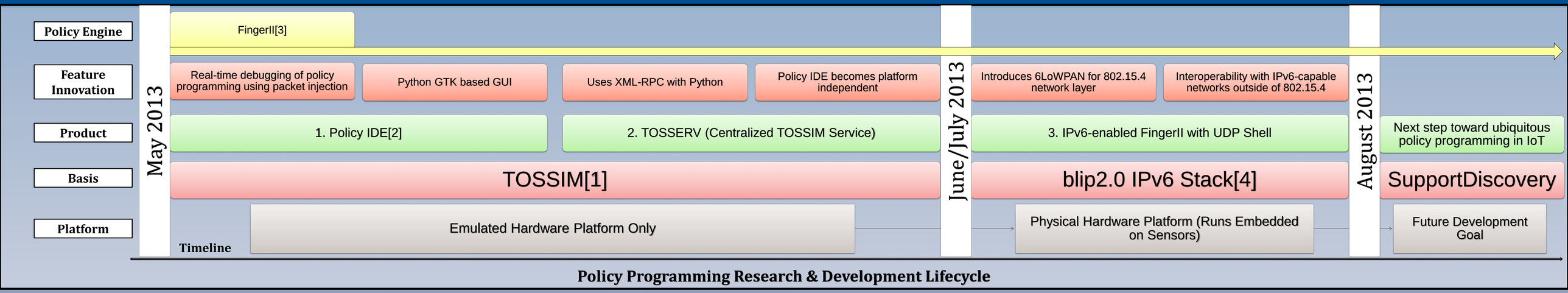
Facilitating the Internet of Things with Policy Programming

Daniel Smullen, Ramiro Liscano

Faculty of Engineering and Applied Science ■ University of Ontario Institute of Technology ■ 2000 Simcoe Street North, Oshawa, Canada, L1H 7K4



Policy Programming Research & Development Lifecycle

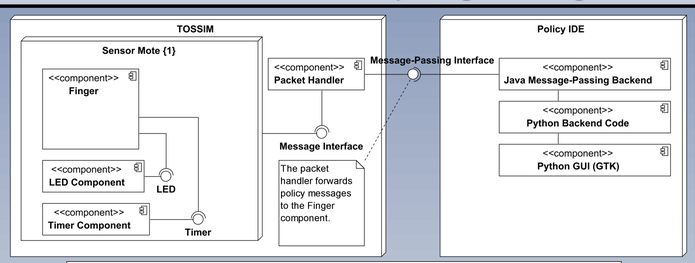
Research Objectives

Policy programming enables autonomic, customized behavior in wireless sensor networks (WSN), augmenting basic subroutines with extensible control logic. In WSN based on the tinyOS platform, state introspection and debugging is challenging and opaque; instrumentation embedded in WSN application nesC code is required, and the process of collecting data is difficult. Our state of the art Policy IDE[2] began with limited simulators to collect and display debugging data interactively.

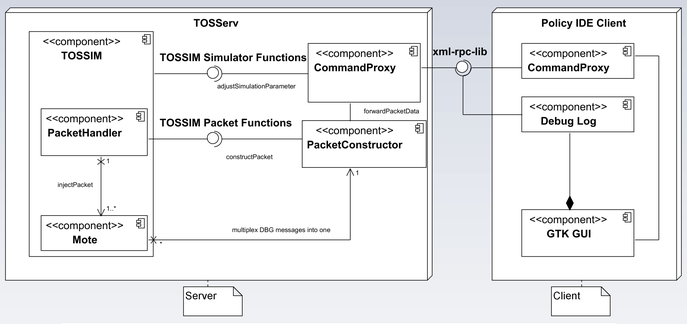
Development has moved towards creating more robust Policy IDEs. Our latest approach pairs policy programing with advanced inter-node network communications protocols at the transport layer, replacing link-layer protocols' low-level communication with a new network stack. Policies can now be used on IPv6, extending WSN interoperability. Policy interaction between sensors, network appliances, desktop and mobile computing platforms will define the types of machine to machine communications on the future Internet. Additional benefits include simpler debugging, visible execution trace, and a shortened WSN application development lifecycle using real sensor motes which can be debugged in development or in the field.

Background

The Internet of Things (IoT) is a novel paradigm which is shaping the evolution of the future Internet. The goal is to enable interaction among mobile devices, desktop computers, network devices, and sensors. This interaction occurs throughout the Internet, local area, and personal area networks - in the home, vehicles, and a multitude of other environments. A simple and proven programming approach for the management of distributed systems is Policy Programming[3], and this has proven particularly effective in surmounting design challenges associated with tinyOS based WSN devices. Developing energy efficient devices that operate on small batteries autonomously for long periods of time is a primary WSN design objective[4]. Reprogramming devices with policies yields less network and processing overhead, which saves power on battery operated sensors[3].



Original Policy IDE Deployment Diagram[2]

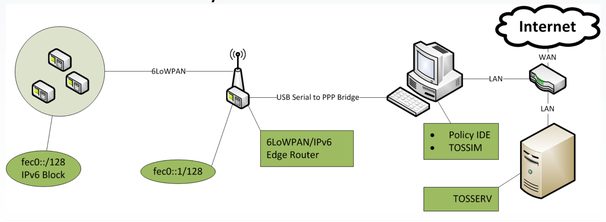


Offloading Emulation/Simulation using TOSSERV

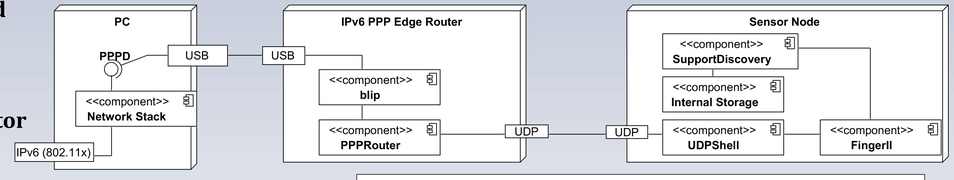
To date, most WSN applications are designed for one specific application, using programming designed to fulfil one limited role only. Policy programming enables these same hardware devices' programming to become more adaptable and more flexible in changing circumstances while maintaining overall system autonomy and availability[3]. Developing tinyOS nesC applications is challenging - there are few libraries and only one set of development tools[2]. Engineering applications for WSN toward applications involving the IoT is on the cutting edge of wireless technology, internet protocols, and autonomic computing.

Products and Results

1. **The Policy IDE was developed by Nidal Qwasmi and Daniel Smullen[2]. This concept was originally designed to allow for policy programming to be debugged, reducing the opacity in the development process. Debug messages were collected from the TOSSIM emulator[1] and displayed on a GUI in real time. Viewing these messages while manipulating policies on simulated devices allowed introspection into the results of policies. Policy programming was conducted using emulated sensor motes running the FingerII policy engine core[3].**
2. **The TOSSERV centralized TOSSIM service was developed, allowing the Policy IDE and TOSSIM device emulator/simulator to become platform independent. This also allowed future versions of TOSSIM to be interoperable with Policy IDE and allowed for simulation to occur across a network when desired, instead of localized on one machine.**
3. **The blip2.0 IPv6 network stack[4] was integrated with the FingerII policy engine and a UDP shell, allowing policy debugging to occur with real sensor hardware. Introspection occurred by logging onto actual motes and directly entering commands into the shell, with feedback returned to the same terminal session in real time. An edge router was used to marshal interoperable traffic between the 802.15.4 6LoWPAN based sensor motes wireless radio and wired/wireless external IPv6 networks.**



IPv6 WSN Configuration with Motes Running FingerII



Early SupportDiscovery Deployment Concept

Future Work

A new concept (dubbed SupportDiscovery) would incorporate a policy discovery component that enables the FingerII engine to perform reflection and report on what device components, events, and tasks are available for policy programming. Development of a detailed architectural model for this software component along with integrating the Policy IDE with the IPv6-enabled FingerII would allow for a more robust IDE to be created. This would allow up to 2^{128} sensors per network layer to be programmed with policies easily.

Specialized policies, policy roles, or applications using policies could be engineered on a large scale.

References

[1] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. 2003. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys '03). ACM, New York, NY, USA, 126-137. DOI=10.1145/958491.958506 http://doi.acm.org/10.1145/958491.958506

[2] N. Qwasmi, D. Smullen, R. Liscano, "Integrated development environment for debugging policy-based applications in wireless sensor networks," The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks, 2013.

[3] Yanmin Zhu; Sye Loong Keoh; Sloman, M.; Lupu, E.; Yu Zhang; Dulay, N.; Pryce, N., "Finger: An efficient policy system for body sensor networks," Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on , vol., no., pp.428,433, Sept. 29 2008-Oct. 2 2008 doi: 10.1109/MAHSS.2008.4660033

[4] "Evaluating the Performance of RPL and 6LoWPAN in TinyOS" by Jeonggil Ko, Stephen Dawson-Haggerty, Omprakash Gnawali, David Culler, and Andreas Terzis. In Proceedings of Extending the Internet to Low power and Lossy Networks (IP+SN 2011), April 2011.

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Funding provided by: NSERC CRSNG

UOIT Student Research Showcase 2013