Ad Hoc Networks 9 (2011) 1270–1286

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

A dynamic reconfigurable routing framework for wireless sensor networks ☆

Mukundan Venkataraman [a,*], Mainak Chatterjee [a], Kevin Kwiat [b,1]

[a] Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, United States
[b] Air Force Research Laboratory, Rome, New York, United States

ARTICLE INFO

Article history:
Received 8 March 2010
Received in revised form 13 January 2011
Accepted 17 January 2011
Available online 22 February 2011

Keywords:
Wireless sensor networks
Multi-hop routing
Dynamic routing

ABSTRACT

Sensornet deployments of the future are expected to deliver a multitude of services, ranging from reliable sensing, real time streams, mission critical support, network reprogramming and so on. Naturally, no one routing protocol can sufficiently cater to the network layer functionalities expected. Severe resource constraints further limit the possibility of multiple routing protocols to be implemented. Further, vertically integrated designs of present protocols hinder synergy and code-reuse among implementations. In this paper, we present an architecture that allows applications to send different *types* of flows, often with conflicting communication requirements. A flow's requirements are made visible to our framework by using just 3 bits in the packet header. The core architecture is a collection of highly composable modules that allows rapid protocol development and deployment. We show that our framework can provide: (i) flow based network functionality that ensures each flow gets an application specific network layer which is dynamically knit as per the flow's needs, (ii) modular organization that promotes code-reuse, run time sharing, synergy and rapid protocol development and (iii) pull processing that allows flows to dictate their traffic rate in the network, and implement flexible scheduling policies. This creates a framework for developing, testing, integrating, and validating protocols that are highly portable from one deployment to another. Using our framework, we show that virtually any communication pattern can be described to the framework. We validate this by gathering requirements for one real world application scenario: predictive maintenance (PdM). The requirements are used to generate a fairly complete and realistic traffic workload to drive our evaluation. Using simulations and 40 node MicaZ testbed experiments, we show that our framework can meet the deployments demands at granularities not seen before in sensornets. We measure the costs of using this framework in terms of code size, memory footprints and forwarding costs on MicaZ motes.

1. Introduction

Protocol development in sensornets have to meet the challenges of correctness and efficiency, while functioning on a resource constrained hardware platform. Hence, typical sensornet deployments thus far have focused on performance more than functional flexibility by building deployments that perform limited tasks. In fact, most sensornet deployments today have a highly focused and rather singular objective. Technology trends indicate that

☆ This research was supported, in part, by the Air Force Office of Scientific Research (AFOSR) under the federal Grant No. FA9550-07-1-0023. Approved for Public Release; Distribution Unlimited: 88ABW-2011-0368 dated 28 Jan 2011.

* Corresponding author. Tel.: +1 321 274 7591.
E-mail addresses: mukundan@eecs.ucf.edu (M. Venkataraman), mainak@eecs.ucf.edu (M. Chatterjee), kevin.kwiat@rl.af.mil (K. Kwiat).

[1] An earlier version of this paper appeared in IEEE WCNC 2009 [19].

1570-8705/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2011.01.007

sensornets of the future will be expected to do multiple, and often conflicting, tasks simultaneously. For example, a comprehensive deployment solution could simultaneously send periodic sense-and-disseminate flows, real time streams, mission critical alerts, network reprogramming data, patched updates, interactive queries, commands and so on. Naturally, no *one* routing optimization can sufficiently cater to such diverse communication requirements. Accommodating arbitrary communication patterns while conserving limited resources calls for dynamism in route selection.

Providing a dynamic network layer functionality is, however, non-trivial. Extreme scarcity of resources makes full implementations of multiple protocols infeasible. Further, existing protocols are highly optimized for a given deployment and its associated traffic type. This optimization has come at a high cost: limited portability, limited code-reuse, and non-extensible designs. To promote synergy and maintain adaptability, there is a need to bring flexibility closer to protocol design.

In this paper, we present an architecture that allows an application to generate multiple *types* of flows, often with conflicting communication requirements. Our architecture provides an *application specific* network layer for each flow based upon its needs. To make this possible, a flow's requirements are made visible to the framework using three descriptive bits in the packet header. Likewise, protocol modules inside the framework carry a similar 3 bits description that advertises their suitability for a particular flow. The core of our framework is a modular software architecture for implementing network layer functionalities composed of "modules". Modules are highly composable, lightweight components that implement small network tasks. Functionalities in the network layer are built by arranging a sequence modules in the form of a graph. New protocols can be written by extending, adding or re-arranging the modules already present. We regularize the various interface assumptions and data structure requirements by the use of a shared neighbor table, and decouple core protocol features from resource management functionalities. In effect, we integrate the vast body of work in network protocols available in sensornets by distilling core protocol features for various traffic types. This would foster tremendous synergy across various deployment efforts since the core component protocols can be easily migrated from one effort to another. While we present specific implementation choices for both the preamble bits and the component protocols, we note that this is *only one* instance of our proposed framework. We envision various preamble bits and accompanying protocols based on the need at hand. In other words, this paper's contribution is a frameworks that can handle conflicting requirements and not the component protocols themselves.

Specific properties of this framework make it a powerful tool for building, testing and deploying complete sensornet applications. *Pull processing* allows elements closer to the transmitting interface regulate packet flow on a need basis. *Flow based network functionality* allows inspection of a flow's needs, and helps construct an appropriate set of functionalities for that flow by knitting relevant modules. It provides a deeper insight to providing reliability for certain flows, speedy delivery for certain other, aggregation for delay tolerant flows and so on.

We evaluate the overhead of using this framework in terms of code size, memory footprint and processing overhead by implementing various combinations of modules on MicaZ motes [27]. Our experience shows that we achieve upto 30% reduction in code size and more than 40% reduction in memory footprints compared to full implementations. Our framework can forward upto 4437 bytes/sec before a livelock [13] occurs.

We also bring out the practical benefits of using this framework by conducting a case study of a fairly large scale industrial application of predictive maintenance (PdM). We show how our framework can adapt to the various communication requirements of an operational PdM, and offer custom network layer function for every instance. We gather and meet PdM's operational demands on a 40 node wireless MicaZ testbed for 2 days, and a 9-week worth simulation time on a 1000 node simulation to analyze behavior at scale.

2. Related research

This paper builds upon two previous approaches by us with a dynamic network framework for sensornets [19–21], by providing a comprehensive network layer architecture and framework for flow based network functionality. We also provide implementation insights, performance evaluation and a case study to substantiate previous work.

There has been an excellent series of work that address the problem of promoting synergy and providing sufficient abstraction for rapid protocol development. Work by Culler et al. [3] argued that the abstraction that IP provides in the Internet can be provided at the link layer for sensornets, and this was substantiated by the SP architecture [16]. The SP architecture deals with the link layer, and as such, is complimentary to our work. In fact, our dynamic routing framework could very well be implemented *on top* of the SP architecture. The case for a modular network layer, where core network layer protocols were decomposed into an abstract set of modules, was then proposed by Cheng et al. [1]. They address issues of code-reuse and runtime sharing, and postulate a co-existence of routing protocols. However, they do not address how multiple flows can co-exist, and how a unified framework could provide flow based network functionality or pull processing. The issue of code-reuse and ease of protocol development has been investigated in [7,9]. Though we share the same motive as some of these approaches (promoting a programmable framework for rapid deployment, unifying various research efforts in routing data), our approach and methodology are different. Unlike other approaches, our scheme revolves around a preamble that describes communication intent of a flow, with the intent specifying requirements such as reliability, speed of delivery, and nature of payload.

Modularizing network layer functionality has also been witnessed with Internet architectures such as the *x*-kernel [10,15] and the Click [14] modular IP router. The *x*-kernel