http://dx.doi.org/10.21172/ijiet.81.002

A Review on reprogramming approach in wireless sensor nodes

Amritpreet Kaur
Department of Electronics and Communication Engineering
Amritsar College of Engineering and Techno logy, Amritsar, Punjab, India

Guneet Kaur
Department of Electronics and Communication Engineering
Amritsar College of Engineering and Technology, Amritsar, Punjab, India

Abstract— As the applications of Wireless Sensor Networks increase rapidly, the number of deployed sensor devices proliferates, which prompts the research community to work towards their integration in the so-called "Internet of Things" to gather real time information and make the maximum out of their use towards enhancing the user experience. The capability to reconfigure/reprogram them remotely not only enables easy maintenance and code updates, which is mandatory in large sensor network deployments, but also provides an unprecedented flexibility regarding the use of all available resources of different types. However, the design of a reliable dissemination protocol is a real challenge and the reason is threefold: the desired reprogramming requirements differ from use case to use case (e.g. tolerated reprogramming time, affordable overhead), the wireless medium is characterized by low reliability, and the devices are severely resource constrained. For this reason, in this paper we first explore the reprogramming requirements and the intricacies of WSNs and then review the already proposed network protocols for reprogramming wireless sensor networks placing emphasis on the their features to guide both prospect users and designers efforts.
Keywords— Wireless sensor networks, reprogramming, data dissemination protocols.

I. INTRODUCTION

As the applications of Wireless Sensor Networks (WSNs) proliferate, the population of sensor nodes increases and the need to manage them remotely becomes more prominent. The need for reprogramming the nodes stems from the fact that such systems must often operate for extended periods of time unattended, while adjustments to the environment after deployment as well as code maintenance and update are needed (e.g. to improve security or robustness). Remote management is also required to fulfill the application requirements which may change in time and space [1]. In emerging WSNs which consist of hundreds or even thousands of nodes, reprogramming them one-by-one requires both physical access to each of them (which is not always feasible) and consists an extremely time-consuming procedure, impeding any real-time reprogramming without human intervention. The ability to add new functionality or perform software maintenance without having to physically reach each individual node is already an essential service. Although solutions for remotely programming communication devices attached to specific infrastructure exist, programming resource constrained nodes over the wireless medium imposes different challenges than traditional network programming approaches. Transmitting the code that the node will execute over the air represents a real challenge due to the enhanced reliability required (since in this case the complete image must reach all the nodes) while the wireless medium is inherently unreliable. Moreover, due to limited memory, the involved nodes cannot store large files of programming code and due to limited processing resources the complexity of the employed code dissemination schemes has to be carefully assessed to ensure feasibility. If the image cannot fit into a single packet, it must be placed in stable storage until the transfer is complete, at which point the node can be safely reprogrammed. Another concern is the low throughput of WSNs: while the broadcast nature of the wireless medium can be exploited to compensate for the low communication throughput, special actions need to be taken in case of lost segments (e.g. due to collisions or errors) either when all or just one of the receiving nodes has not successfully received a packet. Hence, the design of a reliable data dissemination protocol for propagating large data objects from one or more source nodes to many other nodes over a multi-hop, wireless sensor network is the real challenge.

Unlike the unicast case where requirements for reliable, sequenced data delivery are fairly general, different multicast applications have widely different requirements for reliability. For example, some applications require that delivery obey a total ordering while many others do not. Some applications have many or all the members sending data while others have only one data source. Some deployments have all the network nodes executing the same applications and hence code dissemination is relevant to all nodes, while other consist of groups of nodes supporting different applications. Mission critical applications impose severe time-constrains while in other applications prolonged network lifetime is a top priority requirement. In any case, the dissemination process should ensure that no service interruptions to a deployed application and the debugging and testing cycle will occur. Another great challenge is imposed by the dynamic network membership and thus it must be ensured that all nodes receive the newest code since network membership is not static: nodes come and go. And while handling all these intricacies, the dissemination protocol designer should keep in mind that the dissemination must tolerate node densities which can vary by factors of a thousand or more. Such differences affect the design of a reliable multicast protocol with respect to the considered implications of the available design choices based on state-of-the-art network protocols for reprogramming purposes. We anticipate that this work will help both protocol designers providing them guidelines and prosper system designers/administrators to choose the appropriate solution. The procedure of node reprogramming / re-tasking can be split in two steps: first, decide when reprogramming is needed and second, disseminate the program. The schemes addressing the former are usually referred to as Code Consistency Maintenance Protocols (CCMP) and the latter are usually referred to as data dissemination protocols. (Integrated solutions addressing both issues have also been proposed.) Thus, the rest of the paper is organized as follows: in section II code consistency protocols are discussed while section III is devoted to dissemination protocols. Finally in section IV, an assessment and designer guidelines conclude the paper.

II. CODE CONSISTENCY MAINTENANCE PROTOCOLS

The first step in reprogramming a sensor node is to decide when reprogramming is needed and which part of the code needs update (if partial reprogramming is supported). Depending on the application, this can be decided and initiated by the system user, or automatically (in a distributed manner) by the nodes themselves. In the former case, the system user issues a reprogram command along with a set of attributes and the nodes operate in a slave-like mode. In the latter, the nodes should realize whether a code update is needed on their own.
An interesting solution for the first case is presented in [2]. The main goal of the Sensor Network Management System (SNMS) is the monitoring and control of the node and network status by humans. SNMS provides two core services: a query system to enable rapid, user-initiated acquisition of network health and performance data and a logging system to enable recording and retrieval of system-generated events. For this reason, a logical tree-topology for reporting the status of the nodes and the network is constructed and each time the system operator decides to check the health of the network, it issues several queries. In response to these queries the nodes provide status information and if a new code version needs to be downloaded, DRIP is used as the dissemination protocol. The command to switch over the new code uses (a different) named dissemination scheme. SNMS provides also naming instructions for the attributes that may need to be reported. The first contribution of the SNMS networking stack is a collection tree construction protocol that minimizes state requirements by not requiring a neighbor table, and minimizes network traffic by requiring explicit initiation of tree construction. However, the cost of maintaining a tree construction can only be afforded in static or semi-static sensor networks. If high mobility has to be supported, the cost of maintaining a tree just for checking the network status becomes high. Coming to the second case, a straight forward solution would be to periodically announce a profile of the code they run, so that their neighbors can compare the received information with the version of the code they run to figure out whether a code update is needed. As this way overhead is introduced, multiple schemes trying to reduce it have been proposed. A first attempt in this direction was proposed in 2004 and is widely cited. Trickle [3] is an algorithm for propagating and maintaining code updates in wireless sensor networks. Borrowing techniques from the epidemic/gossip, scalable multicast, and wireless broadcast literature, Trickle uses a “polite gossip” policy, where motes periodically broadcast a code summary to local neighbors but stay quiet if they have recently heard a summary identical to theirs. When a mote hears an older summary than its own, it broadcasts an update. Instead of flooding a network with packets, the algorithm controls the send rate so each mote hears a small trickle of packets, just enough to stay up to date. This simple mechanism can scale to thousand-fold changes in network density, propagate new code in the order of seconds, and impose a maintenance cost on the order of a few sends an hour.

III. DATA DISSEMINATION PROTOCOLS

Once one or more nodes have recognized the need for code update, the dissemination protocol is triggered. The design space for data dissemination protocols is large and includes: selection of nodes to transmit data. (For example, having multiple nodes transmitting the same data in the same transmission range area is not a wise option in the energy and throughput constrained environment of sensor networks.). Reliability assurance scheme would define how lost segments are identified, that are responsible for repair. To decide which node will provide the updated program, acting as the source of the updated program different options arise. Even if initially only one node has the updated code, if it broadcasts it, the set of neighbors in its transmission range can become the transmitting nodes in the sequence. (An example is shown in fig. 1a, where nodes 2, 3 and 4 receive all pages of a program image.)

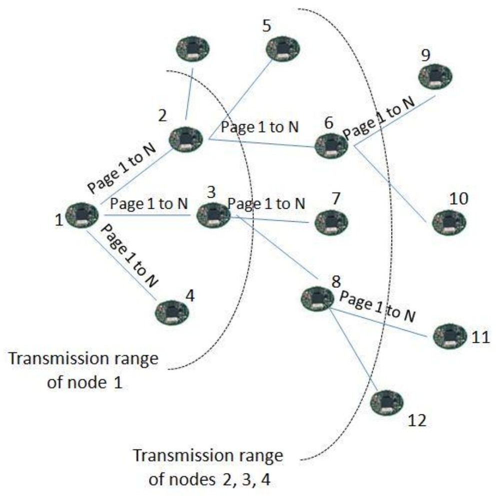

Figure 1.Code dissemination in a neighborhood-per neighborhood manner

In this case where multiple nodes can transmit the code, a specific protocol to choose the source for each neighbor has to be defined. (In fig. 1, it is obvious that node 4 may save energy since node 8 can receive the updated code from node 3 and the same holds for node 7.) Solutions ranging from choosing the latest node that advertised the program to more sophisticated publish-subscribe schemes have been proposed. The source selection scheme is tightly coupled with the way the program is disseminated (e.g. fragmented or not) and with the way that lost segments are retrieved. For example, fragmentation allows for spatial multiplexing which results in reduced time required for the dissemination of the program, as shown in fig. 2, where node 6 disseminates page 1, when node 1 disseminates page 2, assuming that the code has been split in pages. As regards lost segments retrieval, the use of negative acknowledgements is usually most suited to resource constrained WSNs. If one node has lost a certain packet, it can either request the retransmission of the packet, or wait to hear it again as part of the dissemination of the code to a next round. In the former case, the request can be unicasted to the source, multi-casted to improve the probability of receiving back the packet or even broadcasted, to make sure the packet will be received at the cost of energy waste from multiple nodes and increased congestion probability. Other ways to enhance reliability include the adoption of forward error correction (FEC) to avoid (to the extent possible) retransmissions or use link quality estimates to improve decisions or even use fountain codes to transmit data.

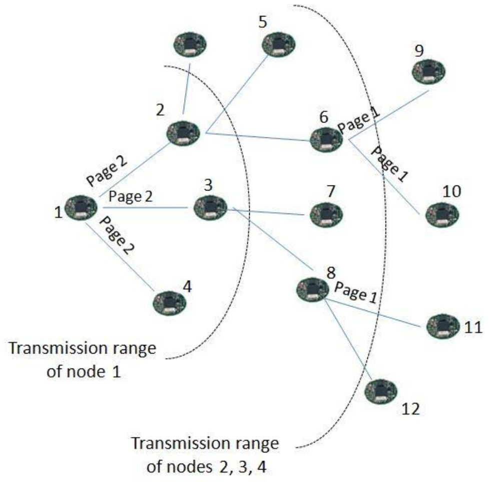
5
9
2
6
Page 1
Page 2
Page 1
3
Page 2
7
10
1
Page 2
8
Page 1
4
11
Transmission range of node 1
12
Transmission range of nodes 2, 3, 4

Figure 2.Spatial multiplexing: when node 6 disseminates page 1, node 1 disseminates page 2, reducing the total code dissemination time

The performance metrics reflecting the target of the dissemination protocols [7] include efficiency in terms of overhead (its reduction directly translates to better utilization of the low available throughput) and of energy (which affects the network lifetime, an important parameter in this mainly battery operated environment) as well the time required to detect that an update is needed and the time required to disseminate the update throughout the network.

IV.ASSESSMENT AND CONCLUDING REMARKS

The problem of reliable data dissemination has been addressed with multiple diverse approaches from the research community. The reason is that, while a common set of performance metrics could be agreed upon, the priority assigned to each of them depends on the specific use case. Additionally, the limited node and network resources prohibit the realization of a sophisticated, possibly even modular scheme that would be activated on demand, i.e. activate the most suitable component based on the realized application. So, any system designer should firstly clearly define the top performance metrics of interest and then choose the appropriate code maintenance and dissemination protocol.

Time-critical applications: when a system to support time critical applications is designed, it is absolutely necessary to avoid any service disruption (during code updates) and to switch to the new code in minimal time. In this case, it is the completion time (the time required for all the nodes to receive the updated code correctly) that counts most, and should guide the selection of the dissemination protocol.
Moreover, to cope with the rapid code updates, code consistency maintenance protocols that minimize the time required to identify a new version should be used. Such applications could be met at military missions or terrorist attack defense, where a sensor system may undertake the responsibility for gunshot localization and thus rapid dissemination of information to combatants is critical. To achieve low completion times and low version detection times, higher overhead is required which increases the energy consumption and reduces the bandwidth utilization.

Frequent code updates: When reprogramming is used to change the functionality of the sensor network in the day or between the days of the week, then the latency and the completion time no longer represent the top requirement. Instead, the energy consumption is more important in this case, since lower energy consumption means longer network lifetime. For example, reprogramming a sensor network to execute a stronger encryption scheme may change twice every day or to change specific parameters of the application (how often and what types of sensed data are sent to the sink) does not impose latency constraints (e.g. in agricultural monitoring where nodes operate in unattended node in wide areas and may need to monitor the temperature once every 5 or ten minutes in spring but more often in winter to prevent the fruit freeze). To safeguard the network lifetime, the energy consumption which is tightly coupled with the overhead of the code consistency and dissemination protocols should be the basic decision factor.

Lost segment recovery: The scheme employed to recover from lost segments affects both the completion time and the overhead produced by the dissemination protocol. As such, it could be seen as a parameter already addressed previously. However, the use of fountain codes or random linear codes has been shown to require light-weight implementations and to also reduce both completion time and overhead. The research community seems to find a common direction on this issue.

Support of multiple applications in the sensor network: As emerging sensor networks comprise of nodes executing multiple applications, the need to support their remote reprogramming becomes essential. In this case, each node should be able to differentiate its role with respect to the different supported applications and become either source/destination of an updated code of interest or just a forwarder for other nodes. This need slightly increases the processing requirements, and thus the implementation of such a scheme should be restricted to systems where needed, leaving more hardware resources for the execution of more sophisticated application or security schemes. The evolution of sensor node hardware platforms is expected to alleviate such problems and enable the wide implementation of dissemination protocols supporting multiple applications concurrently.To this end, with the current sensor platforms a code consistency and dissemination protocol there is no one-fits-all cases solutions. For this reason, the prospect protocol designer or system implementer has to carefully consider and prioritize.

V. CONCLUSION

Reprogramming is important in facilitating the management and maintenance of WSNs, as well as enabling adaptive sensor applications. It becomes a crucial service to the success of WSNs. Currently WSNs are still in the state of active development. In the process of WSN revolution, we will see new hardware platforms (Mica, Telos), new operating systems (Tiny OS, SOS), and new applications (well controlled bridge monitoring, randomly

REFERENCE

[1] E. Ladis, I. Papaefstathiou, R. Marchesani, K. Tuinenbreijer, P. Langendörfer, T. Zahariadis, H. C. Leligou, L. Redondo, T. Riesgo, P. Kannegiesser, M. Berekovic, C. J. M. van Rijn , “SMART: Secure, Mobile visual sensor networks ArchiTecture”, IEEE SECON2009, 22-26 June, 2009 Rome Italy.

[2] G. Tolle, D. Culler, “Design of an application-cooperative management system for wireless sensor networks”, European Workshop on Wireless Sensor Networks, Feb. 2005.

[3] P. Levis, N. Patel, D. Culler, S. Shenker “Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks” Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1 San Francisco, California, 2004.

[4] K. Lin, P. Levis, “Data discovery and dissemination with dip”, International Conference on Information Processing in Sensor Networks (IPSN 2008), Washington, DC, USA, IEEE Computer Society (2008) 433–444.

[5] T. Dang, N. Bulusu, W. Feng, S. Park, "DHV: A Code Consistency Maintenance Protocol for Wireless Sensor Networks", In Proc. of the 6th European Conference on Wireless Sensor Networks (EWSN 2009), Cork, Ireland, Feb 2009.

[6] W. Li, Y. Zhang, and B. Childers, “MCP: an Energy-Efficient Code Distribution Protocol for Multi-Application WSNs”, 5th IEEE International Conference on Distributed Computing in Sensor Systems, LNCS 5516, Springer-Verlag, pages 259-272, Marina Del Rey, California, June 2009.

[7] M. Horsman, M. Marin-Perianu, P.G. Jansen, and P.J.M. Havinga, “A Simulation Framework for Evaluating Complete Reprogramming Solutions in Wireless Sensor Networks”, 3rd International Symposium on Wireless Pervasive Computing, 7-9 May 2008, Greece. pp. 6-10.

[8] S. Floyd, V. Jacobson, C. Liu, S. McCanne, L. Zhang, “A Reliable Multicast Framework for Light-eight Sessions and Application Level Framing”, IEEE/ACM Transactions on Networking, December 1997, Volume 5, Number 6, pp. 784-803.

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

[10] J. W. Hui, D. Culler, “The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale”, Sensys 2004

[11] R.K. Panta, I. Khalil, and S. Bagchi, “Stream: Low Overhead Wireless Reprogramming for Sensor Networks,” IEEE Conference on Computer Communications (Infocom), 2007.

[12] 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”, IEEE SECON 2008, San Francisco, California, US. June 16-20, 2008.

[13] K. Maier, A. Hessler, O. Ugus, J. Keller, D. Westhoff, “Multi-Hop Over-The-Air Reprogramming of Wireless Sensor Networks using Fuzzy Control and Fountain Codes” SOMSED'09, Self-Organising Wireless Sensor and Communication Networks Hamburg, Germany 8 - 9. October 2009.

[14] W. Dong, C. Chen, X. Liu, J. Bu, Y. Gao, “A Light-Weight and Density-Aware Reprogramming Protocol for Wireless Sensor Networks”, IEEE Transactions on Mobile Computing, Dec. 2010.

[15] Y. Yu, L. J. Rittle, V. Bhandari, J. B. Lebrun, “Supporting concurrent applications in wireless sensor networks”. 4th int. conference on Embedded Networked Sensor systems, SenSys 2006, pp. 139-152.