David E. Culler *University of California, Berkeley*

Editors: Carla Schlatter Ellis

Photo, istockphoto.com

THE ONCE AND FUTURE INTERNET OF EVERYTHING

I am often asked these days, "Is IoT really as transformative as all the buzz suggests, or is it a lot of hype?" My answer is simply, "Yes." And the road to here has been a fascinating interplay of academic research accomplishments, industry advances (often with other drivers), and standardization processes (for lack of a better term). It was predictable that today we would be poised for the "next tier" of the Internet to take off, but the endgame seems to be even messier than expected.

For some transformative advances, like the Web, the research community finds itself asking, "How could we have missed this?" For the emerging Internet of Things (IoT), we absolutely saw it coming and took on tackling the hard problems as necessary to bring it into existence – so that today we could take them largely for granted. In preparing this retrospective it was nostalgic to uncover the slides from 1999, such as images in Figure 1: the first from the "Invisible Computer" workshop, organized by Gaetano Borriello (1958-2015), and the second from the first research retreat of Berkeley's part of the DARPA Expeditions program on Information Technology for the 21st Century.

Along with project Oxygen at MIT, Portolano at University of Washington, and others, in 1999 we imagined a world ahead with billions of networked sensors and actuators, connected personal devices, and pervasive cameras and displays, all supported by rich, planetary-scale Internet services, with intervening middle-boxes in some cases. The Web was six years old and search had just appeared; laptops, the Palm Pilot, and cellular phones were prevalent; 802.11 was just appearing and radio modems existed, but the iPhone was still eight years off. Fifteen million Furbys had been sold – a massive deployment of inexpensive sensors. The Wii would appear in 2006. It was predicted then that "in fifteen years a complete computer, with processing, storage, and communication, would fit in a cubic millimeter," creating a sense of urgency to solve the multitude of software and networking challenges necessary to fully utilize such novel machines. Classical operating system and network architectures were seen as constraining the solution space so much so that they needed to be set aside to allow new ideas to flourish. A renaissance of systems and networking research emerged, addressing severe resource constraints and uncertainty, challenges of unattended operation, in-network processing and wholly new application demands.

In 2007, Deborah Estrin and I gave a joint keynote at the Federated Computing Research Conferences entitled "Wireless Sensing: the Internet's Front Tier" in which we laid out the enabling systems research that had made the decade-old vision a reality and described the tremendous societal and scientific value to come from embedded network sensing – what we now largely term IoT. In 2008, with the publication of Hui's thesis and Sensys paper [HuCu08] showing how all these advances could naturally reside within the Internet architecture, I truly thought the Internet of Everything had arrived. So what happened? Why is it taking so long? How close are we nearly another decade hence?

This retrospective touches on four fundamental advances that came out of the broad wireless sensor network research community and are essential to the eventual success of the IoT, along with a brief assessment of where we are today and a call to action.

LOW-POWER, COMMUNICATION-CENTRIC SYSTEM DESIGN

Hardware is the easiest development to track, because it is so tangible. For the predecessors of the IoT, this process was a rich interweaving of research programs, industrial developments, and standards efforts, with the introduction of *open source hardware*. Figure 2 illustrates the ecosystem of embedded wireless sensor network development around the Berkeley Motes before mainstream platforms were available off-the-shelf, as they are today.

The seeds of the IoT can be found in DARPA's Distributed Sensor Networks program of the late '70s and Aloha Net, but really in the DARPA sensIT program in 1999. Also DARPA's Expeditions program sowed seeds of pervasive computing, adding networked sensors, cameras and displays into our physical spaces, pushing storage and computation into what we now call the cloud, and putting connected personal devices into people's hands.

The early development of wireless sensor networks comprised two complementary schools of thought. The "SoCal school" focused on compact versions of desktops (e.g., PC104) and laptops (e.g., winCE) with the newly emerged 802.11 for communication, expecting that, with time, these would be reduced into small, ubiquitous devices. The "NorCal school" focused on the microcontroller (MCU), radio, and flash technology that was already ubiquitous in game controllers, key fobs, etc., and down inside 802.11 cards to create small wireless embedded devices that brought into sharp focus the challenges of constrained resources, large numbers of unattended devices, micro-sensors, and the uncertainties of being embedded in the harsh, noisy physical world. The latter school gave rise to a novel open-source hardware event – the Rene' mote – designed and implemented by Berkeley, funded by DARPA, and then manufactured and sold by Crossbow and others to the research community and industry. Constraints were severe –8 kB of program ROM, 512 B of RAM, and a 19.2 kbps narrow-band, on-off-key radio requiring physical layer modulation in software – which drove the design of an operating system that could handle many concurrent radio, sensor, and application processing events in a tiny footprint, TinyOS. Conventional IP routing was neither possible nor desirable, as these embedded networks were expected to operate as an intelligent ensemble, not merely a transport amongst hosts.

Both these schools framed the problem that fundamentally shapes the IoT: *doing nothing well*. Estrin's group, while working on novel routing protocols for sensor networks, observed that when the communication rate is low, energy consumption is dominated not by communication, but by the cost of just listening. Low-power radios consume almost as much power when listening, as when they are actually transmitting or receiving. But, reception can occur only if the receiver is powered on, listening, during the transmission. How does the receiver know to turn the radio on just when there is something to hear?

Commercial hardware modules failed to do nothing well in numerous ways. The hallmark of the mote design was utilizing an MCU with a very low sleep current, essentially that of battery leakage, with an external clock and logic to reawaken it. Design care was required to transition rapidly out of sleep into action. This was carried throughout the operating system, so the entire platform could drop into deep sleep even between bits of a packet or samples of a sensor and wake up to handle the next event.

The DARPA Network Embedded Systems Technology program (2002) leveraged this momentum and integrated research teams throughout the country

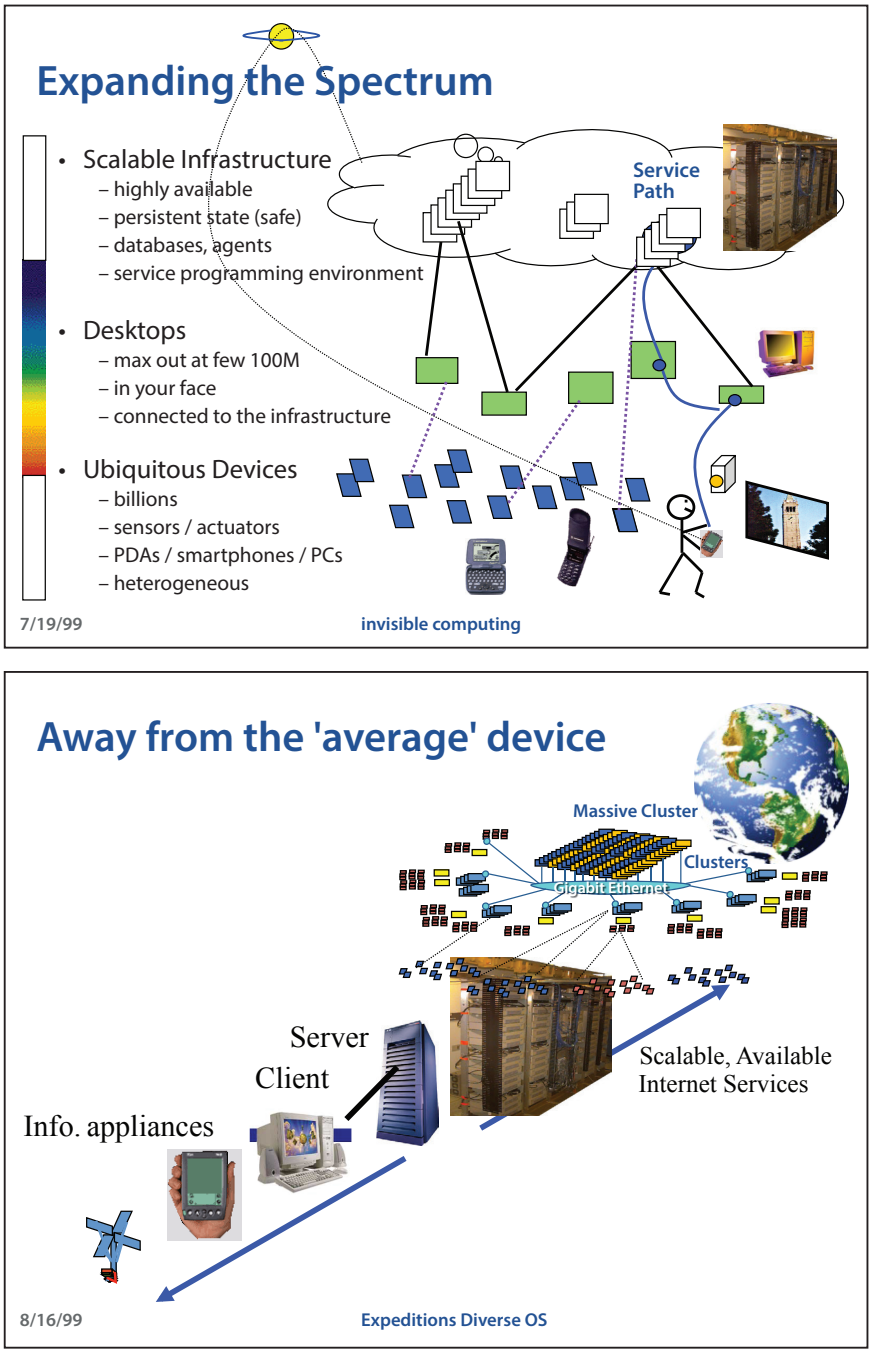

FIGURE 1. Articulations of an Internet spanning from tiny embedded wireless devices to planetary scale services in 1999.

around a sequence of common platforms to develop networking, sensor processing and application tiers with very large-scale field demonstrations, creating the rich ecosystem indicated in Figure 2.

Bluetooth arose from work within Ericsson at the beginning of this development with an impressive physical layer, but protocol restrictions that made it only useful as a peripheral interconnect, not a network link. It would be a decade before calls from the research community for a promiscuous beacon would be answered with the introduction of Bluetooth Low Energy (BLE) – and then only in a very limited manner. Instead, the watershed was the introduction of IEEE 802.15.4 in 2004. Motes were designed for it before chips shipped. Along with the appearance of the TI MSP430 MCU, the capabilities were finally in place to create capable, low-power, wireless networks of just about any smart thing.

Today the IoT remains behind the research platforms of a decade ago. Arduino's introduction brought simplicity and a wonderful groundswell of activity, but completely ignored energy and reliability, while treating communication as an add-on. Structured event-driven processing was reduced to a single acquire-and-act loop. The challenge in doing nothing well depends on getting every little thing right; while recent hardware platforms in the Arduino family have the potential to get there, it will take time and effort to reorient system and application software. Our smartphones do nothing somewhat better, but they do so by relying very heavily on the interaction with the person to do so. Only limited continuous sensing or communication actions can happen in the background. Instead, the illusion of continuity is (partially) provided by piggybacking on the activation of the entire system at the point of user interaction to take samples. And their human will dutifully plug them in at nightfall. Some wearables and BLE tags do achieve low-power continuous operation, but only as peripherals to a master device. They are not a network of things, much less an internetwork of things, but rather a wireless USB of things interacting with and extending a phone. With the incredible cost-reduction of WiFi and complete host computers, such as Raspberry PI, the Internet does extend to things, but only to things with electrical plugs – or people to continually provide for their power needs.

LOW-POWER WIRELESS LINKS

Armed with devices that could rapidly transition from under 10 uW idle with the radio off to above 10 mW with the radio on in a setting where communicating 0.1% of the time is typical, the research community set about to tackle the idle listening problem in a manner that would permit general purpose networking over links that are almost always off. The 802.15.4 MAC defined a complex slotted power management protocol from its origins within Motorola Labs, but it was ill-suited to networking. Across a huge collection of studies, three broad solutions emerged.

- *Scheduled listening*, drawing from TDMA techniques, seeks to arrange in advance a way for each potential transmitter to determine when its desired receiver will be listening. Potential receivers listen in their scheduled slots just in case there is something to hear.
- *Sampled listening*, drawing from CSMA techniques, has receivers turn the radio on for a tiny bit and take a sample to tell if something might be trying to transmit to them. The transmitter extends each, albeit infrequent, transmission to cover the sampling period and allow the receiver to wake up and get the message [Pol*04].
- *Listen-after-send*, drawn from many ad hoc scenarios, requires an always-on node within reach of every low-power node; a low-power node wakes up, sends a packet and then continues to listen to pick up anything queued up for it.

Eventually the 802.15.4 committees reconsidered the standard in light of these solutions and produced 802.15.4e in 2012, which incorporated all three techniques with a big switch to select among them. This might have been an opportunity for the research community to reengage and develop a unified low-power MAC integrating the three techniques, but unfortunately that window closed without much attention. Interestingly, sampled listening is incorporated in Energy Efficient Ethernet, allowing switches to power off rarely used ports while sampling if the "thing" on the other end is trying to transmit. Oddly, almost no progress has been made in this direction with WiFi, despite numerous generations of 802.11 to provide greater bandwidth or coverage, plus its predominant use in smartphones, personal devices, and things at diminishing cost. BLE retains Bluetooth's original scheduled channel hopping and power management, with a single central device (the phone) managing a small collection of directly connected peripherals (things). But a thing can emit a tiny beacon, in case something might be listening. 802.15.4 (LoWPAN) has not achieved the ubiquity we associate with the billions of connected things.

ROUTING WHEN THERE IS NO GRAPH

The classic formulation of routing – given a graph of router nodes and links describing which routers can communicate directly, find a path from a source node to a destination – breaks down for wireless networks spread over a region of physical space, because there is really no *a priori* graph. Devices may serve as routers, and to determine what neighbor links exist, they try to

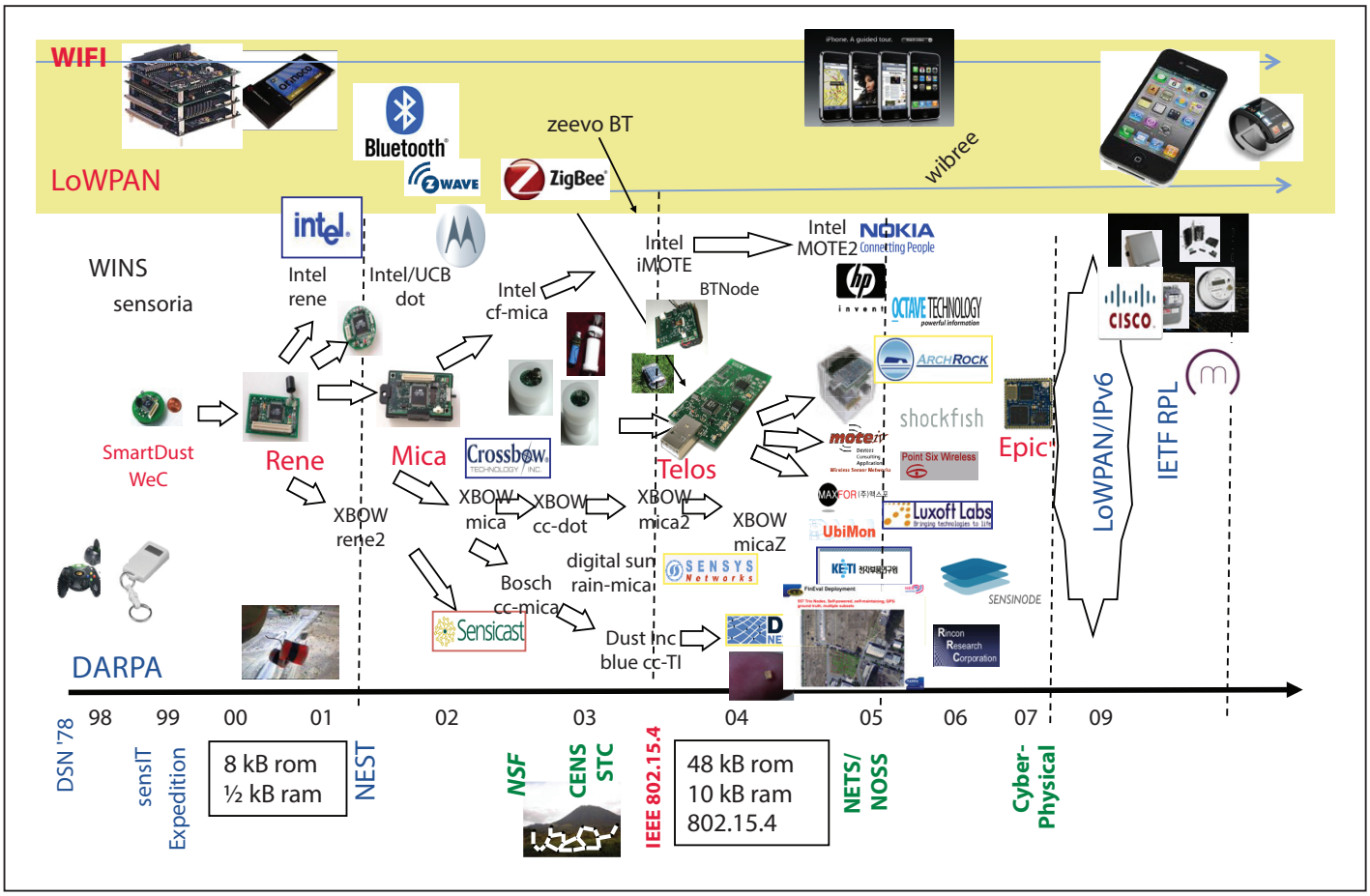

FIGURE 2. Mote/TinyOS and related developments relative to research programs.

communicate and see what happens. At any time, the presence of a link depends not just on the distance between nodes, but antenna orientation, noise due to other devices and electrical appliances, interference due to other transmitters, signal attenuation due to humidity and other phenomenon, occlusion due to obstacles, and (fundamentally) whether the receiver is listening. Thus, even if the devices are all stationary, the topology of links is continuously changing and the only way of determining whether a path is present is to try it.

Many early WSN protocols, like many MANET protocols, sought to discover paths by flooding and tree reversal - a node broadcasts, some nodes receive it, they re-broadcast, and the process grows as a tree flooding over the network. These techniques generally fail to build robust topologies for a host of reasons. When a node transmits it is likely that some node far away will get lucky and receive the packet, but it is unlikely that the reverse direction will be useful, or even repetition of the forward direction. Worse, flooding prefers such unreliable links, while contention during the flood may hide reliable ones. Industry standards, including Zigbee, followed this approach, producing multiple protocol revisions that were deprecated before reaching adoption. Zigbee also created substantial confusion because it served both as a marketing organization for IEEE 802.15.4 (like the WiFi consortium does for IEEE 802.11) and as a protocol design and standardization body, while drawing little from the open research around it and without the rigorous empirical methodologies of the open literature.

Robust protocols construct a good approximation of the graph by having each node continually obtain statistics on the quality its neighbor links. But, this presents many challenges. Recording link history requires an entry in a neighbor table. With very limited memory, this table may have to be quite small, while the number of neighbors (the local density) might be large. The placement of nodes, and implicitly the degree of the graph, is dictated by where sensing or actuating needs to occur, not communication convenience. A node is likely to pick up spurious packets from far away, but when a packet is received from a potential neighbor that is not in the table, there is no way to know if it is a good one, because no history is recorded. To learn more about the potential neighbor it must be added to the table, which may evict an existing entry. The most important ones to keep track of are those that are important for routing, so the network protocol layer needs to guide the link layer and vice versa. As environmental conditions change or nodes move, these structures must detect the changes and adapt. The cost of communication for maintaining these structures needs to be small compared to the application-level communication. Plus, all the nodes have their radios turned off almost all the time, so one cannot assume the radio is a broadcast medium.

Thus, topology formation and route determination need to work in concert with the power management. Based on the discovered topology, various routing algorithms, e.g., distance vector, can be realized. Eschewing the traditional separation of hardware, link, network, transport, and application layers, the research community formulated several elegant solutions to this knotty collection of inter-related challenges to achieve reliable routing over low-power and lossy links.

A unique aspect of this work is the inclusion of routing diversity to overcome the intermittent nature of links and the slightly stale routing state that is inevitable in low-power wireless networks. Reliability in communication is achieved through a combination of diversity and redundancy. For example, frequency diversity is utilized at the physical layer by spreading information over a range of frequencies, either through wide-band channels or channel hoping, with coding redundancy used to recover gaps that may occur at some frequencies. Temporal diversity is utilized at the link layer through retransmission of lost packets. But, in embedded networks links come and go even with these techniques, say, when people get in the way, when the fog rolls in, or when more powerful wireless devices hog the air. Laptops and smartphones utilize spatial diversity to address this problem, because the frustrated human who is so attached to them moves to some place where coverage is better. But embedded networks typically have many unattended devices, i.e., things, without the ready human to solve their communication problem. Indeed, a common failure mode of current IoT devices using BLE or WiFi is losing connectivity while the human is away, to be frustrated upon return.

The presence of many nodes forming a network spread over a physical area naturally creates spatial diversity. When one path is blocked, another might be usable. Traditional Internet routing protocols recognize that links and routers come and go and provide a means of detection and repair, but assume that these occurrences are rare (compared to application traffic) and routing tables must be kept up to date and consistent. For networks embedded in the physical environment, especially with low-power communication, neither assumption holds. Links routinely come and go, and keeping the routing state consistent with these changes may require a great deal of communication. This is yet another instance of doing nothing well. If a link disappears and returns between uses, no one needs to know. If it is being used and is working, it need not be tested. As it is not possible to know if a neighbor link is still there since last probed, it is better to maintain a few

WHY BEMOAN HOW THE IoT TODAY HAS FALLEN SO SHORT OF THE POTENTIAL LAID IN PLACE BY THE BROAD RESEARCH THAT LED UP TO IT? BECAUSE IT IS POISED FOR A RENAISSANCE OF DEEP EXPLORATION

options and sort out which one to use on demand than to try to keep close track. Basically, each router maintains a couple of candidates for the next hop and, then, picks a different alternative for the next node if there is a link level failure on the previous attempt. The best way to learn about links is to use them; so traffic is spread across the candidates. Exploration and local repair are a part of every communication, rather than an exception.

The further challenge is that routing table state is also at a premium. The complexity of a routing protocol is represented by the amount of extra-application communication to maintain these tables and the size of them, as a function of the size, diameter, and density of the network. When both communication and storage are precious, scalability in any dimension cannot be taken lightly. The research community developed elegant means of addressing these trade-offs in the context of the organizational structures and communication patterns that predominate the use of this class of networks. They are almost always edge networks, with little or no through-traffic, so their points of connection to the powered Internet, i.e., border routers, define a natural orientation. The routing topology can then be viewed as a directed-acyclic graph (DAG) – not just a tree – relative to these points of connection. Routing complexity, in both state and communication, can then be traded for stretch in path length, neighbor tables can be safely bounded, and loop detection and mitigation is vastly simplified.

TRICKLE, DON'T FLOOD

Many of the distributed algorithms underlying the IoT involve network-wide dissemination of information to establish some consistency property, say broadcasting a value, collecting readings, updating code, or forming a spanning topology. Much of the early research and many industrial solutions sought to achieve this by flooding. A unique algorithmic contribution of the research community was replacing the flood with a trickle [Lev*04]. Wireless things forming a network need to be polite, always listen before they speak, and, if they have nothing new to add, remain quiet. This is yet another dimension of doing nothing well. The trickle algorithm recognizes that while communication capacity is finite in any region of space, the density of devices can vary dramatically. Through listening, devices can estimate the local density of the network and adjust their communication rate as the reciprocal of density. Thus, the transmission rate becomes roughly constant over space, despite variations in device density. However, when the radio is functioning as a broadcast medium, the rate of reception is proportional to density. Devices either transmit or listen at some rate, and the estimated density determines whether they transmit or just listen. The rate is determined by whether there is new information to share. When the consistency property appears to have been obtained, say, there is no new information to disseminate, the rate of announcing it decreases, approaching zero if there is nothing new anywhere. The detection of some change causes the rate to be increased so the new information can be rapidly disseminated and then return to a quiescent state.

This technique eliminates many of the "voodoo" parameters in protocols, such as how often should advertisements and solicitations be initiated. It is adaptive in both space and time. When new information is introduced, it allows a flurry of activity to propagate. Where lots of devices are close together, few need to speak up, most can just listen. Then it all quiets down.

Photo, istockphoto.com

FROM HERE

In 2008 it was clear that all these techniques could be brought together within the classical Internet architecture, especially with the advances of IPv6 and with few extra provisions. The Internet of Things was at hand, the Routing Over Low-Power and Lossy networks (ROLL) working group formed and the RPL (Routing Protocol for Low Power and Lossy Networks) protocol was eventually approved [RPL]. Success was at hand. 6LoWPAN solved the problem of huge IPv6 headers in small packets through context-based header compression. Routing diversity, local repair, and trickle were incorporated into the core of the design. So why did it fail? Why is RPL so absent from networked things? Why are wireless things still either just peripherals to a host, rather than networked entities, or simply WiFi-based host computers in a new form factor?

The answers to questions such as these are complex and full of nuance. Partly, as

the effort moved into the standards arena, the research community stepped aside. The clarity and rigor of the open technical literature that was so key in developing solutions to the challenges described here were no longer brought to bear on the challenges that arose in finishing the job, which would have brought the fruits of the research into adoption. Numerous questions arose in the working group for which the research was simply non-existent, such as properties of piecewise source routing, backtracking on route failure, stretch detection and mitigation, and so on. Within the IETF process, there was little appetite for serious scalability and complexity assessment amongst protocol candidates, each of which had supporting constituencies [Lev*09]. Consultants and industry representatives with interests in the outcome besides producing a technically superior solution tended to dominate the process. Typically, this translated to advocation for features with tenuous connections to business cases, rather than clarity or simplicity, and without the competitive assessment process, there was little basis for elimination. Level of participation in the process trumped technical justification.

One example is routing metrics, the reasonable candidates of which had been extensively studied. RPL defines an entire infrastructure for definition and use of application-specific routing metrics, completely missing that each distinct metric would mean complete replication of precious routing state. Not only were they never utilized, to my knowledge, no implementation supports more than the defaults. Another is decomposing the natural DAG into a forest of DAGs, each with a single root node, termed a direction-oriented DAG (DODAG). This represents only partially overcoming the misunderstandings of the naïve tree advocates. To implement it would involve nodes maintaining distinct neighbor and routing table state for each of the DODAGs their potential parents are a part of. As a result, implementations only support networks with a single border router serving as the root – a large step back from the solutions in the prior research. These and other "contributions" of the process resulted in bloat and complexity that undermined adoption.

Recently we see a refreshing turn in the introduction of Open Thread (https://github.com/openthread), which draws on simpler, established, techniques. Sadly, this development took place entirely within the opaque world of companies participating in the consortium, out of view of the critical eye of research and drawing little interest from the community. It takes a "low-hanging fruit" approach, completely separating issues of wireless routing (in a powered backbone) from low power operation (in the analog of host devices), thereby sidestepping the hardest challenges that shaped the research, but also raising deployment challenges.

Why bemoan how the IoT today has fallen so short of the potential laid in place by the broad research that led up to it? Because it is poised for a renaissance of deep exploration, just when much of the current research activity is focused on relatively shallow innovations, taking the available product landscape as a foundation. The reopening of networking in Open Thread is but one example. In the past six months the technology became available that make the Mote that we wished we could build throughout the past 20 years actually buildable for less than $10. Price is much more of an enabler than size ever was. And this is a real system with capable compilers that allow programming systems to flourish, with the extremely low idle power that permits unattended use. Programming the ensemble, rather than the individual devices, was never solved, and now they function in a rich ecosystem with non-trivial behaviors. Protection mechanisms and storage capacity within the device put classic operating systems techniques back on the table, while begging the question of whether we ever really move past systems that look like Unix with all its TTYs. A huge swath of security and privacy issues remain completely open, while new approaches, such as attribute, rather than identity based, authorization and computing on encrypted data, are at the fore. New, rich relationships between embedded devices, local tiers of powerful middle-boxes, and global reach of the cloud go far beyond offloading computation, recognizing the particular role of physical premises in privacy and in distributing components of learning and inference across the tiers. Clever interplay between link and physical layers allows WiFi to enable backscatter communication for passive tags with potential that RFID could never reach. Even localization is being transformed with millisecond-scale sampling of GPS, when combined with IP communication of approximate time and place and post-computation. And all of these technological opportunities are taking place in rich context-sensitive application scenarios, offering the potential to improve the sustainability of the built environment, transform health care, and enhance productivity. The visions that opened the research area nearly 20 years ago of the potential of physical information in a connected world to enrich our lives are indeed at our fingertips. It is a time for the research community to engage, rather than settle for the half measures that today populate the commercial IoT landscape. ■

David Culler is the Freisen Professor of EECS and Faculty Director of CITRIS Sustainable Infrastructures at the University of California, Berkeley. He received a M.S. and Ph.D. from MIT in 1985 and 1989. He is a member of the National Academy of Engineering, and received the 2013 Okawa Prize, ACMs Sigmod Outstanding Achievement Award, and Test-of-Time awards from Sensys, Usenix, NSDI, SIGCOMM, PLDI, HPDC, and ISCA. He was PI of the DARPA Network Embedded Systems Technology project that created the open platform for wireless sensor networks based on TinyOS.

REFERENCES

[HuCu08] IP is dead, long live IP for wireless sensor networks, Jonathan Hui and David, Culler, 6th ACM Conference on Embedded Network Sensor Systems (Raleigh, NC, USA, November 05 - 07, 2008). SenSys '08. ACM, New York, NY.

[Pol*04] Versatile Low Power Media Access for Wireless Sensor Networks, Joe Polastre, Jason Hill and David Culler, Second ACM Conference on Embedded Networked Sensor Systems, SenSys 2004: 95-107, Nov. 2004.

[Lev*04] Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks, Philip Levis, Neil Patel, David Culler, and Scott Shenker, First Symposium on Network Systems Design and Implementation, NSDI 2004, Mar. 2004.

[RPL] RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, IETF RFC 6550, ROLL WG (https://tools.ietf.org/html/rfc6550) Mar. 2012.

[Lev*09] Overview of Existing Routing Protocols for Low Power and Lossy Networks, IETF Expired drat (https://tools.ietf.org/html/draft-ietf-roll-protocols-survey-07) Apr. 2009.