The Physical Side of Computing

By Feng Zhao

Wireless sensor networks (or sensornets) represent a new computing platform that blends computation, sensing, and communication with a physical environment such as a bird habitat, bridges, or power grid. This new class of networked embedded computers requires new programming models, abstractions, and management tools. They change the way we think about computation and challenge the design of the next generation Internet that not only connects people together but also connects people with the physical environment.

This computing platform is characterized by the embedding in the physical world and (often) unattended operation for years, severe constraints in resources especially energy, unreliable hardware and communication links, and the need to respond to time-critical events. The implications are twofold. A sensornet must gather and act on sensor data in a timely manner. The value of information and window of opportunity for action may dwindle as time elapses. Consequently, sensornets should support reliable and timely data collection and dissemination despite significant link and data variability and hardware flakiness. Second, because of limited battery capacity, a sensor node limits the amount of onboard memory and uses low-power microprocessor and low-date-rate radio with power-saving dials. The data collection and dissemination must be handled in an energy-efficient manner.

A fundamental computer science question arising from sensornets is the role of energy and how we think about it in relation to performance and quality metrics such as latency and data yield. Much of CS has been built on the analysis of the time and space complexity of algorithms that has informed the design of processor, memory, and I/O in computing systems. Only recently have we confronted the energy problem head on, in designing high-performance servers as well as low-power sensornets (supercomputing addressed the cooling problem before). Multi/many-core is one answer in the upper tier of the computing ecosystem. The tiny computers in sensornets expose another rich area where energy trades with performances in a decentralized, fine-grained way. For example, communication in data dissemination may be delayed, to reduce collision and hence energy due to excessive retransmission, at the expense of a larger latency. Sensor data may be locally compressed at the node, to reduce the data volume sent over the wireless, trading the communication energy with that of processing. This points to the need for establishing a theory of "energy complexity" in computing that provides models for energy and its tradeoffs with other system metrics.

The decade of sensornet research has produced a rich collection of algorithms, protocols, system architectures, tools, and several generations of hardware platforms. The energy constraints, for example, led many to design extremely efficient systems that break the traditional networking and systems layers in order to squeeze the last Joule out of the operation. Naturally, one asks, what are the reusable building blocks and common abstractions that emerge from these works? Some of the techniques address the deeper problems of energy complexity, system scalability, and robustness. Others may just be artifacts of the current hardware limitations.

Levis *et al*. answers the question with Trickle, a building block for algorithms that move data around quickly in a sensornet while conserving its limited energy. Realizing that the one-to-many and many-to-one data dissemination and collection in a network rely on a common primitive to detect when the state of a node becomes inconsistent in a network of shared variables and to propagate the information when inconsistency arises, they propose an epidemic-style algorithm that does so on an as-needed basis. It was originally designed for distributing code in a sensornet, as in re-tasking or code patching. To detect whether a node has the latest version, each node declares to others which version it currently has. An inconsistency triggers the propagation on demand, suppressing the transmissions of others, thus more energy efficient than flooding.

The key idea behind Trickle is to maintain a constant number of message transmissions per area, and use a feedback mechanism to regulate that as node density changes. A node only decides to transmit if it has not heard from a sufficient number of its neighbors. This way, the more nodes in an area, the less likely each node will decide to transmit as the likelihood of others already having advertised increases. Trickle provides the dials to trade energy expenditure of the network with the speed of the propagation. This self-regulation mechanism is similar to how nature regulates the population of a species, where growth is self limiting because of the finite sustainable food supply.

A useful primitive finds itself in many applications. Trickle is promising; since the publication of the original paper, the idea of Trickle has found itself in data dissemination as well as data collection algorithms, including TinyOS 2.0 CTP, a data collection protocol.

Gordon Bell posits every decade or so a new computing platform emerges due to advantages in form factor, interface, and functionality/price. The wireless sensor network is such a new computing platform. I expect emerging primitives and abstractions like Trickle, being developed by the research community, to help us conceptualize and modularize the design of this new platform and to become part of a standard TTL-like catalog for building scalable, reliable, and energy-efficient sensornet systems.

Feng Zhao (zhao@microsoft.com) is a Principal Researcher at Microsoft Research, Redmond, WA.