smaller fragments, and then sending each fragment as a separate packet. Fragmentation occurs because different networks connected through the Internet can have different maximum sizes or MTUs. For instance, the maximum size of the 802.11 WiFi data payload is 2312 bytes, but the maximum size of the data payload carried by many variants of the 802.15.4 standard is 104 bytes. Packets are fragmented when they arrive on a link that has a smaller maximum size than the

packet itself. In IP, each fragment is retransmitted in a separate packet, and the entire packet is reassembled at the end-host, once all of the fragments are received.

The 6LoWPAN protocol is an adaptation layer typically sitting between the 802.15.4 link layer and the network layer performing fragmentation and reassembly [16]. One major difference between fragmentation in IPv4 and IPv6 versus fragmentation in 6LoWPAN is that the latter does *not* copy IP header information into each packet. This means that each device may only be able to reassemble fragments originating from one original packet at a time. Fragments from other packets would need to be dropped.

Supporting packet fragmentation in a resource constrained network is a difficult problem. In addition to increased physical layer impairments, higher layers encounter problems handling fragmentation/reassembly. Considering the many-to-one mode, only the sink would need to reassemble the packets. Since all messages are directed towards the sink, simultaneous packet reassembly from multiple senders is necessary. Increasing buffer size or adding additional buffers may not be viable solutions for resource constrained networks. Careful buffer management is necessary for devices in these networks to support this paradigm. For instance, Contiki's 6LoWPAN implementation keeps one reassembly buffer.

The one-to-many mode is effected by fragmentation/reassembly in a different way. Since the message is intended for all devices in the network, each device needs to reassemble the packet. Once reassembled, the device can then retransmit the message to neighbors, eventually flooding the message to all devices. In resource constrained networks, this is very expensive, particularly from an energy perspective. Contiki's Rime Network Flood [17] protocol (one-to-many) does not provide fragmentation/reassembly capability. The Contiki 6LoWPAN implementation provides a local neighbor broadcast fragmentation/reassembly capability, but does not provide the one-to-many capability.

2.2. Performance Impact of Packet Loss in Tree-based Networks

The performance impact, along with mitigation approaches to the problem of packet loss in LLN wireless networks, has been extensively studied. This work has largely focused on the general impact of lossy links and low transmission speeds on application level performance, along with the techniques for eliminating some of these effects [18, 19, 20, 21, 22, 23]. Much of this centers on current hardware designs or generic LLN protocols and algorithms and does not address our concern about redesigned IP-friendly protocols operating in an LLN.

We focus on the two basic communication patterns in tree-based LLNs, one-to-many and many-to-one [5, 6]. There have been a very large number of research papers devoted to one-to-many communication in the context of reliable or semi-reliable broadcasting. Broadcasting is an important network activity, since it is used by base stations to deliver commands or network management information to the devices in the system. Approaches to broadcasting in LLNs include methods for fully reliable or probabilistically reliable techniques [24, 25]. These techniques generally aim to improve the delivery rate of broadcasting to all nodes in the system without requiring excessive flooding. Although these techniques are often quite powerful, they require varying levels of protocol complexity and message overhead.

To study the impact of fragmentation on one-to-many broadcasting, we designed and implemented a semi-reliable protocol called Radiate. The principles underlying this protocol are heavily influenced by earlier work in lightweight broadcast and data dissemination, including protocols such as Rime [17] and Trickle [26]. In particular, we use techniques such as eavesdropping and selective retransmission to improve reliability and reduce message overhead while minimizing complexity. As will be seen in Section 3, Radiate is designed to tradeoff reliability with overhead, and can be used effectively in different modalities, depending on the rate of packet broadcast.

This paper extends the results presented in [27]. That earlier work focused primarily on the performance impact of many-to-one communica-

tion and did not address one-to-many rebroadcast paradigms.

3. Reliable Broadcasting with *Radiate*

Broadcasting is the basic communication operation to support tree-based one-to-many communication. To improve reliability and decrease the overhead associated with a purely flooding strategy, we designed the Radiate protocol. Radiate combines elements from IPv6 with carefully tuned retransmission timers. In particular, Radiate uses the multicast capability of the IPv6 protocol. The sender broadcasts messages to the link-local, all nodes multicast address, which is ff02::1. Radiate adds a small data structure to these messages. This structure contains the sequence-number, origin, source, and length of the payload.

When a device has data to send, it increments the sequence-number and the data is copied into the payload. The device transmits the message and then sets a timer to send the message again at some random time in the next second. Randomization is a common technique designed to reduce the chance of collisions. When the timer expires, the message is again randomly broadcast, doubling the timer interval (e.g. 1, 2, etc.) until a maximum of 4 seconds is reached. Typically, a total of 4 broadcasts are performed for each send execution.

Normally, this broadcasting would be excessive, so Radiate mitigates this by increasing the timer if a device overhears the same sequence-number that it is currently broadcasting, effectively cancelling the current timer. Thus, if a device broadcasts a message and three neighbors successfully receive and then also broadcast, the device will have increased beyond the maximum and no longer rebroadcasts. The heuristic causes devices in dense topologies to likely only broadcast the message once. The heuristic causes devices in sparser topologies, including those at the edge of the network, to rebroadcast closer to the maximum number of times.

Devices receiving the broadcast check the sequence-number to ensure whether it is more recent and, if so, start broadcasting in a similar fashion. Eventually, all devices will receive the broadcast with the latest sequence number and the re-broadcasting stops.

Figure 1 illustrates a four node topology and time line using the Radiate protocol. Using node *B* as an example, it receives a message with sequence-number 1 from *A*, so *B* schedules a broadcast of the same message to be transmitted (note that this broadcast cannot be cancelled). After the broadcast is sent, node *B* overhears *C* broadcast the same message, so *B* cancels its next scheduled broadcast. Node *B* then overhears node *A* broadcast the same message and again cancels the next scheduled broadcast. Finally, node *B* overhears the same message from node *C* and node *B* stops scheduling rebroadcasts, ignoring any subsequent messages with the same sequence-number. Note that each node transmits at least once and no more than four times.

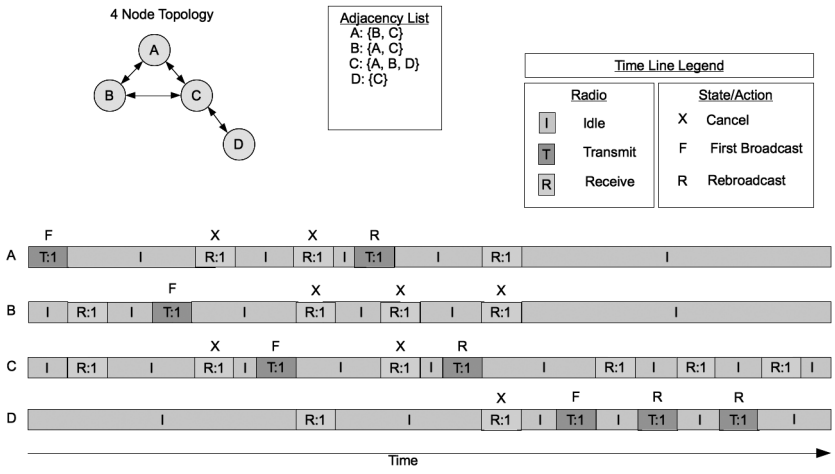

Figure 1. Radiate example showing selective cancelation.

3.1. Radiate Protocol

Following the rebroadcasting process described above, the Radiate algorithm is next detailed using the notation introduced in Table 1.

m	Radiate Message
m_{seq}	Sequence number of Radiate Message
m_{src}	Neighbor source id of Radiate Message
n	Node in network
$n.m$	Current message for Node n (i.e. last m set)
n_{id}	Identifier of Node
$wait$	Current wait
MAX_WAIT	Maximum wait, default 4000 ms
$REBROAD\_INT$	Time interval first rebroadcast, default 1000 ms
$WAIT\_START$	Initial time to start waiting, default 1000 ms

Table 1. Radiate Protocol Notation.

Algorithm 1 Radiate Originator Send

1: //Input: Message m from application
2: set $n.m = m$
3: increase $n.m_{seq}$
4: set $wait = WAIT\_START$
5: call radiateBroadcast

Algorithm 2 Radiate Receive

1: //Input: Message m from neighbor
2: **if** $n.m_{seq} < m_{seq}$ **then**
3: set $n.m = m$
4: notify application new message received
5: set $wait = WAIT\_START$
6: randomWait=rand(REBROAD_INT)
7: scheduleBroadcast(randomWait)
8: **else if** $n.seq == m.seq$ **then**
9: increaseWait($wait$)
10: **else**
11: // message is ignored
12: **end if**

Algorithm 3 Scheduled Timer Expired

1: **if** $wait \leq MAX\_WAIT$ **then**
2: call radiateBroadcast
3: **else**
4: // rebroadcasting stops
5: **end if**

Algorithm 4 Radiate Re/Broadcast

1: set $n.m_{src} = n_{id}$
2: transmitMessage($n.m$) to neighbors
3: randomWait=rand($wait$)
4: scheduleTimer(randomWait)
5: increaseWait($wait$)

In algorithm 1, the sink initiates the flooding by incrementing the sequence-number and broadcasting to its neighbors. The radiate message origin and sequence-number fields remain constant along each hop with the source changing to the node broadcasting. Algorithm 2 is invoked when a radiate message is received by a node from a neighbor.

In algorithm 2, line 2, the receiving node compares the sequence-number number of the message to its own to determine if the message is newer. If not, the message is simply ignored. If newer, the message is set on the node, the wait timer is reset/initialized to a starting value, and rebroadcast timer is scheduled to expire in a random time (uniform distribution) between 0 and $REBROAD\_INT$. The randomness avoids collisions that might result from neighboring nodes rebroadcasting at the same time. The $REBROAD\_INT$ defines an upper bound on how long it takes the flooding to move through the network. In our case, $REBROAD\_INT = 1000ms$ so assuming the network diameter to be 12 hops, the maximum amount of time for the radiate message to traverse the network is 12 seconds.

Algorithm 3 is executed whenever a previously scheduled timer has expired. It simply checks to determine if the maximum wait time has been exceeded. If not, then a broadcast is sent. If it has been exceeded, the timer scheduling ceases and no more broadcasting occurs until another newer message is received.

Algorithm 4 ensures that the $n.m_{seq}$ is set to the node's id and transmits the message to neighbors. It then schedules the timer to potentially send again at the current *wait* time and increases the *wait* time.

The increaseWait function doubles the passed value each time it is invoked.

Line 9 in algorithm 2 increases the *wait* value if the message sequence-number overheard is the

same as the node's current message sequence-number. This increases the *wait* without transmitting a message, therefore, effectively cancelling a pending rebroadcast. This is called the *cancelling* heuristic. Nodes always rebroadcast at least once the first time. Subsequently, while waiting to rebroadcast, nodes may overhear the same message enough times to exceed the *MAX_WAIT*. In this case, when the timer expires no additional transmission is sent because of the check in algorithm 3. If no messages are overheard, then the timer expires calling algorithm 4 which transmits the message and increases the timer. In all cases, eventually the maximum wait time is exceeded and the rebroadcasting ceases.

4. Evaluation

The goal of our evaluation was to conduct performance studies using a realistic set of application level programs. We constructed a simulation environment using the ContikiOS Cooja Simulator [28]. For devices we used the Zolertia Z1 series built using a MSP430 processor with 8kB of RAM and using the CC2420 radio. The system we programmed used a simple data reporting application for the many-to-one-mode, and the Radiate protocol carrying periodic control messages for the one-to-many mode. The evaluation first examines the impact of rebroadcasting for the many-to-one mode using Radiate. The evaluation then compares the impact that fragmentation has for the many-to-one and one-to-many modes.

The configured network stack for the many-to-one modality was as follows. The mesh networking protocol was not necessary for one-to-many, however, for realism, it was allowed to run during the Radiate experiments.

- IEEE 802.15.4
- Contiki's 6LoWPAN
- Contiki's mesh networking protocol
- IPv6
- UDP

The Unit Disk Graph Medium Distance Loss model with a 50-meter transmission range and a 100 meter interference range was used in the simulations. We created three different network sizes, 16 devices, 36 devices and 64 devices. The 64 device topology is shown in Figure 2. The topology is based loosely on a grid structure, with an average of 30 meters between devices. Within this structure each device was randomly placed within a proportionally smaller square area. This introduces some variability while still maintaining a connected network. The sink was placed in the middle of the network.

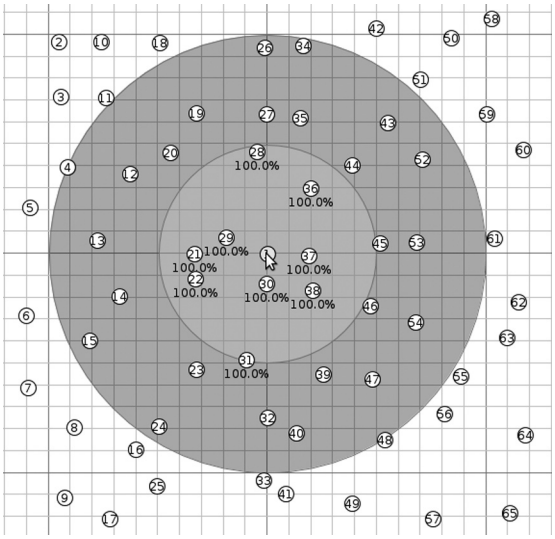

Figure 2. 64 device topology with the gateway in the center.

For each network size, a set of experiments was performed to determine the PDR for application messages using varying rates for both many-to-one (unicast) and one-to-many (broadcast) communication modes. The PDR for each device was calculated by taking the number of application messages received, divided by the number of application messages sent. The average was then taken over the devices to determine the PDR for the scenario.

Note that during this time normal network routing and management traffic continued to be transmitted. Each rate is run for three minutes at which point the rate in increased by five and again run for three minutes. Therefore, a rate of 35 runs for a total of 21 minutes.

Fragmentation is induced by setting the payload threshold to 75 bytes instead of 100 with approximately 25 bytes reserved for lower layer headers. The unicast and broadcast messages

are 85 bytes. The fragmentation scenario results in two fragments/message for the sample messages versus only requiring one for the non-fragmentation scenario.

4.1. Radiate Evaluation

The Radiate protocol was used to compare the impact that one-to-many rebroadcast mechanisms have on PDR. The rates were measured up to 60 messages/minute. In addition to PDR, overhead results were determined. The overhead for each node was determined by counting the number of times each node broadcasts (or rebroadcasts) a Radiate message. The average of the nodes was then taken for each rate.

The Radiate broadcast protocol is modified with the following variants, according to the number of repeated broadcasts.

- radiate_one
- radiate_can
- radiate_all

The radiate_one simply rebroadcasts the received newer message once. The radiate_can is the *cancelling* heuristic described previously and is expected to rebroadcast between one and four messages. The radiate_all does not attempt to minimize the overhead and always rebroadcasts four messages.

Rebroadcasts should result in a higher PDR. However, as the rate increases, the rebroadcasts' additional congestion may diminish the PDR improvements.

Figures 3 and 4 show the PDR and overhead for 16 nodes. The radiate_all PDR does well from 10-20 rate, but thereafter does worse than both other protocols. The radiate_one drops to 90% PDR at rate 15, however, performs well with regard to the other protocols. The overhead between the protocols is somewhat expected with radiate_all incurring the most, radiate_can in between, and radiate_one with the least overhead. At rate 30 radiate_all has approximately 44% more overhead than radiate_can. At rate 60 radiate_can has approximately 39% more overhead than radiate_one.

Figures 5 and 6 show the PDR and overhead for 36 nodes. The radiate_one initially does

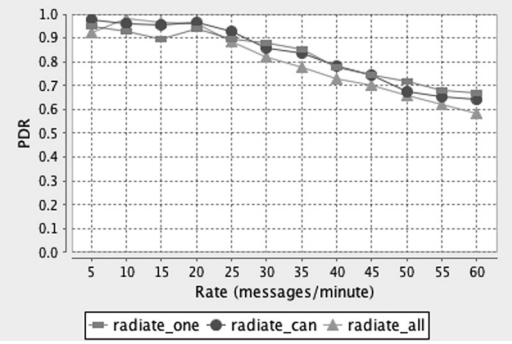

Figure 3. PDR Radiate 16 Devices.

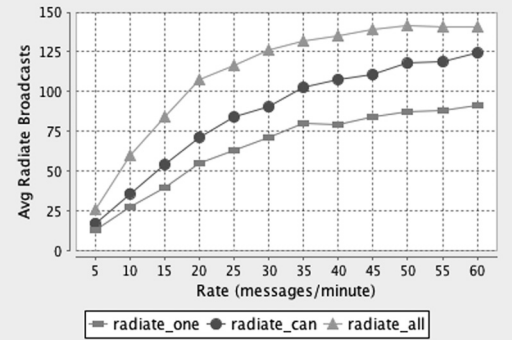

Figure 4. Overhead Radiate 16 Devices.

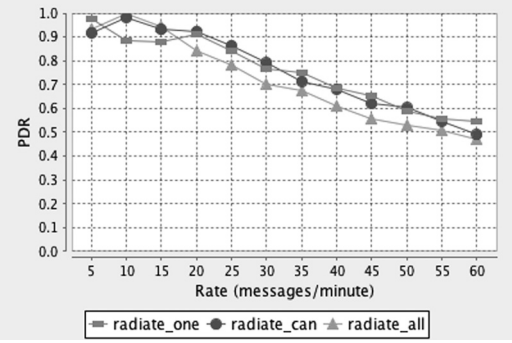

Figure 5. PDR Radiate 36 Devices.

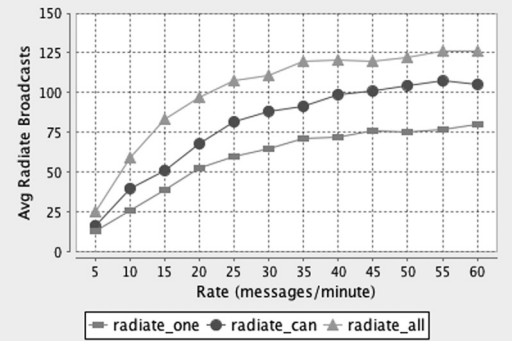

Figure 6. Overhead Radiate 36 Devices.

worse from the 10-20 rate, but is approximately 5% better at rate 60 than either protocol. The radiate_all again does well early, but degrades with almost 10% worse PDR at rate 30. The overhead is 60% more for radiate_all versus radiate_can at rate 15. The radiate_can protocol has approximately 38% more overhead than radiate_one at rate 40.

Figures 7 and 8 show the PDR and overhead for 64 nodes. The PDR and overhead results are similar to the 16 and 36 network size results.

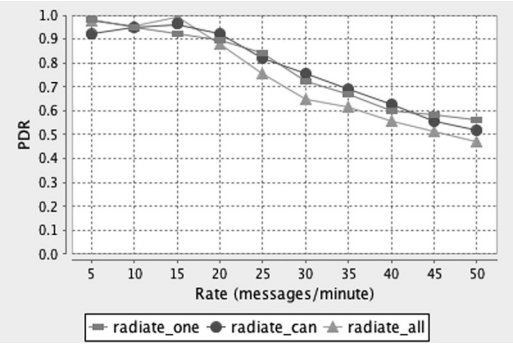

Figure 7. PDR Radiate 64 Devices.

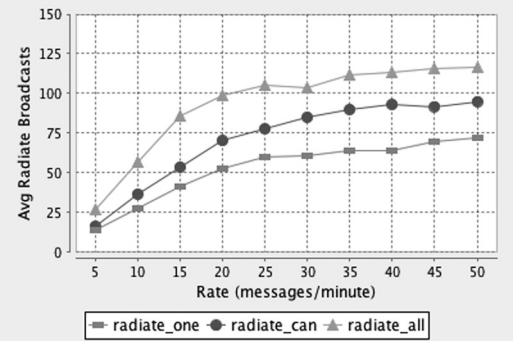

Figure 8. Overhead Radiate 64 Devices.

The PDR results suggest that radiate_one does poorer for lower rates 10-20, but generally does well thereafter. The radiate_all variant initially performs well for rates 10-15, but drops off abruptly at rate 20. Interestingly, radiate_can appears to first correlate with radiate_all and then switches, roughly between rates 15-25, to correlating more with radiate_one. At rate 60 it appears that the variants somewhat converge to the same PDR. The results suggest that for lower rates rebroadcasts do improve PDR. However, for higher rates, the additional rebroadcasts actually become counterproductive, producing worse PDR results.

The relative overhead results between the protocols is as expected. There appears to be a diminishing of the overhead as the rate increases. This is believed to be in conjunction with the reduced PDR; dropped packets will result in fewer rebroadcasts/overhead.

4.2. Fragmentation Evaluation

The many-to-one unicast scenario results are depicted in Figures 9, 10, and 11. The fragmentation scenario is labeled as fragunicast and fragbroadcast; the non-fragmentation scenario is labeled nofragunicast and nofragbroadcast. All three graphs clearly show that fragmentation severely degrades the PDR for the different network sizes. As the rate increases, the PDR for both scenarios appear to be effected proportionally. Figure 9 shows an increasing difference between fragmentation and non-fragmentation from approximately 18% for a rate of 5 messages/minute to over 30% difference for rate 35 messages/minute.

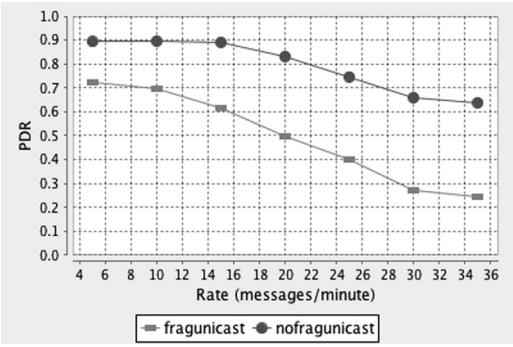

Figure 9. PDR Unicast 16 Devices.

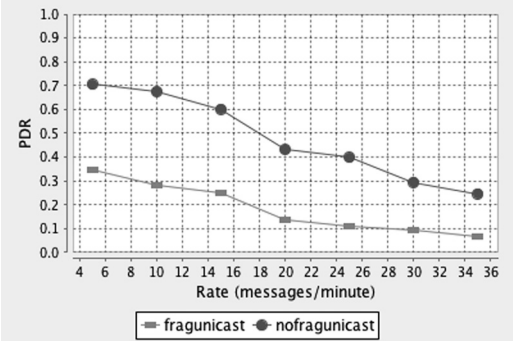

Figure 10. PDR Unicast 36 Devices.

The non-fragmentation scenario appears to be less affected by the rate increases.

Figures 10 and 11 both show the stress both scenarios experience as the rate increases in larger networks. Figure 10 indicates that the non-fragmentation scenario consistently delivers 20% more messages for all rates. Figure 11 shows that eventually both scenarios essentially fail to deliver any many-to-one messages to the sink. We believe that this is due to feeder routes near the sink becoming expectedly congested.

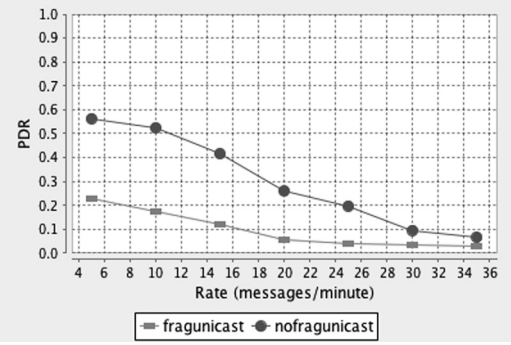

Figure 11. PDR Unicast 64 Devices.

The unicast results clearly show the tremendously negative impact of fragmentation on packet delivery rates. This strongly suggests that network engineers should modify their protocols to either avoid excessive fragmentation or provide additional reliability mechanisms.

The broadcast results, using radiate_one, are shown in Figures 12, 13, and 14. Surprisingly, it appears that fragmentation has less of an effect on the PDR.

Figure 12 shows that both scenarios successfully deliver 90% or more of the messages for all rates except the last which still achieves approximately 86% at 35 messages/minute. For larger networks and higher rates, Radiate still delivers good performance. For rates up to 25 messages/minute, the PDR is at or greater than 85%. However, Figures 13, and 14, show that fragmentation begins to denigrate the PDR towards 70% for the rates of 30 messages/minute.

We investigated the log files for both fragmentation and non-fragmentation scenarios and determined that, indeed, fragmentation was occurring when expected and messages were being dropped due to reassembly errors. We believe

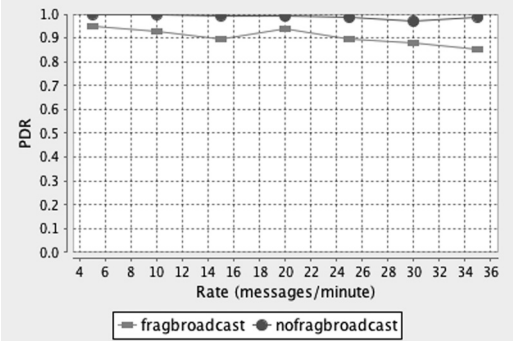

Figure 12. PDR Broadcast 16 Devices.

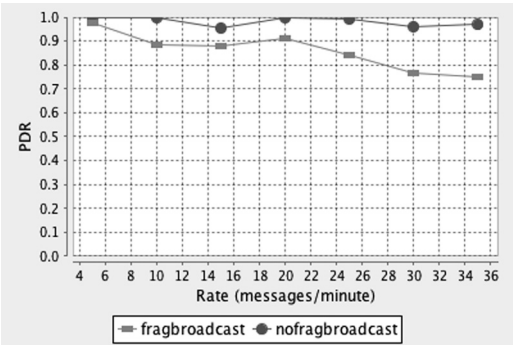

Figure 13. PDR Broadcast 36 Devices.

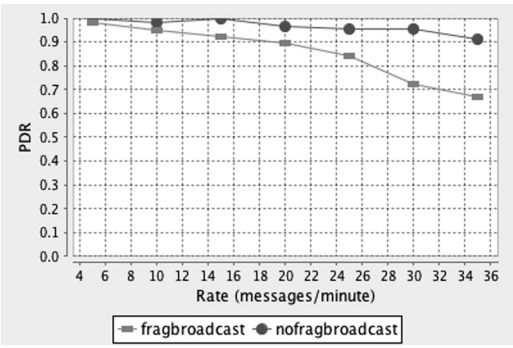

Figure 14. PDR Broadcast 64 Devices.

the difference between the unicast results and the broadcasts results are caused by two reasons:

1. Broadcast messages reassembled each hop
2. Neighbors rebroadcast packets allowing correct reception of previously missed fragments

Because the messages are reassembled by the neighboring devices, for each neighbor the full message can then be transmitted again. Even if a message is not received due to a fragmentation reassembly error, the device very likely

has several more opportunities to receive the message.

Finally, we note that the rate appears to have a much greater effect on PDR than the network size. Network size and topology certainly play a role, however, as Figures 13 and 14 show, there is little difference between their PDR for the varying rates.

5. Conclusion

This paper examined the impact of packet fragmentation at the data link level on the end-to-end performance of IP-based protocols designed to support IoT systems. Using a mixture of applications and network layer routing functions, our results show that fragmentation can seriously degrade the performance of the typical IoT device to gateway communication modality. The results show that better fragmentation/reassembly support is required for IoT systems or, alternatively, implementations simply forbid fragmentation in the first place. For one-to-many communication, we presented the Radiate broadcast protocol. Our results showed that with proper design of data broadcast mechanisms, gateway-to-device communication can maintain relatively high performance levels.

Acknowledgments

This work is supported by NSF under grants CNS-1116122 and CNS-1205453.

References

[1] L. ATZORI, A. IERA, G. MORABITO, The internet of things: A survey. *Computer Networks*, **54**(15) (2010), 2787–2805.

[2] S. DUQUENNOY, F. ÖSTERLIND, A. DUNKELS, Lossy Links, Low Power, High Throughput. In *Proceedings of the ACM Conference on Networked Embedded Sensor Systems, ACM SenSys 2011*, (2011), Seattle, WA, USA, Nov. [Online]. Available: `http://dunkels.com/adam/duquennoy-11lossy.pdf`

[3] P. LEVIS, A. TAVAKOLI, S. DAWSON-HAGGERTY, *Overview of existing routing protocols for low power and lossy networks*, IETF Draft Report, (2009). [Online]. Available: `http://tools.ietf.org/html/draft-ietf-roll-protocols-survey-07`

[4] J. VASSEUR, M. KIM, K. PISTER, *Routing Metrics Used for Path Calculation in Low Power and Lossy Networks*, IETF Draft Report, (2011).

[5] Y. ZHANG, S. GANDHAM, Q. HUANG, Distributed minimal time convergecast scheduling for small or sparse data sources. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, (2007), pp. 301–310.

[6] L. MOTTOLA, G. P. PICCO, Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Comput. Surv.*, **43**(3) (Apr. 2011), 19:1–19:51. [Online]. Available: `http://doi.acm.org/10.1145/1922649.1922656`

[7] Y.-T. PARK, P. STHAPIT, D.-H. LEE, Y.-S. CHOI, J.-Y. PYUN, Data fragmentation scheme with block ack in wireless sensor networks. In *Multimedia and Ubiquitous Engineering (MUE), 2011 5th FTRA International Conference on*, (2011), pp. 79–83.

[8] J. KO, J. ERIKSSON, N. TSIFTES, S. DAWSON-HAGGERTY, A. TERZIS, A. DUNKELS, D. CULLER, ContikiRPL and TinyRPL: Happy Together. In *Proceedings of the Workshop on Extending the Internet to Low Power and Lossy Networks (IP+SN 2011)*, (Apr. 2011), Chicago, IL, USA. [Online]. Available: `http://dunkels.com/adam/ko11contikirpl.pdf`

[9] *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE Computer Society Std. 802.15.4, 2006.

[10] D.-M. HAN, J.-H. LIM, Smart home energy management system using ieee 802.15.4 and zigbee. *Consumer Electronics, IEEE Transactions on*, **56**(3) (2010), 1403–1410.

[11] S. DEPURU, L. WANG, V. DEVABHAKTUNI, N. GUDI, Smart meters for power grid – challenges, issues, advantages and status. In *Power Systems Conference and Exposition (PSCE), 2011 IEEE/PES*, (2011), pp. 1–7.

[12] V. GUNGOR, B. LU, G. HANCKE, Opportunities and challenges of wireless sensor networks in smart grid. *Industrial Electronics, IEEE Transactions on*, **57**(10) (2010), 3557–3564.

[13] A. WILLIG, Recent and emerging topics in wireless industrial communications: A selection. *Industrial Informatics, IEEE Transactions on*, **4**(2) (2008), 102–124.

[14] T. WATTEYNE, A. MOLINARO, M. RICHICHI, M. DOHLER, *From MANET To IETF ROLL Standardization: A Paradigm Shift in WSN Routing Protocols*. Ieee Communications Surveys and Tutorials, 2010.

[15] J. KO, A. TERZIS, S. DAWSON-HAGGERTY, D. CULLER, J. HUI, P. LEVIS, Connecting low-power and lossy networks to the internet. *Communications Magazine, IEEE*, **49**(4) (2011), 96–101.

[16] J. HUI, D. CULLER, Ipv6 in low-power wireless networks. *Proceedings of the IEEE*, **98**(11) (2010), 1865–1878.

[17] A. DUNKELS, F. OSTERLIND, Z. HE, An Adaptive Communication Architecture for Wireless Sensor Networks. In *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys 2007)*, (November 2007).

[18] D. S. J. DE COUTO, D. AGUAYO, J. BICKET, R. MORRIS, A high-throughput path metric for multi-hop wireless routing. *Wirel. Netw.*, **11**(4) (Jul. 2005), 419–434. [Online]. Available: http://dx.doi.org/10.1007/s11276-005-1766-z

[19] H. ZHANG, A. ARORA, Y.-R. CHOI, M. G. GOUDA, Reliable bursty convergecast in wireless sensor networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad-hoc Networking and Computing*, ser. MobiHoc '05. (2005), New York, NY, USA: ACM, pp. 266–276. [Online]. Available: http://doi.acm.org/10.1145/1062689.1062724

[20] M. Z. N. ZAMALLOA, B. KRISHNAMACHARI, An analysis of unreliability and asymmetry in low-power wireless links. *ACM Trans. Sen. Netw.*, **3**(2) (Jun. 2007). [Online]. Available: http://doi.acm.org/10.1145/1240226.1240227

[21] C.-Y. WAN, S. B. EISENMAN, A. T. CAMPBELL, J. CROWCROFT, Overload traffic management for sensor networks. *ACM Trans. Sen. Netw.*, **3**(4) (Oct. 2007). [Online]. Available: http://doi.acm.org/10.1145/1281492.1281493

[22] B. LATRE, P. D. MIL, I. MOERMAN, N. V. DIERDONCK, B. DHOEDT, P. DEMEESTER, Maximum throughput and minimum delay in IEEE 802.15.4. *Proceedings of the International Conference on Mobile Ad-hoc and Sensor Network (MSN 05)*, 2005.

[23] J. KIM, X. LIN, N. SHROFF, P. SINHA, Minimizing delay and maximizing lifetime for wireless sensor networks with anycast. *IEEE/ACM Transactions on Networking*, **18**(2) (April 2010), 515–528.

[24] N. RAHNAVARD, F. FEKRI, Crbcast: a collaborative rateless scheme for reliable and energy-efficient broadcasting in wireless sensor networks. In *Proceedings of the 5th international Conference on Information Processing in Sensor Networks*, ser. IPSN '06. (2006), New York, NY, USA: ACM, pp. 276–283. [Online]. Available: http://doi.acm.org/10.1145/1127777.1127820

[25] F. WANG, J. LIU, On reliable broadcast in low duty-cycle wireless sensor networks. *Mobile Computing, IEEE Transactions on*, **11**(5) (2012), 767–779.

[26] P. LEVIS, E. BREWER, D. CULLER, D. GAY, S. MADDEN, N. PATEL, J. POLASTRE, S. SHENKER, R. SZEWCZYK, A. WOO, The emergence of a networking primitive in wireless sensor networks. *Commun. ACM*, **51**(7) (Jul. 2008), 99–106. [Online]. Available: http://doi.acm.org/10.1145/1364782.1364804

[27] J. POPE, R. SIMON, The impact of packet fragmentation on internet-of-things enabled systems. In *Proceedings of the 35th International Conference on Information Technology Interfaces*, ser. ITI '13, (2013), pp. 13–18.

[28] F. OSTERLIND, A. DUNKELS, J. ERIKSSON, N. FINNE, T. VOIGT, Cross-level Sensor Network Simulation with Cooja. *Proceedings of the Local Computer Networks, 2006 31st IEEE Conference*, (November 2006).

Received: August, 2013
Revised: September, 2013
Accepted: September, 2013

Contact addresses:

James Pope
Department of Computer Science
George Mason University
Fairfax
Virginia
United States
e-mail: james.h.pope8@gmail.com

Robert Simon
Department of Computer Science
George Mason University
Fairfax
Virginia
United States
e-mail: simon@gmu.edu

JAMES POPE ia a Ph.D. student of information technology at George Mason University, Fairfax, Virginia. He received his M.S. in telecommunications from George Mason University and a B.S. in business administration from Auburn University at Montogomery. He is a member of The Honor Society of Phi Kappa Phi. His research areas include embedded wireless systems and developing routing algorithms and protocols that effectively support application requirements for sensing and information quality.

ROBERT SIMON is an Associate Professor in the Department of Computer Science at George Mason University, Fairfax, Virginia. He received his Ph.D. in computer science from the University of Pittsburgh. He also has a B.A. in history and political science from the University of Rochester. His research specialization is in the field of networks, mobile and wireless computing, performance modeling and simulation. He teaches undergraduate and graduate courses in these areas.