Can P2P swarm loading improve the robustness of 6LoWPAN data transfer?

Marcel Bosling, Torsten Redmann, Jean Tekam, Elias Weingärtner and Klaus Wehrle
RWTH Aachen University

***Abstract*—The recent arrival of 6LoWPAN, an IPv6 variant for low-power wireless devices, allows for the development of IP-based applications for low-power wireless networks like sensor networks. As these networks often suffer from unreliable radio channels or frequent node failure, the question arises how 6LoWPAN-based applications can be hardened against such issues. In this paper, we examine if well-known concepts from the research domain of P2P networks can be applied for increasing the robustness of 6LoWPAN networks. For this purpose, we design and implement a straightforward P2P system for data sharing that is adapted to the low system capabilities of a typical 6LoWPAN device. Our preliminary evaluation shows that P2P mechanisms like swarm loading are able to improve the robustness of a 6LoWPAN application.**

I. Introduction

Wireless Sensor Networks (WSNs) are changing more and more from a mere research domain into a key technology with a high commercial potential. Companies such as Philips, Dust Networks and ArchRock (aquired by Cisco in 2010) are developing or are already marketing sensor networks software and hardware products. One important driving force behind the rising commercialization of WSNs is 6LoWPAN [8], [11]. It brings IP-based communication to embedded network devices and WSNs and hence facilitates their integration into common IP-based infrastructures, most notably the Internet.

In the meanwhile, peer-to-peer (P2P) [12] services for VoIP (e.g. Skype) and for file-sharing (e.g. BitTorrent [2], Gnutella [5]) have established themselves as important technologies for content delivery in the Internet of today. The reason is that P2P mechanisms effectively distribute resources, such as bandwidth capacities and storage, among participating nodes. One major application for P2P systems in the Internet is bulk data transfer.

The major advantage of P2P over client-server distribution systems lies in the ability to download partial content from different clients in parallel. This makes these schemes robust against node failure and improves content availability. Admittedly, transferring bulk data is less important for the execution of sensor network applications. However, one exemplary exception are software updates over the air: Here, binary images ranging in size from a couple of kilobytes to a few megabytes need to be distributed to the WSN nodes. This data can be considered bulky, as its size by far exceeds the capacity of a network packet.

Given the resource constraints of WSNs, this leads to the important question whether P2P mechanisms can be applied to wireless sensor networks. Therefore, applying off-the-shelf P2P protocols like BitTorrent or Gnutella for wireless sensor networks directly, is difficult. Typical issues are too bulky packet formats that exceed the frame capacity or the potential use of cryptographic operations that are too costly to compute on WSN hardware. In this paper, we address these challenges. More specifically, the contributions of this paper are the following:

- In Section II we present our ongoing work on a lightweight P2P architecture for bulk data distribution in wireless sensor networks. The design builds on top of 6LoWPAN and hence allows for the bulk data exchange also in heterogeneous 6LoWPAN deployments with different MAC layers and sensor devices in place. We also highlight different challenges related to the actual implementation of P2P systems on sensor hardware.
- We present an overview over our corresponding implementation based on TinyOS 2.X and the BLIP 6LoWPAN stack in Section III.
- In Section IV we evaluate the efficiency of our P2P distribution system in a real-world WSN deployment. We particularly investigate the effect of the swarm loading feature, which enables the parallel download of bulk data from multiple P2P sensor nodes, on the robustness of the data transfer.

We briefly discuss related work in Section V before concluding the paper in Section VI.

II. P2P over 6LoWPAN

We now discuss the general aspects and the design of a P2P system for 6LoWPAN networks.

A. Short Introduction to 6LoWPAN

IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [11] is a specification for implementing IP based communication over IEEE 802.15.4, a common PHY/MAC layer for WSNs. The main challenge is the maximum MAC frame size of 102 octets supplied by the IEEE 802.15.4 protocol in contrast to an MTU of at least 1280 octets specified by IPv6 [9]. Further issues are the limited reliability of devices in WSNs and the difference in functionality of full function devices (FFDs) and reduced function devices (RFDs) which may penalize link- and mac-layer security functions. The challenges addressed by 6LoWPAN are widely discussed in [10]. The main advantage of 6LoWPAN are the inherited stateless auto configuration of IPv6 networks, the large address

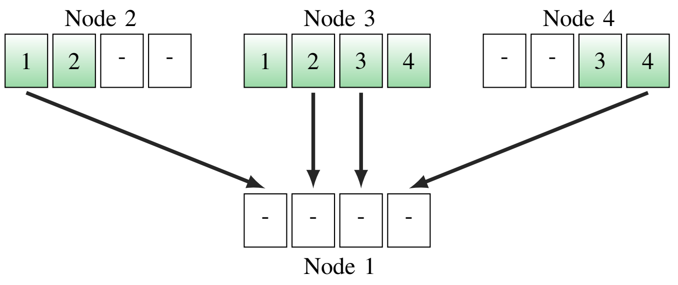

Fig. 1. Swarm loading: downloading blocks of data from different peers in the network.

space and the integration of WSNs into heterogeneous IPv6 networks.

B. A simple P2P protocol for 6LoWPAN

The main goal of P2P networks is the efficient use of all communication partners for speeding up the distribution of data throughout the network. In WSNs this is interesting for (autonomous) distribution of new firmware images and other bulk data. This can be accomplished by a technique known as *swarm loading* (segmented downloading, multi-source downloading) where different blocks of data are retrieved from different peers within the network as shown in Figure 1. In order to enable swarm loading on WSNs a P2P logic has to be implemented to coordinate the download of data blocks. Our P2P logic is depicted in Figure 2. It relies on two distinct communication channels, a designated command channel and a separate data channel. An additional shell-based channel is used for user control. The protocol itself operates as follows:

1) To signal a node that it should acquire a file, a `Gather` command is sent over the shell. This command contains the name of the file.
2) The requesting node will then send a `Query` to all reachable nodes using multicast. The query message also contains the file name.
3) All nodes that have the file answer with a `Hit` message containing data about the file and what blocks they hold.
4) The requesting node then selects a source and sends a `Request` for one block to the source node.
5) A UDP datagram containing the block and necessary management information (e.g. a block CRC) is returned to the requesting node.

This process is repeated until all blocks have been retrieved. This assures that blocks that were not reachable at known sources before, but have been acquired by those nodes since then, will also be downloaded.

III. Implementation

We now describe important aspects and the architecture of our 6LoWPAN-based P2P system.

A. Implementation environment

The application is built on top of TinyOS 2.X, an operating system for wireless sensor networks based on components and written in the nesC language. Our implementation builds

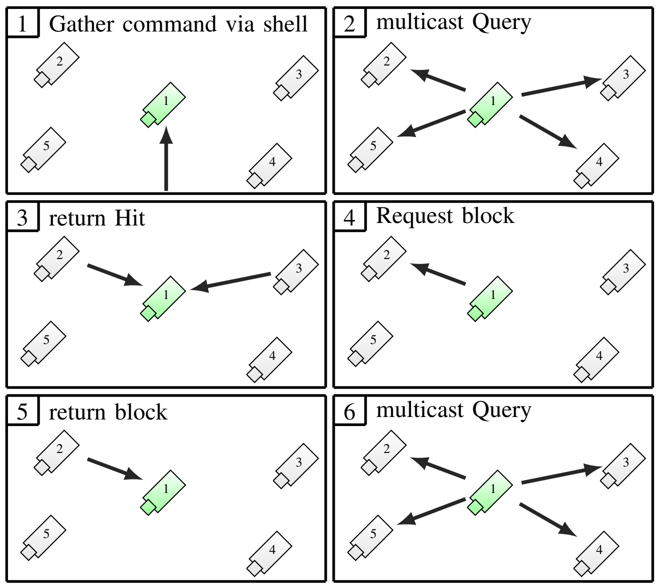

Fig. 2. Steps of the P2P protocol. 1.: A node receives a `Gather` command over the user shell. 2.: The node sends a `Query` for the file using multicast. 3.: It will eventually receive `Hit` messages for blocks of the file from nodes holding the file. 4.: The gathering node will then `Request` a block from answering nodes. 5.: The node receives a requested block. 6.: If the file is still incomplete it sends subsequent `Queries` for the file.

on top of the Berkeley Low-power IP stack (BLIP) [1], which implements 6LoWPAN for TinyOS. BLIP features a dynamic, multi-hop capable topology. Because of apparent stability issues with BLIP's TCP implementation, we use its UDP implementation for all communication tasks instead. In addition, we rely on BLIP's UDP shell functions to implement the user shell, which is mainly used for instructing nodes to gather a file.

B. System Architecture

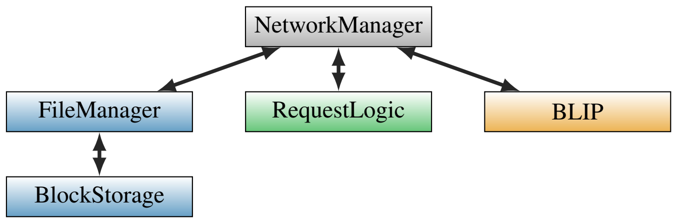

Fig. 3. High-level architecture of our 6LoWPAN system: The functionality is encapsulated into a small set of TinyOS components.

Figure 3 depicts the high level architecture of our TinyOS implementation. The following components are used to put our P2P protocol into action on top of BLIP:

- The main component and focal point, connecting the other components, is the **NetworkManager**. It parses all incoming messages and composes all outgoing messages. For this, it can access the other components to initiate storage/retrieval of data, or to forward data from one component to another.
- The **FileManager** handles everything pertaining to files. Files are identified by their name (considered to be

unique) and have a size measured in blocks as well as a CRC value for integrity checks. Regarding storage of the actual file data, the FileManager only saves a single address, pointing to the first data block in storage. Data is accessed using the **BlockStorage** component, which wraps around TinyOS' own block storage implementation and extends it. We chose block storage because it allows direct access to the flash memory by addresses. Additionally, we chose a data model that complements the swarm loading aspect even further, by partitioning the files into the same blocks that are sent over the network. Only the starting addresses of these blocks and their length are used. We perform integrity checks on every incoming block before it is saved, discarding corrupt blocks entirely.

- All information from incoming Hit messages is handled by the **RequestLogic** component. It stores data about viable source nodes for a file and generates requests for blocks that are missing.

IV. Evaluation

We now investigate the performance of our P2P system for 6LoWPAN networks. The metric used in all evaluation runs is the average completion time for the nodes to complete the retrieval of a file.

A. Evaluation setup

We evaluated the performance of our swarm loading approach in a deployment of 12 TelosB nodes, with one of them serving as the base station for BLIP. The tests were conducted by loading a test file of 5.4 kilobytes (separated into blocks of 64 bytes) onto one node and then setting a specific number of randomly chosen nodes to start acquiring the file simultaneously. For this purpose, we modified the program so that the nodes themselves measure the time it took to acquire the file. The nodes report these measurements back to the base station. This way, there is no effect of the time needed to send the command to the node or the answer back to the base on the measurement. Additionally, as it was our intent to test the performance of the swarm loading approach, the measurements were made with two different settings for the RequestLogic component. The first setting was a non-swarm loading setting where nodes select only one source, but request four blocks from this source at a time. The swarm loading setting was configured so that nodes would request blocks from a maximum of four source nodes with one block per source at a time. This means that the amount of messages generated by requesting nodes should on average be the same for both settings, resulting in a comparable amount of network traffic.

The routing tree maintained by BLIP changed often, providing a realistic simulation of a rather chaotic environment and thereby enabling us to draw more generalized conclusions.

B. Experimental results

During our evaluation deployment, measuring loading times turned out to be challenging, as the network became increasingly unstable with increasing number of active nodes. When using more than seven nodes trying to load files simultaneously, routing breakdowns happened often. This resulted in nodes becoming unreachable and therefore significantly hindering the tests. We attribute this behavior to the broadcast storm that is caused by a higher number of nodes if they are instructed to gather files concurrently.

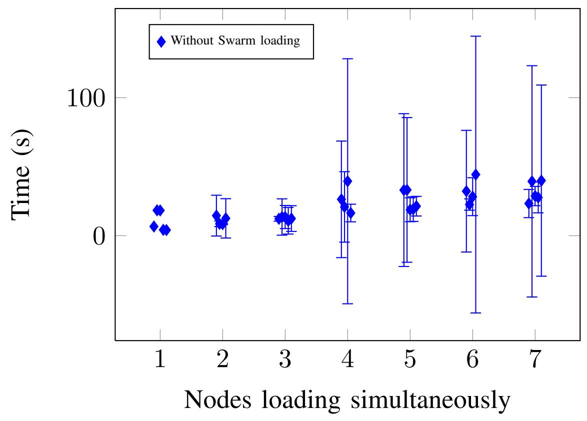

Fig. 4. Data for individual measurements of the non swarm loading setting. Values are slightly offset around their x-values to be discernable. The averaged individual measurements are close together, however, the standard deviation for each test is enormous, indicating bad performance.

Performance without swarm loading: In Figure 4 we depict the result for the non swarm loading tests. Times were averaged over the different test runs. For an increasing number of concurrently downloading nodes, the network becomes unstable. In fact, it was not possible in our deployment to perform reliable tests with more than seven loading nodes at the same time: crashes were inevitable. This can be explained by the overwhelming of bottleneck nodes: the sheer amount of requests to a single source node seems to disrupt the messages BLIP uses to build the routing tree, and therefore causes it to drop routes to nodes. The measurements that could be made consistently show a high standard deviation, meaning that although average loading times are similar, the individual times are very different, with some nodes taking many times as long as others.

Swarm loading enabled: Figure 5 contains the result for the swarm loading tests, showing that individual measurements have small standard deviations. We can see that while the time needed for all nodes to finish loading differs more if more nodes are active, almost always all nodes finished roughly at the same time. The file's blocks propagate through the network, becoming accessible from more nodes, which are then selected as sources by other nodes. There is also a serious improvement of the reliability of the network, making stable test runs with at least ten nodes possible.

Comparison: Figure 6 compares both measurement runs, with swarm-loading enabled and disabled, respectively. In both cases, we now average over all values for one node count. We clearly see that the standard deviation of the download only slowly grows if swarm loading is enabled. If we disable swarm

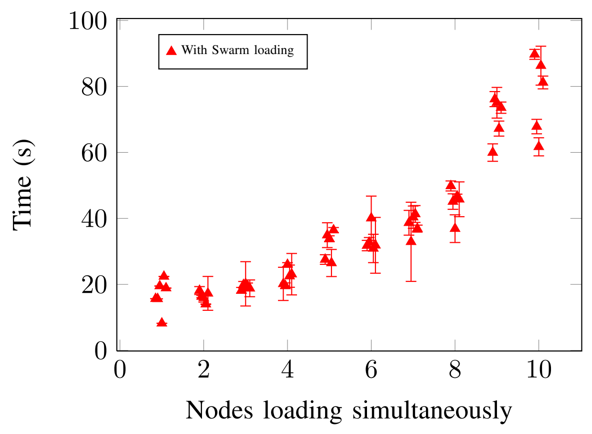

Fig. 5. Graph showing the data for individual measurements of the swarm loading setting. The diagram shows that for higher node counts, the measurements differ greatly, but the values in that measurement do not.

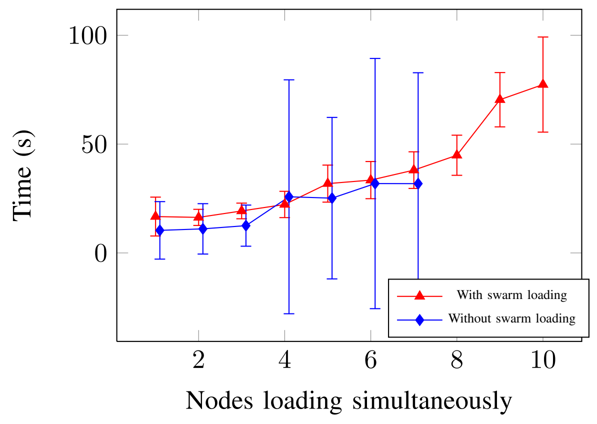

Fig. 6. Time for loading the test file averaged over all measurements for a specific setup. The graph clearly shows increased stability for the swarm loading tests, making measurements with more nodes possible.

loading, however, we observe very high standard deviations for measurements with more than 4 nodes, indicating a high variance in the individual download times observed by the nodes. We conclude from these measurements that enabling swarm loading dramatically increases the performance and the stability of our P2P file-sharing system.

V. Related work

To our knowledge, this is the first specific P2P system for data sharing over 6LoWPAN networks. It is, however, not the first P2P system for WSNs. TinyTorrents [7] is a BitTorrent [2] based architecture for WSNs. TinyTorrents employs well-established P2P mechanisms also found in our system, most notably, the segmentation of bulk data into blocks and the parallel transfer of such. In contrast to BitTorrent, which relies on torrents to keep track of file distribution, our flooding-based search strategy is reminiscent of Gnutella [3]. As both BitTorrent and Gnutella are designed to operate on standard IP networks, their native packet formats mostly exceed the maximum transfer unit of the IEEE 802.15.4 standard; this is an additional motivation behind our custom P2P system, which shares some similarities with WSN data dissemination frameworks such as Deluge [4] or MNP [6]. While these approaches use similar mechanisms (e.g. parallel transmission of blocks), they're realized as native protocols on top of the WSN's MAC protocol. By contrast, our approach is implemented on top of 6LoWPAN in order to investigate the efficiency of P2P mechanisms for improving the robustness of 6LoWPAN networked systems.

VI. Conclusion and Future Work

In this paper, we have sketched the design and the implementation of a P2P system for block data transfer on top of a 6LoWPAN network stack. To our knowledge, this is the first P2P system to operate on 6LoWPAN that resembles common P2P systems like Gnutella or BitTorrent. The main motivation behind this endeavor is the question if classic P2P mechanisms, in this case swarm loading, can improve the performance of 6LoWPAN applications. Our preliminary results show that swarm loading is able to increase the stability of a BLIP-based file sharing service against issues such as node failure or a disruptive network topology.

We consider the work presented in this paper as a preliminary first step towards a more comprehensive analysis of P2P mechanisms applied to 6LoWPAN-based network architectures. Other important questions to be investigated in the future include the operation of distributed hash tables on such networks and studies of energy efficiency.

References

[1] Berkeley IP implementation for low-power networks (blip). http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip (accessed Sept. 2011).

[2] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, 2003.

[3] Gnutella Developer Forum. The annotated gnutella protocol specification. http://rfc-gnutella.sourceforge.net/developer/stable/index.html (accessed Sept. 2011).

[4] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 81–94, New York, NY, USA, 2004. ACM.

[5] T. Klingberg and R. Manfredi. The gnutella protocol specification v0. 6. *Technical specification of the Protocol*, 2002.

[6] S. Kulkarni and L. Wang. Mnp: Multihop network reprogramming service for sensor networks. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 7 –16, june 2005.

[7] C. McGoldrick, M. Clear, R. S. Carbajo, K. Fritsche, and M. Huggard. Tinytorrents: integrating peer-to-peer and wireless sensor networks. In *Proceedings of the Sixth international conference on Wireless On-Demand Network Systems and Services*, WONS'09, pages 109–116, Piscataway, NJ, USA, 2009. IEEE Press.

[8] G. Mulligan. The 6lowpan architecture. In *Proceedings of the 4th workshop on Embedded networked sensors*, EmNets '07, pages 78–82, New York, NY, USA, 2007. ACM.

[9] Network Working Group. RFC 2460: internet protocol, version 6 (ipv6) specification. Technical report, IETF, 1998.

[10] Network Working Group. RFC 4919: ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals. Technical report, IETF, 2007.

[11] Network Working Group. RFC 4944: transmission of ipv6 packets over ieee 802.15.4 networks. Technical report, IETF, 2007.

[12] R. Steinmetz and K. Wehrle. *Peer-to-Peer Systems and Applications*. Lecture Notes in Computer Science. Springer, Oct. 2005.