Reprogramming over the Air and Sensor Island Management through SYNAPSE++

Nicola Bui, Cristiano Tapparello, Michele Rossi and Michele Zorzi
Dept. of Information Engineering, University of Padova, 35131 Padova, Italy.
Email: {bui, tappare1, rossi, zorzi}@dei.unipd.it

I. Introduction

Wireless sensor networks (WSNs) have been one of the main topics of networks research of recent years and now they are widely considered as a viable communication solution for resource constrained environments with moderate perfomance requirements. WSNs are often deployed in regions where it is difficult or too expensive to collect and redistribute the nodes for maintenance. However, there is often a need to reprogram all the nodes in the network, either during application test phases on deployed networks, or to support software upgrades. Therefore, a reliable method of sending a relatively large amount of data to each node in the network is required to support these functions. Also, such a feature moves forward the obsolescence time of the network, since new applications can be installed on the devices without any on site intervention.

The challenge to designing such in-network node reprogramming protocols lies in the potentially large amount of energy required to successfully transmit an entire program image to every node of the network. Typical issues related to the wireless channel include loss probability, fading effects and collisions. The use of unicast retransmissions for error correction for each node can be prohibitive in terms of traffic generation and hence transmission cost. Additionally, such retransmission techniques are known to result in feedback implosion [1] in dense networks. Coding solutions allowing different errors at various nodes to be corrected with single packet transmissions are preferable. However, many such techniques, using forward error correction codes (FEC), tend to be inefficient for wireless sensor networks. This is mainly due to the inherent computational complexity of standard codes (e.g., Reed Solomon). In addition, standard block codes have a fixed code rate, that cannot be changed on the fly according to channel errors or number of receivers. As a solution to these problems, in this demo we present SYNAPSE++, a tinyOS [2] reprogramming system on SYNAPSE [3] that efficiently copes with the above requirements through the use of rateless Fountain Codes [4] (FCs). These allow for high performance in dense as well as noisy environments, and substantially mitigate the feedback implosion problem. The novelties of this reprogramming system are:

This material is based upon work partially supported by the European Commission under contract number INFSO-ICT-215923 (SENSEI), by the Italian Foundation Cassa di Risparmio di Padova e Rovigo (CARIPARO).

- We design a new dissemination protocol consisting of an original pipelining strategy, coupled with a novel and distributed channel access mechanism, called soft-TDMA.
- SYNAPSE++ improves the FC implementation of SYNAPSE by a joint design with the forwarding mechanism so as to maximize the number of errors that are corrected through overhearing, thus limiting the number of explicit retransmissions.
- SYNAPSE++ features advanced bootloader and memory management modules, which allow the dissemination of binary images written in any operating system and make application and reprogramming software completely independent in terms of memory and variables.
- This demo will demonstrate how SYNAPSE++ can efficiently disseminate different applications to sensor nodes and allows users to easily interact with them.

II. System Description

Next, we present the data dissemination and error recovery algorithms of SYNAPSE++. Our aim is to disseminate a program image to all nodes of a WSN. Due to the inherent memory constraints of sensor nodes an efficient dissemination requires splitting this file into B *transport blocks* (TB) of K packets each, so that they can be processed in the available RAM. They are thus encoded through a dedicated Fountain Code into $K' = K + \delta$ packets each (δ is the number of redundant packets) before being transmitted. We tested SYNAPSE++ effectiveness against the standard de facto in WSN reprogramming, Deluge [5]: during our extensive experimental campaign in actual multi-hop deployments SYNAPSE++ showed an average dissemination times at least 10% shorter than that of Deluge (see fig. 1) with performance gains as high as 50% in dense networks. Below, we briefly describe SYNAPSE++'s protocol elements.

ADV/REQ handshaking: Any given node n maintains a bit-mask $b(n)$ indicating the TBs that it correctly received so far. As done in [5]–[7], either periodically or whenever a new TB is received, any node n advertises its status $b(n)$ by sending an ADV. This informs its neighbors about the TBs that this node can provide and allows an out-of-order delivery of transport blocks within the network. Interested neighbors respond with a REQ message including the smallest identifier among the blocks they need.

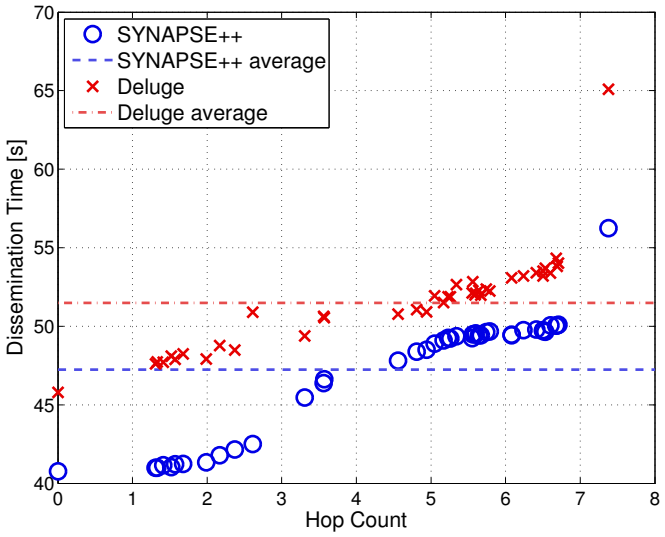

Fig. 1. Dissemination time as a function of the hop distance.

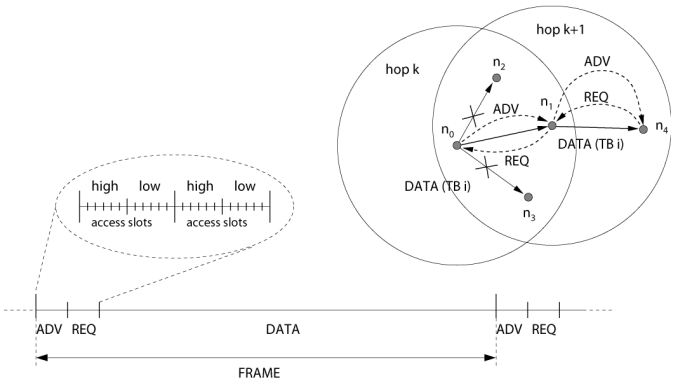

Fig. 2. ADV/REQ/DATA contention example and time *frame* structure.

Fountain codes: fountain code (FC) encoding is applied to each TB. That is, given the K original packets of a given TB, FC produces a slightly larger number of packets K', where the overhead δ is picked to have a successful recovery with probability greater than a minimum threshold at the first transmission of the TB. Each time a node n has a TB to send, it picks a suitable random seed s and obtains the K' encoded packets for this TB. To achieve good performance, random seeds have been carefully selected.

Pipelining: SYNAPSE++ implements pipelining through a novel approach as 1) packets are encoded through a suitable FC, 2) transmission turns among nodes are coordinated through an original pipelining scheme exploiting soft-TDMA schedules for improved efficiency (see Fig. 2 illustrating an example of ADV/REQ/DATA contention in two subsequent hops) and 3) pipelining and FCs are coupled through the selection of proper seeds to encode and transmit data over subsequent hops. In particular SYNAPSE++ is designed so that residual errors within hop i are corrected while forwarding the image to the nodes in the next hop $i+1$.

Synchronization and priority: in order to reduce the number of collisions and avoid idle times, which are typical of pure CSMA schemes, we opted for a loosely synchronized channel access scheme, which we call soft-TDMA. According to this technique ADV/REQ/DATA phases follow a specific time *frame* structure (see Fig. 2). Specifically, the transmission is subdivided into three time intervals: the first two are dedicated to the transmission of ADVsand REQs, respectively, and the last is allotted to DATA transmission and decoding. Synchronization is maintained by including in ADVs the remaining time to the next REQ interval. We refer to this *frame* synchronization as "loose" as only those nodes within the same transmission range must be synchronized at the *frame* level for correct reception; in this case a precision of a few milliseconds suffices (time skews are accounted for adding guard intervals among ADV, REQ and DATA periods). ADV and REQ intervals are further subdivided into access slots with two priority levels to push newest data towards unexplored portions of the network.

ADV/REQ suppression: in order not to produce too much control messages, ADVs and REQs related to transport block with lower or equal than those already advertised or requested in the same time *frame* are dropped.

Failure management: In case a node is unable to recover from losses through overhearing (i.e., its decoding matrix is rank deficient), it explicitly ask for incremental redundancy. These requests are served with low priority so as to promote the advancement toward unexplored portions of the network.

Back-off policy: a node that receives no response to its ADV defers the transmission of its next ADV according to a random timer whose maximum value is doubled at each transmission attempt (up to a maximum back-off time) and reset upon the reception of a REQ.

Energy conservation: due to the imposed *frame* structure, each node can infer whether the next TB to be transmitted will be useful or not interpreting eithr the preceding ADVs and REQs or the first DATA packet. Hence it is possible to switch-off the radio during the DATA interval in case of not needed TBs.

Bootloader for reprogramming WSNs: the Boot Loader manages many operations on the external storage (i.e., formatting, saving new applications and exchanging data between the internal and the external memories). Before issuing the *reboot* command an application writes in the information memory, which is non-volatile, information about the operation that has to be executed at the next reboot from the Boot Loader. Possible commands are: format, store, load.

We designed our system to maintain a complete independence between SYNAPSE++ and the third-party applications that are distributed and executed in the network. The advantages of this approach are manyfold: first, the reprogramming system does not interfere with the running program and second, this allows a better utilization of the scarce memory resources of the nodes. Additionally, even if SYNAPSE++ is written using TinyOS 2, there is no O.S. or language limitation for third-party applications. SYNAPSE++ can distribute and load any binary code, without jeopardizing

the network functionality as long as all applications include a procedure to reboot the node (the code needed to do this only takes a few bytes). After a reboot the Boot Loader will load the SYNAPSE++ image, that is always present in the first partition of the EM, and the network will be ready to distribute, load and execute any new application.

III. Demo Setup

This demonstration will show SYNAPSE++ as it disseminates new applications in a reliable fashion over multiple hops, activates these new programs in the network and performs management tasks such as checking nodes status (e.g., installed applications, memory usage).

In order to show SYNAPSE++'s true potential, during the demo we will use typical WSN applications, such as data gathering and a topology analyzer. Finally, we will demonstrate the user-friendliness of the system thanks to our graphical user interface: this extremely intuitive software provides easy access to every SYNAPSE++'s feature while hosting a visualization tool that shows data coming from the installed applications.

A small testbed will be deployed for the demonstration consisting of 20 TmoteSky sensor nodes (telosb achitecture [8]), one of which (the *sink* node) will be connected to a laptop pc running the graphical user interface. The whole setup requires a standard exhibition desk (about 1 square meter). The demo begins with all nodes running SYNAPSE++ and having their external memory empty (no applications installed). Subsequently, the sink node disseminates the application images to the other nodes, checking that it is received correctly by all sensors. The new applications will be loaded by all nodes and their output will be showed thanks to the visualization tool. Finally, after resetting the network to SYNAPSE++ further dissemination runs can be executed and the memory of the nodes can be managed.

References

[1] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-based Loss Recovery for Reliable Multicast Transmission," *IEEE/ACM Trans. on Networking*, vol. 6, no. 4, pp. 349–361, 1998.

[2] "TinyOS: an open source OS for the networked sensor regime," Last time accessed: March 2009. [Online]. Available: http://www.tinyos.net

[3] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. F. Harris III, and M. Zorzi, "SYNAPSE: A Network Reprogramming Protocol for Wireless Sensor Networks using Fountain Codes," in *IEEE SECON*, San Francisco, CA, US, Jun. 2008.

[4] D. J. C. MacKay, "Fountain Codes," *IEE Proceedings – Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[5] J. W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," in *ACM SenSys*, Baltimore, Maryland, USA, Nov. 2004.

[6] S. S. Kulkarni and L. Wang, "MNP: Multihop Network Reprogramming Service for Sensor Networks," in *IEEE ICDCS*, Columbus, Ohio, USA, Jun. 2005.

[7] R. K. Panta, I. Khalil, and S. Bagchi, "Stream: Low Overhead Wireless Reprogramming for Sensor Networks," in *IEEE INFOCOM*, Anchorage, Alaska, USA, May 2007.

[8] "Crossbow," Last time accessed: October 2008. [Online]. Available: http://www.xbow.com