Energy and Quality Scalable Wireless Communication

by

Rex Kee Min

B.S. in Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 1998

M.Eng in Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 1999

Submitted to the Department of Electrical Engineering
and Computer Science in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 21, 2003

Author .....
Department of Electrical Engineering and Computer Science
May 21, 2003

Certified by .....
Anantha P. Chandrakasan
Associate Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Professor of Electrical Engineering and Computer Science
Chairman, Department Committee on Graduate Students

Energy and Quality Scalable Wireless Communication

by

Rex K. Min

Submitted to the Department of Electrical Engineering and Computer Science on May 21, 2003, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and Computer Science.

Abstract

Nodes for emerging, high-density wireless networks will face the dual challenges of continuous, multi-year operation under diverse and challenging operating conditions. The wireless communication subsystem, a substantial consumer of energy, must therefore be designed with unprecedented energy efficiency. To meet this challenge, inefficiencies once overlooked must be addressed, and the system must be designed for energy scalability—the use of graceful energy vs. quality trade-offs in response to continuous variations in operational conditions.

Using a comprehensive model framework that unifies multi-disciplinary models for energy consumption and communication performance, this work explores multi-dimensional trade-offs of energy and quality for wireless communication at all levels of the system hierarchy. The hardware "knob" of dynamic voltage scaling is implemented on a commercial microprocessor and integrated into a power aware, prototype microsensor node. Power aware abstractions encourage collaboration between the hardware, which fundamentally consumes the energy, and software, which determines how the hardware acts. Accurate models of hardware energy consumption reveal inefficiencies of routing techniques such as multihop, and the models are fused with information-theoretic limits on code performance to bound the energy scalability of the hardware platform. An application-specific protocol for microsensor networks is evaluated with a new, interactive Java simulation tool created expressly for energy-conscious, high density wireless networks. Close collaboration between software and hardware layers, and across the research disciplines that compose wireless communication itself, are crucial enablers for energy-efficient wireless communication.

Thesis Supervisor: Anantha Chandrakasan
Title: Associate Professor of EECS

This page intentionally left blank.

Acknowledgments

My first and foremost thanks to Prof. Anantha Chandrakasan, who is the primary reason I stayed at MIT instead of escaping to the balmier west coast. Words cannot describe my gratitude for his numerous interesting research ideas (several of which formed this thesis) and opportunities to present them in print and in travels. I thank him for pushing me to achieve higher standards and for his unconditional support through my successes, setbacks, and changes of research direction.

I wish to thank my readers, Prof. Muriel Médard and Dr. Chris Terman, for agreeing to supervise my work and tolerating the last-minute schedule. Prof. Medard is the inspiration behind the integration of coding theory and energy scalability, which led to the integration of theory and simulation present in the Java simulation tool. The tool itself owes much to Dr. Terman, who has offered insights into optimization of the code and steered the tool in the directions of education and intuition-building, rather than mere simulation. His influence is reflected in the tool's graphical interface and interactive capabilities. I sought a diverse committee who could evaluate my work from different technical perspectives, and I sincerely thank Prof. Medard and Dr. Terman for accepting this task and offering enthusiastic support throughout the process.

On the technical front, sincere thanks goes to Kevin Atkinson, who roomed and worked with me for two weeks at HP Labs in Palo Alto and was a bright and personable research partner. I gratefully acknowledge Mark Smith, Susie Wee, John Ankcorn, Mitch Trott, and John Apostolopoulos of HP Labs, who provided us the opportunity to work with energy-scalable 802.11b at their Palo Alto campus and provided valuable feedback throughout our two-week residency. Travis Furrer skillfully wrote the embedded software that made the dynamic voltage scaling results possible. Nidhi Sharma contributed a great deal of code to the simulator, most notably the user interface. For their technically insightful discussions and suggestions on all manners of topics, I thank Deborah Estrin and Mani Srivastava of UCLA, Jim Reich of PARC, David Brown of MIT LIDS, and Twan Basten of TUE in the Netherlands.

On the less technical front, equally sincere thanks go to Belinda Escanio, Jason Strautman, and Saul Blumenthal, three personal friends who have provided companionship and counsel through the most joyful and most stressful periods of graduate life.

I can not say enough about the friendliness and aptitude of my fellow students in AnanthaGroup. I'm thankful to have been given the chance to be a part of it. I offer each of you my heartfelt thanks: Margaret Flaherty, our dependable, honest, good-natured, dependable, conscientious (did I mention dependable?) administrative assistant, who provided invaluable support for travel, purchasing, and communications; Nathan Ickes, an engineer's engineer and the quickest dry wit this side of the Charles; Manish Bhardwaj, a wonderfully obsessive thinker who has shaped several foundations of my personal philosophy; Amit Sinha, whose work inspired me to pursue a thesis that spans hardware and software; Alexandra Kern, who wins the hard-earned title of Friendliest Person in the Office; Johnna "Dawn" Powell, a seamless fusion of ambition and humility; Julia Cline, our Ph.D. of social intelligence; Curt Schurgers, a dependable and insightful tutor for all aspects of wireless communication; David Wentzloff, who *never* has a bad day and thus made a terrific cube-mate; Ben Calhoun, a reminder that family is, and always will be, the most valuable of all priceless things; Theodoros Konstantakopoulos, a great coalition-builder and entertainer; Frank Honoré, who reminds me to live well and live happily; Fred Lee, whose proud personal convictions earn my admiration; Raul Blasquez-Fernandez, a mild-mannered gentleman on the outside, a torrent of mathematical talent on the inside; Puneet Newaskar, a dedicated workhorse who is immune to 20-hour days; Alice Wang, a versatile technical leader and a repository of all-around wisdom; Eugene Shih, a disciplined, philosophically-minded researcher and a great housemate to boot; Seong-Hwan Cho, the first Ananthagroup Korean and yet another of the group's signature soft-spoken geniuses. Together, they have not only provided me with all manners of technical assistance, but they have given me a chance to experience the best in human nature.

The institutions that provided funding for myself and my research are the NDSEG Fellowship program, DARPA's Power Aware Computing and Communication (PAC/C) program, the Air Force Research Lab, the Army Research Lab Collaborative Technology Alliance through BAE Systems. Their support is sincerely appreciated. I would briefly like to cite the three non-academic institutions who provided the greatest positive impact on

my quality of life during my graduate years at MIT: Apple Computer, Amtrak, and the Boston Symphony Orchestra.

And finally, I save my deepest gratitude for my father and mother, Byung and Soon Min. The greatest luck I've received so far is to have been born to, and raised by, these people. Their guidance, support, love, vision, and values are the foundation that enable every accomplishment of mine. And before I forget, thanks Mom and Dad, for bringing a computer into our modest apartment in 1985. The more things change...

About the Author

Rex Min received the S.B. and MEng. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (MIT) in 1998 and 1999 respectively. During his undergraduate years at MIT, he won design competitions in 6.004 (computation structures) and 6.111 (digital systems laboratory) and was Chairman of the MIT Lecture Series Committee. In 1998 and 1999 he served as a teaching assistant for 6.312 (Acoustics) under Prof. Amar Bose. As a graduate student, he has co-authored thirteen IEEE and ACM publications, and three chapters for books and edited volumes. In September 2003 he joins the Department of Defense to pursue research in information systems security for mobile and wireless systems. He is a member of Eta Kappa Nu, Sigma Xi, and Tau Beta Pi, and is a National Defense Science and Engineering Graduate (NDSEG) Fellow.

This page intentionally left blank.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Scope

Ranger Smith enters the station at the edge of his forest preserve, triggering a flurry of activity from the thousands of climate microsensors in the building. Having kept the building a trifle warm in his absence, the nano-nodes open innumerable vents to condition the hallway as he walks through. His demand to see the thermal profile of the reserve ripples across roughly 100 square miles of wilderness, waking up nearly a quarter of the million sensors that were air-dropped a few years back. "Paint" on the wall receives and interpolates the data, rendering a visual picture through actuating nodes. Bad news—the thermal map indicates hot spots, and a million-node cluster is not responding. If only nano-nodes could put out fires...[1]

1.1 Energy Demands for Emerging Wireless Networks

While paintable "smart dust" [101] lies well within the realm of science fiction, networks of hundreds to thousands of pill-sized, ubiquitous computational elements are rapidly approaching our grasp. Many of the necessary components and technologies are already available. Microscopic MEMS motion sensors are routinely fabricated on silicon. Digital circuits shrink in area constantly; a complete data processing unit can fit on a pinhead. Entire radio transceivers—including the associated digital electronics—have been fabricated on a single chip [68]. Refinements of these enabling technologies will soon yield the form factors practical for networks of truly tiny wireless devices, such as the microsensor networks discussed above.

Small, densely placed nodes harbor serious energy consumption considerations. Small nodes imply limited physical space for batteries, and high density implies that periodic battery replacement will be tremendously inconvenient—and more likely, impossible. A state-of-the-art lithium primary battery offers an energy density of about 2 kiloJoules per cm^3 [30]. Assuming that 1 cm^3 is available within the node for the battery, the resulting battery capacity for the node is 2 kJ. Put in perspective, this amount of energy would allow for the transmission of 600,000 ten-kilobit packets (nine minutes of transmission) over a commercial 802.11b transceiver [29], the Viterbi decoding of 230,000 packets on a com-

1. This concept for this opening paragraph comes from Manish Bhardwaj; this particular paragraph is taken from [53], where it also serves as an introduction.

mercial low-power processor [87], or 20 seconds of illumination from a 100 Watt light bulb.

While 2 kJ of energy seems fairly ample from the perspective of active communication, lifetime requirements provide a fundamental challenge. For a desired device lifetime of one year—fairly short for autonomous applications—the average power dissipation must be less than $(2000\text{J})\left(\frac{1\text{ year}}{365\text{ days}}\right)\left(\frac{1\text{ day}}{24\text{ hrs}}\right)\left(\frac{1\text{ hr}}{3600\text{ s}}\right) = 63.4\ \mu\text{W}$. As this value exceeds the standby power of most digital systems, energy dissipation is of paramount concern. Moreover, Moore's law simply does not apply to batteries: the energy density of batteries has only doubled every five to twenty years, depending on the particular chemistry, and prolonged refinement of any particular battery chemistry yields diminishing returns [76]. Energy conservation strategies are therefore essential for achieving the lifetimes necessary for viable applications.

1.2 Distributed Microsensor Networks

The distributed microsensor network [1,80], illustrated in Figure 1.1, is among the most

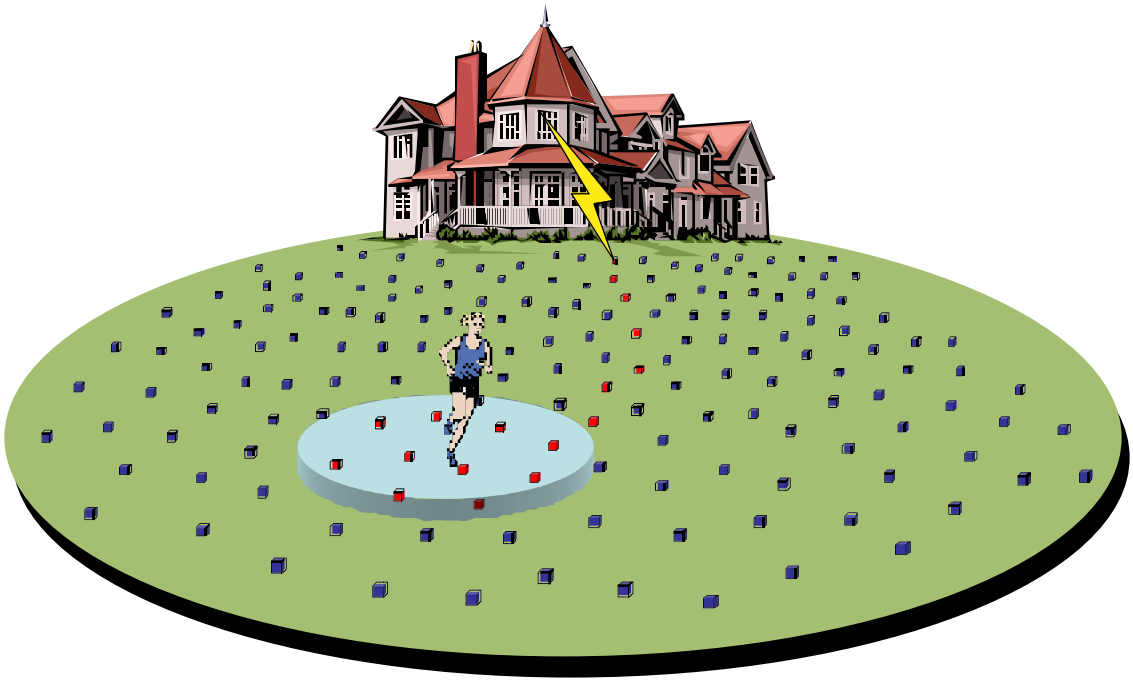


Figure 1.1: A distributed microsensor network consists of hundreds to thousands of nodes that cooperatively gather environmental observations (footsteps of the jogger) and forward them to a remote base station (within the house).

anticipated applications of ultra-high-density wireless networking. Microsensors promise to revolutionize spatial data gathering. Driven by data aggregation—the fusion of multiple observations from different perspectives—a spatially distributed network of nodes returns a rich, high-resolution, multi-dimensional picture of the environment that is not possible with single sensor or small group of sensors. The sheer number of nodes naturally leads to the network's robustness and fault-tolerance to the loss of individual nodes, making maintenance unnecessary. As the nodes self-organize into ad hoc networks, deployment can be as easy as sprinkling nodes about the region of interest or dropping them by air, and setting up a conveniently located base station to which the nodes will relay their observations. These advantages, and the nodes' diminutive size, make sensor networks ideal for any number of inhospitable or unreachable locations where deployment is difficult, wires impractical, and maintenance impossible. The cramped confines of an appliance, facilities that produce toxic radiation or chemical vapors, the lands of extreme desert or Arctic climates, and even the surface of foreign moons and planets, are excellent candidates for easily deployed, maintenance-free microsensor networks.

1.2.1 Microsensor Node Hardware

Figure 1.2 presents a generalized architecture for a microsensor node. Observations about the environment are gathered using some sensing subsystem consisting of sensors connected to an analog-to-digital (A/D) converter. Once enough data is collected, the pro-

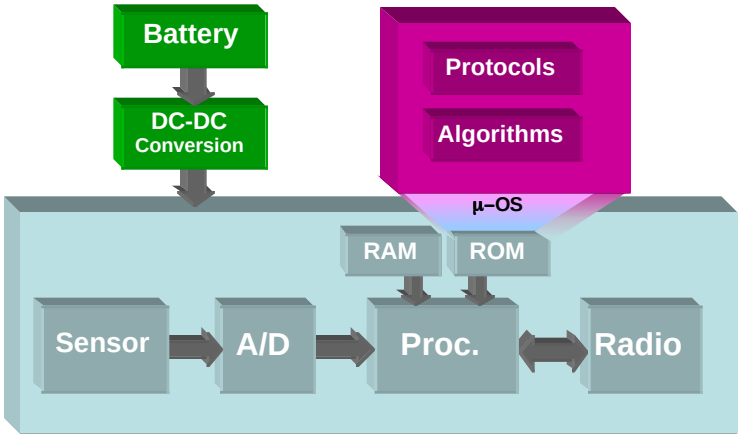


Figure 1.2: Block diagram of a microsensor node.

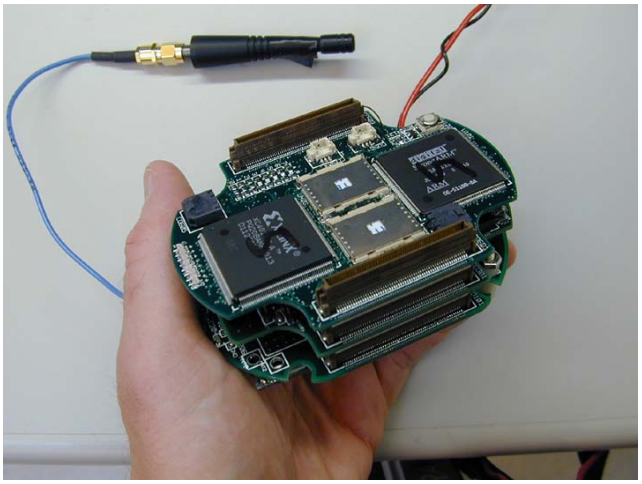


Figure 1.3: Berkeley PicoRadio testbed node [4].

cessing subsystem of the node can digitally process the data in preparation for relay to a nearby node (or faraway base station). Portions of the processing subsystem that are microprocessor-based would also include RAM and flash ROM for data and program storage, and a "μ-OS", an operating system with light memory and computational overhead. Code for the relevant data processing algorithms and communication protocols are stored in ROM. In order to deliver data or control messages to neighboring nodes, data is passed to the node's radio subsystem. Finally, power for the node is provided by the battery subsystem with DC-DC conversion to provide the voltages required by the aforementioned components.

1.2.2 Examples of Microsensor Nodes

The PicoRadio project [79] aims to create a low-cost, integrated radio with an energy dissipation of under 5 nJ per communicated bit and an average power dissipation of 100 μW. PicoNodes, consisting of this radio integrated with digital processing circuits on a single die, will form self-configuring sensor networks. These aggressive energy and power targets are to be achieved through system-on-chip integration and event-driven operation. Figure 1.3 illustrates a prototype PicoRadio testbed node developed with off-the-shelf components.

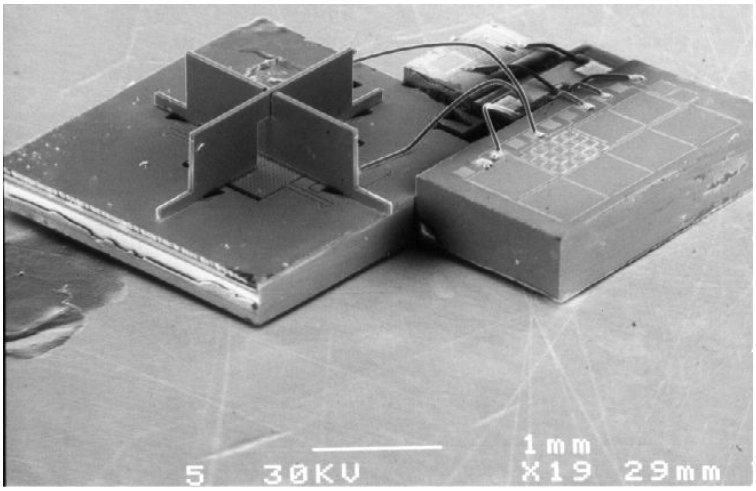


Figure 1.4: Micrograph of Smart Dust "mote" featuring optical communication [72]. Total node volume is 4.8 mm^3.

The Smart Dust project sacrifices some degree of functionality for small size. Smart dust *motes* (many, tiny nodes), easily the smallest microsensor prototypes in development, are targeted for ubiquitous applications. Particularly noteworthy is the choice of optical communication using lasers and micromirrors. These optical communication structures require substantially less energy and physical space than radio communication due to what is effectively a high antenna gain from the focused laser beam. While optical communication systems can consume well under 1 nJ of energy—over two orders of magnitude lower the RF-based systems—these systems are restricted to line-of-sight communication and require that the laser beam be aimed at data recipients [44]. The prototype "Golem Dust" mote illustrated in Figure 1.4 senses light and acceleration, and communicates by reflecting an incident laser from a basestation.

The Adaptive Multidomain Power-Aware Sensors (μAMPS) project at MIT is designing a series of microsensor nodes that highlight power aware system design techniques. Much of this thesis is based on, and will contribute to, the efforts of μAMPS. While the eventual goal of μAMPS is a sensing system-on-chip, the first prototypes have been built with off-the-shelf components for rapid demonstrations of power aware design techniques. The μAMPS-1 node, a recent prototype node for the μAMPS project, is composed of an acoustic sensor, StrongARM SA-1110 processor, low-power static and non-volatile mem-

ory, and a 2.4 GHz transceiver [87]. Many communication energy models utilized in this dissertation are derived from measurements of the μAMPS-1 node.

1.2.3 Challenging Operational Characteristics

The microsensor network is a domain with operational demands unlike any current paradigm in wireless communication. Consider a network specification for a machine monitoring application [100]. This application scenario specifies up to 12 nodes per square meter and a maximum radio link of 10 meters. Nodes are expected to process about 20 two-byte radio transmissions per second and to operate for 5 to 10 years with an AA-size battery. Practically every specification is a departure from the norms of current wireless technology. As illustrated by Table 1.1, the microsensor node is the antithesis of high-

Table 1.1: Operational characteristics of microsensor networks.

	Wireless LAN	**Microsensors**
Battery Capacity	20 kJ - 2 MJ	2 J - 2 kJ
Transmission Range	10 m - 1000 m	1 m - 100 m
Node Density	Low	Very High
Data Rate	1 - 54 Mbps	1 - 54 Kbps
Tolerable Packet Loss	Low	High
Node Movement	Moderate	None

bandwidth or long-range communication: node densities are higher, transmissions shorter, and data rates lower, than any previous wireless system. The energy of radio transmission, normally the largest burden of the system, will likely be dwarfed by other components. As a low data rate, low duty cycle device, the node will be subject to unprecedented operational diversity, characterized by long idle periods interspersed with brief spurts of activity and communication.

1.3 Sources of Communication Inefficiency

The challenging operational demands and energy constraints posed by the microsensor application can expose energy inefficiencies that were previously not of concern. In particular, the non-ideal behaviors of a real wireless system cause the actual performance of communication to differ substantially from idealized assumptions. Such inefficiencies arise from a number of sources at many levels of the system hierarchy; several prominent

examples are illustrated in Figure 1.5.

1.3.1 Environmental Inefficiencies

The environment into which wireless nodes are placed introduces non-idealities that ultimately reduce the energy-efficiency of communication. Many visions of microsensor networks assume nodes that are dropped by air or sprayed onto a surface. The lack of uniform spacing between nodes could increase the communication burden on nodes placed in sparse portions of the network.

Even if the distribution of nodes were uniform, spatial variations in the environmental would bring out similar inefficiencies. Uneven terrain, obstacles, and RF interference sources cause the communication channel to vary with location. These variations, coupled with real-world antenna propagation patterns, present the node a communication channel that varies with direction as well.

Uneven distributions in node density and channel characteristics present the nodes with operational diversity that can not be predicted prior to the nodes' deployment. As a result, communication systems are traditionally "over-engineered" to provide sufficient performance for a worst case scenario, at the expense of additional energy consumption.

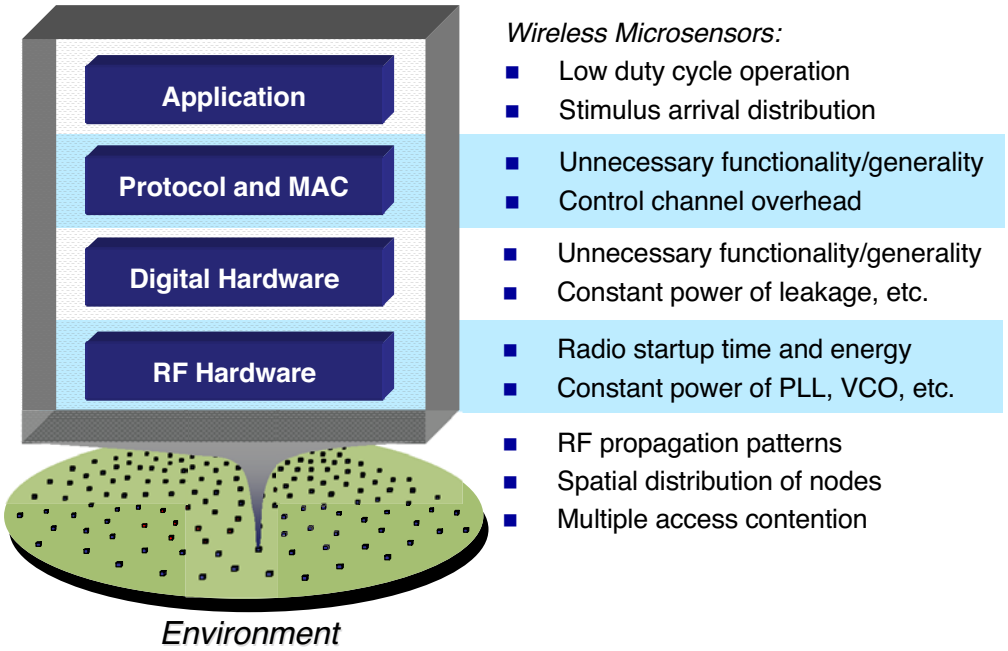


Figure 1.5: Sources of inefficiency in communication across the system hierarchy.

1.3.2 Digital Processing Energy

A node's digital processing circuits are typically used for digital signal processing of gathered data and for implementation of the protocol stack. Energy consumed by digital circuits consists of dynamic and static dissipation as follows:

$$E_{digital} = CV_{DD}^2 + tV_{DD}I_0 e^{\frac{V_{DD}}{nV_T}} \quad (1.1)$$

The dynamic energy term is CV_{DD}^2, with C representing the switched capacitance and V_{DD} the supply voltage. The dynamic energy of digital computation is the energy required to switch parasitic capacitors on an integrated circuit. The static dissipation term, originating from the undesirable leakage of current from power to ground at all times, is set by the thermal voltage V_T, and constants I_0 and n that can be measured for a given process technology.

Note that, for a constant supply voltage V_{DD}, switching energy for any given computation is independent of time while leakage energy is linear with time. While switching energy has historically exceeded leakage energy for modern CMOS applications [104], the trend is beginning to reverse with recent semiconductor process technologies. Each new process generation increases leakage threefold; leakage in advanced process technologies will soon approach 50% of a digital circuit's operating power [17].

Even in a process technology that is typically dominated by switching energy, a microsensor's long idle periods impose a low duty cycle on its processor, encouraging the dominance of leakage energy. Figure 1.6 illustrates this possibility with measured data from the StrongARM SA-1100 microprocessor [19]. Graph (a) shows that leakage energy begins to dominate over switching energy as the processor's duty cycle is reduced. Leakage energy is proportional with time, whether or not the processor is doing useful work. As a result, slowing down the clock actually increases the amount of leakage energy within each clock period, causing the energy per operation to increase. This is illustrated by graph (b). A reduction in clock frequency reduces the number of idle cycles, but leakage nonetheless remains proportional to time.

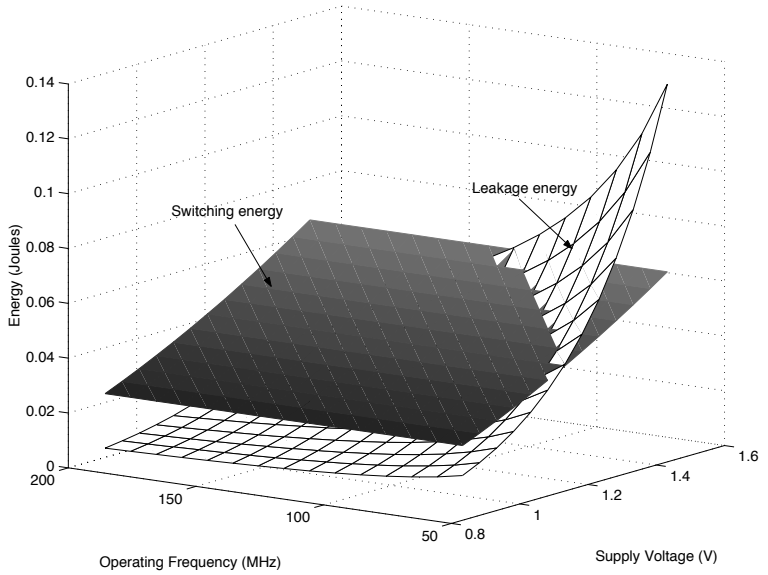


Figure 1.6: Comparison of leakage and switching energy in SA-1100 [89].

The easiest way to reduce leakage is to shut down the power supply to the idle components. For relatively simple circuits such as analog sensors, circuits can be powered up and down quickly, with no ill effect. Shutting down more complicated circuits, however, requires a time and energy overhead. For example, powering down a processor requires the preservation of its state. If the processor should be needed immediately after it has been powered down, the energy and time required to save and restore the state is wasted. In choosing any shutdown policy, the hidden time and energy cost to shutting down a circuit must be balanced with the expected duration of shutdown.

1.3.3 Radio Transceiver Energy

The issues of static power and the cost of shutdown, two key concerns for the node's digital circuits, emerge analogously in the node's radio. The energy consumption of the radio consists of static power dissipated by the analog electronics (analogous to leakage in the digital case, except that these bias currents serve to stabilize the radio), and the radiated RF energy. The radiated energy, which scales with transmitted distance as d^2 to d^4 depending on environmental conditions, has historically dominated radio energy. For closely packed microsensors, however, the radio electronics are of greater concern.

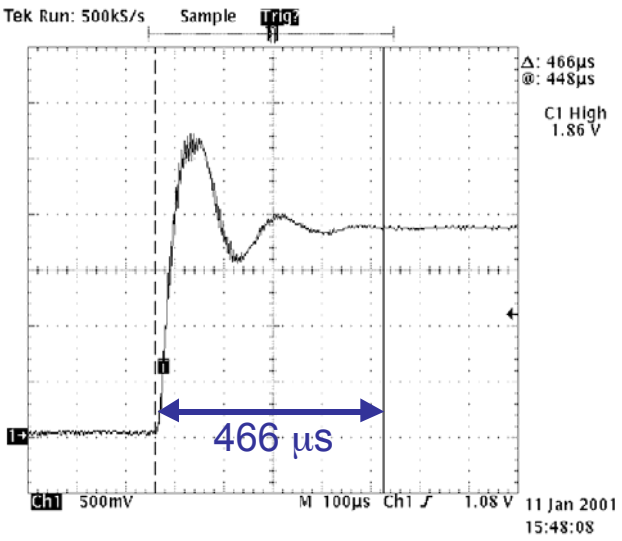


Figure 1.7: 466 µs startup transient from the µAMPS-1 2.4 GHz radio, measured at the input to the radio's VCO.

The average power consumption of a microsensor radio can be described by:

$$P_{radio} = N_{tx}[P_{tx}(T_{on-tx} + T_{st}) + P_{out}T_{on-tx}] + N_{rx}[P_{rx}(T_{on-tx} + T_{st})] \quad (1.2)$$

where $N_{tx/rx}$ is the average number of times per second that the transmitter/receiver is used, $P_{tx/rx}$ is the power consumption of the transmitter/receiver, P_{out} is the output transmit power, $T_{on\text{-}tx/rx}$ is the transmit/receive on-time (actual data transmission/reception time), and T_{st} is the startup time of the transceiver. $N_{tx/rx}$ will largely depend on the application scenario and the media-access control (MAC) protocol being used. Furthermore, $T_{on\text{-}tx/rx} = L/R$, where L is the packet size in bits and R is the radio's data rate in bits per second. The power amplifier is assumed to be on only when communication occurs.

The startup time T_{st} is a significant concern. During the startup time, no data can be sent or received by the transceiver, for the internal phase-locked loop (PLL) of the transceiver must be locked to the desired carrier frequency before data can be demodulated successfully. Figure 1.7 plots the measured startup transient of a commercial 2.4 GHz low power transceiver [60]. The control input to the voltage-controlled oscillator (in volts) is plotted versus time.

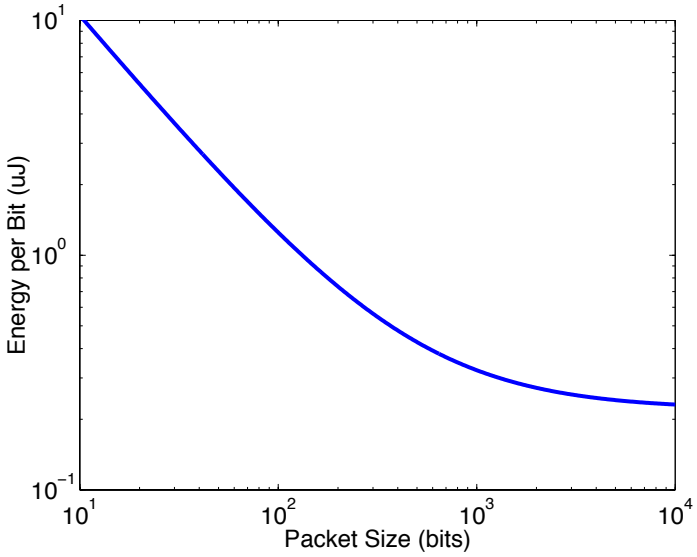


Figure 1.8: The total radio energy per bit required to transmit packets of various sizes. The radio has a data rate of 1 Mbps, startup time 466 μs, and the active transmitter electronics consume 220 mW.

The startup time can significantly impact the average energy consumption per bit E_b for systems that tend to communicate with very short packets, such as low-rate microsensor networks. Turning off the radio during idle periods is a natural and intuitive approach to reducing the radio's energy consumption. Unfortunately, when the radio is needed again, a large amount of energy is dissipated during the initial startup time of hundreds of microseconds. Figure 1.8 illustrates the effect of the startup transient for P_{tx} = 220 mW. Energy consumption per bit is plotted versus the packet size. For short packets, the fixed energy cost of radio startup is amortized over fewer bits, resulting in a far higher average energy per transmitted bit.

1.3.4 Protocol Overhead

Media access and routing protocols introduce additional overhead to wireless communication. Routing protocols typically require periodic control packets to establish or maintain protocol state across the network. Control packets typically implement channel reservation, route discovery and maintenance, authentication, and (somewhat ironically) power management. While these are often necessary and useful functions, each packet that does not carry application data contributes to protocol energy overhead. As an example,

the 802.11b media access protocol specifies four types of data frames and *21* types of management and control frames.

Moreover, any functionality that not useful at any given time is wasteful overhead. Many applications, such as multimedia streaming, do not require explicit guarantees of reliable end-to-end communication for all packets. If a protocol that implements reliable communication through persistent retries is used in conjunction with such applications, energy and bandwidth are wasted due to the retransmissions and associated control channel messages. As 802.11b and TCP/IP both implement persistent retransmissions, this type of overhead emerges frequently in practice.

Finally, the energy overhead of packet headers can sometimes be significant. Just as radio startup time becomes a significant overhead for short packets, a protocol with long packet headers produces an identical inefficiency.[1] The most extreme example possible with TCP/IP is the [fairly common] one-character Telnet packet; such a packet would carry a 4000% overhead relative to the amount of data actually carried [59].

1.4 Prior Art in Energy-Scalable Communication

The problem of energy-scalable communication has been approached from the viewpoints of hardware and software. Hardware-oriented research has identified "knobs" that may be varied to induce trade-offs of energy *vs.* performance. Software-oriented research has focused on protocol designs that explicitly optimize for minimum energy consumption.

1.4.1 Hardware "Knobs" for Energy Scalability

Hardware designers have identified a rich collection of "knobs" to enhance the energy scalability of a wireless communication subsystem, which consists of both digital and analog subsystems.

For energy-scalable digital processing, one popular knob is dynamic voltage scaling (DVS). Reducing processor voltage slows computation, permitting a graceful exchange of energy for delay. As described by Azevedo *et al.,* DVS is a "known effective mechanism

1. And the two inefficiencies mask each other well: does the fact that a 1 Mbit/s radio was on for 500us to transmit a 2-byte packet suggest that startup time (a hardware overhead) or packet headers (a software overhead) are the problem? One cannot immediately tell without more details about the complete communication system.

for reducing CPU energy consumption without significant performance degradation" [2]. DVS is a feature of many low-power processors today including the Intel XScale [45] and Transmeta Crusoe [96], validating the efficacy and commercial viability of this power aware technique. Researchers have considered compilers for DVS scheduling [20, 37], improved real-time scheduling algorithms [73, 75], and variable-voltage regulation [18]. *Voltage overscaling* [32] is the scaling of voltage below the level required for error-free digital signal processing, coupled with circuitry to counteract the accuracy degradation introduced by errors. Pouwelse *et al.* [75] present a StrongARM-based DVS testbed similar to the one described in Chapter 2.

Further energy reductions are possible by coordinating a second parameter, the device threshold voltage V_{TH}, with the supply voltage [26,98]. Device thresholds can be adjusted through substrate biasing in triple-well CMOS technology. Just as V_{DD} scaling exploits the trade-off between propagation delay and switching energy, V_{TH} scaling can exploit a trade-off between propagation delay and sub-threshold leakage power, as discussed in Section 1.3.2. For every clock frequency, there exists an optimal pair of parameters (V_{TH}, V_{DD}) that minimizes energy. Dynamic threshold and supply voltage scaling form a two-dimensional trade-off between energy and circuit latency.

In the analog subsystem, a popular energy-scalable knob is an adjustable power amplifier that allows energy to scale with transmission range [29,41,58,62]. Reducing the output power of the transmitter ensures that excess energy is not wasted by transmitting beyond the range of the intended receiver. The modulation technique may be varied from binary to *M*-ary to effect trade-offs between energy and throughput [85, 100]. *M*-ary modulation provides higher throughput for higher *M*, at the expense of more complex modulator hardware that requires more energy. Finally, the strength of the forward error correction (FEC) may be similarly varied to produce trade-offs between energy and reliability [62, 87].

1.4.2 Energy-Conscious Communication Protocols

Energy-saving medium access and routing protocols have become an increasingly popular research topic within the networking community. Most protocol-based work in this field attempts to exploit one of two hardware enablers for communication energy reduction: shutting down the radio receiver and controlling the transmission power.

Protocols for radio shutdown attempt to maximize the amount of time that radio receivers are shut down by reducing redundant receivers. Inspired perhaps by time-division multiplexing, which naturally provides periodic idle time that is ideal for receiver shutdown, BECA [106] imposes a somewhat periodic shutdown on the radio, in coordination with the routing protocol. AFECA [106] and SPAN [13] utilize radio shutdown to approach the minimum density of receivers required to maintain network connectivity. The Geographical Adaptive Fidelity (GAF) algorithm [107] leverages an explicit knowledge of node locations to shut down redundant nodes. Each of these techniques utilize an additional energy-aware layer between existing MAC and protocol layers to coordinate receiver shutdown. Protocols that take advantage of radio shutdown are not necessarily energy-*scalable*, but their number and popularity require their discussion here.

Protocols for transmit power control, on the other hand, do take advantage of an energy scalability knob. Such protocols seek to reduce the transmission power to the minimum necessary for reliable communication. The common power protocol (COMPOW) [61] sets a minimum but uniform transmit power level for all nodes in a network that ensures connectivity over the network. The protocol described in [105] utilizes a control channel for per-packet negotiation of transmit power; a similar approach (but with a “busy tone” approach rather than RTS-CTS based collision avoidance) is utilized Power Control Multiple Access (PCMA) [58]. Power control protocols that are based on the RTS-CTS mechanism can be incorporated into 802.11b if care is taken to avoid the re-introduction of the hidden terminal problem. A comprehensive discussion is available in [41].

In summary, recent work in low-power and energy-scalable communication protocols has greatly advanced the notion of *optimization* for minimum energy consumption in any given network. However, *dynamic* adaptations to the operational diversity created by the user and environment has received much less emphasis.

1.4.3 Cross-Layer Explorations

Most prior work on energy-scalable communication has been isolated to the design or manipulation of individual “knobs” for scalability from the perspective of a single level of the communication hierarchy. This weakness is addressed by several farsighted works have considered larger subsections of the communication subsystem, such as the interac-

tion between coding and output power [62]. Others have considered interactions among multiple layers of the communication hierarchy, such as the interactions between variable *M*-ary modulation and queueing protocols [81]. Such works have integrated multiple concepts horizontally (multi-dimensional hooks) or vertically (hardware-software interactions). While these are significant advances, a single coherent and cross-disciplinary picture of energy-quality scalability has been largely missing from the literature.

1.5 Scope of this Dissertation

The central goals of this dissertation are the analysis and implementation of cross-disciplinary, cross-layer enablers that provide graceful, dynamic, and energy-efficient responses to operational diversity. Managing and exploiting the interactions among the layers and subcomponents of the communication subsystem is a crucial enabler for the energy-scalable wireless node. Compared to previous works on low energy communication, this dissertation therefore places heavy emphases on graceful energy and performance scalability and the interactions between hardware and software layers. Hence, this exploration necessarily encompasses the domains of circuits, intermediate abstraction layers, and protocols.

Moreover, a wide scope of disciplines is required to derive accurate models of communication performance and a comprehensive framework for energy and quality scaling. Figure 1.9 illustrates the broad scope of disciplines reflected by the parameters and models that are relevant to this work. Performance parameters such as range d and packet reliability P_M of communication are related through models for the channel, modulation, radio hardware, and error-correcting codes. Calculating these parameters—as well as the energy E_{tot} and delay T_{tot} of communication—requires knowledge of parameters such as the transmitted power P_{rad}, code rate R_c, packet length N, and digital supply voltage V_{DD}. While each "block" and parameter in Figure 1.9 will be discussed in depth in the following chapters, the model framework is introduced here to emphasize the cross-disciplinary nature of this work. This dissertation unifies models of digital and analog hardware, coding and information theory, and communication protocol design into a single framework that enables a rich exploration of the trade-offs underlying energy and quality scalable wireless communication.

1.6 Outline of Key Contributions

1.6.1 Motivating Energy Scalability

Energy-scalable systems can gracefully degrade performance to conserve energy. Energy scalability is crucial because the performance demands of a wireless application will change over time, and vary from node to node. Perhaps a few nodes are accidentally deployed more closely together than expected; if they are microsensors, these nodes will generate redundant data. Suppose the user chooses to tolerate a few additional milliseconds of latency; this relaxes many performance requirements of the system.

How can the node automatically take advantage of these new conditions to save additional energy? A node that cannot adapt to relaxed performance demands with energy reductions is wasting precious battery energy. An energy-scalable system continuously

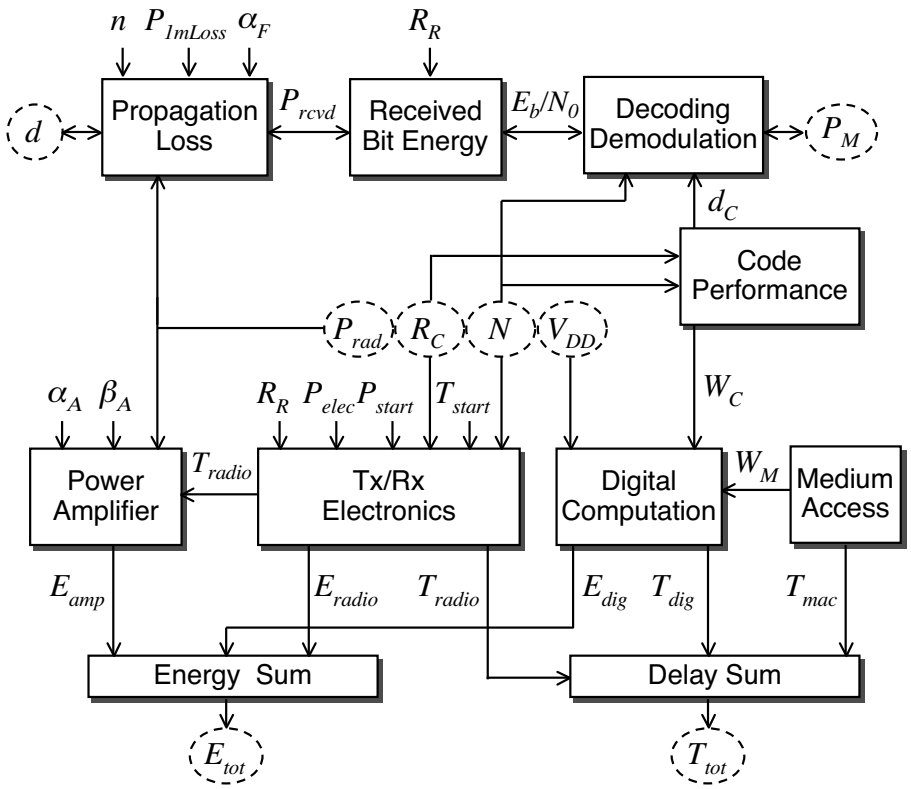


Figure 1.9: An accurate characterization of communication energy and performance requires models from a wide range of disciplines. This figure will recur throughout this dissertation as a roadmap for each exploration into energy-quality scalability.

monitors performance requirements and consumes just enough energy to achieve that level of performance—and not one decibel, byte, or Hertz more. An energy-scalable system demands collaboration among the components of the system hierarchy, for energy trade-offs enabled at the circuit-level are useless unless they are exploited by algorithms and protocols.

1.6.2 Roadmap for the Dissertation

Investigations in hardware will expand the utility of dynamic voltage scaling, a technique for trading digital processing energy for latency through supply voltage adjustments. Voltage scaling will be the centerpiece of a prototype microsensor node that demonstrates power-aware concepts despite construction from off-the-shelf components. These topics are discussed in Chapter 2.

As application designers that utilize wireless communication typically do not wish to concern themselves with processor voltages and the like, it is imperative that an API bridge the gap between low-level "knobs" for energy scalability and those performance parameters more accessible to an application. Hence, this thesis will define a high-level application programming interface (API) for energy management and demonstrate how a "middleware" layer must translate the API calls into hardware parameters for minimal energy consumption. Chapter 3 introduces the API and middleware.

Chapter 4 fuses the analysis of energy-quality scalability with the domains of routing protocols and information theory, resulting in a number of surprising results. Hidden energy overheads of routing techniques such as data aggregation and multihop routing are revealed, with especially dramatic consequences to multihop. Next, the incorporation of BCH block coding and information-theoretic bounds on code performance yield results that counter the conventional wisdom of wireless communication.

Chapter 5 moves fully into the "upper layers" of the application and protocol, focusing specifically on multihop routing for spatially large, low-rate, low-energy microsensor applications. Using a Bellman-Ford shortest path routing methodology, traditional addressed routing is challenged by an unusual "unaddressed" forwarding technique designed specifically for microsensor network topologies. This performance comparison

is carried out using a powerful and interactive Java simulation tool designed specifically for high-density, energy-conscious networks. This tool is the subject of Chapter 6.

Chapter 2

Power-Aware Digital Hardware

As a wireless node is physically composed of digital and analog hardware, energy scalability fundamentally begins with hardware that can make graceful energy *vs*. performance adaptations. This section formalizes energy scalability through a rigorous definition of *power awareness* pioneered by Manish Bhardwaj and presents a practical enhancement to the power awareness of a microprocessor through dynamic voltage scaling. Finally, the dynamic voltage scaled microprocessor is incorporated into an innovative prototype for an energy-scalable microsensor node.

2.1 Power Awareness through Energy Scalability

A hallmark of high-density wireless systems is their operational diversity. At different points in time, a node may be idle, listening for control information, relaying data, or running a FIR filter. The node's operational role adjusts constantly in response to variations in external stimuli, whether from an environmental sensor or incoming radio transmissions. Furthermore, the performance demands of the application itself may change over time, and vary from node to node. Perhaps a few nodes are accidentally deployed more closely together than expected; if they are microsensors, these nodes will generate redundant data. Suppose the user chooses to tolerate a few additional milliseconds of latency; this relaxes many performance requirements of the system.

Great inefficiencies can occur when a node's time-varying operational demands are mapped onto hardware whose energy consumption is invariant to operational diversity. This section asserts that power awareness is fundamentally tied to *energy scalability*, the design of circuits, architectures, and algorithms that can gracefully trade-off performance and quality for energy savings. The key realization is that a node's high operational diversity will necessarily lead to energy inefficiency for a node that is optimized for any one operating point. A node that cannot respond to relaxed performance demands with energy

reductions is utilizing energy in an inefficient manner. *Power-awareness*, then, is really an awareness of the exact performance demands of the user and the environment, and the *graceful scaling of energy consumption to attain that level of performance*.

2.2 Formalisms

The power awareness of a system can be defined formally through a comparison to a system with theoretically "perfect" power awareness. This section builds this definition using a simple motivational example from [6], the 16-bit multiplier.

The first step to evaluating power awareness is the definition of the *scenarios* over which the system will operate. This quantifies the operational diversity. In the case of the multiplier, a simple, one-dimensional *scenario basis* is the operand size of the inputs. Hence, all multiplications are assumed to be n x n, where n completely specifies the scenario. The scenario basis can be made more complicated (and more faithful to the real-world) by considering n x k multiplications, or by considering the *previous* operand size since short-operand multiplications are guaranteed to generate many zeros within the multiplication circuitry. Nevertheless, a one-dimensional scenario basis will suffice for this discussion.

Once the scenario basis is chosen, *´energy curves* are plotted for each system under evaluation. The energy curves plot the energy consumption of a system *versus* the scenario, resulting in a $N+1$ dimensional graph for an N-dimensional scenario basis. Figure 2.1 illustrates two energy curves: one for a monolithic 16 x 16 multiplier, and one representing the *perfectly scalable system*—a theoretical lower bound on energy consumption across all scenarios. This "perfect system" is constructed by considering, for each scenario, a monolithic point system optimized for minimum energy consumption for that scenario. For an n x n bit multiplication, the minimum-energy point system is a n x n bit multiplier. A smaller multiplier could not reliably generate the correct result for all operands, and a larger multiplier would contain combinational paths that are *never* excited, resulting in wasted switching and leakage energy. Hence, a conceptual realization for the perfect multiplier for this scenario basis is an array of sixteen n x n bit multipliers, $1 \le n \le 16$, where incoming multiplications are routed, without energy overhead, to the smallest multiplier that can compute an accurate result. Figure 2.2 illustrates this system.

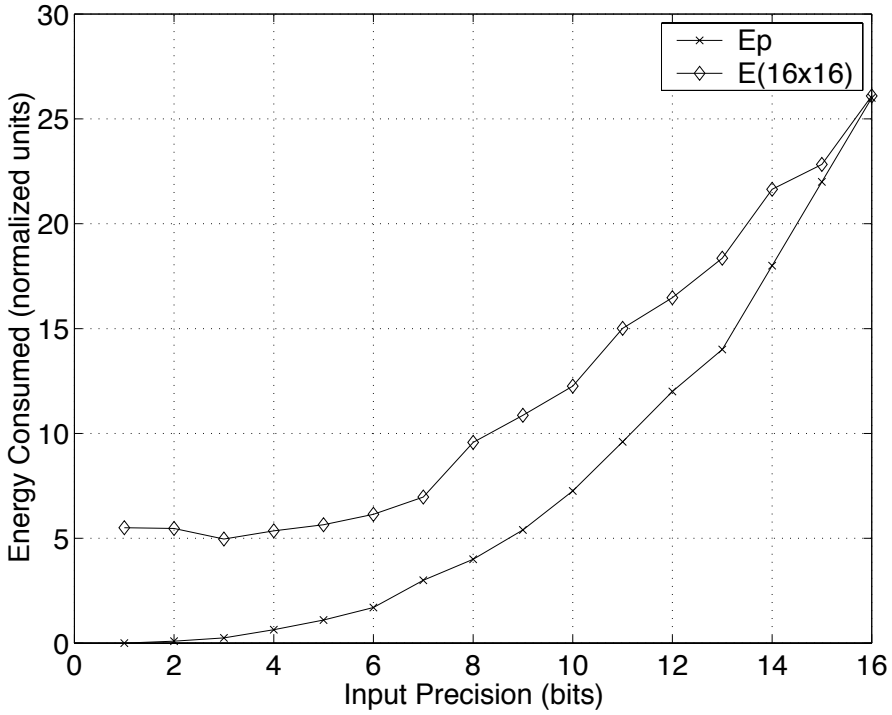


Figure 2.1: Energy curves for *n* x *n*-bit multiplication on a monolithic 16 x 16-bit multiplier, and a theoretical multiplier with perfect energy scalability.

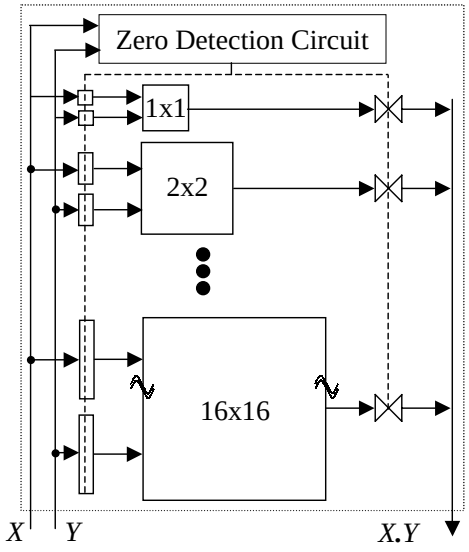


Figure 2.2: For the *n* x *n*-bit multiplication scenario, the "perfect" system can be conceptualized as an ensemble of sixteen multipliers, 1 x 1-bit to 16 x 16-bit, where each multiplication is routed to the appropriate multiplier with no energy overhead.

The key phrase “without energy overhead” is why perfect systems can never be realized in practice (though tangency to a point is always possible).

The power awareness of a system, then, should intuitively be a metric of how close its energy curve approaches the energy curve of the perfect system. The metric defined in [6] is

$$\phi = \frac{\sum_{i=1}^{|S|} E(H_{perfect}, s_i) d_i}{\sum_{i=1}^{|S|} E(H, s_i) d_i} \tag{2.1}$$

where $E(H,s_i)$ evaluates to the energy curve for system H over the scenarios s_i. The term d_i represents the probability that the scenario s_i occurs in the application in question. The d_i term correlates the energy for each scenario with its probability of occurrence, so that the scenarios that appear more frequently in practice are given greater weight. Values for d_i should be distilled from the application(s) expected of the system formalize the system’s operational diversity.

ϕ ranges from 0 to 1 with $\phi = 1$ defined as the perfect system. Note that ϕ can be viewed as the normalized system lifetime compared to the perfect system, assuming a fixed, lossless energy source. A monolithic 16 x 16 multiplier operating under a scenario distribution d_i that is typical of speech processing [90] has $\phi = 0.42$.

While ϕ above represents the absolute power awareness of a system H, the *relative* power awareness between two systems H_1 and H_2 is computed as

$$\phi_r = \frac{\sum_{i=1}^{|S|} E(H_1, s_i) d_i}{\sum_{i=1}^{|S|} E(H_2, s_i) d_i} \tag{2.2}$$

with ϕ_r now representing the expected system lifetime advantage of H_2 over H_1. This form is useful when comparisons to a perfect system are not relevant or not easily defined.

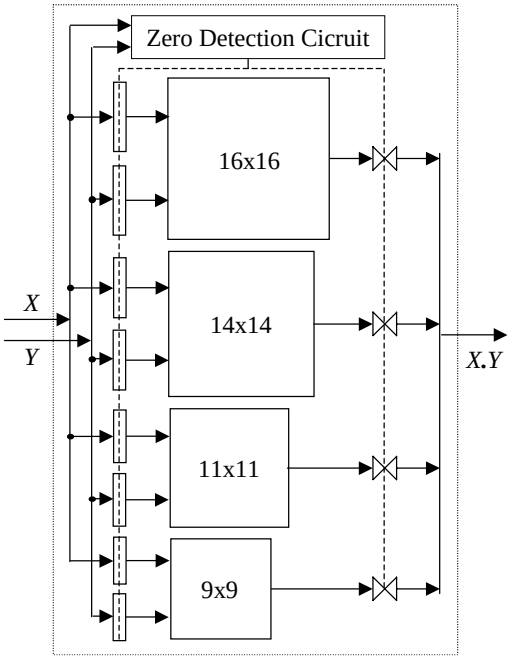


Figure 2.3: The above *ensemble of point systems* provides better power awareness than a monolithic multiplier. This ensemble can be viewed as a carefully chosen subset of all possible point systems (Figure 2.2), accounting for operand frequency and routing overhead.

2.3 Improving Power Awareness

The previous section suggested the construction of a perfectly scalable system as an array of individually optimized point systems, one for each scenario. Such a system would be terribly inefficient in practice due to the overhead of classifying and routing inputs to the appropriate subsystem. However, this notion of assembling an *ensemble of point systems* can improve the power awareness of practical systems. Figure 2.3 illustrates an ensemble of four point systems, an array of four of the sixteen multipliers present in the perfect system. Incoming operands are routed by a zero detector to the smallest multiplier that can generate a correct result. Choosing a subset of all possible point systems provides the benefits of smaller multiplier circuits with a modest operand routing overhead. The choice of point systems is an optimization problem with the number of points and the input distribution d_i as inputs; many examples are presented in [5].

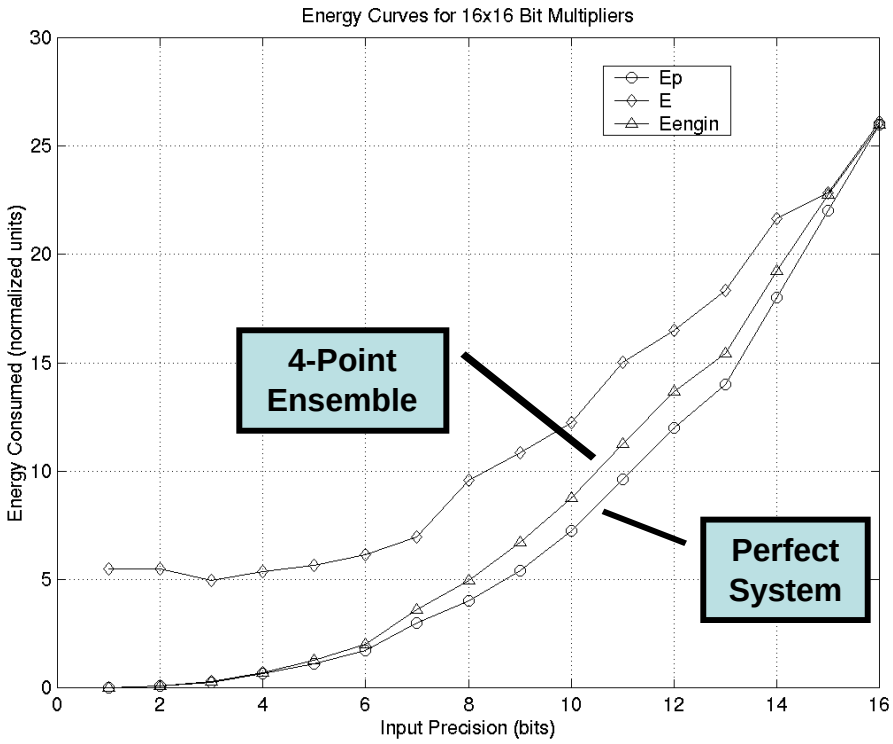


Figure 2.4: The energy curve of the four-point ensemble (Figure 2.3) approaches that of the perfect system. Energy efficiency versus the monolithic multiplier is substantial for the smallest operand sizes.

Figure 2.4 compares the energy curve of the above ensemble to perfect and monolithic systems. This energy curve approximates the perfect system more closely than the monolithic multiplier. Under the scenario distribution discussed above, the ensemble achieves $\phi = 0.92$, even with all energy overheads taken into account.

To summarize, power awareness is defined through a metric that exposes a system's energy scalability over all scenarios that the system is expected to encounter. Power awareness can be enhanced by forming an ensemble of point systems, each optimized for a subset of the scenario space.

2.4 Dynamic Voltage Scaling for Power Awareness

The power awareness of digital computation can be enhanced by varying the supply voltage of the circuits. The circuits' latency can be traded for energy savings in CMOS circuits by varying the circuit's supply voltage. As discussed in the introduction, a higher supply

voltage results in a shorter propagation delay through CMOS gates. The result is faster computation with the cost of quadratically higher switching energy and exponentially higher leakage current. A lower supply voltage thus offers dramatic energy savings at the expense of additional latency. A microsensor node's high operational diversity all but ensures that an application's average performance demands will fall well below a circuit's peak performance at maximum voltage. Therefore, it is often desirable to trade latency for energy savings.

A *dynamic voltage scaled* circuit operates with a higher voltage when peak performance is required and is scaled back to a lower voltage when latency requirements are less strict. Reducing the supply voltage "stretches out" a computation over a longer period of time, saving power and eliminating the need for component shutdown. A lower clock frequency, which relaxes the latency constraints on critical paths, allows the processor to operate with lower supply voltages. Dynamic voltage scaling requires static logic so that logic ones on intermediate nodes scale in tandem with the supply voltage.

The fully static design of the SA-1100 [19] enables the use of DVS on this processor. Figure 2.5 illustrates the measured energy for the voltages and clock frequencies at which

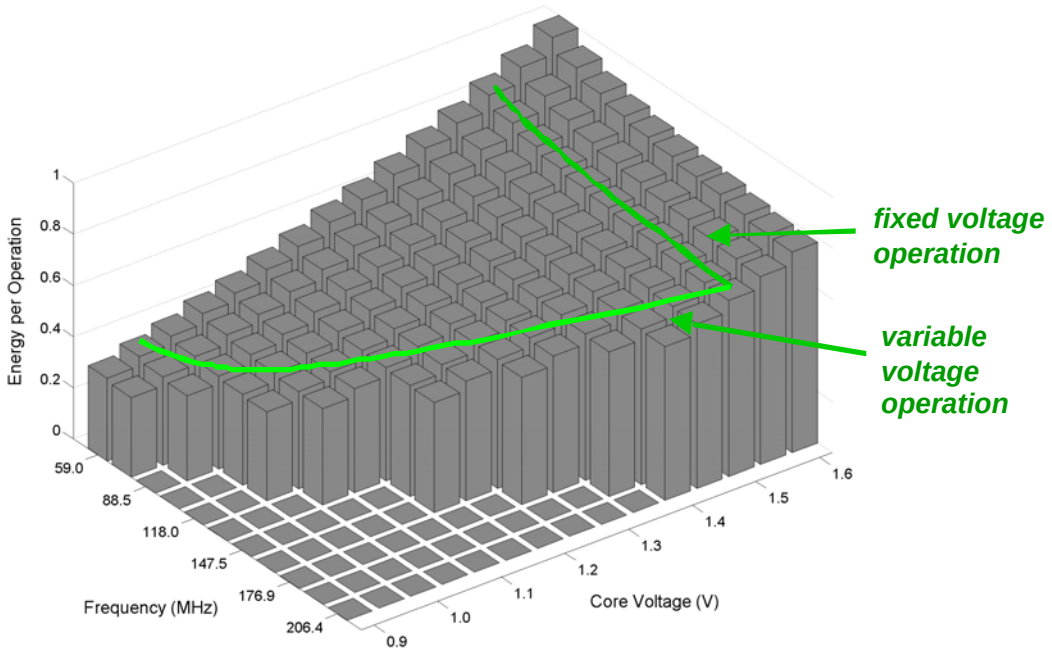


Figure 2.5: Measured energy per operation, versus clock frequency and supply voltage, *or*, a three-dimensional shmoo plot for the SA-1100. Results include regulation losses.

a SA-1100 processor can operate, assuming reasonable environmental conditions. The processor is operating at 100% load. The absence of a bar indicates that the SA-1100 did not function at a particular frequency-voltage combination, meaning that our results can also be interpreted as a three-dimensional shmoo plot.

Equation (1.1), which models the energy consumption of digital computation, provides theoretical insight for the behavior of Figure 2.5. This equation is repeated below for convenience:

$$E_{digital} = CV_{DD}^2 + tV_{DD}I_0 e^{\frac{V_{DD}}{nV_T}} \tag{2.3}$$

An ideal system with DVS would operate at the lowest voltage possible for each supported frequency. In terms of the figure, the system would operate along the "cliff" along the diagonal of the graph. To allow for process and temperature variations in real-world SA-1100 devices and operating scenarios, slightly higher voltages should be chosen for each clock frequency. Interpreting (2.3) as the energy per clock cycle, simultaneous voltage and frequency scaling conserves energy by reducing the switching energy CV_{DD}^2 quadratically. In the leakage term, the reduction in V_{DD} competes with an increase in cycle period t. However, switching energy tends to dominate leakage, and in the SA-1100, the positive impact of a lower V_{DD} dominates the negative impact of higher t even in the leakage term [88]. Supply voltage scaling reduces leakage both by reducing the power-to-ground potential and through drain-induced barrier lowering [104].

The trace of *fixed voltage operation* in Figure 2.5 corresponds to a policy reducing the clock frequency (and therefore t) while keeping V_{DD} fixed. Slowing down a computation without adjustments to V_{DD} actually *increases* energy consumption since the switching term CV_{DD}^2 remains constant while the leakage energy increases linearly with t. Frequency scaling alone does not save energy; voltage scaling is the essential enabler. Similar reasoning also justifies the energy savings of DVS over shutting down a processor during idle periods. While shutdown eliminates the leakage term entirely, the switching term is unaffected since neither the problem size (which corresponds to C) nor the supply voltage are changed.

The fundamental lower limits on V_{DD} scaling may be obtained from physical and mathematical intuition. Physically, the thermal noise floor (26 mV at room temperature) sets a lower bound on V_{DD} for the reliable operation of digital circuits [104]. Mathematically, (2.3) may be partially differentiated with respect to V_{DD} to solve for the minimum-energy V_{DD}, with the proviso that the solution may be below the intended capabilities of the process technology or circuit design.

We conclude this section by forging a connection with the energy scalability formalisms presented in Section 2.2. Each (frequency, voltage) pair is effectively a single point system with distinctive energy consumption characteristics. Hence, a dynamic voltage scaled system is, in itself, an ensemble of point systems. The difference between a DVS-based system and the more traditional multiplier example of Section 2.2 lies in the mechanics of reconfiguring the system. The routing of multiplicands within an ensemble of point multipliers is reconfiguration in space, for each point system that constitutes the ensemble occupies an exclusive physical area in silicon. Dynamic voltage scaling, then, is reconfiguration in time. All point systems are effectively time-multiplexed onto a monolithic computational circuit and a flexible voltage regulator.

2.5 A Dynamic Voltage-Scaled Processing System

Energy-quality scalability with DVS has been realized through a complete, functional demonstration with the SA-1100 processor. A DC-DC converter circuit with a digitally adjustable voltage delivers power to the SA-1100 core and is controlled by a multi-threaded, power-aware operating system.

2.5.1 Hardware Implementation

Figure 2.6 presents a schematic diagram for a DVS demonstration system consisting of a SA-1100 and a DC-DC regulator that can supply a dynamically adjustable voltage to the SA-1100's core. A buck regulator composed of discrete components is driven by a commercial step-down, DC-DC switching regulator controller. This controller is programmed with a 5-bit digital value to regulate one of 32 voltages between 0.9 and 2.0 Volts. The operating system running on the SA-1100 commands the core voltage as a 5-bit digital value that is passed to the regulator controller. Digital logic between the SA-1100

and the regulator controller limits the requested voltage to 1.6V and thus protects the SA-1100 core from receiving a voltage beyond its rated maximum. The operating voltages selected for each clock frequency of the SA-1100 are selected and illustrated in Figure 2.7. Each of these (voltage, frequency) points provide reliable operation in a laboratory setting.

The demonstration system itself consists of two subsystems: a custom circuit board containing the regulation circuits, and the commercial SA-1100 "Brutus" evaluation sys-

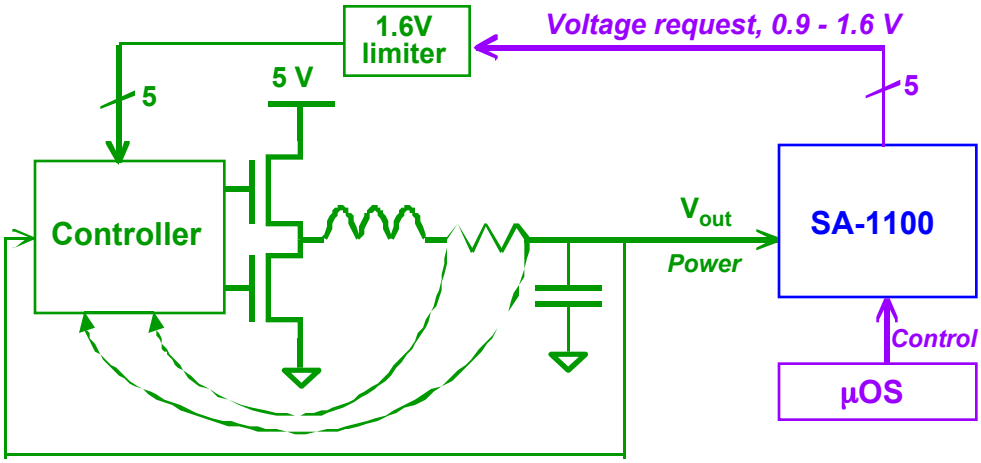


Figure 2.6: Dynamic voltage scaling control system for SA-1100.

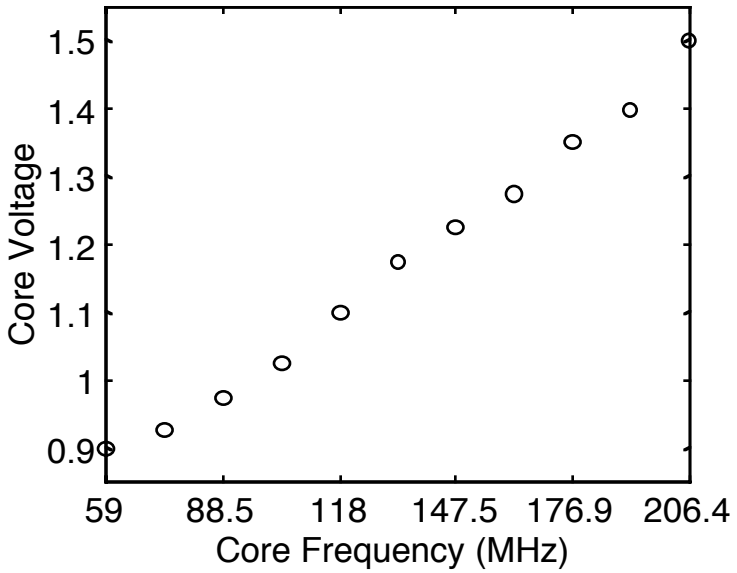


Figure 2.7: Assignments of (frequency, voltage) pairs for DVS on SA-1100.

tem consisting of a SA-1100 and supporting circuitry. The custom printed circuit board pictured in Figure 2.8 implements the DVS-capable regulation system. A Linear Technology LTC1736 is chosen as the controller. While this component is intended to drive 1-10 A of average current for processors such as the Intel Pentium III, it can be applied with reasonable efficiency to the relatively low-current SA-1100 through the judicious choice of MOSFETs and inductor. Compared to a higher-current application, lower currents mean that less peak-to-peak ripple can be tolerated and that switching losses in the MOSFETs may overshadow I^2R series losses. Hence, a larger inductor and narrower MOSFETs are chosen. Voltage limiting is achieved digitally through a 22v10 PAL that bounds any voltage request values above 1.6V.

The SA-1100 “Brutus” evaluation system includes the SA-1100 microprocessor and its accompanying RAM and ROM. The SA-1100 itself operates at a nominal core supply voltage of 1.5 Volts and is capable of on-the-fly clock frequency changes in the range of 59 MHz to 206 MHz. Each frequency change incurs a latency of up to 150 ms while the SA-1100’s on-board PLL locks to the new frequency. The µOS described below is written into the on-board ROM. Power from the regulator board is delivered to the core power supply pins on the SA-1100 evaluation board, and the outputs of five SA-1100 general-purpose I/O (GPIO) pins deliver a five-bit voltage code to the regulator board.

2.5.2 Operating System Considerations

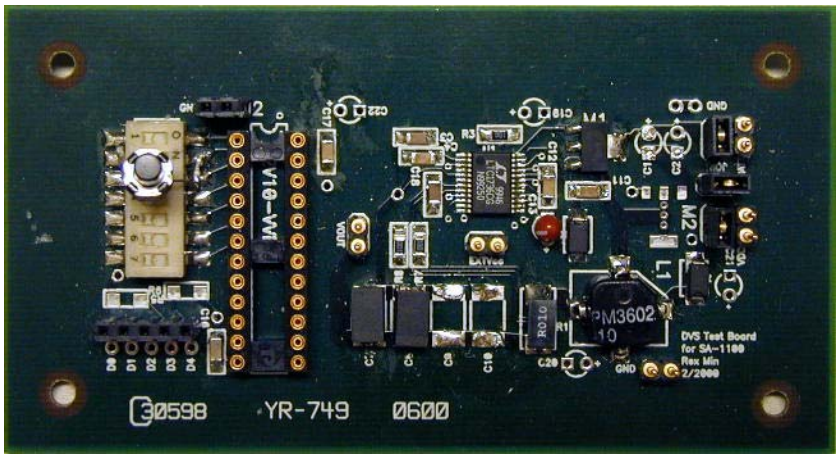


Figure 2.8: Digitally adjustable variable-voltage supply board.

A power-aware operating system based on the Cygnus eCOS kernel [22] has been developed for the SA-1100 by Travis Furrer. This "μOS" supports preemptive multitasking, allowing threads such as data fusion, data packetization, network protocol handling, and voltage scheduling to operate simultaneously within a microsensor node. The μOS resides within a bootable instruction ROM.

In this implementation, the μOS monitors load on the processor and adjusts the clock frequency and supply voltage to meet throughput requirements imposed by the tasks. In the initial implementation, the μOS samples the workload over a brief window of computations. If substantial idle time is detected, the frequency and voltage are stepped downward. If the processor is fully utilized and real-time deadlines are missed, the frequency and voltage are stepped upward. The frequency and voltage are varied according to the pairs chosen in Figure 2.7. These dynamic adjustments are visible on the Brutus color LCD display, illustrated in Figure 2.9.

The majority of throughput requirements, computational latencies, and real-time deadlines on a microsensor network are likely to be known *a priori*. Hence, this particular application domain will likely be amenable to pre-computed voltage schedules that allow all deadlines to be met with minimum energy. Otherwise, more sophisticated load prediction algorithms [70,103] would improve voltage scheduling over the simple windowing scheme described above.

2.5.3 Regulator Functionality

All measurements in the following sections are taken using a Keithley Model 2400 SourceMeter as the power supply and ammeter. The efficiency of the regulator includes all components on the regulator board except for the programmable logic that implements the voltage limiter. Unlike the discrete analog components on the regulator board, a legacy 22v10 PAL device is not representative of the energy consumption of a real-world regulator.

Figure 2.10 plots the efficiency of the regulator board for several output voltages. The efficiencies peak over a range of 50-150 mA, suggesting that MOSFET and inductor values have been appropriately chosen for the SA-1100, which draws from 40-200 mA. Efficiencies are in the 75-85% range, with the lower values occurring at lower output voltages.

Losses from this regulation can be traced to three sources: CV^2 switching losses in the power MOSFETs, resistive losses in the MOSFETs and inductor, and the LTC1736. The operating power of the LTC1736, which is responsible for sensing the output voltage and switching the power transistors, is the largest source of overhead. Due to the constant power consumption of this component, regulation efficiency drops as V_{out} is reduced and less output power is delivered. Newer DVS implementations for the SA-1100 have incorporated the MAX1717 in place of the LTC1736, raising efficiencies at V_{out} = 0.9 V above 80%.

Figure 2.11 below traces the dynamic voltage adjustments commanded by the μOS. This is an actual oscilloscope trace of the voltage supplied to the SA-1100 core over a thirty second interval. This μOS coordinates these changes to the core voltage with simultaneous changes to the clock frequency. The system continues to run when the processor is

Figure 2.9: Photograph of Brutus LCD screen during a DVS demonstration. The lower graph (green on a color rendition of this document) depicts the varying workload imposed on the processor, chosen by the user. The upper (white) graph indicates the voltage and clock frequency level that is set by the system in response to workload changes. The voltage and frequency track the workload with a slight latency.

moved from (59 MHz, 0.9 V) to (206 MHz, 1.5 V) in a single step, the largest change supported by our hardware.

2.5.4 Performance over Two Application Scenarios

The DVS-enabled demonstration system, consisting of the SA-1100 and variable-voltage regulation board, is evaluated with respect to a fixed-voltage system over two scenario bases. The first scenario trades energy for computational latency on a fully loaded processor, and the second trades energy for output quality on a constant-throughput FIR filter. All

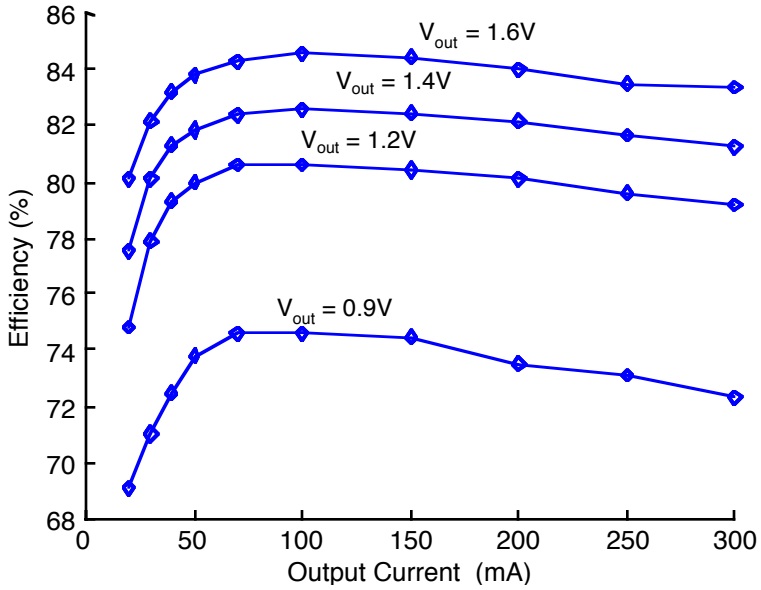


Figure 2.10: Measured efficiency of the regulator board shown in Figure 2.8 with a 5V input.

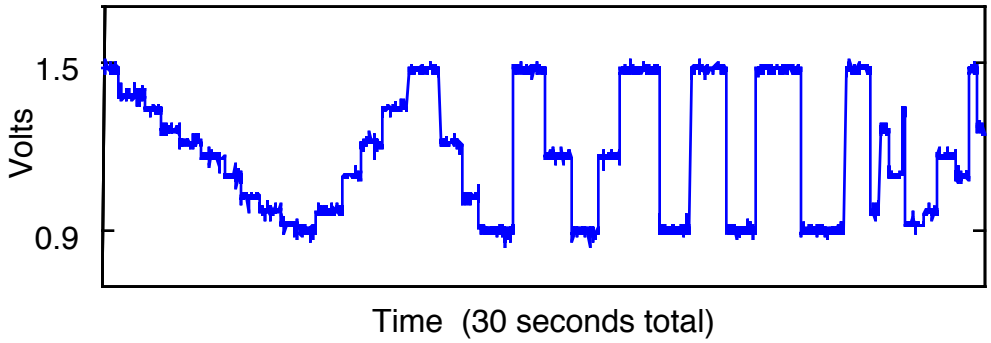


Figure 2.11: Thirty-second oscilloscope trace of voltage supplied by the regulator board. Voltage levels were commanded by software running on the SA-1100.

energy measurements below are based on measured current into the regulator circuit and therefore include losses in the variable-voltage regulator.

The first scenario basis is perhaps the simplest one of clock frequency for a fully-utilized SA-1100 processor. Figure 2.12 plots energy curves for the demonstration system (SA-1100 with regulation overhead) over all supported operating frequencies, both with and without voltage scaling. The upper curve, representing a system with fixed voltage, is the cross-section along $V_{dd} = 1.5$ from the previous graph. As discussed in Section 2.4, the energy per operation is actually *greater* for lower frequencies since the energy lost to leakage is distributed over fewer computations. The lower curve represents the voltage scaled system, the ensemble of (*frequency, voltage*) points depicted in Figure 2.7. Our initial system demonstration of voltage scaling exhibits up to 60% in energy savings over a fixed-voltage system. It is noteworthy that the losses of the custom regulator board, which performed suboptimally at low voltage, are included in these and all measurements.

To illustrate DVS with energy-scalable algorithms, a bandpass FIR filter application is programmed and run under the μOS. The scenario basis this time is the number of filter

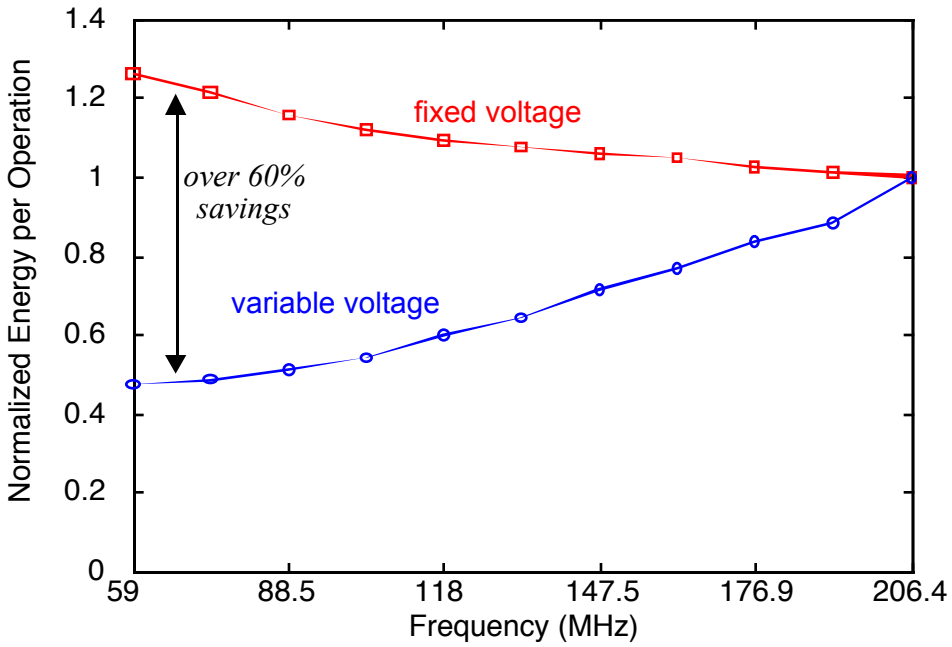


Figure 2.12: Energy savings realized by dynamic voltage scaling, with processor at full workload for a given clock frequency. Note that a substantial regulator loss is included.

taps, which can be scaled to alter the output quality of the filter. Figure 2.13 shows the effect of impulse response length on the stopband rejection of bandpass filter. A longer impulse response improves stopband rejection, but processor workload increases linearly in FIR filtering.

The FIR filter is run at a constant throughput while the impulse response length is varied. Since changing the impulse response length changes the amount of computation required to filter at a constant throughput, the μOS dynamically adjusts the core voltage and frequency to meet the throughput requirement with the lowest possible energy. Figure 2.14 again plots energy curves for fixed-voltage and variable-voltage systems. Again, DVS provides a comparable energy savings to the prior example in Figure 2.12. The discrete series of steps in both plots are due to the eleven discrete frequencies supported by the SA-1100. From Equation 2.2, the relative power awareness ϕ_r of the fixed-voltage system is 0.63, compared with the variable-voltage system.

The difference in shape between the energy curves of these two scenarios is due to differences in what is being measured. Figure 2.12 measures energy per clock cycle—a constant amount of computation—while Figure 2.14 measures energy per filtered data block, a computation whose size varies with the filter length. Discretization aside, dividing the

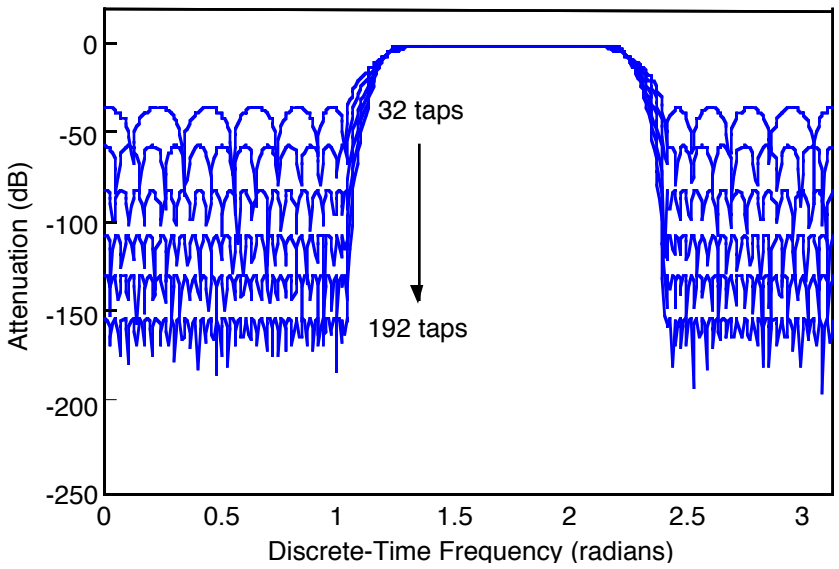


Figure 2.13: Effects of filter tap length on the quality (stopband rejection) of a bandpass FIR filter.

energy measurements in Figure 2.14 by the tap length, effectively a derivative "with respect to workload," would basically result in the energy curves in Figure 2.12.

2.6 µAMPS-0: A DVS-Enabled Microsensor Node

Dynamic voltage scaling has been incorporated into a complete, power aware embedded system. µAMPS-0, the first prototype of the µAMPS project, is designed with commercial off-the-shelf components for rapid prototyping and modularity. The µAMPS-0 sensor node is designed to be both a proof-of-concept for a microsensor node a platform for demonstrating and evaluating power-aware methodologies. The node consists of subsystems for power delivery, sensing, processing, and communication. Figure 2.15 illustrates the major components of these subsystems.

A StrongARM SA-1100 microprocessor is selected for its low power consumption, sufficient performance for signal processing algorithms, and static CMOS design. The memory map mimics the SA-1100 "Brutus" evaluation platform and thus supports up to 16 MB of RAM and 512 KB of ROM. The lightweight, multithreaded "µOS" running on the SA-1100 is an adaptation of the eCOS microkernel [22] that has been customized to support the power-aware methodologies such as DVS. The µOS, data aggregation algorithms, and networking firmware are embedded into ROM. The SA-1100 core is powered

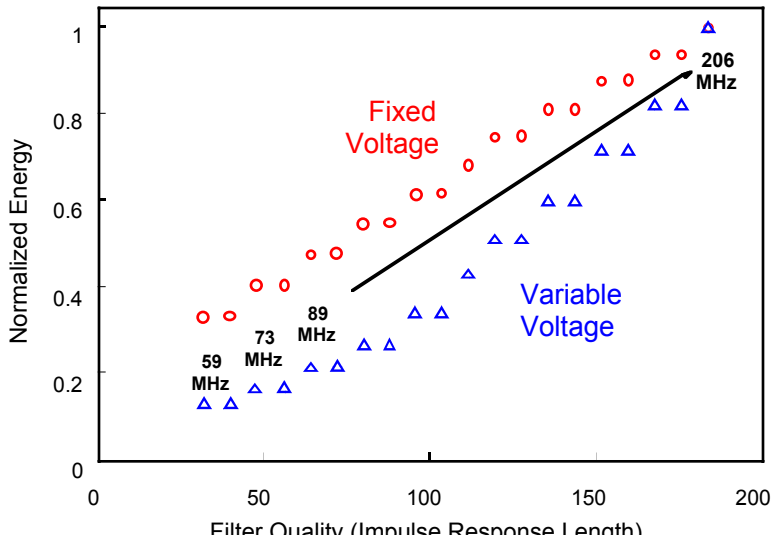


Figure 2.14: Energy *vs.* quality comparison for FIR filtering, with and without DVS. The clock frequency is scaled with workload in both cases.

by a digitally adjustable switching regulator that can provide 0.9V to 1.6V in twenty discrete increments. The digitally adjustable voltage allows the SA-1100 to control its own core voltage, enabling dynamic voltage scaling.

The node includes seismic and acoustic sensors. The analog signals from these sensors are conditioned with 8th-order analog filters and are sampled by a 12-bit A/D. All components are carefully chosen for low power dissipation; a sensor, filter, and A/D typically requires only 5 mA at 5 V. The node's radio module interfaces directly to the SA-1100. The radio is based on a commercial single-chip transceiver optimized for ISM 2.45 GHz wireless systems. The PLL, transmitter chain, and receiver chain are capable of being shut-off under software or hardware control for energy savings. Power for the sensor node is supplied by a single 3.6V DC source, which can be provided by a single lithium-ion cell or three NiCD or NiMH cells.

Figure 2.16 illustrates the μAMPS-0 sensor/processor board designed and fabricated for this work. The board incorporates the power, sensing, and processing subsystems into an area roughly 6" x 6". The variable-voltage regulation circuitry of the regulator board in

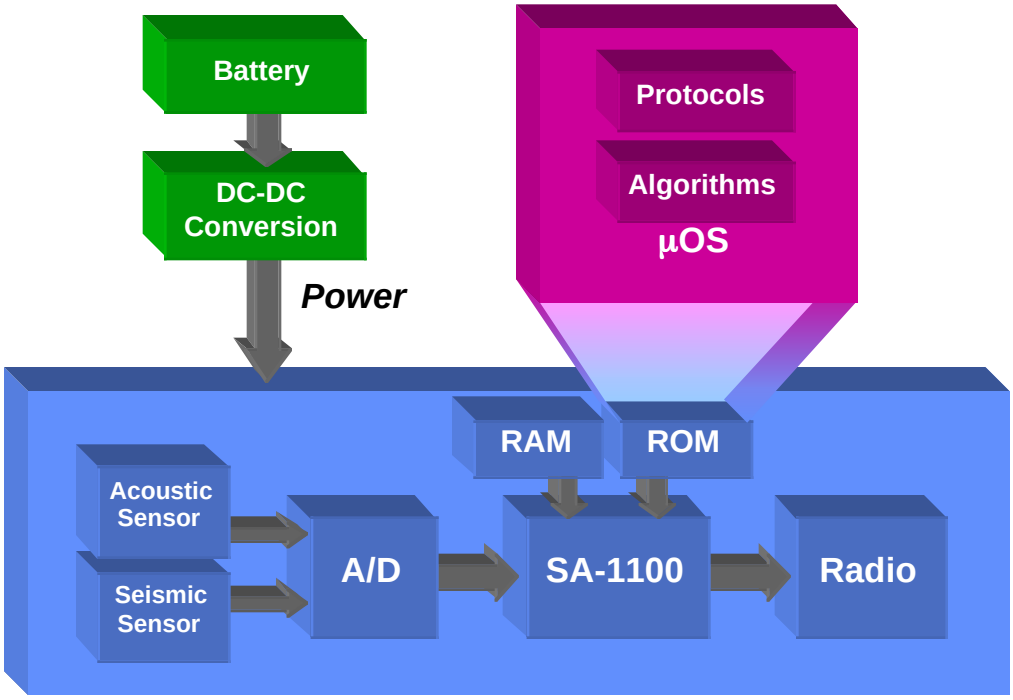


Figure 2.15: Block diagram of the μAMPS-0 sensor node.

Section 2.5 are incorporated into this design. An external connection is available for the radio subsystem, which has been fabricated by Eugene Shih and Fred Lee as two similarly-sized boards.

Each subsystem has successfully been tested for functionality. Notably, dynamic voltage scaling is functional over the same 0.9 - 1.5 V range of the demonstration system, and the SA-1100 has shared data with the radio subsystem using variable-strength forward error correction. Peak power consumption for the board is nearly 1 Watt; several significant sources of inefficiency include copious, off-the-shelf DRAM and flash ROM, and the use of linear regulators. In practice, the board operates for one full day using three 1.2 V, 1450 mAh NiMH cells.

2.7 Summary and Impact

The energy dissipation of a system depends on its inputs, whose variations are termed *operational diversity.* Systems that adapt gracefully to operational diversity are known as *power aware systems*, and the power awareness of the system may be captured through a formal metric that is a function of the system's energy consumption over a given input dis-

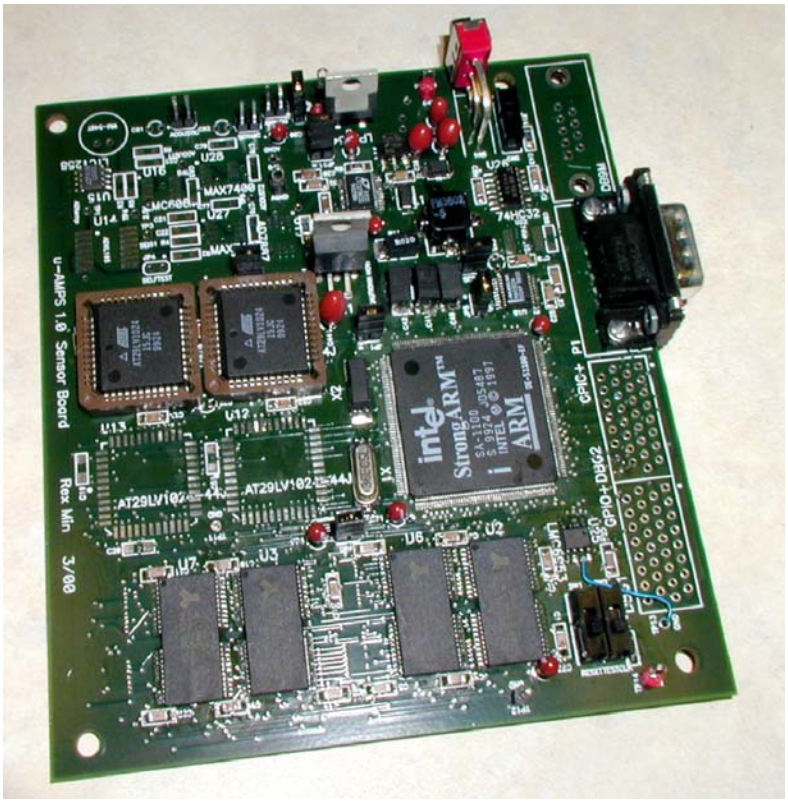


Figure 2.16: Digital processor board for µAMPS-0.

tribution.

Dynamic voltage scaling is a practical enhancement to a microprocessor's power awareness. Operational diversity arises from relaxed latency constraints or periods of light workload, and a DVS-enabled system reduces the speed and voltage of computations to reduce energy consumption. DVS has been demonstrated on the SA-1100 processor and incorporated into μAMPS-0, a power aware microsensor node designed and constructed from off-the-shelf components. This implementation of dynamic voltage scaling on the SA-1100 processor is, to the best of the author's knowledge, the first demonstration of a dynamic voltage scaled commercial processor.

The μAMPS-0 node described above has served as the template for the next-generation μAMPS-1 node [87], which miniaturizes the area of the circuit boards, integrates newer components such as the StrongARM SA-1110, and reduces power consumption.

Chapter 3

A Middleware Abstraction for Energy-Scalable Communication

While power-aware hardware is a significant stride toward energy-efficient communication, application designers that utilize wireless communication typically do not wish to concern themselves with low-level parameters such as processor voltage or transmit power. Since energy conservation is so crucial to wireless nodes, however, the application must be provided with convenient means to exploit hardware energy scalability. Figure 3.1 previews the approach proposed in this chapter. A middleware layer is introduced to bridge the gap between low-level "knobs" for energy scalability and performance metrics more relevant to an application. Performance metrics for communication are expressed through an API established by the middleware and are translated into energy-efficient parameter settings for the communication hardware.

Since our goal is to bridge hardware-centric notions of energy agility and software-centric notions of communication performance, we begin by considering the hardware and

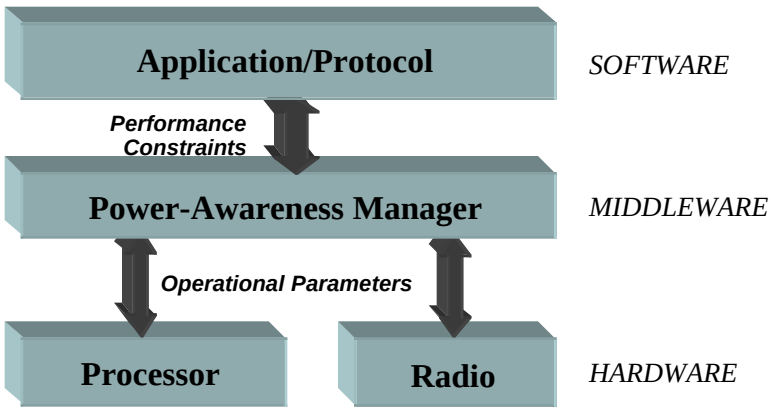


Figure 3.1: The notions of hardware energy scalability and communication protocol performance are unified through an abstraction layer that encourages energy-quality scalability.

software worlds in isolation. From Section 3.3 onwards, we begin to span the divide with evaluated expressions for communication performance and energy consumption using energy and performance models for the μAMPS-1 microsensor node. This chapter concludes with an implementation of dynamic range scalability for μAMPS-1.

3.1 Bottom-Up View: Hardware Energy Scalability

Power aware hardware reacts gracefully to constantly changing operational demands. As performance demands increase or decrease, power aware hardware scales energy consumption accordingly to adjust its performance on-the-fly. Graceful energy scalability is highly desirable for any energy-constrained wireless node since the operational demands on a real-world node constantly change, and the peak performance of the node is rarely needed. Energy agility is effected by key parameters that act as knobs to adjust energy and performance simultaneously.

The μAMPS-1 is an energy-scalable microsensor node that supports the hardware "knobs" of processor voltage scaling, variable FEC, and adjustable radio power. The processing subsystem, implemented primarily by a SA-1110 low-power microprocessor, is customized to support dynamic voltages scaling from 0.9-1.5 V. Forward error correction, implemented through convolutional encoding and Viterbi decoding at the SA-1110, is scalable by altering the constraint length and puncturing of the base code. (Note that an FPGA or dedicated hardware would be a lower-energy solution.) The radio subsystem consists of a 2.4 GHz, 1 Mbps FSK transceiver with time-division media access and features two power amplifiers for low (+0 dBm) or high (+20 dBm) power transmission.

Voltage scaling, convolutional code strength, and radio transmission power are three crucial knobs for power awareness that will be increasingly present in modern wireless nodes. The remainder of this chapter will analyze the impact of these hardware knobs.

3.2 Top-Down View: Protocols and Communication Performance

3.2.1 Application View of Communication

Application designers that utilize wireless communication typically do not wish to concern themselves with processor voltages, transmit power, or the constraint lengths of convolutional codes. It is therefore imperative that an application programming interface

Table 3.1: Model parameters and values for Section 3.5.

Symbol	Description	Value
α_{amp},	amplifier inefficiency, constant term	174 mW
α_c	switched cap/bit, exponential base	2.62
α_t	decode time/bit, exponential base	2.99
β_{amp}	amplifier inefficiency, linear coeff.	5.0
C_0	switched cap/bit, linear coefficient	51.6 nF
c_{proc}	processor f,V_{DD} relation, const. term	659 mV
d	transmission range	0-100 m
f	processor frequency	59-206 MHz
f_{max}	maximum processor frequency	206 MHz
I_0	proc. subthresh. leakage, linear coeff.	1.196 mA
K_C	conv. code constraint length	3, 5, 7
K_{proc}	processor f,V_{DD} relation, linear coeff.	245 MHz/V
N	bits per transmission	1000
n	path loss exponent	3.5
n_0	subthresh. leakage, exponential term	21.26
P_{1mAtt}	path loss, one-meter attenuation	30 dB
P_{amp}	transmit amplifier power	179, 674 mW
P_b	bit error rate (BER) at receiver	10^{-3} to 10^{-9}
P_{rxElec}	receive electronics power	279 mW
P_{start}	radio startup power	58.7 mW
P_{txElec}	transmit electronics power	151 mW
R	radio transmit data rate	1 Mbps
R_C	convolutional code rate	1/2, 2/3
T_0	decode time/bit, linear coefficient	219 ns
T_{start}	radio startup time	470 μs
V_{DD}	processor supply voltage	0.9-1.5V
V_T	thermal voltage (room temperature)	26 mV

(API) bridge the gap between these low-level “knobs” for energy scalability and those performance parameters more accessible to an application.

An application's view of communication can be summed into four basic questions:

1. *Where* is the communication going?
2. *How reliable* need be the delivery?
3. *How soon* does it need to get there?
4. *At what cost* should the communication be made?

It is the author's belief that these questions fundamentally define communication of any kind. For the purposes of this work, these four questions can be further refined into four fundamental parameters of wireless transmission: *range*, *reliability*, *delay*, and *energy*. These parameters, therefore, are the appropriate fundamental bases for an application's specification of its communication needs. An API for power-aware communication, then, must speak in these terms.

3.2.2 The Protocol Stack and its Limitations

Having established the application cares about a communication's range, delay, reliability, and energy, it is surprising to find that these properties are poorly modularized in a classic protocol stack. Figure 3.2 lists the four application concerns and the layers of the protocol stack that primarily determine each of them. While the stack provides a valuable abstraction for protocol designers, the abstraction boundaries hinder the control of crucial performance parameters of a communication.

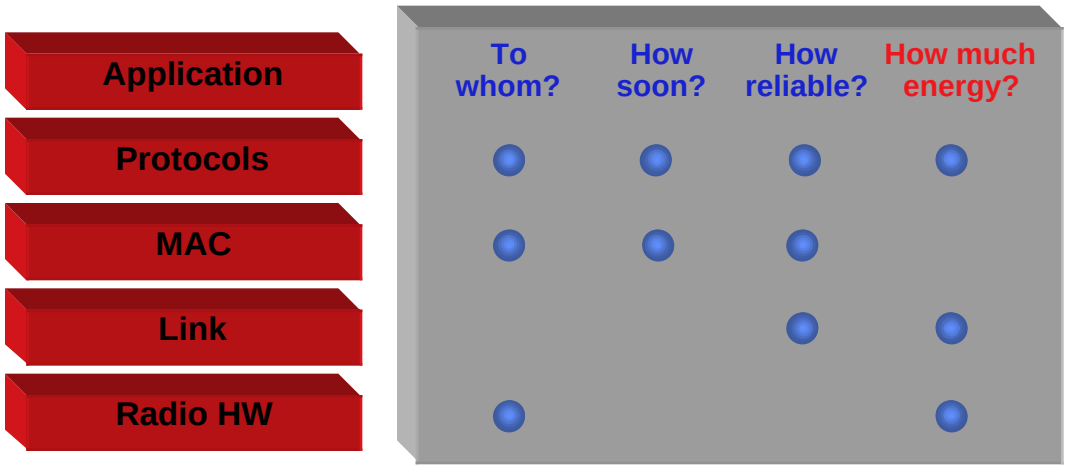


Figure 3.2: Multiple layers of a protocol stack drive the answers to the four fundamental parameters of communication: destination, delay, reliability, and energy.

The design of any resource-constrained system presents a fundamental tension between abstraction and efficiency. Abstraction facilitates the design of complex systems by hiding functional details within the proverbial black box. Awareness of the implementation within the box, however, typically yields more efficient designs. *Power* awareness is no exception; the inefficiency of compiled code running on (black box) microprocessors *versus* application-specific integrated circuits is a dramatic example. Custom integrated circuits have been shown to consume six orders of magnitude less energy than compiled—or even assembled—code [27].

But awareness and abstraction need not conflict. Power awareness does not demand that an entire implementation be exposed. Previous chapters have honed the notion of power awareness "knobs," the adjustable parameters of a system that have a significant impact on energy consumption. Exposing only the "knobs" of a functional unit, as well as their precise impact on the unit's performance, preserves the unit's abstraction boundaries. Higher levels of a system need not be concerned about the mechanics of internal energy-quality trade offs.

The classic protocol stack, however, exposes power awareness hooks poorly. Hardware, which determines energy consumption, is separated by numerous abstraction layers from the application, which determines how the hardware will behave. These abstraction boundaries inhibit the cross-layer cooperation that is essential to power awareness. Hence, this work replaces the notion of a protocol *stack* with the paradigm of self-contained functional blocks as illustrated in Figure 3.3. Operations in the "middle" of the protocol stack, such as media access, error correction, and queueing, are treated as a self-contained black boxes. Abstraction boundaries are preserved *between* modules to facilitate design. As all modules are at the same level of hierarchy, only *one* set of boundaries exists between any two modules, facilitating *awareness* through parameter sharing. With the protocol stack cast aside, awareness and abstraction can work in concert.

With interaction between communication objects defined, the final task is to determine how the application will interact with the communication framework. The approach, also illustrated in Figure 3.3, is a power awareness manager between the application and the object-oriented communication framework. This power awareness manager supports an application programming interface (API) that allows the application to expose the commu-

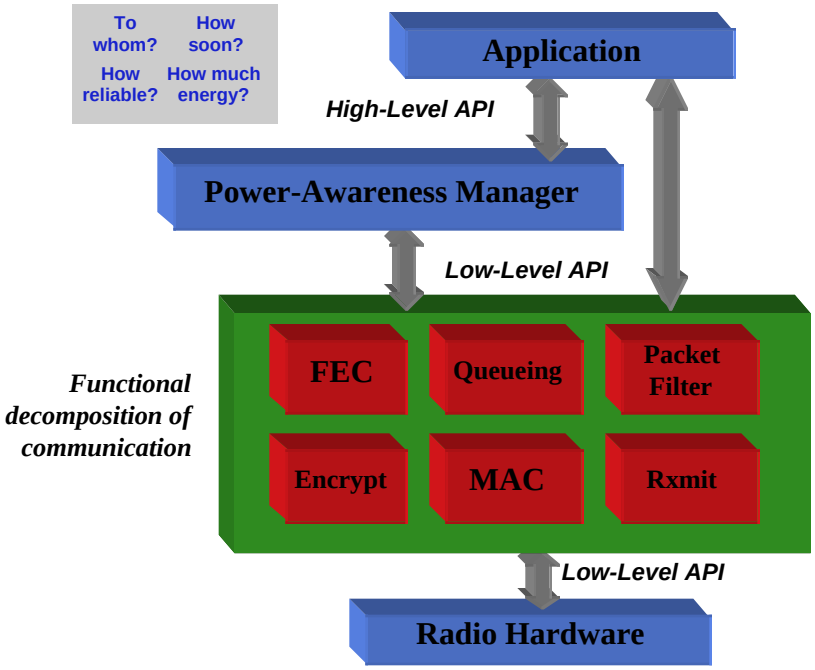


Figure 3.3: A power aware approach to communication may be achieved through integrated power management, combined with a module-based approach to communication.

nication performance (delay, reliability, range) that it needs. This communication API delivers "hints" from the application to the communication subsystem that suggest what kinds of performance degradation are tolerable in exchange for energy savings. Utilizing these hints, the power awareness manager chooses the best settings for the hardware and communications objects to minimize energy consumption. An analogy can be drawn to the operating system of a dynamic voltage scaled processor, which can utilize hints of upcoming workloads from the application to schedule the processor's voltage and frequency.

3.3 Bridging the Gap: Hardware to Software

3.3.1 Parameter Bound-Based API (A Starting Point)

Designing a power-aware API framework for wireless communication entails two key steps. First, the API calls are *defined* in a form that is practical for the expected applications. Second, the calls are *implemented* with respect to the energy consumption characteristics of the hardware. This section presents a constraint-based API that encourages applications to bound the performance requirements of inter-node communication. These

bounds are translated by the API implementation into low-level hardware policies for the hardware, using detailed knowledge of the hardware energy consumption characteristics. This process is presented using the μAMPS-1 sensor node as a practical example.

The required range of a transmission is determined by its destination(s). To encourage an application to take advantage of range scalability, we allow it to specify the explicit destination or destinations of its communication, whether it be a unicast, multicast, or broadcast message.

With cooperation from a protocol layer that maintains approximate distances to—and numbers of—neighboring nodes, the communication range desired by an application can be expressed through four API calls:

```
set_destination(Node n)
set_destination(Nodes n[])
set_range(int numberOfNearestNeighbors)
set_range(Distance d)
```

The `set_destination` calls allow the range of a communication to be expressed in terms of its intended recipients. The unicast form directly implies a transmission range equal to the distance to the receiver. The multicast form, which eliminates the need for redundant unicast calls by the application, additionally implies a range that varies with direction, a fact that can be utilized by clever algorithms to generate asymmetric multi-hop routes.

The `set_range` calls allow an explicit specification of transmission radius and are especially useful primitives for broadcast. Range is specified either in terms of the number of nearest neighbors reached by the transmission, or as an outright distance in meters.

The remaining parameters of delay, reliability, and energy should be boundable by the application. This is achievable through the following calls:

```
set_max_delay(double usecs)
set_min_reliability(double ber)
set_max_energy(double ujoules)
```

Any or all bounds can be specified, but an appropriate exception is thrown if the combination of bounds requested exceeds the capability of the system. To guide the user, reasonable defaults and parallel `get` calls to fetch property values should be provided.

A power-aware communication API liberates the application designer from direct manipulation of the hardware's power-aware hooks. This burden is transferred to the API implementor, who views each API call as an instance of operational diversity that demands adaptation by the hardware. Choosing the best operational policy for the hardware based on the application-level settings brought down from the API requires detailed knowledge of the hardware's energy consumption characteristics. More specifically, the energy consumption of hardware—which is typically expressed in terms of physical parameters such as voltage—is evaluated as a function of the application level parameters of delay, reliability, and range.

3.3.2 The Middleware Problem

We may now precisely define the goal of dynamic energy-quality scaling for communication. Given required standards for communication quality, we aim to set the low-level knobs such that the standards are met with minimum energy consumption. In general, for a model with k low-level energy scalability knobs x_1 to x_k, our goal is to find

$$\arg\min E(x_1, x_2, \ldots, x_k) \tag{3.1}$$

subject to one or more of the constraints

$$\begin{aligned} T_{tot}(x_1, x_2, \ldots, x_k) &\le T_{request} \\ d(x_1, x_2, \ldots, x_k) &\ge d_{request} \\ P_M(x_1, x_2, \ldots, x_k) &\le P_{Mrequest} \\ R_{tot}(x_1, x_2, \ldots, x_k) &\ge R_{request} \end{aligned} \tag{3.2}$$

This is the fundamental problem statement for algorithms that dynamically scale communication performance and energy consumption. While (3.1) and (3.2) often produce convex relations, yielding a tractable optimization, the problem is nonetheless a nonlinear program (NLP) with nonlinear constraints, the most general class of minimization problems. The investigations of this chapter therefore consider a subset of these relations.

3.4 Energy-Quality Models for the μAMPS-1 Microsensor Node

To provide some insight into the middleware problem, this section reviews and develops model relations that will fuse our parameterization of communication performance with the energy consumption characteristics of hardware.

3.4.1 Review of the Model Framework

To review, we begin by identifying parameters that quantify the "quality" of communication as follows:

• **Range** d: the required transmission distance

• **Reliability** as a bit error rate P_B or packet error rate P_M: the probability that a transmitted bit or packet will be decoded in error, or otherwise not received correctly.

• **Delay** T_{tot}: the end-to-end communication delay

• **Throughput** R_{tot}: (uncoded) data bits transmitted per second

• **Energy** E_{tot}: total energy consumed per uncoded data bit

Low-level link and physical layer "knobs" are adjusted to provide scalability in one or more of these performance parameters. The four knobs considered in this work are:

• **Radiated power** P_{rad}, the power transmitted by sending node

• **Packet length** N, the number of useful bits per packet

• **Supply voltage** V_{DD} of the digital circuits

• **Code rate** R_C, the packet expansion due to error-correcting code

• The complexity of the error-correcting code

For convolutional codes, the constraint length K_C is the relevant measure of code complexity. The complexity of block codes is highly code-dependent, but the Hamming distance d_{min}, the minimum number of unique bits between codewords, may reflect on code complexity.

We are now prepared to quantify the impact of the above knobs on the energy and performance of wireless communication. Figure 3.4 repeats the model framework presented in the introduction as Figure 1.9. The four high-level communication parameters d, P_{tot}, E_{tot}, and T_{tot} appear at the edges of the block diagram while the four low-level parameters P_{rad}, R_c, N, and V_{DD} appear at the center. The parameters and models relevant to this initial exploration are shaded. This is a convention that will be followed in future chapters to

place each exploration in context with the complete, interdisciplinary framework for energy-quality scalability.

The energy and performance of point-to-point wireless communication is modeled using energy measurements from prior work with the SA-1100 processor and the μAMPS-1 node [87,88,99]. Many of these relations will be utilized in following chapters.

3.4.2 Radio Transmission Energy

Components of the radio such as the PLL and VCO consume a constant power while on, and require a non-negligible startup time to resume operation from a powered down state [54]. This portion of the radio energy, present in both the receiver and transmitter, can therefore be modeled as

$$E_{radio}(T_{on}) = P_{start}T_{start} + P_{elec}T_{on} \tag{3.3}$$

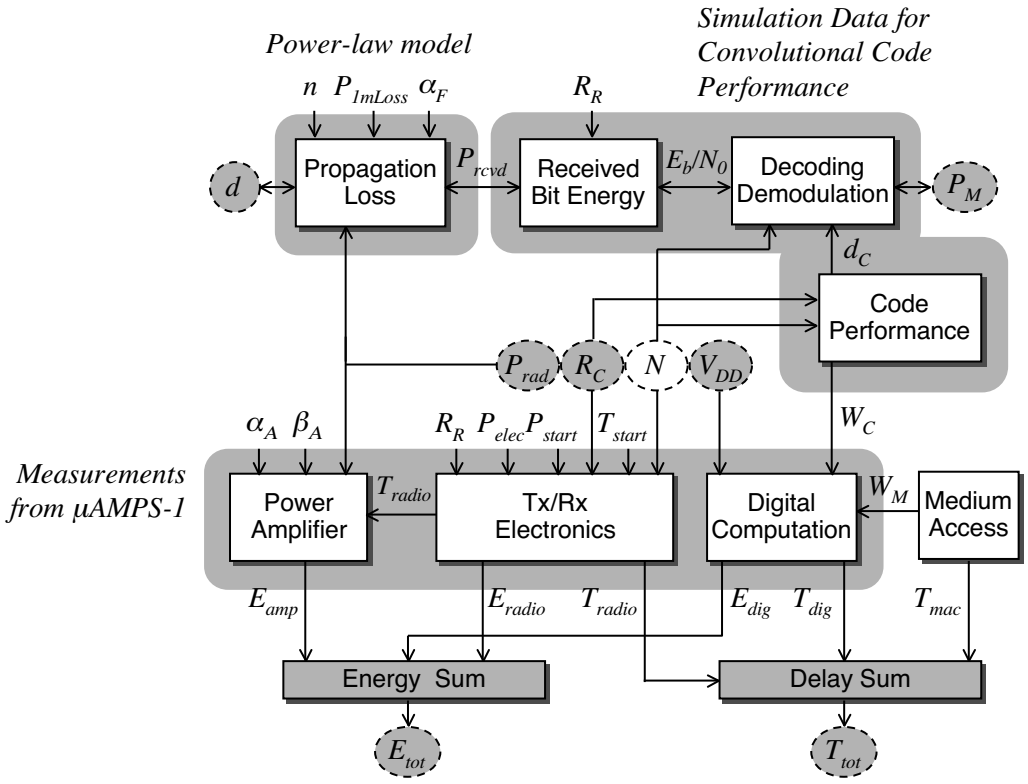


Figure 3.4: Graphical overview of model relations in this section, linking communication performance (energy E_{tot}, delay T_{tot}, range d, and packet error rate P_M) with three low-level scalability knobs.

for a startup power and time of P_{start} and T_{start}, a constant power of P_{elec} while on and an on-time T_{on} that represents the startup delay of the radio electronics for a single transmission. Typically, $P_{start} < P_{rxElec}$, since only the VCO and PLL require a settling time T_{start}. The remainder of the transmission circuit, such as the power amplifier, starts quickly enough to be omitted from the startup energy term.

The on-time T_{on} is simply the length of the uncoded packet N, the radio bit rate R_R and code rate R_C:

$$T_{on}(N, R_c) = \frac{N}{R_R R_C} \tag{3.4}$$

Receiver energy and the static (output power independent) portion of the transmitter's energy are modeled by (3.3) and (3.4), with parameters chosen appropriately for each section. The energy consumed by the transmit power amplifier increases in relation to the radiated power P_{rad}. Power amplifiers typically have efficiencies around 20%, an overhead that cannot be ignored. Hence the amplifier energy is modeled as

$$E_{amp}(P_{rad}) = T_{on}(\alpha_{amp} + \beta_{amp} P_{rad}) \tag{3.5}$$

where α_{amp} and β_{amp} linearize the inefficiencies of the amplifier. Figure 3.5 illustrates the true energy cost of the power amplifier for μAMPS-1.

Combining (3.3)-(3.5), the total energy of the *transmitter* is then

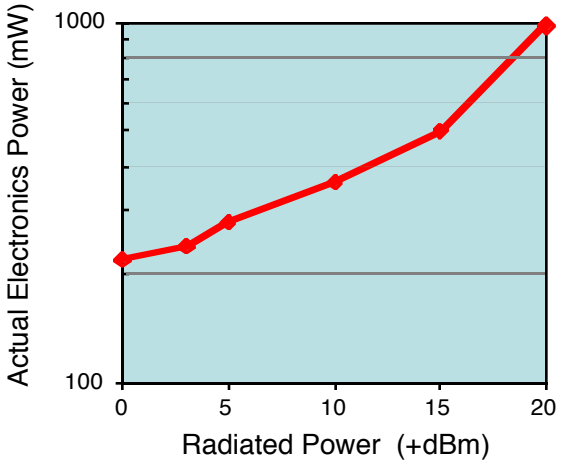


Figure 3.5: Due to the inefficiencies of linear power amplifiers, a radiated output power of 100 mW for the μAMPS-1 node requires an input of nearly 1 Watt to the power amplifier.

$$E_{tx}(N, R_c, P_{amp}) = P_{start}T_{start} + \frac{N}{R_c R}(P_{txElec} + P_{amp}) \quad (3.6)$$

(Total receive energy is derived in Section 3.4.4 below.) In (3.6), the energy of transmission is expressed as a function of the low-level parameters R_c and P_{amp}. Our goal now is to relate these values to the application level parameters for range and reliability: the transmission distance d and bit error rate P_b. We achieve this by first relating P_b to the power incident at the receive antenna, and then computing the transmit power required to attain this receive power over a distance d.

3.4.3 Range and Reliability with Convolutional Coding

The range and error rate of a transmission are related by airborne propagation loss, the received energy per transmitted bit, and the performance of both the receiver and any error correcting code. The path loss block relates the range d with the required radiated power P_{rad} as

$$P_{rad} = \alpha_F(P_{1mAtt}d^n)P_{rcvd} \quad (3.7)$$

The channel is modeled with three parameters: the attenuation over one meter P_{1matt}, the path loss exponent n, and a random variable α_F representing time-varying fading. The probability distribution of α_F determines the fading characteristic modeled (Rayleigh, Rice, etc.)

An explicit relation between P_b and received power under Rayleigh fading and convolutional coding is beyond the scope of this work[1]. Simulations of convolutional codes' performance under Rayleigh fading [87], plotted in Figure 3.6, are regressed into a function for the received power P_{rcvd} needed to achieve a bit error rate P_b. We denote each function $P_{rcvd}(P_b,R_c,K_c)$. Since R_c and K_c together identify a distinct convolutional code in this work, P_{rcvd} can be viewed as an array of functions on P_b, with R_c and K_c serving as indices to identify the particular BER-to-receive power relationship for each code.

Computing the total amplifier power at the transmitter required to achieve P_{rcvd} at the receiver requires consideration of the path loss and amplifier inefficiencies:

1. Rayleigh fading with *block* codes, however, is considered in Section 4.3.

$$P_{amp}(d) = \alpha_{amp} + \beta_{amp} P_{1mAtt} d^n P_{rcvd}(P_b, R_c, K_c) \quad (3.8)$$

where d is distance, α_{amp} and β_{amp} parameters representing the linearized efficiency of the power amplifier, P_{1mAtt} the attenuation at one meter from the sender including antenna effects, and n the path loss index. The following results are based on P_{1mAtt} = 30 dB and n = 3.5. In practice, P_{1mAtt} and n vary heavily by environment.

Substituting (3.8) into (3.6) yields a complete expression for $E_{tx}(P_b,d,N)$, the energy in terms of reliability and range, for each code (R_C, K_C):

$$E_{tx}(P_b, d, N) = P_{start} T_{start} + \frac{N}{R_c R}[P_{txElec} + (\alpha_{amp} + \beta_{amp} P_{1mAtt} d^n P_{rcvd}(P_b, R_c, K_c))] \quad (3.9)$$

3.4.4 Decoding and Total Receive Energy

The energy required to receive a packet is the sum of the energies dissipated by radio startup, the active receiver electronics, and the digital circuits used by Viterbi decoding:

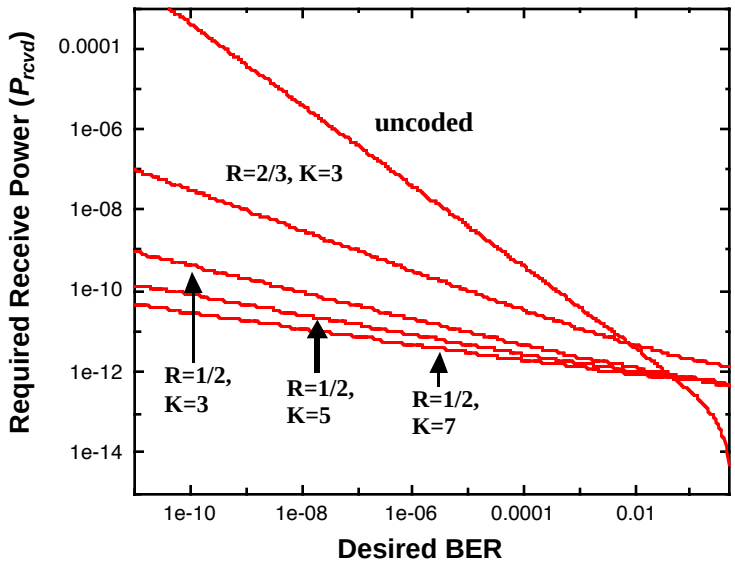


Figure 3.6: Receive power P_{rcvd} required to attain a desired bit error rate for various coding schemes, using performance parameters of a commercial radio [87].

$$E_{rx}(N, R_c, E_{decbit}) = P_{start}T_{start} + P_{rxElec}\frac{N}{R_c R} + E_{decbit}N \quad (3.10)$$

E_{decbit} is the decoding energy per information bit; the other parameters have been introduced in the previous section.

The decoding energy E_{decbit} models the energy consumed by the Viterbi algorithm on digital hardware. The energy consumed per bit is expressed as the sum of digital switching and leakage energies [88]:

$$E_{decbit}(V_{DD}, C_{bit}, T_{bit}) = C_{bit}V_{DD}^2 + T_{bit}\left(V_{DD}I_0 e^{\frac{V_{DD}}{nV_T}}\right) \quad (3.11)$$

C_{bit} is the switched capacitance per bit, V_{DD} the supply voltage, which is adjustable through dynamic voltage scaling, and T_{bit} the computational time required per bit. I_0 and n, which model digital leakage current, are functions of the process technology. V_T is the thermal voltage.

C_{bit} and T_{bit} are themselves functions of the convolutional code and the use of dynamic voltage scaling. The computational workload of Viterbi decoding is exponential with the constraint length K_c [87] since the number of states maintained by the decoding trellis increases exponentially with K_c. As a result,

$$C_{bit}(K_C) = C_0\alpha_c^{K_c} \quad (3.12)$$

and

$$T_{bit}(f, K_C) = T_0\alpha_t^{K_c}\frac{f_{max}}{f} \quad (3.13)$$

where f_{max} represents the maximum clock frequency, and f represents the actual frequency which may have been reduced due to dynamic voltage scaling. The constants C_0, α_c, T_0, and α_t can be regressed for the hardware being modeled; note that C_0 and T_0 will vary by orders of magnitude depending on the implementation fabric (i.e., ASIC versus microprocessor). Figure 3.7 graphically presents the impact of K_c on energy consumption for a SA-1100 processor.

To first order, the relation between core voltage V_{DD} and frequency f is linear, arising from the roughly inverse relationship between and circuit delay.

$$f(V_{DD}) = K_{proc}(V_{DD} - c_{proc}) \tag{3.14}$$

where the constants K_{proc} and c_{proc} adjust the slope and intercept. Now, given a delay constraint of T on the decoding of N useful bits of data, we can solve for the required V_{DD}.

$$V_{DD}(K_C, T) = \frac{f_{max} N T_0 \alpha_t^{K_C}}{K_{proc} T} + c_{proc} \tag{3.15}$$

Substituting (3.12),(3.13), and (3.15) into (3.11) provides $E_{decbit}(T)$, the decoding energy per useful bit in terms of a delay constraint on coding. Assuming an expected MAC overhead time T_{mac} for the data link, we substitute $E(T_{tot} - T_{mac}, K_C)$ into (3.11) to yield $E_{rx}(T_{tot}, N, R_c, K_c)$, the receive energy as a function of the tolerable delay T_{tot} over the link:

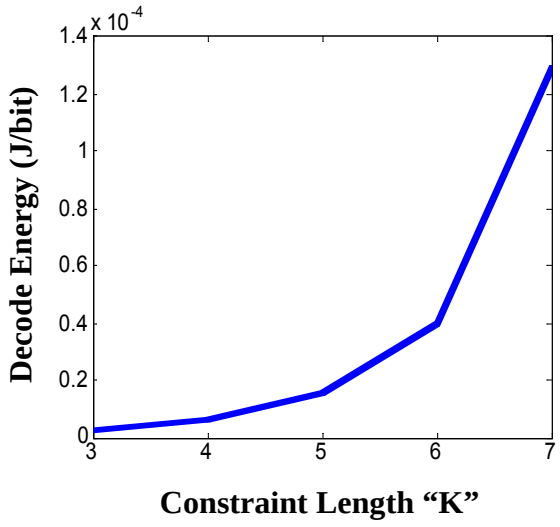


Figure 3.7: Increasing the constraint length of a convolutional code increases the amount of digital computation required for decoding. This graph represents the decoding energy of the Viterbi algorithm on a SA-1100 processor [87].

$$E_{rx}(T_{tot} - T_{mac}, N) = P_{start}T_{start} + N\left(\frac{P_{rxElec}}{R_c R} + C_0\alpha_c^{K_c}V_{DD}^2 + (T_0\alpha_t^{K_c})\frac{f_{max}}{f}V_{DD}I_0 e^{\frac{V_{DD}}{nV_T}}\right) \quad (3.16)$$

with V_{DD} defined in (3.15) above.

Figure 3.8 graphically summarizes the component model equations utilized in this section.

3.5 Middleware Policy Example for μAMPS-1

The energy models derived in Section 3.4 characterize the system energy required to attain the performance dictated by the API of Section 3.3. These expressions dictate the operational policy of the power aware middleware layer that translates the API calls for communication performance bounds into the minimum-energy hardware settings that meet those bounds. We now provide a concrete example of a real-world middleware policy by evaluating our energy models with measured parameters for the μAMPS-1 microsensor node [87]. Figure 3.9 illustrates the architecture of μAMPS-1. The parameter values extracted from μAMPS-1 are listed in Table 1.

3.5.1 Node-to-Base Station Communication

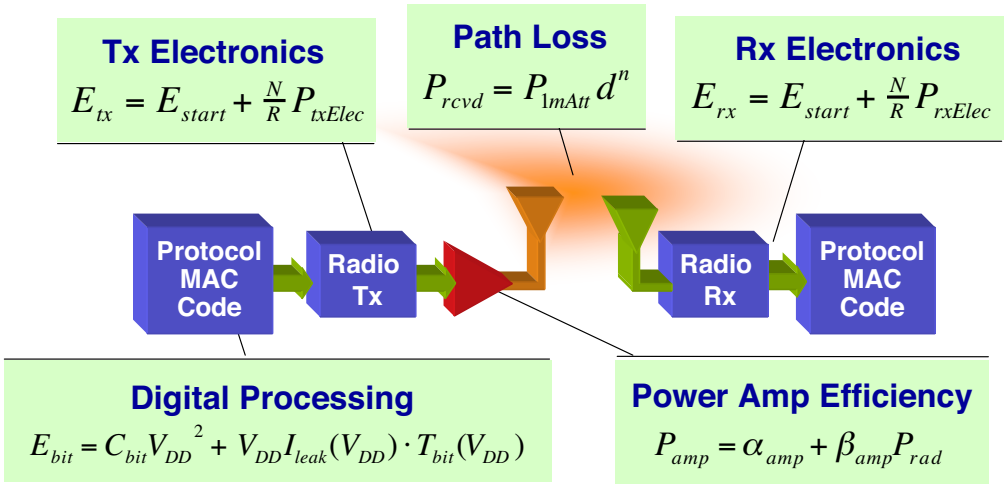


Figure 3.8: Energy models for Section 3.4.

When a node is communicating with an energy-unconstrained base station, only the energy of transmission need be considered. Given d and P_b, we can evaluate (3.9) over all codes supported by the node and choose the lowest-energy code. Back substituting the selected code (R_C, K_C) into (3.8) provides the required amplifier power setting. As convolutional encoding is a trivial computation, and the data rate for μAMPS-1 is high compared to the expected media access delay T_{mac}, delay constraints are decoupled from the hardware operational policy. We have plotted the least-energy coding and transmission power policies in Figure 3.10 for transmission of a 1000-bit packet to a base station, As the evaluated version of μAMPS-1 supports only two power levels, range and reliability

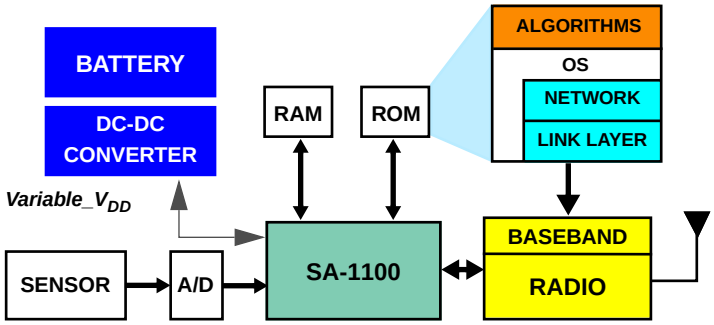


Figure 3.9: Architecture of the **μ**AMPS-1 microsensor node.

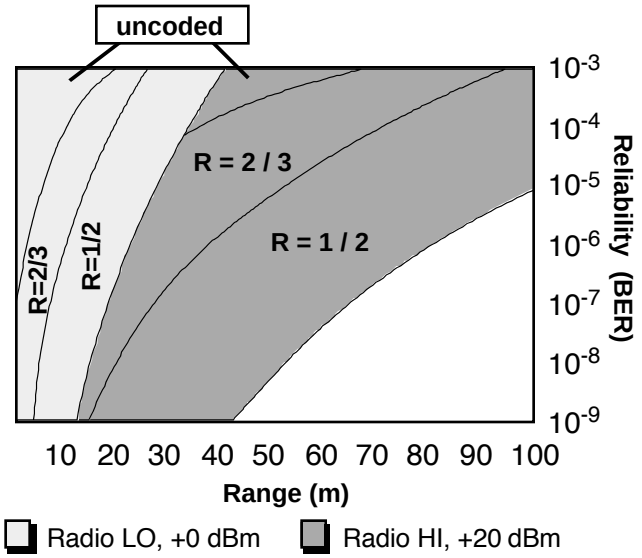


Figure 3.10: Least-energy hardware (transmit power and code rate) policy for single-hop communication given a specified reliability and range, considering *only the energy of transmission*. K_c = 3 for coded communication. N = 1000.

can be increased with greater energy-efficiency by lowering the code rate and prolonging the transmit time, rather than switching to a higher power amplifier.

3.5.2 Node-to-Node Communication

Summing the receive and transmit energies E_{rx} and E_{tx} defined in (3.16) and (3.9), provides the total energy of communication between two energy-constrained wireless nodes:

$$E_{tot}(P_b, T_{tot}, d, N) = E_{rx}(T_{tot} - T_{mac}, N) + E_{tx}(P_b, d, N) \tag{3.17}$$

The minimum energy operational policy is selected by choosing the least-energy code (R_C, K_C) from (3.17), sufficient amplifier power P_{amp} from (3.8), and the decoding processor voltage V_{DD} from (3.15), such that all performance constraints are met.

Figure 3.11 evaluates (3.17) to illustrate range scalability through the variation of transmit power and convolutional coding scheme. Both receive and transmit energy are considered for a 1000-bit packet (N=1000), a BER constraint of P_b=10^{-5}, and no delay constraint. As the required range increases, the transmit power amp is switched from low to high power and higher-rate (and higher K_c) codes are applied. Varying both parameters together has a dramatic impact on range scalability: for communication under 100 meters,

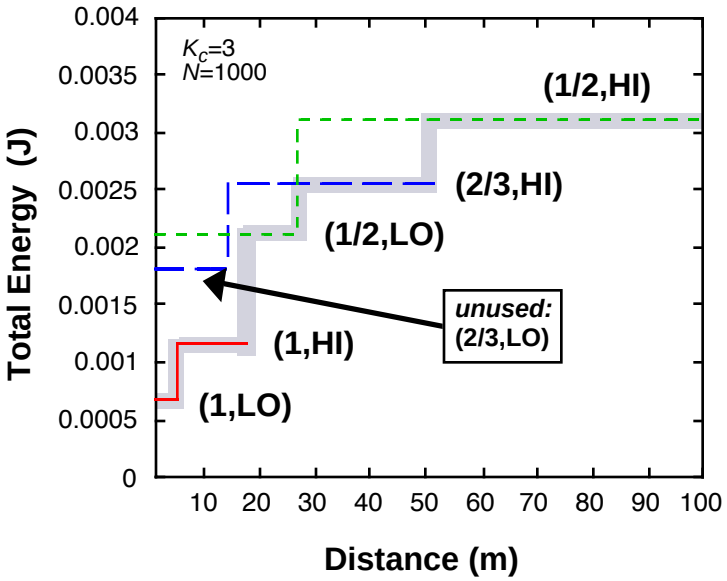


Figure 3.11: As communication distance increases, the operational policy (R_c,P_{tx}) is continuously adjusted for minimum energy. For a target BER of 10^{-5}, the least-energy policy for each d is shaded above. Note that our code and transmit power variation removes path loss effects at these distances.

the total system energy is less than linear with distance. A judicious middleware policy that accounts for the energy consumption characteristics of the hardware ensures that communication range does *not* scale as a power law with distance.

Figure 3.12 extends this analysis to two dimensions, illustrating the least-energy coding scheme given a choice of d and P_b, under no delay constraint. These parameter selections are appropriate for communication between two energy-constrained devices. Due to the high processing energy of μAMPS-1, it is now more desirable to utilize high-power transmission instead of convolutional coding when additional performance is needed for node-to-node communication. The choice between radio and processor scalability is hardware-dependent and highlights the importance of building detailed energy models for an energy-efficient operational policy. Figure 3.13 illustrates the total network energy (sum of the energies dissipated by the sender and receiver) consumed by employing the policies selected in Figure 3.12. In short, energy is scalable over nearly two orders of magnitude, realizing range scalability to well over 100 meters and BER scalability across several decades.

In the previous discussion, no delay constraint is considered, and therefore the processor is run at its lowest operating frequency of 59 MHz. Figure 3.14 illustrates the impact of delay scalability on energy consumption. Note that the vertical axis replaces reliability

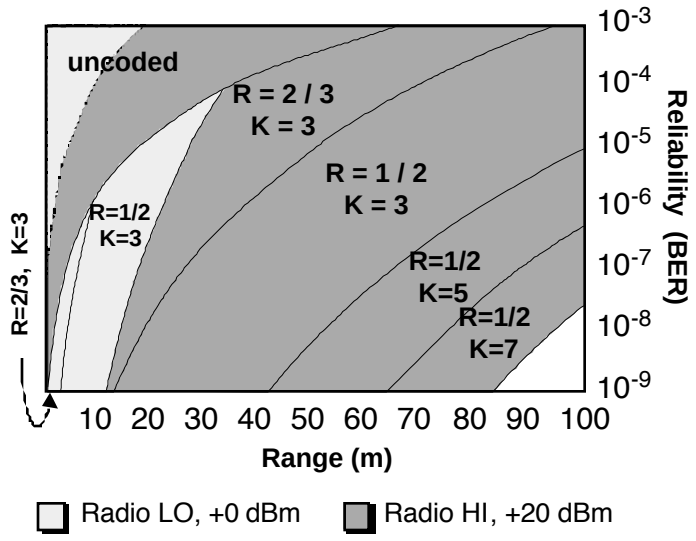


Figure 3.12: Least-energy hardware policy for single-hop communication given a specified reliability and range, considering both transmit and receive energy. N = 1000.

with delay; a fixed BER constraint of 10^{-6} is assumed. Viterbi decoding on the SA-1110 takes a fair amount of time, and codes with higher K_C require longer delays. Expanding that allowable time allows the SA-1110 to be run at a reduced frequency and voltage. The

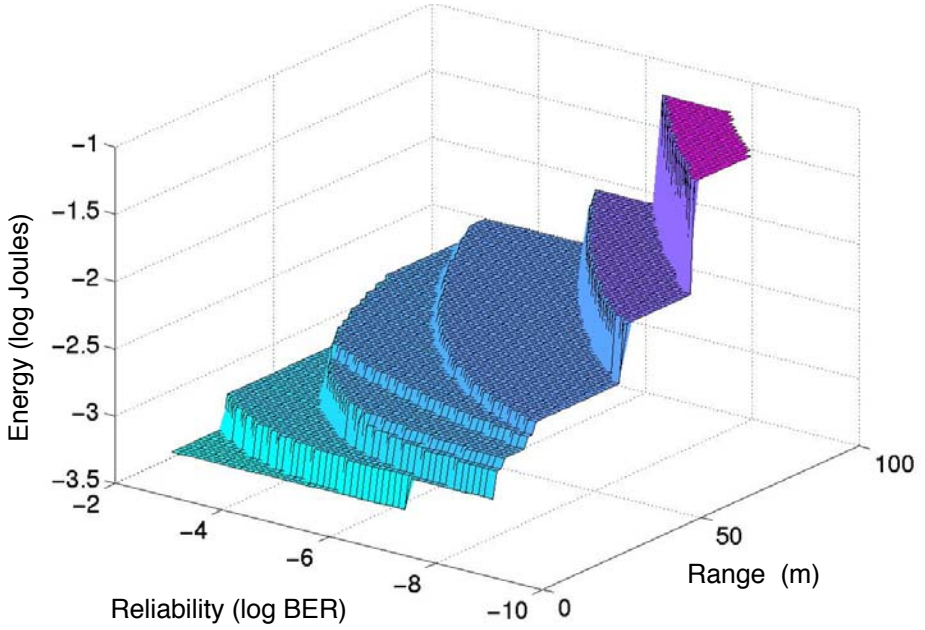


Figure 3.13: Total communication energy as a function of reliability and range for the μAMPS-1 node. Energy is monotonic with communication quality as expected for a gracefully scalable system. (Note logarithmic scale.)

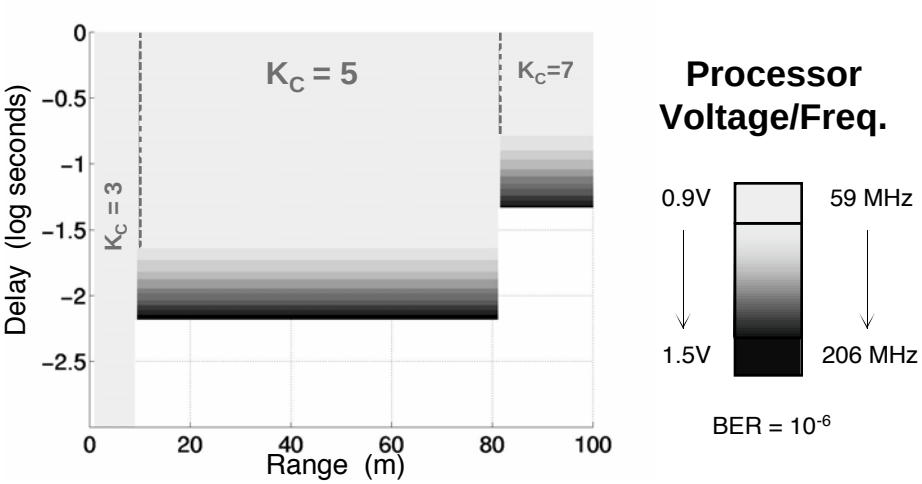


Figure 3.14: When higher delays are tolerable, dynamic frequency and voltage scaling enable an energy-delay tradeoff. The unshaded region corresponds to regions of unattainable performance for a SA-1110; the constraint length is sufficiently high that a 1000-bit packet cannot be decoded in the time allotted.

unshaded region corresponds to regions of unattainable performance for a SA-1110; the constraint length is sufficiently high that a 1000-bit packet cannot be decoded in the time allotted.

Figure 3.15 illustrates the energy consumption for the policy selections of Figure 3.14. The increase in energy with range is due to the use of stronger codes; the increase in energy with tighter delay requirements is due to the increase in V_{DD} for the same code. For $K_c = 7$, relaxing the delay constraint by a factor of four enables a 60% energy savings for the *entire* communication. The jaggedness of the tradeoff curves is due to the eleven discrete voltage/frequency pairs supported by the SA-1110.

3.6 Two Design Explorations

It is instructive to consider the impact of alternate hardware on the middleware policy. For instance, μAMPS-1 utilizes a 2.4 GHz radio; radios in lower frequency bands tend to consume less energy and have a lower data rate R. As a result, the delay and energy penalties of coding would become more significant, and a N/R term (which was previously neglected) would be added to the delay models.

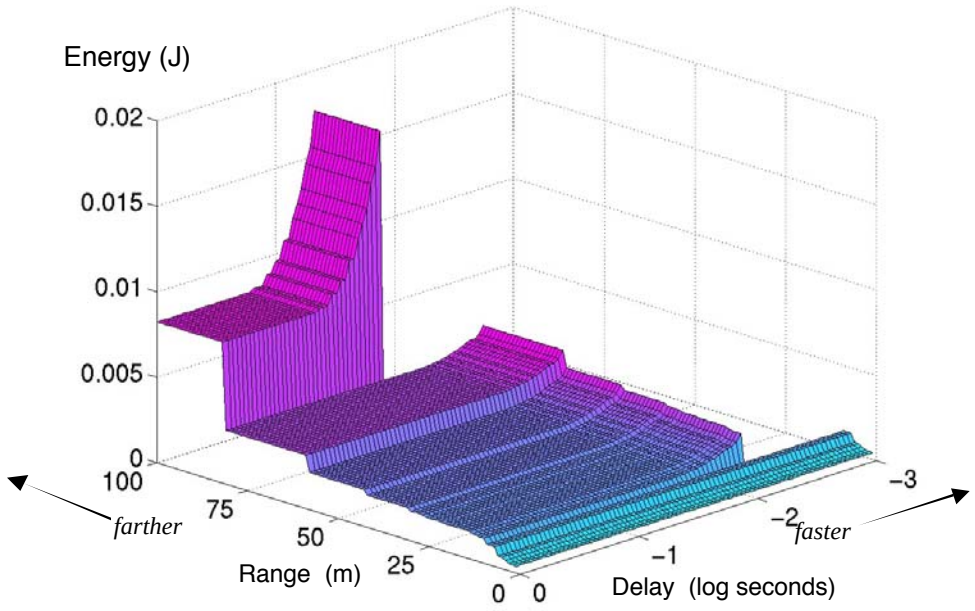


Figure 3.15: Energy savings enabled by decoding with DVS. For K_c=7, the most energy-intensive code considered, extending the tolerable delay by a factor of four allows a 60% total communication energy savings.

This section expands upon the results of Section 3.5 by considering the improved resolution of the hardware knobs and application-specific integration of individual subsystems. These enhancements can be realized in the initial μAMPS-1 design through two design changes: a power amplifier with more output power levels and an application-specific digital circuit for Viterbi decoding.

3.6.1 Hardware Knob Resolution: Six-Step Power Amplifier

From a theoretical standpoint, the optimal (minimum-energy) hardware settings for a wireless node are analog quantities. The discrete number of operational settings in most wireless communication hardware—μAMPS-1 included—is therefore an inefficiency to some degree, as we must "round up" the optimal settings to the nearest discrete setting that is available on the hardware platform. A significant example of this inefficiency on μAMPS-1 is the two-step power amplifier, which offers the choice of a 1 mW or 100 mW output power.

The resolution of the output power level on μAMPS-1 may be increased by replacing the two-step power amplifier with an amplifier supporting six intermediate power output levels, with the minimum and maximum levels unchanged. The new minimum-energy operational policy for the redesigned architecture, considering both transmission and reception energies, is illustrated in Figure 3.16. As the number of possible power and coding policies have increased threefold, the policies for power and coding are presented as separate graphs. The graphs can otherwise be interpreted in an identical manner to Figure 3.12.

Comparing Figure 3.16 with Figure 3.12 illustrates an immediate increase in efficiency. This is most clear in the upper-left hand region where transmission is uncoded. In Figure 3.12, the radio must shift to +20 dBm transmission once +0 dBm is insufficient to meet the range and reliability requirements demanded by the user. However, in Figure 3.16, all intermediate output power levels are successively used as the range and reliability are increased. Transmitting at +3 dBm requires 90 mW of total amplifier power, compared with 830 mW for +20 dBm output, an 89% difference. For the transmission and reception of a 1000-bit packet, the presence of a +3 dBm level alone enables a 64% communication energy savings for uncoded transmission.

Another worthwhile observation arises from the upper-right hand region of the plots from Figure 3.16. Virtually all of this region is effected by a convolutional code of R_C=1/2 and K_C=3, the least complex of the convolutional codes considered. A two-step radio, on the other hand, requires that more complex convolutional codes be used in this region due to the lack of additional radio power levels. The moral of this observation is that refinement of resolution in one hardware knob can therefore impact the selection of the second. More bluntly, a high resolution knob can partly mask the shortcomings of resolution in another.

How much resolution is necessary? In the end, the resolution of individual hardware knobs should be chosen to provide acceptable resolution of energy consumption as quality needs increase. Figure 3.17 illustrates the energy required for communication using the six-step amplifier as a function of range and reliability. While this energy curve offers gen-

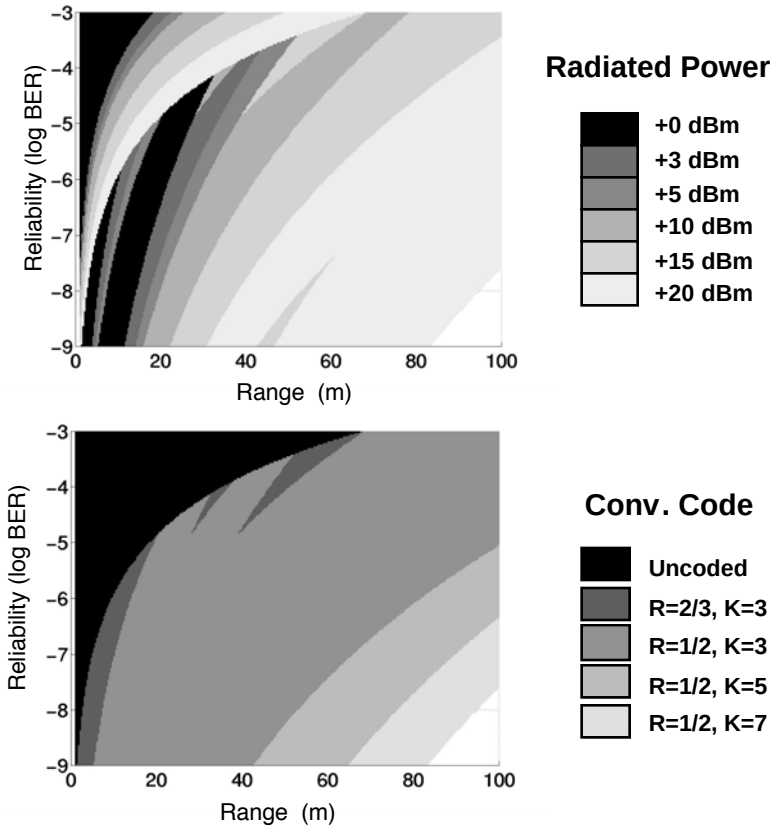


Figure 3.16: A six-step power amplifier provides more fine-grained operational policies compared to Figure 3.12. Note that the presence of additional output power settings encourage rate-1/2 coding to be utilized

tler steps than Figure 3.13, a fairly steep discontinuity remains between the highest quality tiers. Perhaps lower-rate convolutional codes could be considered to alleviate the need for energy-intensive, high-K_C coding. Finally, the overhead of selecting the minimum-energy policy should also be considered. If the middleware is designed such that the energy required to compute the minimum-energy policy increases with the number of available policies, then this would effectively place an upper limit on the number of discrete policies available.

3.6.2 Subsystem Optimization: Application-Specific Decoder

It is well known that application-specific and domain-specific digital circuits are more efficient than general-purpose microprocessors [12,25]. Considering the relative inefficiency of Viterbi decoding on the SA-1110 processor and the communication quality gains achievable with coding, it is worthwhile to move the Viterbi decoding operation onto a dedicated domain-specific processor. As [12] demonstrated a domain-specific processor that consumed three orders of magnitude less energy than optimized assembly code for the SA-1100, this quantity will be utilized for the following example.

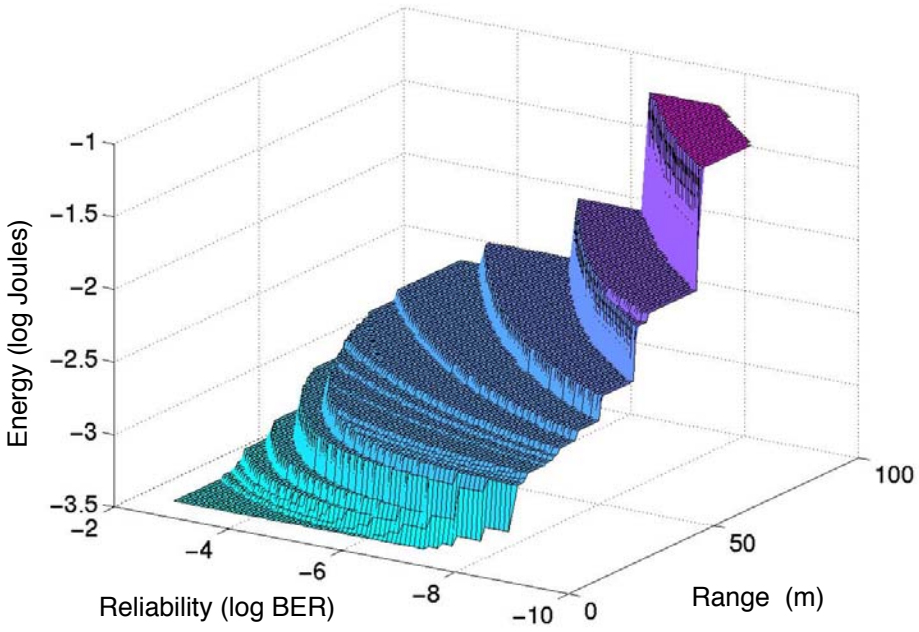


Figure 3.17: Total communication energy as a function of reliability and range, for the μAMPS-1 node with a six-step radio. (Compare with Figure 3.13.)

Figure 3.18 illustrates the minimum-energy policies for coding and output power if the digital circuitry for Viterbi decoding consumed 1000 times less power than a SA-1110 implementation. As one might reasonably expect, the operational policy now favors complex convolutional codes. Now, the use of K_C=7 convolutional codes, the most complex codes considered in this section, is preferred to raising the output power beyond +5 dBm. The energy of decoding has been lowered to the point that the only impact of coding is the increase in communication delay and the radio on-time. This is similar to transmission-only case from Section 3.5.1 that ignores receiver energy, except that receiver energy also increases in this case due to packet length expansion. If an ASIC were employed for Viterbi decoding, the energy of the radio and protocol processor (not modeled in this section) would become dominant.

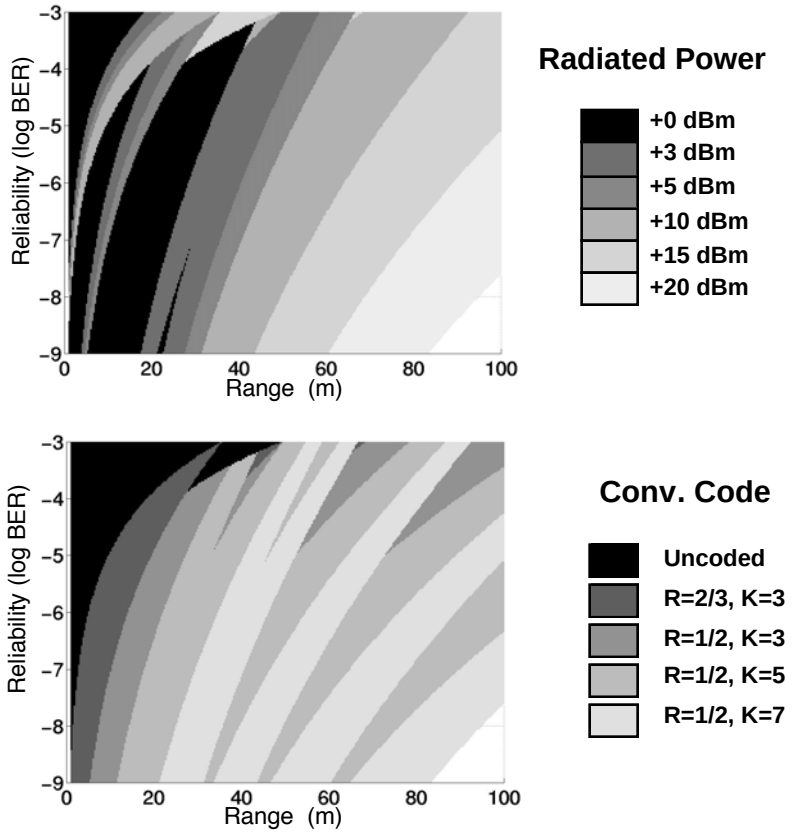


Figure 3.18: With an application-specific Viterbi decoder that consumes three orders of magnitude less power than the SA-1110, the energy penalty of coding is reduced greatly. It is now more energy-efficient to raise the code constraint length K_C to increase communication quality, rather than increasing the output power.

3.7 Real-World Demonstrations of Power-Aware Middleware

This section presents real-world implementations of energy-quality scalable middleware. To demonstrate the immediate potential for energy-quality scaling, the relationship between radiated power, range, and reliability is empirically determined using the μAMPS-1 nodes.

3.7.1 Empirical Range-Reliability Relation

An experiment was conducted with two μAMPS-1 nodes to determine the empirical relationship between radiated power, range, and reliability of node-to-node communication within a long indoor corridor. Using one transmitting and one receiving node, one thousand packets of 2000 uncoded bits each are sent at six power levels between +0 dBm and +20 dBm. The distance separating the nodes is varied between 4 and 40 meters, at intervals of 4 to 8 meters. Figure 3.19 plots the measured packet error rate P_M as a function of range and energy. It is encouraging that P_M is nearly monotonic as a function of both variables, suggesting that an energy-reliability-range trade-off is indeed possible in real-world communication.

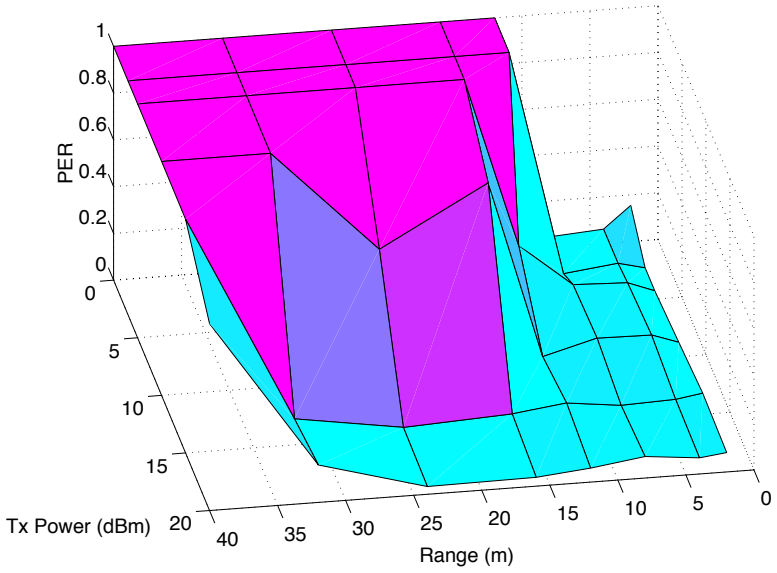


Figure 3.19: Measured packet error for uncoded 2000 bit packets transmitted between two μAMPS-1 nodes.

Figure 3.19 also suggests that the transmission of 2000 consecutive, uncoded bits without error is a challenge, to the extent that the range for reliable packet transmission at +20 dBm is under 50 meters. (The placement of the nodes upon the floor and multipath along the corridor further contribute to this low performance.) With this in mind, the impact of packet length scaling is demonstrated in Figure 3.20. These two curves plot the measured P_M as a function of packet length for two values of range and transmit power that provide borderline performance. The error rates for packets of 10 to 2000 bits are determined by the first-error positions of the 2000-bit packets from the above experiment. As expected for *uncoded* packets, longer packets are more susceptible to corruption than shorter packets since any single bit error corrupts the entire packet. However, longer packets need not be more difficult to transmit reliability. This issue is addressed in Section 4.4.

3.7.2 Middleware for Dynamic Scaling

Section 3.5 solves the middleware problem with model equations that relate hardware knobs to the performance and energy of communication. While an equation-based middleware layer could manage operational diversity imposed by the application (variations in communication performance), such an implementation would not react gracefully to changes in channel conditions. In practice, a middleware layer must also incorporate infor-

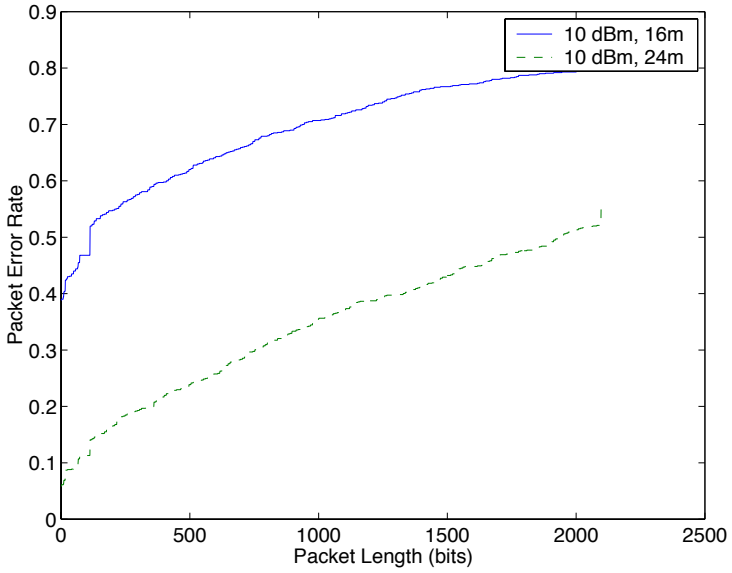


Figure 3.20: Measured error rate P_M *vs.* packet length.

mation about channel conditions of both the local and remote nodes. Figure 3.21 is a conceptual illustration of a middleware layer empowered with channel information. By collecting local and remote information such as the RSSI, signal-to-noise ratio (channel quality), and a history of received and missed packets, the middleware layer implements a feedback loop that is robust to channel variations as well as changing application commands.

3.7.3 Dynamic Range Scaling for μAMPS-1

To illustrate a basic but practical application of energy *vs.* range scaling, the μAMPS-1 nodes were programmed with a dynamic range scaling application. This application varied the transmission power of transmitting nodes to maintain reasonable performance ($P_M < 0.1$) with the minimum possible output power. In this application, packets are generated at a constant rate by the transmitting nodes, which periodically receive the current P_M as feedback from their receivers. If the receivers report a high P_M, or if the transmitters receive no feedback at all, then the transmit power is increased. If the receivers report perfect reception ($P_M = 0$) over a fixed window of time, then transmit power is reduced in order to "probe" occasionally for opportunities to communicate with lower power. Hence,

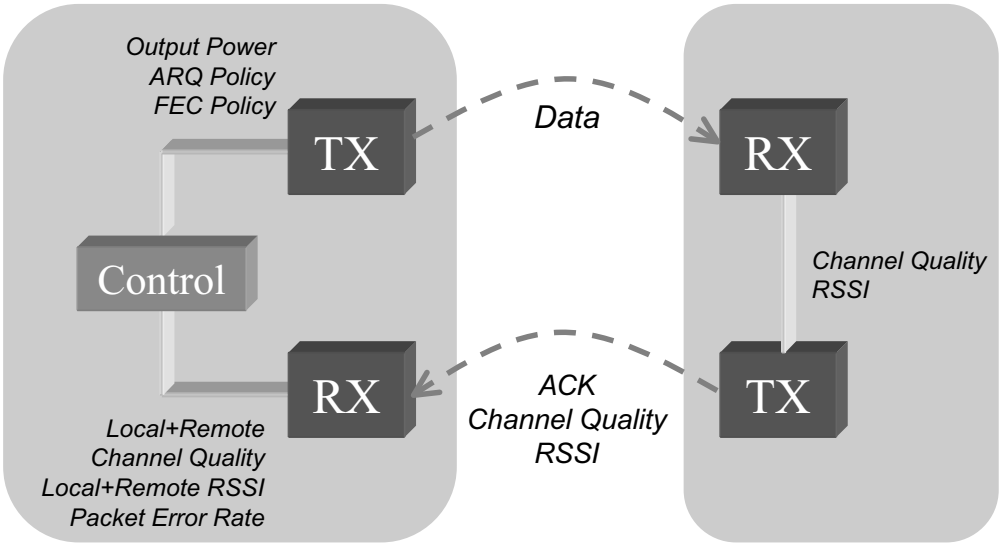


Figure 3.21: An energy-agile middleware layer must be prepared for operational diversity from the channel as well as from varying application performance demands.

as the distance between nodes is varied, or interference introduced, the system adjusts dynamically to maintain reliable communication with minimal transmit power. This implementation is analogous to the power control feedback loop of IS-95 CDMA [94].

Two μAMPS-1 were placed in adjacent rooms separated by a wooden door, and the door was closed and opened during the test period to provide operational diversity in channel attenuation. Figure 3.22 plots the output power selected by the middleware layer over a one minute test period. As the channel attenuation increases, the middleware selects higher output power levels in response to increased error rates. Once the channel attenuation decreases, a lower power level is selected. The frequent oscillation between two power levels indicates the algorithm's "probes" for improved channel quality.

3.8 Summary and Impact

In contrast to works that have considered small sections of the communication hierarchy from the perspective of a single research discipline, the framework presented in this chapter paints a horizontally and vertically comprehensive picture of the performance of a communication subsystem. Multi-dimensional trade-offs of communication performance are encouraged, and multiple dimensions of hardware energy scalability are manipulated to minimize energy dissipation. Energy models for hardware are recast as energy curves for the three-dimensional application scenario space of delay, reliability, and range, and the minimum energy hardware configuration (each a point system) is chosen. The energy models themselves are taken from measurements of representative, implemented hard-

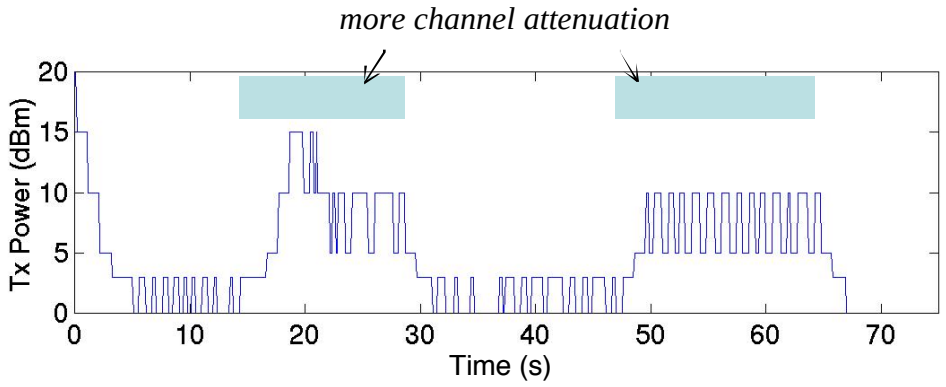


Figure 3.22: A power-aware middleware layer implemented on the μAMPS-1 node enables dynamic output power scaling in response to channel and range variations. Periods of increased channel attenuation are highlighted.

ware—the μAMPS-1 microsensor node. Hence, the accuracy and fitness of the presented energy models exceed those of many contemporary works.

Finally, this work also considers a dimension often overlooked by hardware or software-centric researchers: the interface that enables effective communication between the realms of hardware and software. This work proposes a methodology for application developers to take advantage of energy scalability in the hardware without the need for explicit knowledge about the hardware's energy consumption characteristics. The application specifies its communication performance requirements in terms of delay, reliability, range, and energy of communication, and a middleware layer makes the translation between these parameters and the settings for hardware knobs. While abstractions generally provide clarity and organization at the expense of efficiency, this *power aware abstraction* actually *encourages* the efficient use of energy consumption.

Chapter 4

Cross-Disciplinary Implications of the Energy-Scalable Framework

The previous chapter constructed a cross-layer framework for energy-quality scalability of a single point-to-point transmission, using energy models of the μAMPS-1 hardware to provide a real-world example. This chapter expands our analysis of this framework in two directions. First, we begin to consider communication among multiple nodes in a network. This chapter examines how data aggregation and multihop routing interact with our energy-scalable framework. Second, we generalize our discussion of coding, beginning with explicit relations linking code rate and performance, and concluding with information-theoretic bounds on code performance. This chapter continues to cross disciplines by applying our current model framework, which currently unites concepts from hardware and software, to the realms of communication protocols and information theory.

4.1 Quality Scaling and Data Aggregation

Data aggregation [99] fuses observations from S nodes, each of length N, into a single, high-quality data stream of length N. Hence, as depicted in Figure 4.1, aggregation reduces the number of bits forwarded by the aggregating node from NS to N, suggesting an immediate and substantial energy savings due to reduced transmission and reception time.

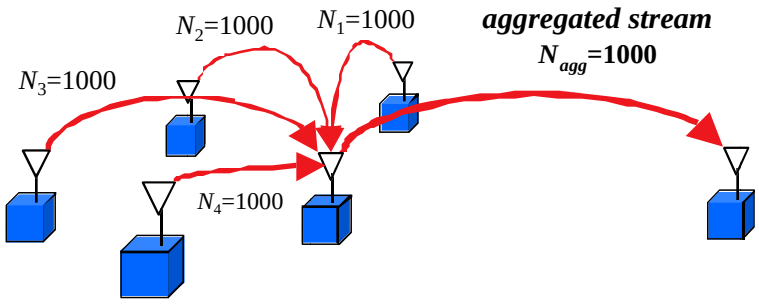


Figure 4.1: *Data aggregation* fuses the observations from multiple sensors into a single, high-quality stream.

From an application's point of view, however, aggregated data is presumably of higher value than a single stream of data. Therefore, we would expect the application to demand a higher reliability bound when aggregated data is being forwarded—perhaps P_b/S. Hence, the operating point in a hardware policy such as Figure 3.12 moves vertically downward, and the total energy consumed for the transmission changes from $E_{tot}(P_b/S, T_{tot}, d, NS)$ to $E_{tot}(P_b/S, T_{tot}, d, N)$. The energy savings from aggregation of *S* data streams is offset by any additional energy required to increase link reliability, an effect not considered by prior work on power-aware data aggregation [34].

A brief quantitative example demonstrates this point. If ten data streams of 1000 bits each are being transmitted over d = 30 meters with reliability $P_b = 10^{-4}$, then the least-energy transmission policy from Figure 3.12 is +20 dBm output power with no coding, resulting in E_{tot} = 11.1 mJ. Aggregating the ten streams into one increases the reliability requirement to $P_b = 10^{-5}$. The least-energy coding policy is now R_c=2/3, K_c=3, resulting in E_{tot} = 2.56 mJ. Although the transmission is 10% of its original length, the energy consumed is *not* 10% of its original value before aggregation, but rather 23%. Processing energy for the aggregation algorithm would further increase this figure.

4.2 Inefficiencies of Multihop Routing

4.2.1 Motivation for Multihop Routing

The fundamental expressions for hardware consumption are summarized in Figure 3.8. With the exception of the startup energy, the sum of all components of transmitter and receiver energies may be summarized as $\alpha + \beta d^n$, where α is a distance-independent term that accounts for the overheads of the radio electronics and digital processing, and β models linear losses of the power amplifier and its antenna. This aggregate expression is sufficient for a discussion of multihop energy. This section will assume that α and βd^n are expressed in units of Joules/bit.

The d^n power-law dependence between transmission power and distance (with a path loss exponent *n* typically between 2 and 4) motivates the use of *multihop routing*, the use of several shorter transmissions via intermediate relay devices [43]. Multihop, illustrated in Figure 4.2, replaces a single, long transmission with *h* shorter transmissions, resulting

in an energy consumption of $h[\alpha + \beta(d/h)^n]$ instead of $\alpha + \beta d^n$. With the ubiquity of devices promised by high-density applications, multihop is especially appealing at first glance, as there are likely to be many nodes available as candidate relays. Figure 4.3 illustrates the ideal result: as the total transmission distance increases, energy is saved by adding intermediate relays to limit the distance covered by each hop.

4.2.2 Energy of Multi-Hop Communication

Figure 4.3 illustrates the energy consumed by *h*-hop communication over a large distance. For the sake of example, the parameters α and β are chosen optimistically (so that multihop indeed shows energy savings) for a next-generation radio ASIC. The introduction of relay nodes is a balancing act between reduced transmission energy and increased receive energy. Hops that are too short lead to excessive receive energy. Hops that are too long

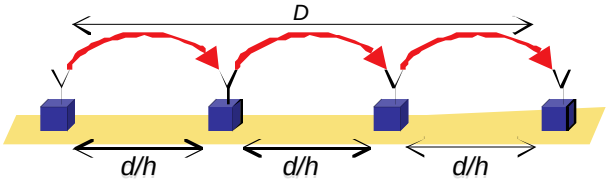


Figure 4.2: Multihop routing replaces a single, long transmission with several shorter transmissions through intermediate relays.

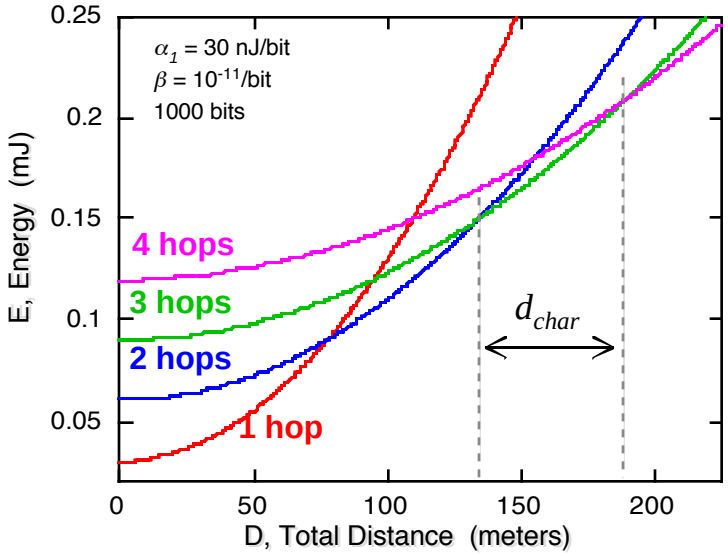


Figure 4.3: Motivation for energy conservation through multihop. As the transmit distance increases, increasing the number of hops improves energy-efficiency.

lead to excessive path loss. The seemingly even spacing of crossover points between *h*-hop and (*h*+1)-hop routing is no coincidence: the existence of a single minimum-energy hop distance, a fact that is proved in [5], yields this periodic behavior. Again from [5], the optimal inter-hop distance, known as the *characteristic distance*, can be computed as $d_{char} = \sqrt[n]{\alpha / \beta}$.

The above observations holds two noteworthy implications for microsensor networks. First, as most microsensor networks utilize a small mean distance between nodes, nearest neighbors are often the wrong candidates for energy-efficient next-hops. Second, large classes of applications exist for which the *entire network diameter* is within the characteristic distance. For these classes of networks, such as those completely enclosed within a room, machine, or small lawn, transmission is most efficient with *no multi-hop protocol at all*.

4.2.3 Energy of Real-World Radios

So far, we have not considered realistic power consumption parameters for commonly used, commercial short-range radios. An examination of these values reveals a surprising argument *against* the use of multihop.

Values of α and β for popular commercial radios are not directly provided by manufacturer's datasheets, but the distance-independent term α can be estimated by choosing a reasonable value for the distance-dependent coefficient β. A higher value for β implies that multihop is more likely to conserve energy over *shorter* end-to-end transmission distances since energy depends *more strongly* upon distance.

To determine the relative magnitudes of the fixed α and distance-dependent βd^n terms, it is instructive to find the *maximum* value of βd^n, which is the value attained by this term at the radio's maximum range, and therefore, output power. Recalling that βd^n has units of Joules per bit, the maximum value attainable by βd^n is found by obtaining the maximum radiated power provided by the radio specifications, multiplying by an estimate of amplifier inefficiency, and dividing by the radio bit rate. In other words,

$$\beta d^n = \frac{P_{radMax} \beta_{amp}}{R} \tag{4.1}$$

for a maximum radiated power P_{radMax}, radio rate R, and amplifier inefficiency factor β_{amp} (as introduced in Chapter 3). The peak *total* power consumption of the radio, also provided by specifications, is then $\alpha + \beta d^n$, and we can therefore solve for α for the transmitter. To consider total communication energy, the total energy required by the receiver—which is independent of distance—is added to α as well.

Using this approach, Table 4.1 lists the values of α and the maximum value of βd^n for a number of commercial radios, using the assumption $\beta_{amp} = 10$. Energy is given in Joules per transmitted bit including the energies of transmission and reception. Since we are trying to demonstrate that multihop does *not* save energy, we choose β_{amp} very conservatively in favor of multihop. This is a reasonable choice for short-range radios, whose maximum output powers are typically 100 mW or less.

Table 4.1: Estimated energy consumption of real-world radio hardware

	Energy per bit	
Radio Type	α	Maximum value of βd^n
2.4 KHz OOK (RFM TR1000 @ 916 MHz [83])	14 μJ	3.1 μJ
115.2 KHz ASK (RFM TR1000 @ 916 MHz [83])	372 nJ	65 nJ
11 Mbps 802.11b (Cisco Aironet 350 @ 2.4 GHz [29])	236 nJ	91 nJ
54 Mbps 802.11a (Atheros ISSCC2002 chip [93])	14.8 nJ	11 nJ

It is surprising, then, that the distance-independent α component of power consumption exceeds the maximum value of βd^n for all of the radios considered in Table 4.1. βd^n is limited by the maximum output power of each transceiver, which is typically set by FCC regulations for each band. Figure 4.4 illustrates the energy consumption of h-hop transmission for the RFM TR1000 [83] and Cisco Aironet 350 [29] radios versus distance, which is normalized using (4.1) to the maximum distance corresponding to the maximum output power. As expected by the results of Table 4.1, multihop communication *always* requires more energy than direct transmission. Additional overheads incurred by the

underlying routing protocol will further reduce the efficiency of multihop. Multihop is only beneficial when the required communication range exceeds the maximum range of the radio.[1]

While multihop remains useful for long-range radios such as microwave relays, this technique must be reconsidered for present-day, short-range radios characteristic of high-density networks. This conclusion holds until radio technologies are developed with low α without a corresponding reduction in β.

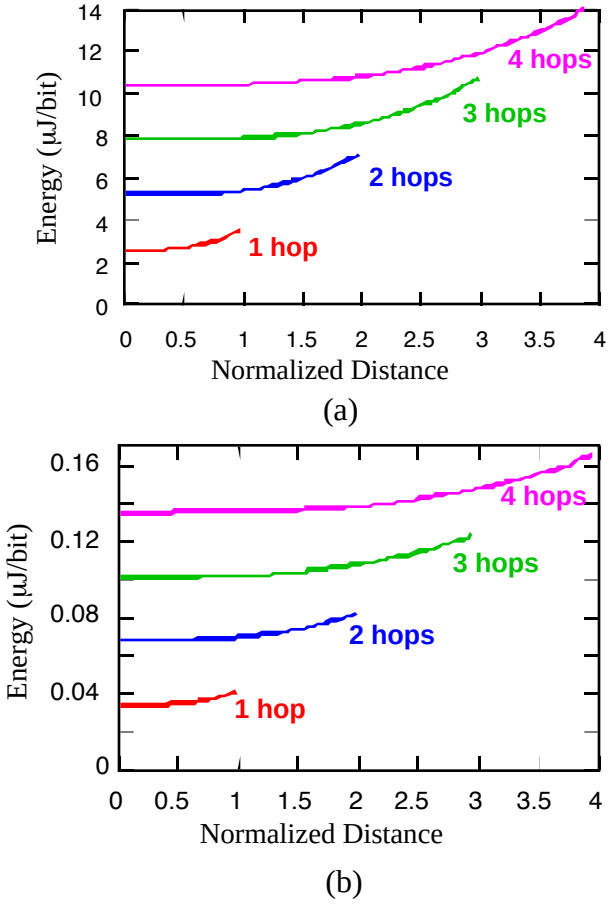


Figure 4.4: Energy consumption as a function of hops and distance for (a) an RFM on-off keyed radio and (b) the Cisco Aironet 350 802.11b radio. Each distance axis is normalized to the range of the radio at maximum output power, which need not be evaluated explicitly.

1. While multihop can improve spatial reuse of spectrum, one must keep in mind that the interference range of a radio transmission can greatly exceed the reception range. Furthermore, capacity is only improved when disjoint portions of the network are communicating with each other. Spatial capacity bottlenecks remain when all nodes are transmitting to a single base station, or when multihop communication paths cross within the interference ranges of the radios.

4.2.4 Energy Scalability vs. Multihop

The present discussion has considered multihop routing from a real-world energy perspective. When incorporated into an energy-quality framework of the previous chapter, the outlook for multihop are even more bleak.

There are two reasons why multihop routing is less desirable in a framework with scalable communication quality. First, both the end-to-end BER and delay grow with the number of wireless hops, so that the total energy consumed by an h-hop transmission changes from $E_{tot}(P_b, T_{tot}, d, N)$ to roughly $hE_{tot}(P_b/h, T_{tot}/h, d/h, N)$. (Obtaining the exact relationship between hop count and end-to-end P_b is rather involved and depends on N; P_b/h is offered for illustrative purposes.) This change in energy is illustrated graphically in Figure 4.5, using the energy curve from Figure 3.17. If an application demands an upper bound on either parameter, a multihop scheme must tighten the bounds for the individual hops, potentially requiring a more energy-intensive operational policy. This is the same effect seen in the aggregation example from Section 4.1.

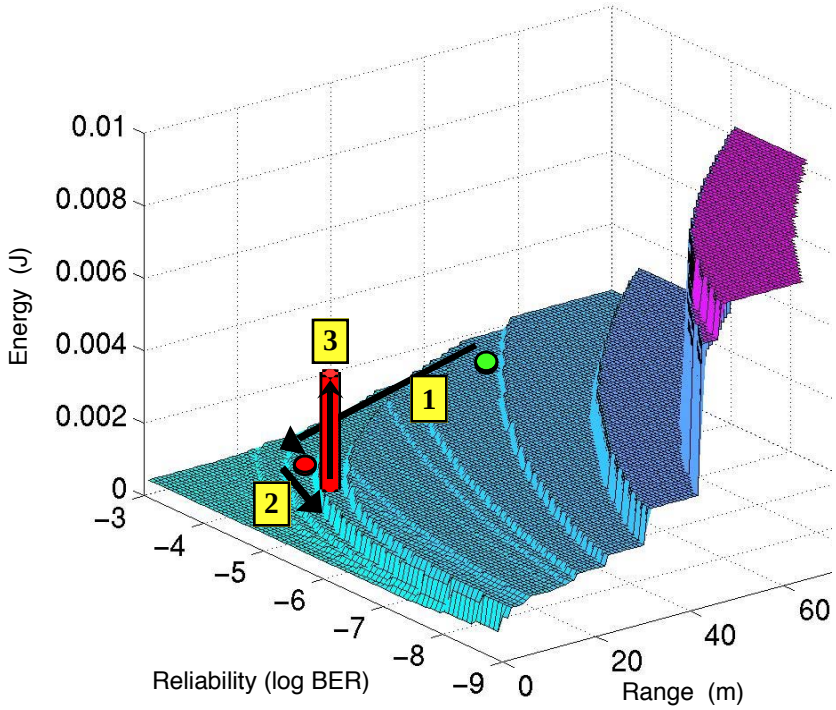


Figure 4.5: Impact of multihop on energy for a framework where multiple dimensions of communication performance are specified. Multihop (1) reduces per-hop transmission distance by h, (2) tightens the reliability constraint by approximately h, and (3) requires h times the energy of a single hop.

The second strike against multihop is that a distance-scalable communication policy using multi-dimensional hardware trade-offs already mitigates transmission path loss over moderate distances. This is apparent in the energy *vs.* distance curve of Figure 3.11 for the initial μAMPS-1 radio. Hence, a 60-meter transmission under the assumptions of Figure 3.11 consumes 3.115 mJ, while four 15-meter hops require a total of 4.636 mJ, even before any reliability constraints are considered. Multihop provides no energy benefits for μAMPS-1 until the desired transmission range exceeds the radio range at +20 dBm with a R_c=1/2, K=3 convolutional code at d > 100 m.

Figure 4.6 illustrates the minimum energy required for h-hop transmission using scalable convolutional coding and transmit power. These results are based on the six-step μAMPS-1 radio described in Section 3.6.1 and can be conceptualized as slices taken from the three-dimensional surface of Figure 4.5. These graphs include the impact of the required increase in per-hop reliability as h increases.

A worthwhile digression arises from the ability of 802.11b radios to transmit at a variable rate. While the fundamental data rate remains 11 Mbps, the data stream is coded to reduce the net bit rate to either 5.5, 2, or 1 Mbps for extended range. While the analysis presented in this dissertation suggests that rate reductions through coding are an energy-efficient way to increase distance, the minimum-energy solution for long-range 802.11b transmission is multihop! The coding scheme employed by 802.11b for "downsampling" the data rate is notoriously inefficient from a coding theorist's perspective—the codes are designed for backward compatibility with lower-rate 802.11 standards rather than for coding gain. Hence, multihop is more energy-efficient than a 5.5-fold or 11-fold decrease in rate and the corresponding increase in energy.

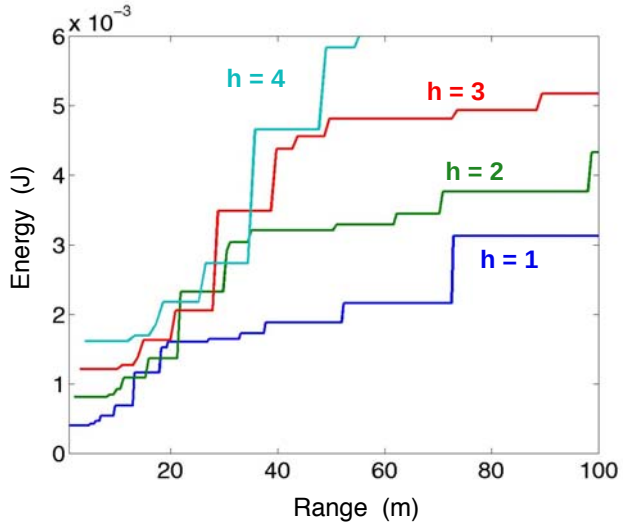
4.3 Information-Theoretic Bounds on Scalability

Chapter 3 related the BER of specific convolutional codes to the received power P_{rcvd} with regressions of simulation data. This section expands and generalizes the coding knob by introducing fundamental performance bounds for block codes to draw conclusions about the *ultimate limits* of energy scalability. As illustrated by Figure 4.7, this next exploration will combine information-theoretic bounds on the performance of coding with the real-world models of hardware energy and performance from Chapter 3. This fusion of

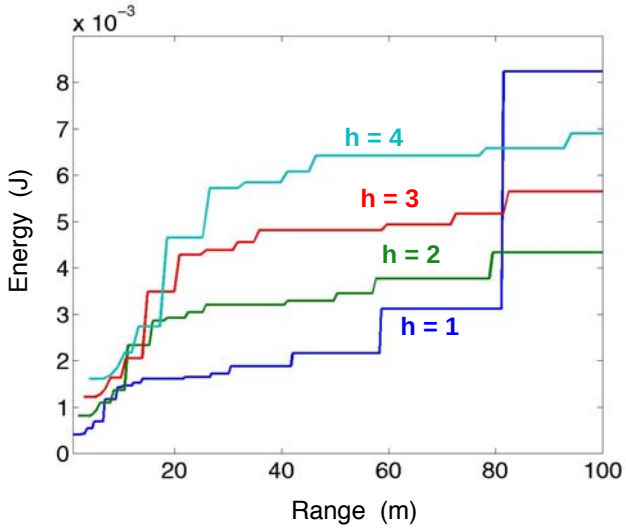
theory and practice enables the derivation of bounds on the ultimate limits of energy-quality scalability for a given hardware platform.

4.3.1 Upper and Lower Bounds on Block Code Performance

A fundamental metric of block code performance is the *Hamming distance* d_{min}. The Hamming distance is the minimum number of bits that distinguish any codeword from any another. Hence, a greater Hamming distance provides more error resilience. Specifically, a



(a)



(b)

Figure 4.6: Energy consumption as a function of hops and distance for (a) an RFM on-off keyed radio and (b) the Cisco Aironet 350 802.11b radio. Each distance axis is normalized to the range of the radio at maximum output power.

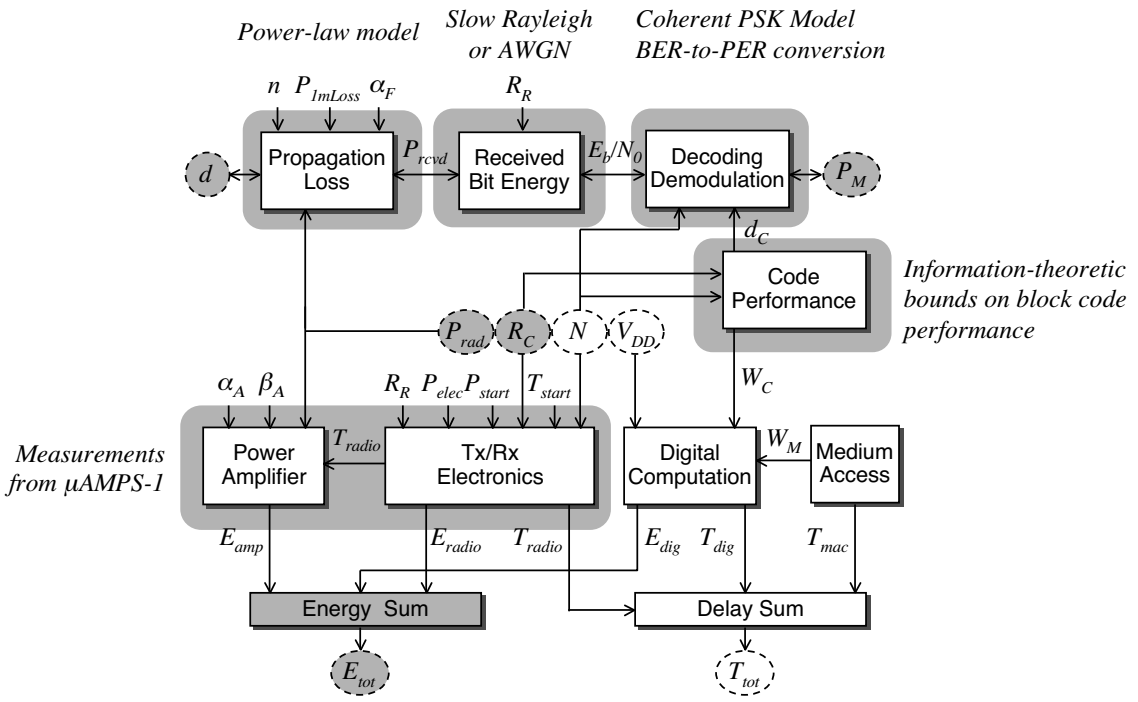


Figure 4.7: Model relations for the incorporation of information-theoretic bounds on block code performance.

code that can correct *t* bit errors requires that $t = 2d_{min} + 1$. An intuitive view is to consider the codewords 00 and 11, which form a two-word code with d_{min}=2. A single bit error will result in the reception of 01 or 10, which enables the *detection* of an error but not its *correction.* A single bit error on the d_{min}=3 code consisting of 000 and 111, however, can correct a single bit error through a simple majority vote. For instance, the reception of 010 due to a single bit error can be corrected to 000.

A metric of a block code's "efficiency" is the relationship between d_{min} and the code rate R_c, given a block length (after coding) of *n* bits[1]. If two codes with the same performance (*n* and d_{min}) have different rates, the higher-rate code—the one which can carry more information bits in the same block—is the more efficient one.

Information theory provides a number of upper bounds on d_{min}, given R_c and *n*. As these bounds basically provide an upper bound on the efficiency of large classes of block

1. In prior chapters, the capital letter *N* was defined as the number of information (pre-coded) bits. Lowercase *n* is temporarily utilized here as the block size *after* coding to conform to the conventions for the information-theoretic code bounds introduced. The energy-quality exploration that follows will utilize the original definition and substitute N/R_C into the bounds in place of *n*.

codes, we can incorporate these bounds into our model framework to find the *ultimate limits* of energy scalability offered by block coding.

This exploration considers four upper bounds on code performance—the singleton, Hamming, Plotkin, and Elias bounds. We will consider the asymptotic forms of the bounds that relate R_c to the normalized Hamming distance d_{min}/n in the limit as $n \to \infty$. These forms greatly simplify the expressions while only nominally loosening the bounds for the fairly large n ($n >>$ 1000 bits) typical of wireless packets. The bounds discussed below are plotted in Figure 4.8 and chosen from [78].

The *singleton bound*, the loosest of the upper bounds, represents the maximum possible Hamming distance for any block code, whether linear or nonlinear. Its asymptotic form is

$$\frac{d_{min}}{n} \leq 1 - R_C \tag{4.2}$$

The repetition code is the only binary code to meet this bound with equality; other nonbinary codes that meet the bound are referred to, appropriately, as *maximum-distance-separable* codes.

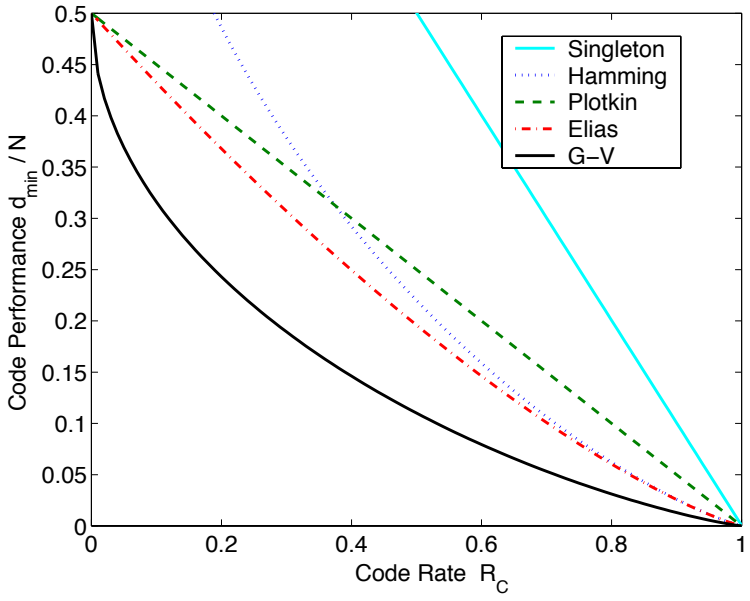


Figure 4.8: Information-theoretic bounds on the performance of block codes for $n \to \infty$.

The *Plotkin bound* applies to linear block codes and is tighter by a factor of two:

$$\frac{d_{min}}{n} \leq \frac{1}{2}(1 - R_C) \tag{4.3}$$

The *Hamming bound* also applies to linear block codes and has the form

$$H_2\left(\frac{d_{min}}{2n}\right) \leq 1 - R_C \tag{4.4}$$

where $H_2(p)$ represents the binary entropy function $-p\log(p) - (1-p)\log(1-p)$. As shown in Figure 4.8, this bound is tighter than the Plotkin bound for higher-R_C codes.

Finally, the *Elias bound* is a upper bound that is asymptotically $n \to \infty$ tight. It is given by

$$\frac{d_{min}}{n} \leq 2A(1 - A) \tag{4.5}$$

where the parameter A is defined as

$$R_C = 1 - H_2(A) \tag{4.6}$$

We reiterate that these bounds are asymptotic for infinite n; codes designed for finite (typically short) n may beat all but the singleton bound. For instance, Proakis [78] notes that several BCH codes with n=63 will beat the asymptotic Elias bound, and several BCH codes with n=31 will beat the asymptotic Plotkin bound.

The previous four expressions were *upper* bounds on code performance. A lower (worst-case) bound on code performance exists in asymptotic form only. The Gilbert-Varsharmor (G-V) bound is expressed as

$$R_C \geq 1 - H_2(d_{min}/N) \tag{4.7}$$

By placing a lower asymptotic limit on block code performance, the G-V bound postulates the existence of *asymptotically good codes* [95]. Such codes have the property that d_{min} scales linearly with N *for arbitrarily large* N. In other words, the number of errors corrected by such a code scales linearly with packet length, resulting in a constant "bit error correction rate." The potentially dramatic implications to our framework are revealed in Section 4.4.

4.3.2 Additional Model Equations

The bounds discussed above will specify the "code performance" block in Figure 4.7. We now specify the models that relate d to the packet error rate P_M in the top row of the figure.

The path loss expression (3.7) is utilized here to convert d to P_{rcvd}. For binary or quadrature modulation (one bit per dimension), P_{rcvd} can be converted into the signal-to-noise energy ratio per bit E_b/N_0 as

$$E_b/N_0 = \frac{P_{rcvd}}{R_R N_R N_{th}} \tag{4.8}$$

for a radio rate R_R, radio noise figure N_R, and thermal noise N_{th} = -174 dBm. Following the conventions of coding theory, bits and bit-rates in this context refer to the *coded* bits output by the radio (bits n per second). If desired, additional in-band AWGN terms may be added to the denominator of (4.8).

E_b/N_0 may be converted to BER with knowledge of the fading characteristics of the channel. For an AWGN channel without fading, this relation is often of the form

$$P_b = Q\left(\sqrt{k\frac{E_b}{N_0}}\right) \tag{4.9}$$

where k depends on the modulation technique used. For instance, $k = 2$ for binary antipodal signaling (for instance, binary PAM and PSK) while $k = 1$ for binary orthogonal signaling [78]. In a slow Rayleigh fading channel, the relation can be approximated as

$$P_b = \frac{1}{k(E_b/N_0)} \tag{4.10}$$

For coherent binary PSK, $k = 4$, and for coherent FSK, $k = 2$. k is reduced by a factor of two for the non-coherent detection of either case [52]. For consistency with the channel model of Chapter 3, the results that follow will assume Rayleigh fading by utilizing (4.10) and $k = 2$.

Converting the bit error rate P_b to the *packet* error rate P_M is fundamentally accomplished through a summation of the binomial cumulative distribution function (CDF):

$$_M = \sum_{k=t+1}^{n} \binom{n}{k} P_b^k (1-P_b)^{1-} \tag{4.11}$$

where t is, as before, the number of errors that can be corrected by the packet's block code and n is the coded packet size. (4.11) sums the probabilities that $t+1$ or more errors will occur for the BER P_b. To facilitate calculations and maintain monotonicity for non-integer values of t, (4.11) is evaluated through the incomplete Beta function. Details of this common procedure may be found in mathematical references such as [77]; the important point is that the end result is a smooth function over real values of t.

4.3.3 Ultimate Limits on Scaling for μAMPS-1

We now have all the relations necessary to fuse performance bounds for block code performance with the energy models (3.3)-(3.6) for μAMPS-1. As in the previous chapter, the goal is to find the minimum energy coding and output power policy (R_C, P_{rad}) that corresponds to a given range and reliability constraint (d, P_M). We will find this policy using each of the five code bounds from Section 4.3.1.

Due to the complexity of the final expression for P_M as a function of d, R_C, and P_{rad}, the optimization is unfortunately not convex. Fortunately, d, P_M and energy are both monotonic with R_C, and P_{rad}, so the minimum-energy selection of (R_C, P_{rad}) may be found through what is effectively a two-dimensional binary search in the (R_C, P_{rad}) space. This approach can bound the performance of any hardware under any channel conditions that can be modeled with tractable equations.

Using this approach, the minimum-energy selections of (R_C, P_{rad}) under five information-theoretic code performance bounds are found for μAMPS-1 and plotted in Figure 4.9. Since applications and protocols often expect packets of fixed [maximum] size, the packet length *before coding* is assumed fixed at $N = 5000$, and the selection of a code rate R_C results in a variable length *after coding* of $n = N/R_C$.

Note that tighter upper bounds on code performance (such as Elias) predict higher radiated power, while looser bounds (such as Plotkin) that bound code performance more optimistically predict lower radiated power. Recall that the singleton bound, while loose for linear block codes, is the ultimate performance limit of *any* linear or nonlinear block

code. Hence, the policy selections of (R_C, P_{rad}) under this bound represent the *minimum possible output power* to transmit at a quality (d, P_M) for the modeled hardware and channel characteristics. The results for the G-V represent the *maximum* necessary output power for asymptotically good codes, with the proviso that such codes are yet to be discovered. The inflections present on the graphs for the G-V bound are due to the 100 mW output

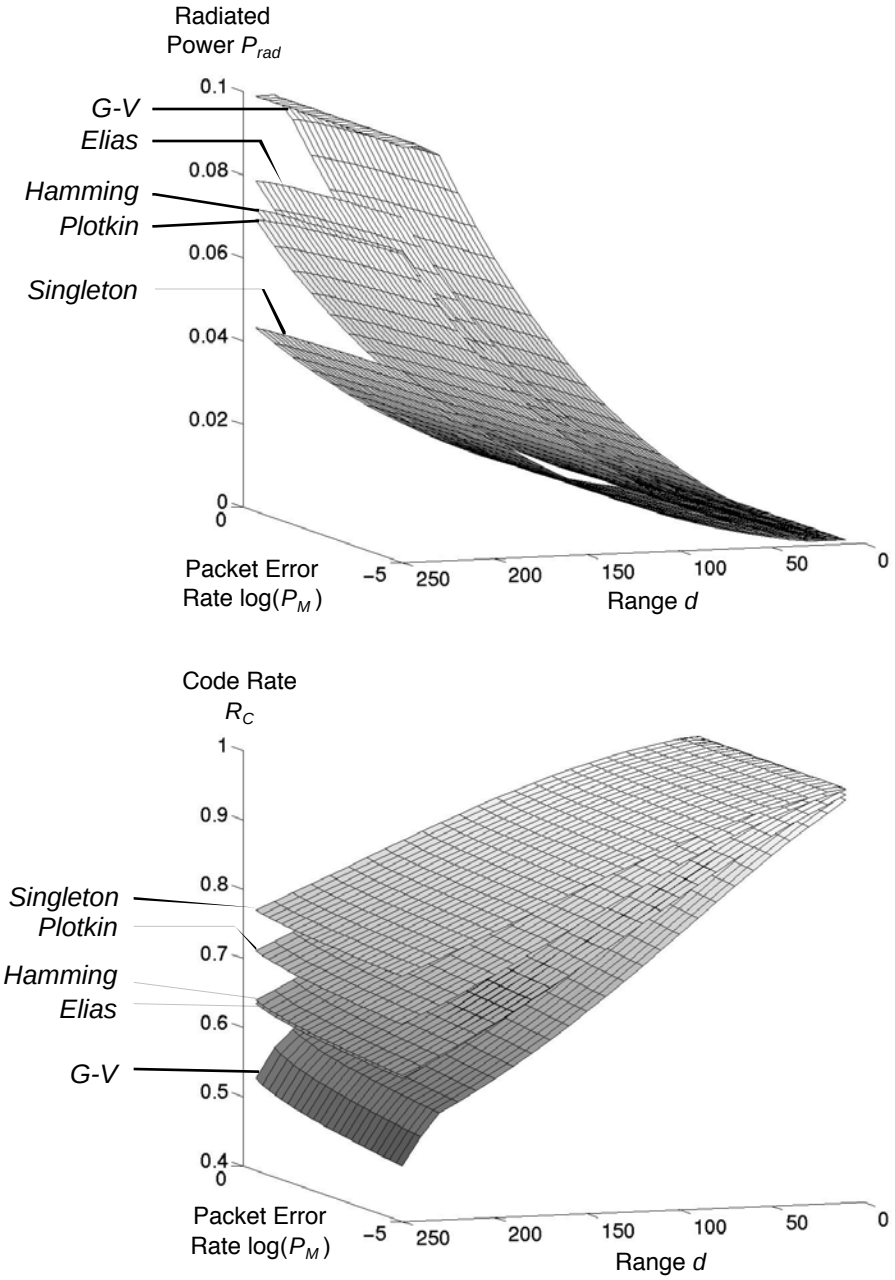


Figure 4.9: Minimum-energy policies for radiated power P_{rad} (above) and code rate R_C (below) using five bounds on code performance.

power limit of μAMPS-1; additional coding must compensate for the output power limit when additional performance is required. Removing limits on output power consumption would remove the inflection as well.

For completeness, the energy consumed by the operational policies of Figure 4.9 is illustrated in Figure 4.10. As shown by the figure, a *lower* bound on code performance provides an *upper* bound on energy while an *upper* bound on code performance places a *lower* bound on energy. By fusing information-theoretic results and real-world hardware models, the fundamental limits of the energy scalability of coding on a real-world system have been derived.

These energy calculations do not include the cost of decoding, as the relationship between performance d_{min} and encoding or decoding complexity vary heavily from code to code. Depending on the energy-efficiency of the digital hardware, including this cost would encourage the use of additional transmission power P_{rad} instead of stronger codes, as seen in Chapter 3.

It is notable that the energy *vs.* performance curves scale more heavily in one dimension than the other. Energy increases with transmission distance in a greater-than-linear fashion, but the dependence between energy and the reliability P_M are nearly indiscernible

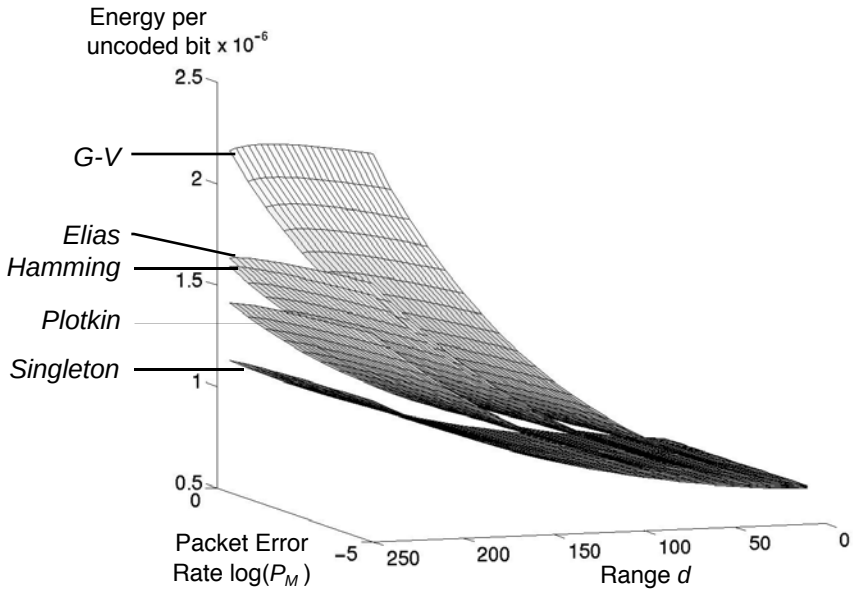


Figure 4.10: Energy consumption using the policies selected by Figure 4.9. These results may be interpreted as fundamental limits on the energy scalability of μAMPS-1.

in comparison. To provide a clearer perspective, Figure 4.11 plots the energy as functions of reliability *or* range, with the second parameter held constant. With the range fixed at $d = 200$, energy scales by less than 9% over four orders of magnitude in packet error rate. With reliability fixed at $P_M = 0.01$, energy can scale by more than a factor of two as the range increases from $d = 50$ to $d = 250$ meters.

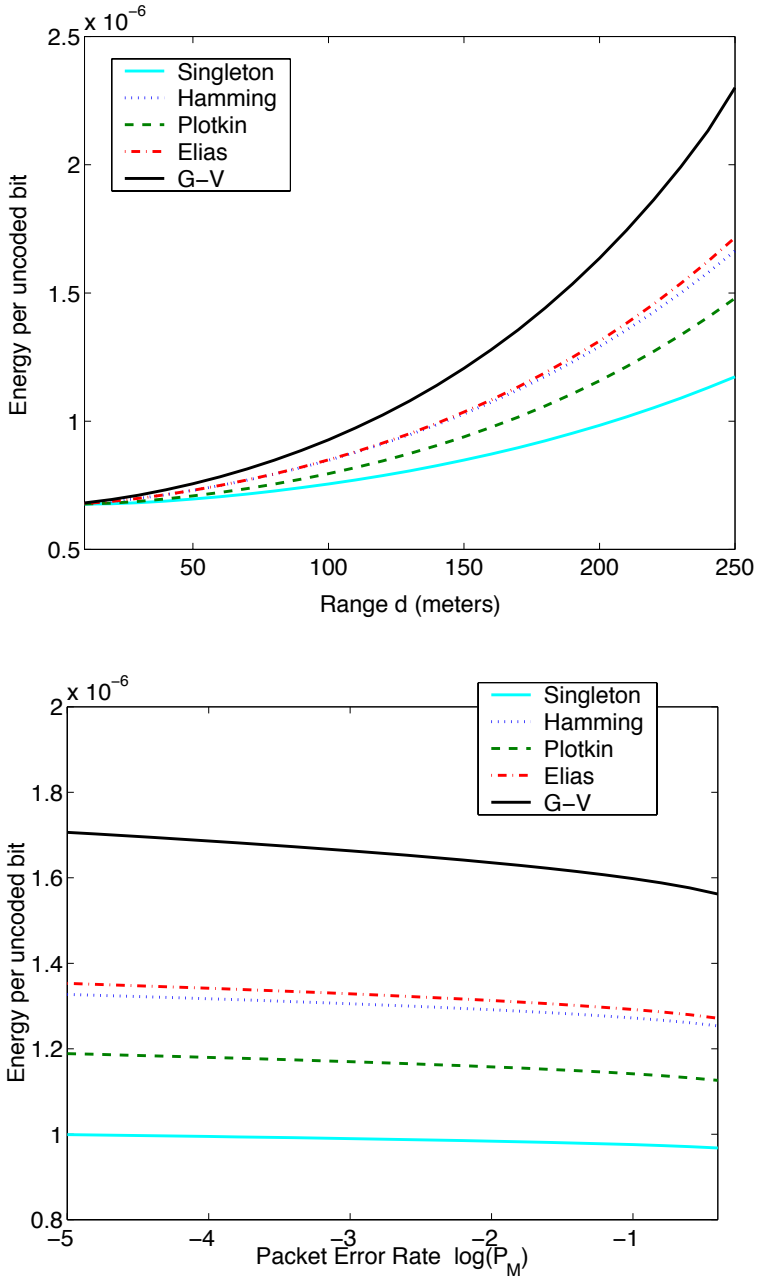


Figure 4.11: (Above) Energy as a function of range d, holding P_M fixed at 0.01. (Below) Energy as a function of P_M, holding d fixed at 200 meters.

The relative independence between reliability and energy suggests that little energy is conserved by scaling back the reliability of communication, a contradiction to the results of Chapter 3. The chief reason is the subtle difference in reliability metrics between the explorations. The reliability metric in Chapter 3 is the bit error rate, which is the most convenient metric for convolutional codes. The reliability metric here is the *packet* error rate for a fixed-length packet before coding, a metric more appropriate to block coding where each packet is encoded as a single block. The hypothesis here is that small changes in the bit error rate are amplified in the packet error rate, so that "a little energy goes a long way."

4.4 Block Coding and Packet Length

The previous section joined the notions of energy-quality scalability and expressions for block code performance that related R_C and d_{min}. This final section extends this idea one additional (and more concrete) step to illustrate how coding may counteract a hardware overhead introduced in Section 1.3.3. This section considers the impact of varying an additional low-level knob—the packet length N—on the high-level communication metrics of energy, reliability, and delay. As is our custom at this point, the parameters and relations relevant to this section are illustrated in Figure 4.12.

4.4.1 Trade-offs of Short Packets

Recent work has demonstrated that reliability can be scaled upward by reducing packet length [47]. For a channel with a constant BER, longer packets provide more opportunities for error introduction. Hence, shorter packets are more reliable than longer ones.

Unfortunately, short packets exact a heavy energy toll on the radio transceiver. It is reasonable to assume that a packet-based radio would shut down between packet transmissions to conserve energy. From (3.6), short packets reduce T_{on} while the radio's startup time T_{start} remains constant. Hence, the overall energy *per bit* increases since a constant startup energy is amortized over fewer bits. Figure 4.13 illustrates the dramatic impact of startup energy using parameters from Table 3.1. For $N < 2000$, the impact of startup energy is far greater than expansion of the packet through error correction.

4.4.2 Models for BCH Block Codes

To provide a specific example of block code for this analysis, the Bose-Chaudhuri-Hocquenghem (BCH) block code is selected. BCH codes are convenient for this analysis

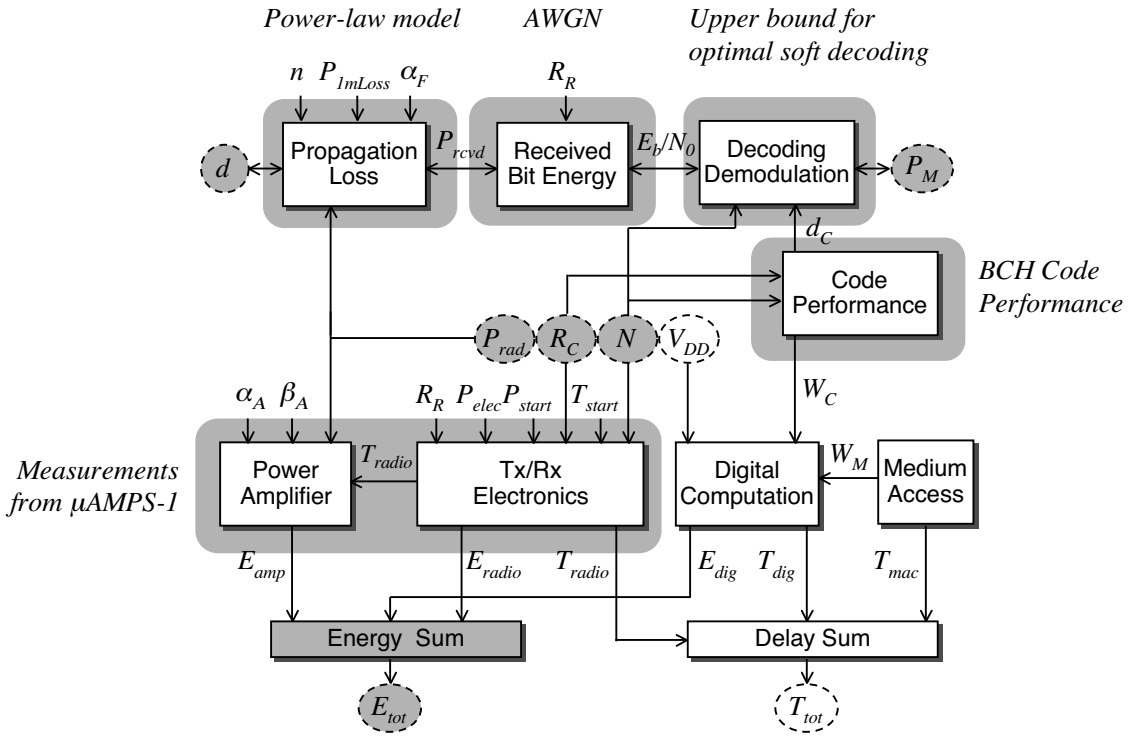


Figure 4.12: Relations and parameters relevant to Section 4.4.

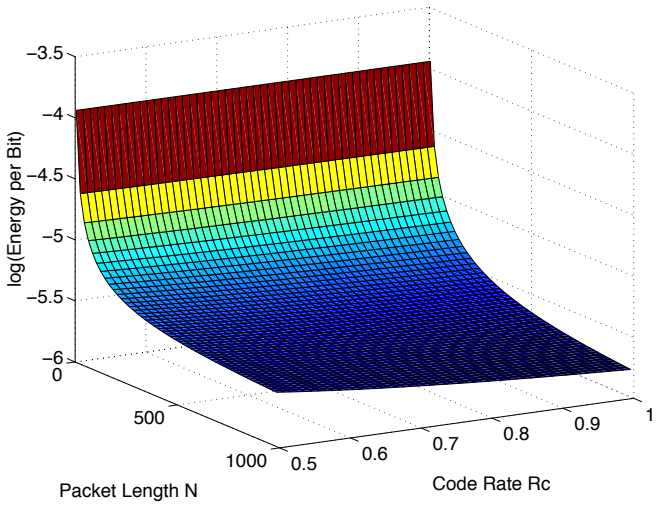


Figure 4.13: Radio energy as function of R_c and N.

since their performance is parameterizable and boundable in terms of R_C and d_{min}, in a manner similar to the bounds in Section 4.3.1. Encoding an N-bit packet using a BCH block code of rate R_C and (coded) block length n provides a performance of

$$d_{min} \geq 1 + \frac{n(1-R_C)}{\log_2(n-1)} \tag{4.12}$$

To simplify the calculations for this section, the expressions (4.8)-(4.11) relating E_b/N_0 with the packet error rate P_M are replaced with the following upper bound on P_M for optimum soft decisions of block codes for coherent phase-shift keying [78]:

$$P_M \leq \frac{1}{2}\exp\left(R_C d_{min}\frac{E_b}{N_0} + nR_C \ln 2\right) \tag{4.13}$$

Tighter, more accurate bounds can be derived for specific classes of codes, but (4.13) is a conservative approximation valid for *all* linear block codes and is well-suited for high-level analysis. All other model relations applicable to Section 4.3 are retained here.

4.4.3 Impact of BCH Coding for Short Packets

Energy is saved by buffering and combining several short packets into a single, longer packet, so that startup energy is dispersed over more bits. This is a simple and effective

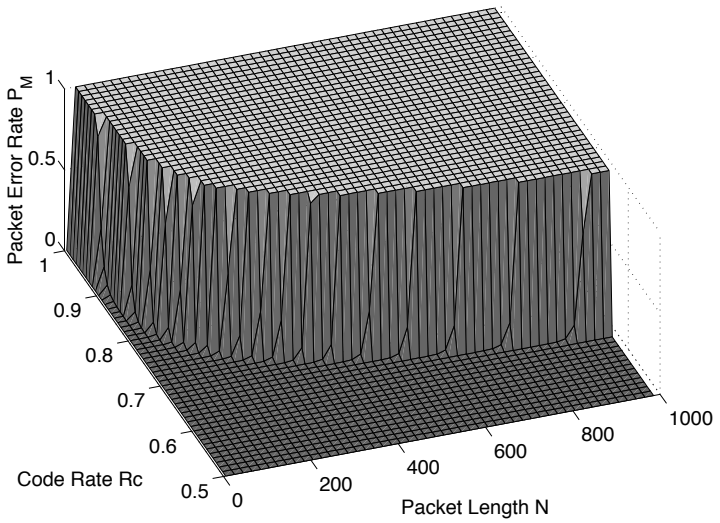


Figure 4.14: Encoding longer packets with lower-rate BCH codes can counteract the reliability issues of longer packets.

energy *vs.* latency trade-off. Unfortunately, packet reliability seems sacrificed as well, for longer packets—at least so far—are less reliable.

We now add the impact of coding. Encoding a packet of length N with a BCH block code of rate R_C reduces the packet error rate P_M as illustrated by Figure 4.14. The key to this result is to utilize a family of R_C-scalable codes, such as BCH, whose performance (d_{min}) increases as R_C decreases. Hence, lower-rate BCH codes result in a lower P_M so that longer packets may be transmitted as reliably—or more reliably—than uncoded, shorter ones. Moreover, d_{min} for BCH codes increases with N to a certain extent, which provides further benefits for longer packets.

Using R_C and N as energy-scalable knobs, we may now effect an energy *vs.* latency trade-off to control startup energy. Consider a source such as a real-time sensor node that generates data at a constant bit rate R_d. The delay of buffering N bits into a single packet can be modeled as a medium access delay of $T_{mac} = 1/R_d$, which contributes to the total latency T_{tot}. Now, if we are *given* latency and reliability requirements of T_{tot} and P_M, we can use (4.13) and (3.6) to compute the code rate R_C that will meet this level of performance with minimum energy. Figure 4.15 illustrates the optimal R_C selection for $P_M = 0.05$ and varying T_{tot}, as well as the energy consumed at each optimal selection. Joining five 100-bit packet transmissions into a single 500-bit transmission reduces E_{tot} by nearly a factor of three. Using coding, we are able to maintain a constant P_M for the longer packets that are essential for dealing with startup energy.

4.4.4 Implications of the G-V Lower Bound

This exploration concludes with the replacement of (4.12), the bound for BCH code performance, with (4.7), the G-V bound. The code rate R_C required to maintain a given packet error rate P_M is plotted in Figure 4.16. In contrast to BCH codes and Figure 4.15, a code that satisfies the G-V bound requires a *constant* rate that is independent of N!

This is precisely the meaning of the *asymptotically good code* that is postulated by the G-V bound. Recall that the G-V bound sets a lower bound on the performance of codes whose performance d_{min}/n remain constant as n asymptotically grows to infinity. The number of errors corrected by such a code therefore scales linearly with packet length, resulting in a constant “bit error correction rate.” If this “bit error correction rate” exceeds

the channel's BER, then the code would asymptotically correct all errors ( ) as N

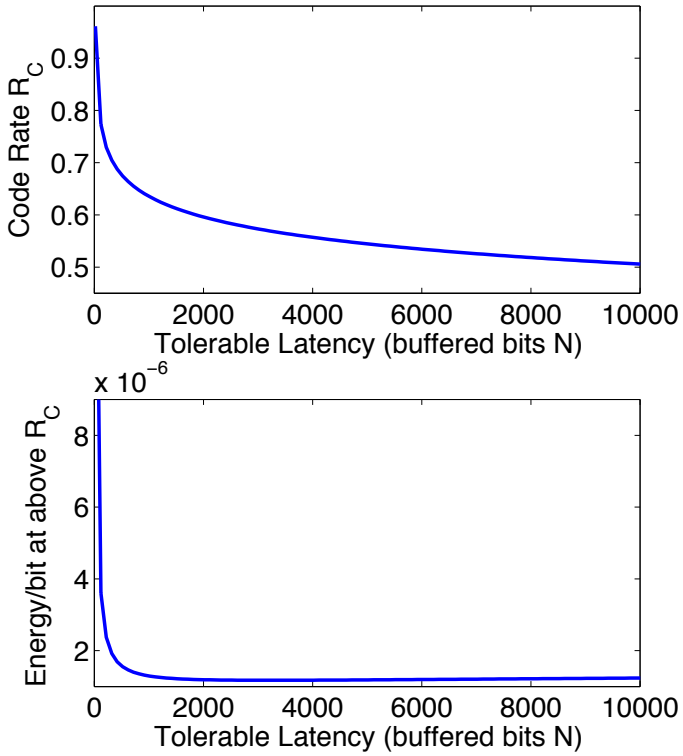


Figure 4.15: Buffering short packets together results in energy savings. Reliability is maintained by selecting R_C to meet P_M=0.05 at minimal energy.

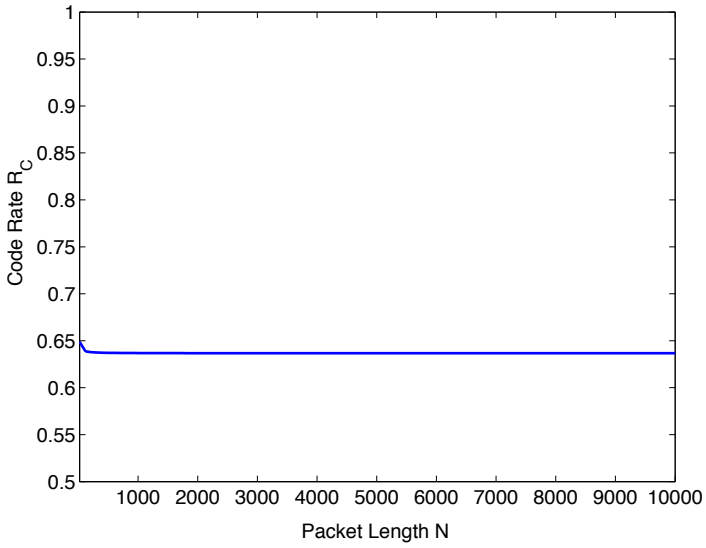


Figure 4.16: An asymptotically good code (as postulated by the G-V bound) requires a constant rate to maintain a given P_M, regardless of N.

tends toward infinity! The discovery of asymptotically good codes that are computationally tractable would have profound implications for quality-scalable, coded communication.

4.5 Summary and Impact

This chapter has applied the framework introduced in Chapter 3 to cross-disciplinary explorations into protocols and coding theory. Beyond the expansion of our energy and performance-conscious framework into new disciplines, the key contributions of this chapter are a series of results that challenge conventional wisdom about energy-efficient communication. For instance, our framework has revealed previously neglected or unnoticed inefficiencies in data aggregation and multihop routing. The multi-dimensional scaling of coding and transmit power provide energy-efficient transmission range scaling that is superior to multihop routing for short- to medium-range communication. Moreover, both multihop and data aggregation have hidden inefficiencies for end-to-end communication with fixed delay and reliability bounds.

Two detailed explorations into the impact of coding on energy scalability reveal additional surprises. The argument that shorter packets are more reliable and therefore more efficient does not hold. The energy consumption characteristics of radios favor longer packets, and coding allows us to maintain reliability for longer packets. Finally, the incorporation of bounds and postulates from information theory allows the derivation of the ultimate bounds on the energy-quality scalability of wireless communication for any given hardware platform.

Chapter 5

Application-Specific Protocols for Microsensor Networks

The previous chapter has considered the energy of communication within a single node, forging ties between power-aware communication hardware and applications with scalable communication demands. This chapter takes the broader view of energy consumption over the entire network; "system energy" is now the sum of the energies of all nodes in the network.

Designing energy-efficient protocols for high-density networks of thousands of nodes can be a daunting task. This work borrows the lesson from application-specific circuits that the reduction of unneeded functionality consequently reduces energy. Tuning a protocol for the application at hand results in an *application-specific protocol* with superior energy efficiency.

5.1 Realities of Emerging High-Density Networks

This chapter moves beyond point-to-point links and assumes a spatially dense network whose radius far exceeds the range of a single node's radio range. Due to this change of context from single-hop communication to a network of numerous and densely placed nodes, preconceptions about transmit and receive power must be reconsidered.

Multihop, while derided in Section 4.2 for its high energy consumption relative to energy scalable approaches, is nonetheless required when the distance between sender and receiver exceeds the maximum transmission range of the node. Section 4.2 reviewed the existence of an optimum characteristic distance d_{char} for each hardware platform at which communication is most efficient in terms of meters-per-Joule. For multihop communication, as the total communication distance approaches infinity, the "meters per Joule" metric is the relevant figure of merit. It therefore follows that the output power should be set to that level which achieves d_{char}, which, for today's short-range radios, equals the radio's maximum output power. Variable radio transmission power is not relevant. It is assumed

that the network is sufficiently dense that there always exists one or more nodes in the direction of communication at a distance of roughly d_{char} from the sender.

A second, more sobering assertion is that *all* of today's short-range radio hardware is unsuitable for low duty cycle, high density wireless applications such as the microsensor network. The reasoning behind this statement is the energy consumption of the radio receiver: the energy consumption of the idle listening mode (scanning for packets) and the active receive mode (receiving a packet) have nearly the same power. For instance, the Cisco Aironet 350 card, an 802.11b radio, consumes 900 mW in active receive and 740 mW while listening for packets [24]. μAMPS-1 behaves similarly. Moreover, the power required during transmission is comparable. Hence, even with aggressive radio shutdown to reduce the spatial redundancy of receivers, a substantial amount of energy is required just to maintain connectivity across the network. Unless receivers are shut down so aggressively that portions of the network are disconnected for many seconds at a time (a substantial delay trade-off that is not acceptable for quick-reaction microsensors), a microsensor network with multi-year lifetimes is simply not possible.

The unfortunate truth is that emerging, high-density wireless applications will require advances in radio receiver technology; namely, a radio capable of an ultra-low-power listening mode. The most commonly proposed approach is a "wakeup radio"—a second, low-power receiver that is minimally designed to detect the presence of wireless communication and consumes orders of magnitude less power than a full receiver [79,86]. A wakeup radio implemented with commercial, off-the-shelf components [86] consumes two orders of magnitude less energy than the main receiver; it is therefore reasonable to expect future wakeup radios to increase this difference to at least three orders of magnitude.

To summarize, we have stated and justified two assumptions about the radio that will be utilized in this section. First, the output power of all nodes will be fixed at that level which maximizes energy efficiency in terms of meters-per-Joule. Second, the idle listening energy for the radio will be assumed as three orders less than the active receiving energy. This ensures that the results from this chapter remain relevant to emerging radio technology designed for high-density, low duty cycle wireless applications. Nevertheless, shutting

down receivers remains relevant for networks that are “overprovisioned” with radio receivers.

5.2 Multihop Routing for Sensor Fields

Given a fixed transmission range and a large set of candidate relays, the problem of power aware routing appears to boil down to a shortest-path routing problem. Hence, this section introduces the classical technique of Bellman-Ford routing for minimum-cost routes, as well as a practical technique for assigning cost metrics that takes advantage of hardware power-aware knobs, and the possibility of “unaddressed” data forwarding.

5.2.1 Distributed Bellman-Ford Routing

Of particular interest to the ensuing discussion are multi-hop routing protocols that route packets along the shortest-cost paths. Such protocols are generally based upon the Bellman-Ford algorithm, which finds the minimum-cost paths from one source to each destination [15]. In this chapter, the path costs will be variations of communication energy and expected remaining lifetime as the path costs, so that “shortest” paths will, in our case, refer to minimum-energy or maximum-lifetime routes.

The operation of the distributed Bellman-Ford algorithm is illustrated in Figure 5.1. Each node maintains a path metric which it believes to be the minimum cost required for a transmission from the source node. Each node’s path metric is initialized to infinity. Next, we “compute” the lowest-cost path from the source to all nodes within one hop of the source. Since there is only a single one-hop path between any two nodes, the minimum-cost path between source and destination at this stage is the *only* path. As illustrated in Figure 5.1b, the path metrics of these nodes are temporarily set to the cost of this one-hop path. Next, we compute the lowest-cost path from the source to all nodes within *two* hops of the source. As shown in Figure 5.1c, the nodes within one hop of the source can advertise their current path metrics to *their* one-hop neighbors, thereby reaching all nodes within two hops of the source. From all of the options presented, the two-hop nodes are then able to choose a minimum cost path, and update their path metrics accordingly. This recursive procedure continues until we have reached the maximum number of hops possible in the network; the final path metrics then reflect the cost of relaying data along the

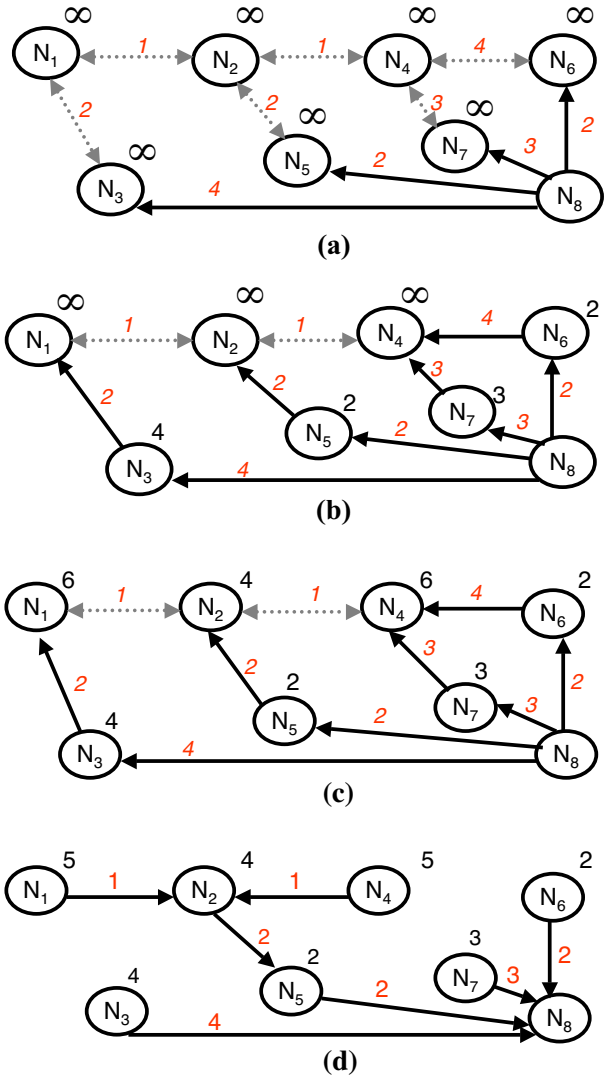


Figure 5.1: Operation of the distributed Bellman-Ford algorithm, courtesy of [50]. This example shows the calculation of shortest paths to N_8. (a) Shortest-path metrics from each node are initially set to infinity. (b) Shortest one-hop paths are calculated. (c) Shortest two-hop paths are calculated. (d) As all nodes are within three hops of N_8, the three-hop solution is the final one. Note that the metrics for N_1 and N_4 have changed as a three-hop solution offers a shorter path than the two-hop solution.

minimum-cost route to the base station. A plot of all the minimum-cost paths, as depicted in Figure 5.1d, is a spanning tree rooted at the source.

The above implementation of the Bellman-Ford algorithm is *distributed* because it can be accomplished by local computations and communications at each node, rather than the transmission of all path costs to a centralized point. Each node chooses the one-hop neigh-

bor that is on the shortest path to the source using only the path metrics passed from these neighbors. Distributed Bellman-Ford is at the heart of distance vector protocols for Internet routing such as RIP [33], and for ad-hoc wireless networks such as Destination-Sequenced Distance Vector (DSDV) [71].

Distributed Bellman-Ford for wireless applications is typically implemented as a subset of the full Bellman-Ford algorithm. The complete Bellman-Ford algorithm requires N iterations of path relaxation for N nodes to allow for the possibility of up to N-hop paths in a general directed graph. The end result is then checked for the presence of negative-cost loops, which would produce an infinitely-negative path [15]. Since the path costs for wireless communication are typically a strictly positive physical quantity (such as energy) that tends to be monotonic with physical distance from the source, the full N iterations and negative-loop check are not necessary in practice.

5.2.2 Establishing the Path Metrics

Microsensor networks can be rapidly deployed by dropping nodes by air or spraying nodes from a moving vehicle. Since these nodes begin without knowledge of their location relative to the base station, path metrics for Bellman-Ford routing must be propagated across the network.

As depicted in Figure 5.2, path metrics can be established by flooding. The base station initially broadcasts a metric of one, and all nodes within radio range of the base station receive, increment, and rebroadcast the metric. By setting each node's metric to the

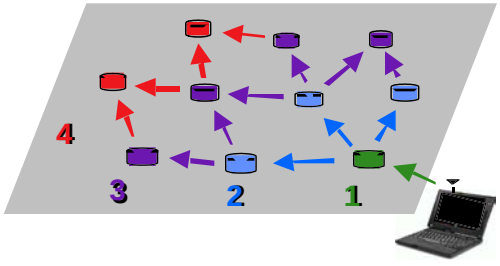


Figure 5.2: Establishing hop-count distance metrics to the base station through flooding. Nodes receive, increment, and forward an integer originating at the base station. The lowest integer heard becomes the metric.

lowest one heard, these metrics approximate the distance to the base station in terms of the number of hops to the base station.

In applications with high node density, however, many nodes are within radio range of the receiver. Hence, under the above scheme, many adjacent nodes are likely to share the same metric. As receivers are shut down, then, it becomes increasingly likely that no active receiver with a lower distance metric is within range of the sender. Figure 5.3 illustrates this problem and its solution. The goal is to reduce the radius of the distance metric transmission. Since many of today's radios feature adjustable transmit power, a natural way to achieve this is to broadcast distance metric messages with a lower radio transmission power than data messages. Specifically, data messages are sent at the optimal output power for a range of d_{char}, while distance metric broadcasts are broadcast over a range d << d_{char}.

5.2.3 Addressed Forwarding

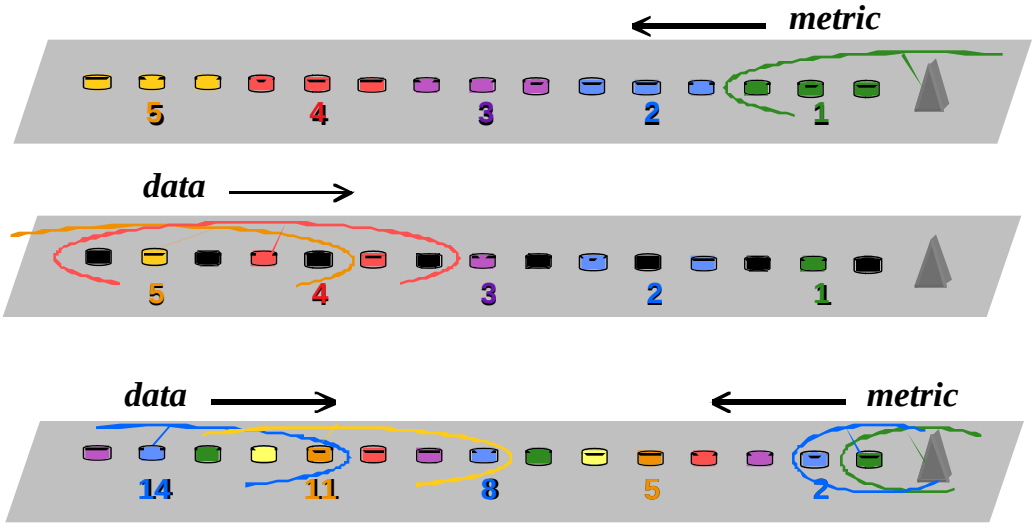


Figure 5.3: (a) Due to the high density of receivers, many nodes are within radio range of a transmission. Hence, distance metrics based on hop counts have poor spatial resolution. As a result, nodes are not always aware of neighbors that are closer to the base station. In (b), no distance-3 nodes are within range of the transmitting distance-4 node. (c) Transmitting distance metric messages with a lower propagation radius will increase distance metric resolution. This is effected by reducing the radio transmit power for these transmissions.

Most wireless protocols and MAC layers in use today utilize unique addresses to route packets to specific destinations, with the expectation that these destinations are actively listening for packets. Packets are addressed to intermediate relays nodes that are chosen to minimize the total number of hops to the destination. Under a Bellman-Ford routing scheme, the next hop node would be the one with the lowest cost metric to the destination.

The Bellman-Ford addressed routing technique simulated in this section utilizes acknowledgments and retries to counteract errors generated by the wireless channel. A transmitting node addresses its packet to the lowest-cost intermediate relay, or to the base station itself if the latter is in range. (When multiple next-hop candidates have the same minimum metric—a frequent occurrence—one is chosen at random to provide spatial dithering of energy consumption.) Packets are resent until an acknowledgment is received or the retry limit of six attempts is reached. After the retry limit, the second-best route is chosen and transmissions are attempted in the same manner.

Acknowledgments from the recipients to the sender can take one of the two forms illustrated in Figure 5.4. In multihop routing, an intermediate relay node that receives a packet will forward it onwards. The forwarding transmission may serve as an implicit

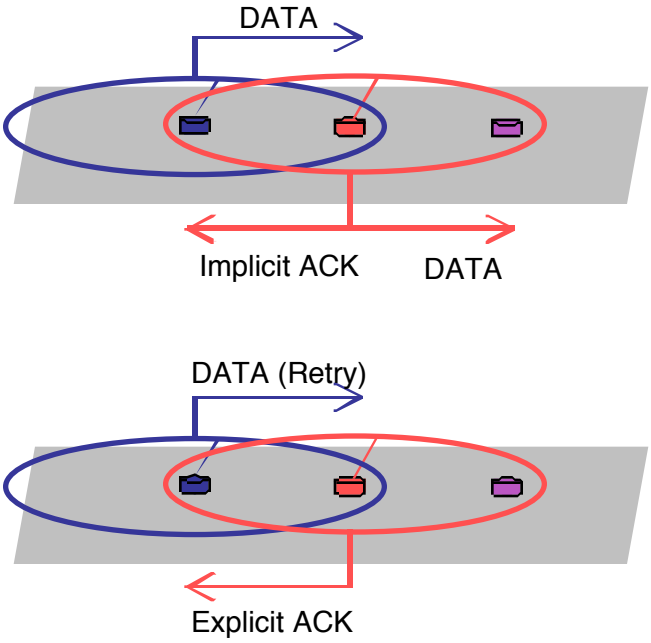


Figure 5.4: Acknowledgment mechanisms for addressed multihop forwarding. (a) Data forwarded by one node serves as an implicit ACK for the previous sender. (b) A duplicate packet is acknowledged by a brief explicit ACK message to suppress further retries.

acknowledgment to the initial sender. If the implicit acknowledgment is missed by the initial sender due to channel variations, the sender's retransmissions will result in duplicate packets at the receiver. If the receiver has already relayed the packet onwards, the receiver then sends a brief, explicit acknowledgment message to suppress further retries and prevent the sender from choosing another route.

5.2.4 Unaddressed Forwarding

When radio receivers are periodically shut down, packets may be addressed to nodes that are not presently receiving packets. As a result, table-driven routing algorithms that explicitly address packets may suffer from routing instabilities. The solution is an *unaddressed* multihop scheme that does not depend on explicit, point-to-point routes.

For microsensor networks and similar high node density applications, communication is decidedly one-way from the observer nodes to a base station. There are many data sources and relays, but few actual sinks. As the individual relays have no need for the data they are relaying, the entire notion of *addressing* a packet to a specific relay node is unnecessary. The only concern is that packets move progressively closer to a base station.

This work proposes a routing methodology based on refinements to unaddressed forwarding, a technique also utilized by gradient routing (GRAd) [74] and gradient broadcast (GRAb) [109]. Nodes employing unaddressed forwarding do not utilize explicit addresses at the protocol or MAC level, but rather a metric representing the number of hops to the packet's ultimate destination. A node with a packet destined for the base station simply broadcasts its packet with its current path metric, with the understanding that some other node with a lower metric will capture and relay the packet. On-the-fly routing decisions minimize hops while remaining resilient to channel variations and radio receiver shutdown. The path metrics utilized by unaddressed forwarding are identical to those used by addressed forwarding. Hence, this technique may be viewed as a modification to traditional Bellman-Ford routing.

As the goal is to mimic the hop-minimizing path selection of traditional routing protocols without explicit knowledge of ever-changing network topology, a "greedy" forwarding solution is utilized. Nodes broadcast all data packets destined for the base station *with no explicit next-hop destination.* The next relay node for a transmission—the receiver

closest to the base station, and therefore farthest from the sender—is self-selected *after* the packet is broadcast. In terms of distance metrics, the next relay for any transmission is the receiver with the lowest metric. Figure 5.5 illustrates this idea; a node with distance 3 from the base station—the farthest active receiver from a transmitting node of distance 6—becomes the next hop for the packet.

Self-selecting relays are achieved using a forwarding timer whose value is inversely proportional to the distance between sender and receiver. Each node receiving a data packet therefore sets its forwarding timer to

$$T_{forw} = \left(\frac{T_0}{d_{send} - d_{recv}}\right)(1 + kU[0, 1]) \tag{5.1}$$

where d_{send} is the distance metric embedded into the data packet by the sender, d_{recv} is each receiver's own distance metric, $U[0,1]$ is a uniform random variable between zero and one, and T_0 and k are constants. As long as the deterministic term is substantially larger than T_1, this local choice of forwarding timer value at each node ensures that nodes farther from the sender will initiate relay transmissions first. The random term allows for arbitration among nodes with the same metric.

The forwarding timer associated with a packet is cancelled when the node hears that packet relayed from another node with an equal or lower distance metric. Ideally, this implicit acknowledgment mechanism ensures that only the node farthest from a packet's

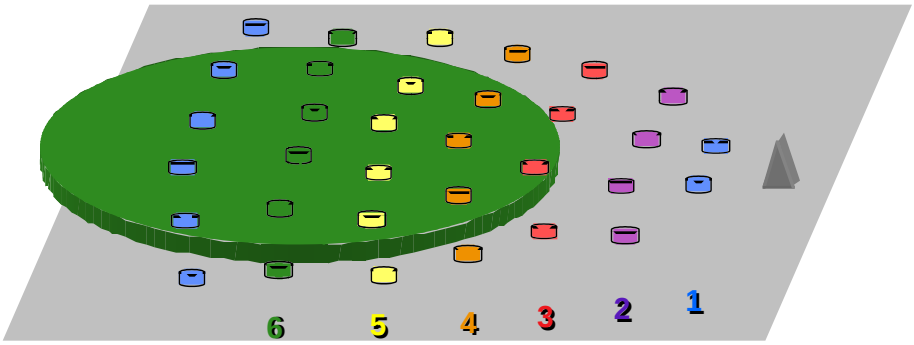


Figure 5.5: High-resolution distance metrics allow hop lengths to be maximized in a greedy fashion. For each hop, the farthest node from the sender becomes the next relay through the use of a deterministic relay timer whose value is inversely proportional to the distance between sender and receiver. Above, the distance-3 should relay.

sender will actually relay the packet, while the others will suppress their transmissions. Readers familiar with scalable reliable multicast (SRM) [25] may note clear parallels to SRM's deterministic and stochastic suppression techniques.

Unfortunately, the implicit acknowledgment of a relay transmission is insufficient to suppress all other candidate relays. The reason is illustrated in Figure 5.6: there is a sizable possibility that other nodes with a lower metric than the sender will hear the sender's transmission but not the relay's suppressing transmission. This would result in the "forking" the packet propagation into two or more branches, as more than one node would relay the initial sender's packet.

This problem closely resembles the hidden terminal problem, which is typically solved through an RTS-CTS exchange. Since different nodes are within receiving range of the initial sender and the receiver that becomes the following relay, *both* of these nodes must generate suppression messages. Consider what would happen if an RTS-CTS exchange were incorporated. Rather than forwarding data immediately, a sending node firsts broadcasts an RTS message. Potential relay nodes respond with a CTS as dictated by the forwarding timer in the protocol. The sender then chooses the node corresponding to the first CTS it hears, and explicitly sends the packet to that node only. Hence, both the receiver's CTS and the sender's data message suppress relays to other nodes. Unfortunately, this

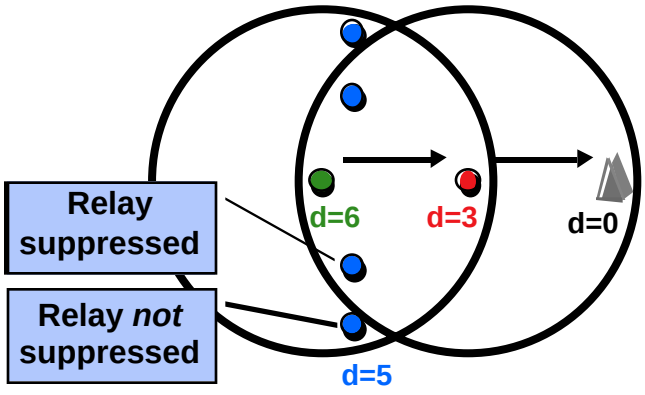


Figure 5.6: The implicit-ACK suppression mechanism fails when candidate relays (the distance-5 nodes, above) do not hear the relay transmission of the distance-3 node. This variation on the hidden terminal problem is countered with an additional ACK step from the initial (d = 6) sender.

scheme requires local addressing and offers no fault-tolerance when the node committing with a CTS becomes unable to forward the packet.

Thus, rather than an RTS-CTS-Data exchange, a “Data-Data-ACK” scheme is utilized for unaddressed forwarding. A sending node, upon hearing any successful relay from any node with a lower distance metric, also broadcasts a brief ACK message with minimal delay. This ACK serves to suppress the other potential relays that did not hear the relay itself. Note that this is a very different ACK from the one generated in 802.11b; this ACK is broadcast by the *initial sender*, not the receiver.

To summarize, unaddressed forwarding allows any active node—rather than one that is specifically addressed—to relay a packet. Routing decisions made hop-by-hop in a greedy fashion are robust to changes in network topology due to radio shutdown or channel variations. These routing decisions are nonetheless based upon the same Bellman-Ford path metrics utilized by an addressed scheme. Hence, both techniques may be employed by the same network without additional setup overhead.

5.3 Experimental Setup and Simulation Models

Addressed and unaddressed forwarding under Bellman-Ford routing are evaluated in simulation using a custom Java wireless network simulator that was co-developed with the protocol implementations. The simulator is discussed in depth in Chapter 6.

Figure 5.7 illustrates the scenario evaluated in this chapter. Five hundred nodes are distributed in uniformly random fashion over a region that is 300 meters by 150 meters. The base station is located at the far right of the landscape, and a cluster of nodes at the opposite end of the landscape generate packets destined for the base station. To simulate a low duty cycle microsensor network, one packet is generated every ten seconds. All graphs in Section 5.4 are based upon simulations of at least 2000 packets over at least five randomly generated topologies.

To minimize the impact of the media access layer on the protocol layer (works such as [3] provide excellent examples of such interactions), the medium access layer is implemented by carrier sense multiple access (CSMA) arbitration [42]. Retransmissions are *not* implemented at this layer to prevent interference with the unaddressed forwarding mechanism. Retransmissions are implemented at the protocol layer of the addressed scheme.

The energy models assume a μAMPS-1 radio augmented with a wakeup radio that consumes three orders of magnitude less energy than the active receiver, as discussed in Section 5.1. Based on measurements of the total transmit and receive packet counts (C_{tx} and C_{rx}), the total active transmitter receiver times (T_{tx} and T_{rx}), and the total idle listening times (T_{listen}) for all nodes, the total energy consumed by the network is calculated as

$$E_{tot} = P_{start}T_{start}(C_{tx} + C_{rx}) + P_{tx}T_{tx} + P_{rxElec}\left(T_{tx} + \frac{T_{listen}}{1000}\right) \tag{5.2}$$

The startup energy $P_{start}T_{start}$ and active receive power P_{rxElec} are obtained from Table 3.1. The transmit power P_{tx} is the sum of the transmit power electronics (P_{txElec}) from Table 3.1 along with the amplifier power, given by (3.5), at the radio's maximum power of +20 dBm.

As a strength of simulation is an analysis of the impact of random variables, the random variable from the path loss expression (3.7) is utilized to simulate a fading channel. This expression is repeated below:

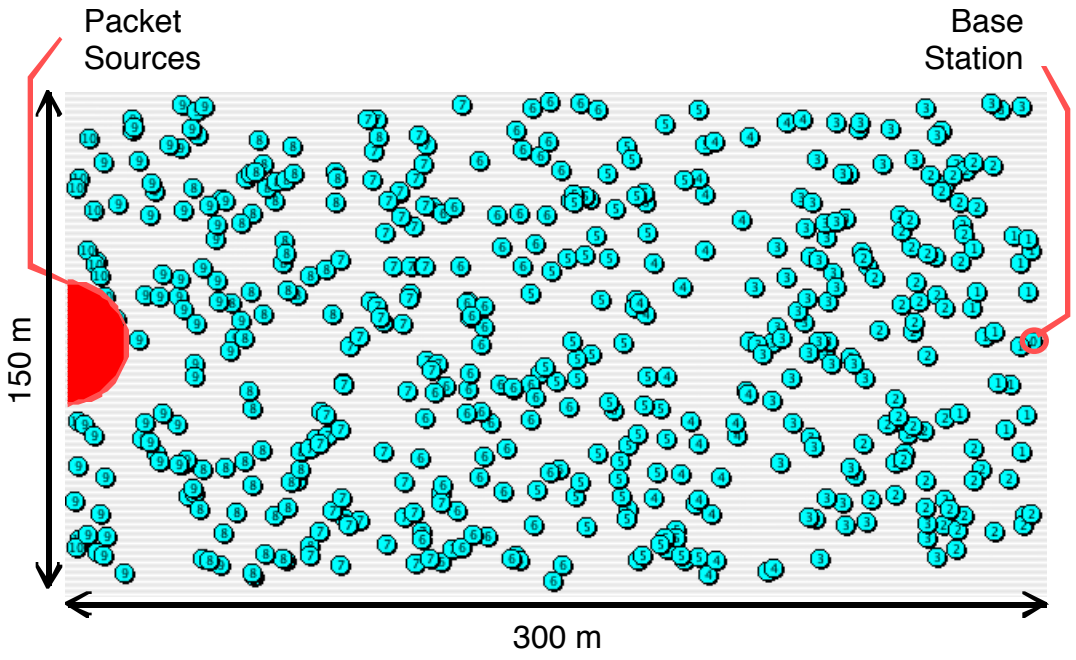


Figure 5.7: Simulation scenario for the evaluation of multihop routing for high-density networks. 500 nodes are randomly distributed over a 300 m x 150 m landscape.

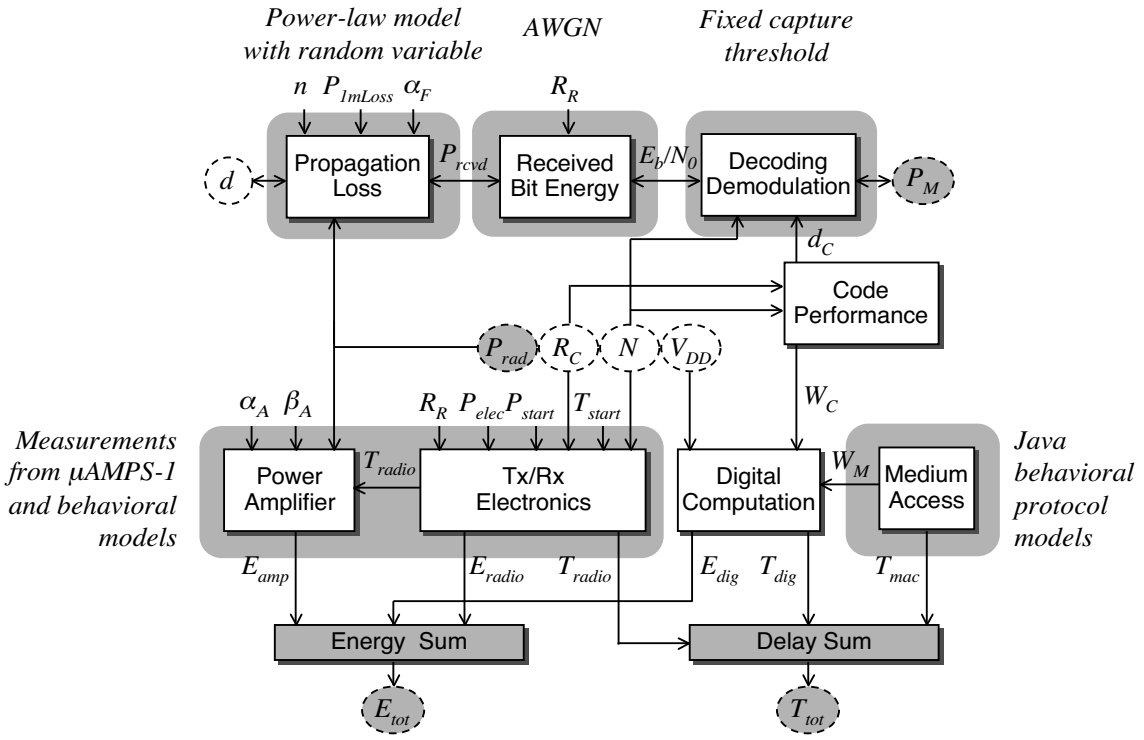


Figure 5.8: Models and assumptions for the comparison of addressed and unaddressed forwarding through simulation.

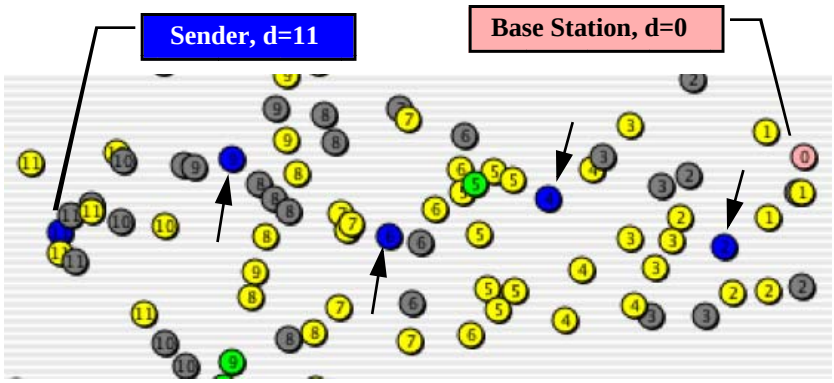
$$P_{rad} = \alpha_F(P_{1mAtt}d^n)P_{rcvd} \tag{5.3}$$

Utilizing an empirical model from [40], the random variable α_F is a Gaussian in the *logarithmic* (dB) domain. With P_{rcvd} now a random variable, the received power from this model is compared to a fixed receiver sensitivity and the interference floor produced by other transmissions to determine whether a packet is successfully captured. This approach essentially relates P_{rcvd} directly to packet error rate P_M. As decisions are made on a packet-by-packet basis, this model is actually a fairly pessimistic one compared to the Rayleigh fading models utilized in Section 4.3.2. As a result, protocol that can transmit successfully through this channel model is likely to be successful in the real world.

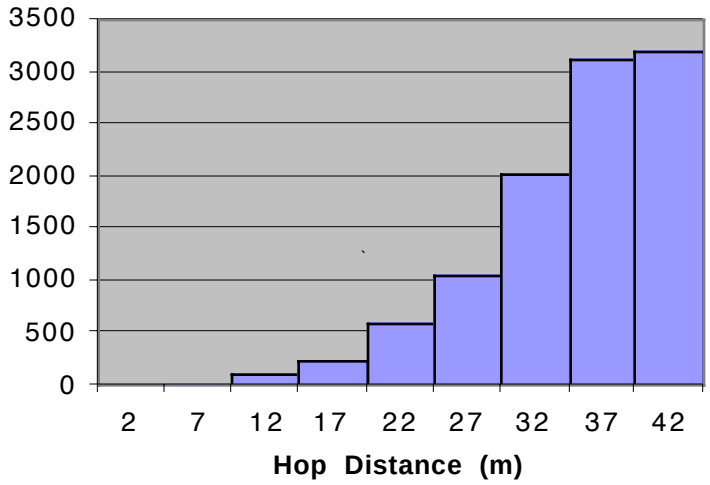
In keeping with previous chapters, Figure 5.8 summarizes the models and assumptions used for the simulation results below.

5.4 Simulation Results

This section presents simulation results that demonstrate the operation of unaddressed for-



(a)



(b)

Figure 5.9: The forwarding timer implementation for unaddressed forwarding consistently chooses long hop distances to minimize the number of hops to the base station. (a) Snapshot of a simulation run demonstrates the selection of five long hops to the base station from a node of distance-11. Blue nodes (also depicted with arrows) are the chosen relays. (b) This histogram aggregates 10,000 hops over ten different random topologies of 500 nodes.

warding, compare the performance of addressed and unaddressed forwarding, and preview two possibilities for energy-quality scalability at the protocol level under unaddressed forwarding.

5.4.1 Operation of Unaddressed Forwarding

Figure 5.9 demonstrates the successful operation of the next-hop selection mechanism in unaddressed forwarding. Figure 5.9a, a snapshot taken from the simulation tool, illus-

trates how a packet from a node with distance 11 has been forwarded to the base station (distance 0) using five regularly-spaced hops. Figure 5.9b illustrates the distribution of hop lengths for networks of reasonable active receiver density (500 nodes at 60% duty cycle). The great majority of hops selected by this forwarding scheme are of long distance, and the median hop length is within seven meters of the maximum radio range. Longer hops usually imply fewer hops, the goal of any forwarding scheme based on Bellman-Ford path metrics. Note that higher active node densities can further improve the hop length.

5.4.2 Performance of Addressed and Unaddressed Forwarding

This section compares the performance of addressed and unaddressed forwarding under two impairments to successful packet transmission: an increase in the variance of the path loss random variable α_F from (5.3), and the periodic shutdown of radio receivers in the network.

Figure 5.10 plots two metrics of communication performance, the end-to-end packet delay and the total network energy consumed per packet, versus the standard deviation of α_F, a log-Gaussian random variable. For an "ideal" channel with no fading, addressed forwarding achieves a lower delay than unaddressed forwarding. Unaddressed forwarding incurs additional delay due to the delay-based forwarding timer utilized to select a next-hop. As the channel becomes increasingly variable, however, the end-to-end delay of addressed forwarding rises while the delay of unaddressed forwarding actually falls.

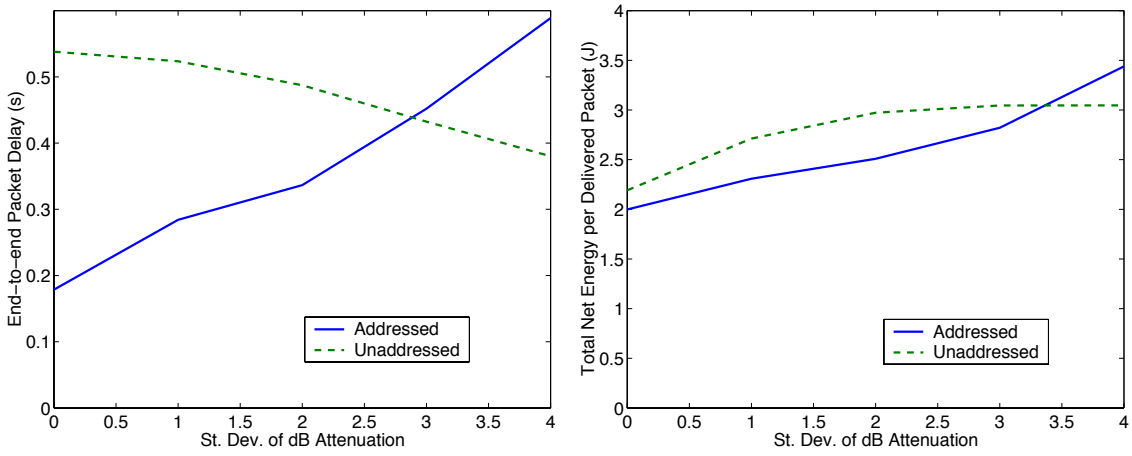


Figure 5.10: Delay and energy of addressed and unaddressed forwarding as channel quality decreases, at increments of 0.5 dB in standard deviation.

The reasons for these behaviors can be determined by plotting the total active times T_{tx} and T_{rx} for the transmitter and receivers, respectively, in Figure 5.11. As the standard deviation of the attenuation random variable increases, addressed forwarding requires progressively shorter hops and more packet retransmissions to achieve end-to-end communication. Hence, both the transmit and receive times increase. Unaddressed forwarding requires additional transmissions as well, but for a very different reason: channel variations cause the packets to "fan out" across the network due to the loss of some acknowledgment and suppression messages. As more nodes hear (and forward) the packets, the transmission and receive totals increase.

The slight *decrease* in delay apparent in Figure 5.10 as the variance increases, suggests that unaddressed forwarding is achieving longer hops when variations in path loss are in the transmitter's favor. The decrease in total transmission time from Figure 5.11 for higher variances is consistent with such behavior. Addressed forwarding, which relies on routing tables, is less capable of exploiting such rapid variations in channel conditions.

5.4.3 Impact of Receiver Shutdown

Also apparent from the axes of Figure 5.11 is the dominance of receiver energy over transmitter energy, due to the fact that many tens of nodes will receive a single transmission by a node. The choice of addressed or unaddressed forwarding is irrelevant to this

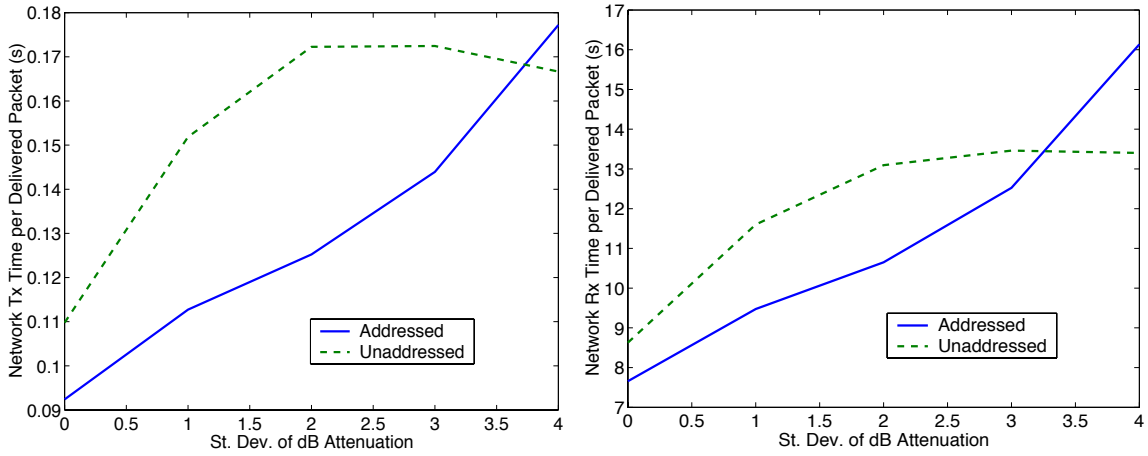


Figure 5.11: Total transmission and receive times of addressed and unaddressed forwarding as channel quality decreases. Decreases in performance for addressed forwarding are due to increases in packet retransmissions.

behavior since packets must be received and decoded before the intended recipients are identified from within the packet. The solution for reducing receive energy is the explicit shutdown of radio receivers. While receiver shutdown reduces idle and active receive energies, shutdown is an "all-or-nothing" operation that results in the radio's ignoring *all* packets, including those packets explicitly addressed to the node.

To investigate the impact of receiver shutdown on protocol performance, all node radios (except for the base station and the initial packet sources) are duty cycled at a fixed period of 40-80 seconds with a random phase offset to simulate worst-case, uncoordinated shutdown. The standard deviation of α_F fixed at 2 dB for the remainder of this chapter.

Figure 5.13 illustrates the delay and energy of both forwarding techniques as radio duty cycles are reduced. In a result more dramatic than that for channel quality above, the delay of addressed forwarding suffers greatly with lower radio duty cycles. A lower duty cycle increases the likelihood that a packet will be addressed to a node whose radio is off, resulting in a retransmission timeout and the selection of a new route (which also may or may not be listening). This is reflected by an increase in the total transmission time in Figure 5.14.

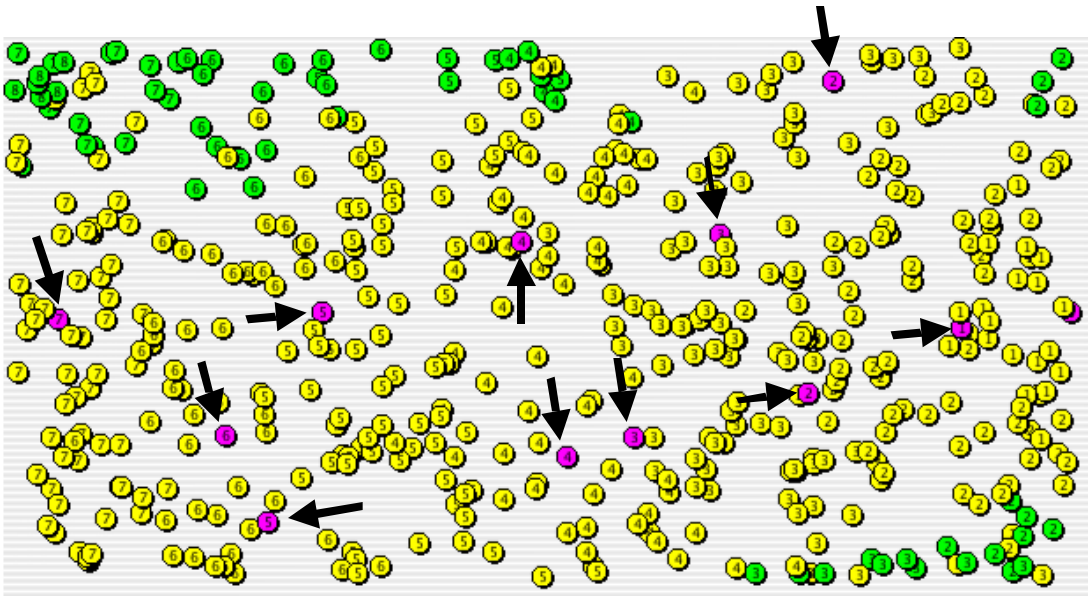


Figure 5.12: Example of an unaddressed forwarding over a highly variable channel. Suppression messages that are not received cause packet transmission paths to "fork," resulting in increased transmission and receive energy.

The delay of unaddressed forwarding, with the lack of explicit relay selection, is immune to radio duty cycling. On the other hand, the total energy per delivered packet reduces for both routing techniques as the radio duty cycle is reduced, reflecting a savings in idle and receive energy, which is also evident in Figure 5.14 as a reduction in the total network receive time.

Despite the potential for higher overall energy, the persistent retries and reroutes utilized by addressed forwarding result in higher overall end-to-end reliability.

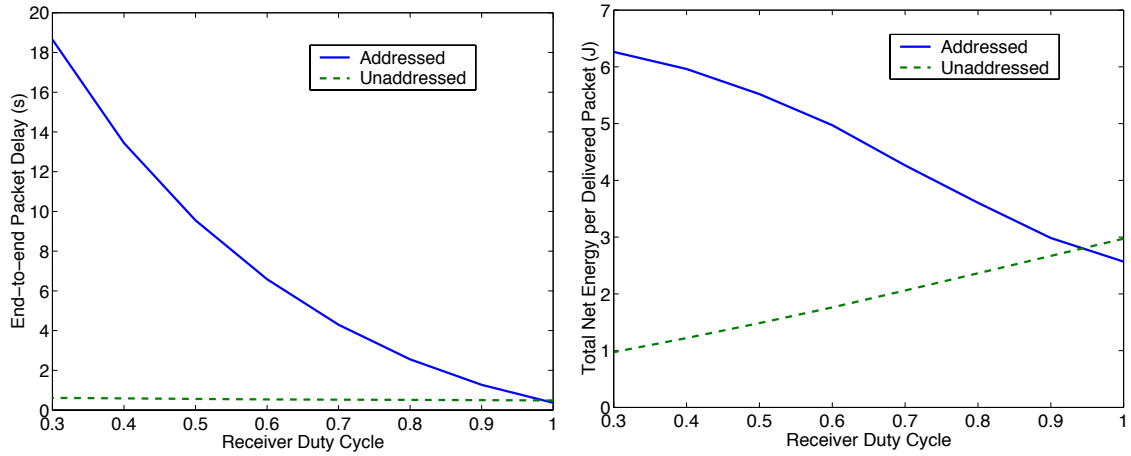


Figure 5.13: Delay and energy of addressed and unaddressed forwarding as a function of radio receiver duty cycle, at increments of 0.1.

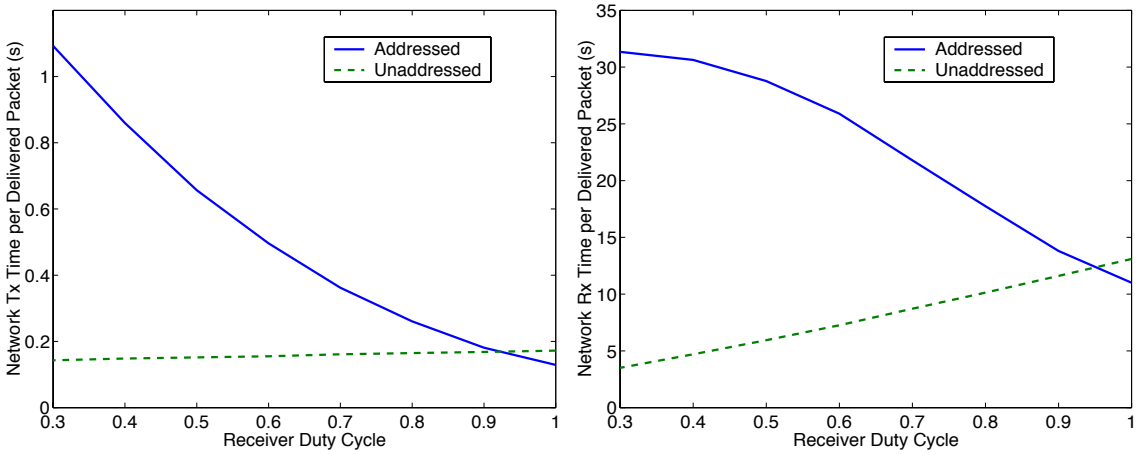


Figure 5.14: Total transmission and receive times of addressed and unaddressed forwarding as a function of radio receiver duty cycle. As duty cycle *decreases*, addressed forwarding is more likely to attempt packet delivery to a sleeping node, resulting in additional retransmissions.

5.4.4 Quality Scalability of Unaddressed Forwarding

The unaddressed forwarding technique, whose resilience to arbitrary channel and topology variations was discussed above, also presents opportunities for energy-quality scalability. This section explores the relationship between the high-level metrics of communication reliability, delay, and energy in the context of unaddressed forwarding.

One adjustable protocol-level "knob" is the constant T_0 in the backoff timer expression (5.1), which is repeated below.

$$T_{forw} = \left(\frac{T_0}{d_{send} - d_{recv}}\right)(1 + kU[0, 1]) \tag{5.4}$$

Reducing T_0 causes a corresponding reduction in the end-to-end delay, but extreme reductions in T_0 will effectively diminish the resolution of the distributed timers, impairing the ability of the distributed timers to select the node farthest from the sender. The results are shorter, more frequent hops and packet collisions.

As seen in Figure 5.16, the end-to-end delay scales linearly with T_0. While not inherently surprising, this is a particularly pleasing result as it implies great consistency across the simulation runs in the number of hops per end-to-end transmission. The impact on energy, however, is less pleasing. Reducing T_0 results in an increase in energy that

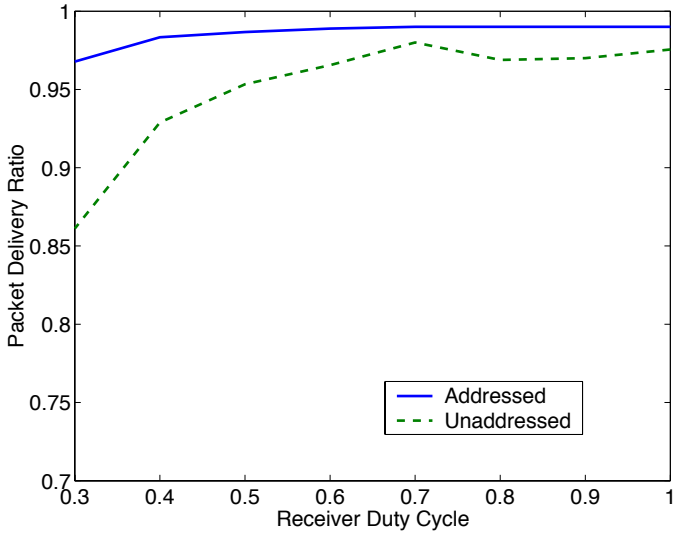


Figure 5.15: The retry and rerouting mechanisms of addressed forwarding result in a higher end-to-end reliability than unaddressed forwarding.

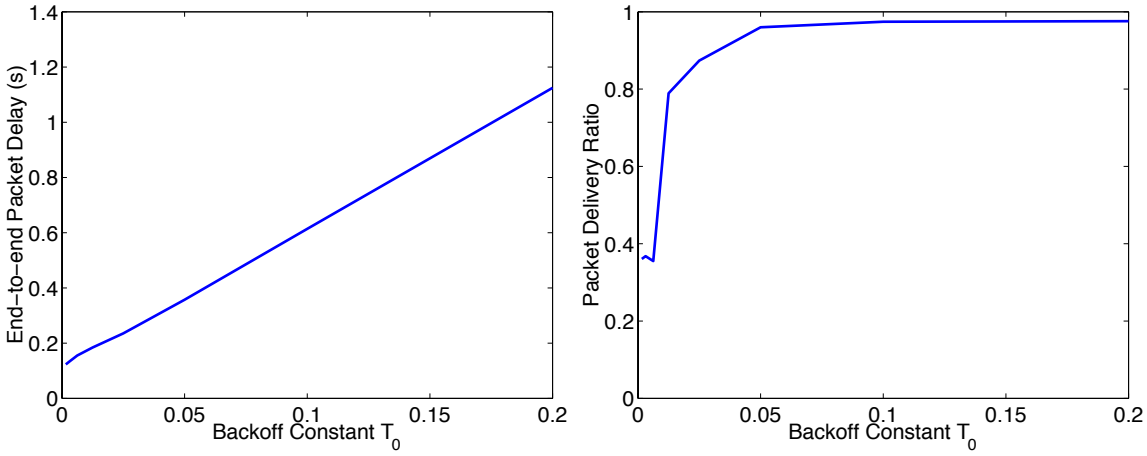


Figure 5.16: Delay and reliability of unaddressed forwarding as the backoff constant T_0 from Equation (5.1) is varied by factors of two.

becomes increasingly dramatic with a shorter T_0. As one might have guessed, the energy per *delivered* packet increases because fewer packets are actually delivered as T_0 increases. Figure 5.17 illustrates the impact of T_0 variation on reliability, the fraction of packets successfully delivered from their initial source to the base station. Hence, T_0 controls a trade-off between the end-to-end packet delay and the fundamental robustness of the forwarding mechanism.

A second energy-quality trade-off is the variation of the radio duty cycle, a trade-off foreshadowed in the previous section. While a low radio duty cycle has been shown to

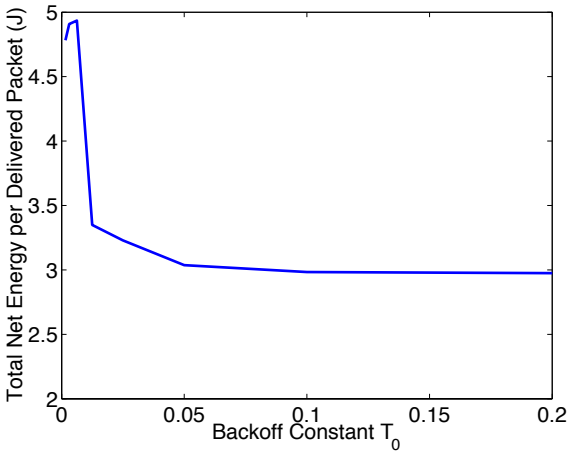


Figure 5.17: Reliability of unaddressed forwarding as the backoff constant T_0 is varied. Shorter values of T_0 reduce the effectiveness of the backoff timer and lead to additional collisions and contention, ultimately reducing end-to-end reliability.

reduce receiver energy, this savings in energy does not come without a cost. Figure 5.18 clarifies Figure 5.13 by improving the axis range and illustrating the relationship between duty cycle and protocol performance for radio duty cycles down to 10%. A lower duty cycle nominally increases packet delay by reducing the likelihood that the farthest nodes within radio range are awake. Moreover, if too many nodes are shut down, no nodes with a lower path metric may be within range of a sender, causing the packet to be lost. The end result is a reduction of end-to-end packet delivery.

For completeness, the resulting network energy *per successfully delivered packet* is plotted in Figure 5.19. The minimum-energy point (which is not exactly at a 20% duty

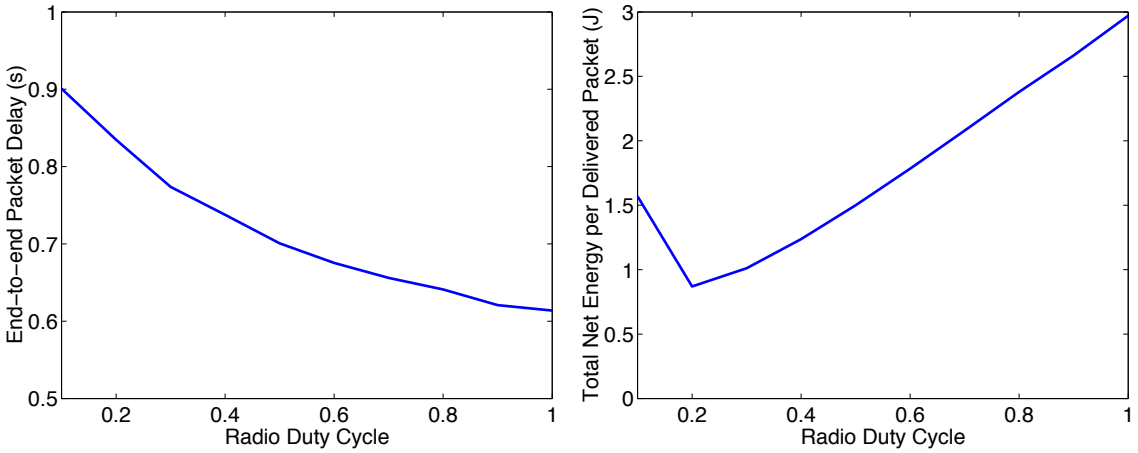


Figure 5.18: Radio duty cycle revisited: delay and energy for unaddressed forwarding as a function of radio duty cycle, in increments of 0.1.

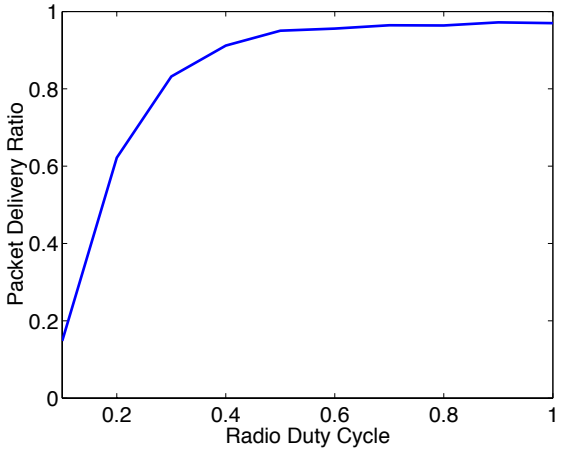


Figure 5.19: Reliability trade-off incurred by varying the radio duty cycle.

cycle as the simulations were run in 10% steps) occurs where the marginal energy impacts of lower receive energy and lower reliability are equal.

5.5 Advantages and Disadvantages of Addressing

Under ideal channel conditions, addressed forwarding is the more efficient than unaddressed forwarding. Fewer transmissions are required, for all acknowledgments are implicit. Unaddressed forwarding requires a brief, explicit suppression message from the original sender. Moreover, addressed forwarding has inherently less delay since next hop nodes are addressed directly. Nodes receiving a packet will forward that packet as soon as possible, while unaddressed forwarding uses delay for distributed arbitration. The ideal approach for radio shutdown, then, is to use global information about the network to shut down the maximum number of nodes possible while maintaining network connectivity, and to utilize addressed routing among the nodes that are awake.

Unfortunately, ideal, time-invariant communication channels and the low-cost, global sharing of network information are not reasonable assumptions for high-density networks. Hence, a robustness to real-world operating conditions makes unaddressed forwarding a compelling choice. Not only is unaddressed forwarding resilient to changing channel conditions and network topologies (as induced by radio shutdowns), but this technique opportunistically takes advantage of fades that increase range. Moreover, if the protocol is designed in such a way that path metrics vary with time, unaddressed forwarding reacts instantly to these changes. Addressed forwarding has a long time constant for adaptation due to the well-known delays of propagating updated path metrics [71] and to ensure that the routing tables are updated in a stable fashion.

Perhaps the greatest advantage of unaddressed forwarding is its simplicity. With no need for routing table maintenance or global state, unaddressed forwarding does not require periodic inter-node communication and is simple to implement. The Java implementation of addressed forwarding for the simulations above required 240 lines of code, compared to 135 lines of code for unaddressed forwarding.[1] A *simpler* protocol requires less time to implement, can be implemented on lightweight hardware platforms, and con-

1. The code for path metric distribution, which is common to both protocols, is not included in this comparison. Metric distribution was implemented with 100 lines of Java code.

sumes less digital processing energy. Finally, unaddressed forwarding decouples the protocol layer from the radio shutdown policy, allowing shutdown to occur without coordination from the protocol layer. The practical benefits of simplicity and resilience are valuable in practice and may outweigh the potential for lower efficiency compared to a more globally-aware addressed routing protocol.

5.6 Summary and Impact

Application-specific protocols improve communication efficiency by tailoring their behavior to the expected needs of the application. Unaddressed forwarding offers a simple and elegant solution to the problem of forwarding sensor data to a base station. Addressed and unaddressed forwarding protocols based on Bellman-Ford cost metrics were simulated an environment with two dimensions of operational diversity: unpredictable channel characteristics and radio receivers that shut down frequently and arbitrarily to conserve energy. Unlike addressed forwarding, the unaddressed technique offered resilience and energy-efficiency that was immune to the operational diversity presented.

While the concept of unaddressed forwarding existed before this exploration [74], this work refines its implementation by exploiting variable output power to propagate higher resolution metrics, resulting in superior performance to prior unaddressed techniques. Given the great emphasis on addressed multi-hop routing protocols over the past five years of research into *ad hoc* networking [84], this work stands in support of an alternative technique that is well-suited to high density microsensor networks due to its simplicity and robustness.

Chapter 6

Interactive Simulation of High Density Wireless Networks

Current simulation tools are not conducive to the design and evaluation of energy-efficient microsensor protocols. Performance can be poor with networks of thousands of nodes, facilities for the development of accurate hardware energy models are underemphasized, and the convenience of real-time control and visualization are not always available. These deficiencies in current tools motivates the creation of an extensible, energy-conscious, and high-performance simulator for high-density wireless networks.

6.1 Lessons from Prior Art

6.1.1 Contemporary Simulation Tools

While *ns-2* [48] is perhaps the premier academic research tool for network simulation, architectural weaknesses limit its extensibility for simulations of high-density, energy-conscious wireless networks. *ns-2* simulations are written in two layers: the simulation core and low-level routines such as packet classification are written in C++, and high-level control scripts are composed in *tcl*. A simulation's data structure are accessible from either layer. In *ns-2*, class inheritance is implemented at the *otcl* layer, not in C++; all C++ simulation objects share the same class. This architectural choice unfortunately encourages development at the lower-performance *otcl* layer. Moreover, developers who make modifications to the simulation core at the C++ layer (to model energy, for instance) tend to do so in mutually incompatible ways.

Two lessons are evident from *ns-2*. First, a carefully thought-out inheritance framework is necessary so that models can be modified without altering the simulation core. Second, the separation of code into interpreted and compiled layers may have been a wise

design decision at *ns-2*'s inception, but hindsight suggests a detrimental impact on abstraction and performance. A monolithic, object-based binary format seems the wiser choice.

The *x-kernel* project [38] offers some guidance on abstraction and API design. *x-kernel* is an early implementation of object-based libraries and interfaces for constructing network protocols. Two functionalities exposed by *x-kernel* that are of particular interest are API calls for deferred actions, such as timers and callbacks, and table structures for lookups and queues. *Deferrable actions* and *tables* are perhaps the two fundamental structures that drive all network protocols and are therefore mandatory components of a simulation tool.

SensorSim [69] is an extended version of *ns-2* designed specifically for sensor networks. It is particularly noteworthy for its co-simulation abilities; virtual nodes in simulation can interact with physical microsensor nodes. Unfortunately, SensorSim's *ns-2* heritage ensures that both tools share the performance and extensibility weaknesses discussed above.

A large number of network simulation tools written in Java suggest the ease of developing for this platform. JavaSim [66], simjava [36], SSFnet [65], and JASPER [97] are each complete network simulation frameworks written in Java. Complete TCP/IP protocol stacks have been implemented for each of these environments, demonstrating their functionality.

6.1.2 Simulation for Java

The proliferation of Java-based simulation tools, and the accessibility and extensibility of Java code, both suggest that Java is a reasonable development platform for this simulation tool. Recent Java APIs provide built-in high-level data structures, double-buffered graphics, and an event-driven user interface. This not only facilitates the rapid development of the tool but also encourages a user-friendly design that is more likely to be accepted by the networking community. The ability to declare standard interfaces and the ease of reusing and extending code through inheritance facilitates the development of new models.

While early implementations of the Java virtual machine may have instilled the notion that Java programs run slowly, recent advances in Java have shown that this is no longer the case. Just-in-time (JIT) compilation technologies such as HotSpot [28] have made Java programs competitive with natively compiled code. In what was likely a best-case scenario, JIT-compiled Java code for a Macintosh PowerPC G3 platform differed in speed by less than a factor of two to natively compiled C++ on the same platform. However, given the development advantages of Java, this performance penalty is easy to justify. Furthermore, the performance of native C++ is strongly offset by its longer development time. Java's programming methodology and rich API will allow development to focus on the simulation architecture, where more advances in simulation speed can be made per programmer-hour. As a result, a Java-based simulator may outperform a C++ based program given the same amount of development time.

6.2 Simulator Design Goals

With the prior art in mind, four design goals were selected to encourage the development of a flexible, usable tool for teaching and research. These goals are as follows:

Accessibility and extensibility. The tool will be accessible to as many users as possible and easy to extend with user-defined models. This naturally implies a multiplatform tool with an accessible and functional API for creating new models. The ulterior motive is widespread acceptance by the networking community.

Real-time interaction. Most simulation tools do not offer interactive control. Once the simulation begins, no parameters can be changed, and visualizations are often *post mortem*. As a primary goal of this tool to build intuition and educate the user of the benefits of energy and quality scalable communication, this simulator will offer significant interactive control. Parameters will be adjustable, nodes movable, and graphs displayed, all in real-time.

Speed. Long simulation time is the primary limitation to simulating distributed microsensor networks; simulations can require overnight processing with current tools. A substantial reduction in simulation time will allow simulation results to be fed back into the design process, enabling the use of simulation in sensor network design. While the initial goals were a two order of magnitude improvement over current simulation tools, the

present design emphasizes the ease of extensibility and the quality of real-time interactions between user and simulator. Hence, speed is a secondary goal.

Rapid development. In order for this tool to make a substantial impact as a design and optimization tool, it must be ready for use well before microsensor networks become commonplace and well understood. As research in distributed microsensor networks is proceeding at an intense pace, a short "time to market" becomes an important design goal.

6.3 Simulator Architecture

This section describes the architecture of the Java simulator and assumes some familiarity with the Java language and the Java API [10].

The simulation tool is composed of five components: the real-time parameter control framework, quantitative performance and energy models, behavioral models, event-driven core, and graphical user interface (GUI). Each of these components is designed with architectures that offer reasonable performance while conforming with traditional Java abstractions to the greatest extent possible. Two examples may clarify this principle. First, numerous data structures are implemented with the Java Collections framework [11]. Collections are used extensively to implement lists or maps whose members may change at run time. The use of collections in this way avoids the somewhat controversial Reflection methods. Second, unique objects such as parameter names and equations follow the *singleton design pattern*. Rather than identifying simulation parameters by string literals, parameter names are implemented as singleton objects. The extra code required to describe such objects is offset by identification of naming "typos" at compile-time and the inclusion of metadata within each singleton object.

6.3.1 Parameter Framework

The parameter framework enables the real-time visualization and modification of virtually any simulation variable. The parameter framework consists of ParameterKey objects that uniquely identify each parameter, and the Parameter interface for wrapped variable values.

ParameterKey objects are singleton objects that uniquely identify each parameter. ParameterKey objects contain the parameter's name and description, both of which are

useful to the GUI. The Parameter interface describes self-contained objects that closely resemble the wrapped Double, Integer, and Boolean classes but provide more functionality. Parameters offer built-in methods for incrementing, decrementing, and converting their values to and from String. The richness of the Parameter interface facilitates cooperation with the simulator GUI. Parameters may be utilized for both input and output purposes; as a general rule, anything that the user may wish to view or edit should be implemented as a Parameter, rather than a primitive data type such as *double* or *int*.

Parameter keys and values are stored in ParameterBlock objects throughout the simulation, as illustrated in Figure 6.1. ParameterBlock objects contain a HashMap that maps ParameterKeys to Parameters, essentially mapping names to values. Each node in the simulation maintains a local ParameterBlock that stores the node's viewable Parameters. Each local ParameterBlock also points to a global ParameterBlock maintained by each simulation. Global ParameterBlock objects store parameters shared by all nodes in a simulation, such as channel characteristics or common energy models. Figure 6.1 illustrates the relationship between local and global ParameterBlocks, which is analogous to methods in a Java subclass that override methods from the parent class.

6.3.2 Quantitative Models

A flexible energy and performance modeling strategy that is robust to a wide variety of wireless network hardware demands a clean separation between the node hardware and the software (protocols and algorithms) running upon it. Algorithms and protocols deter-

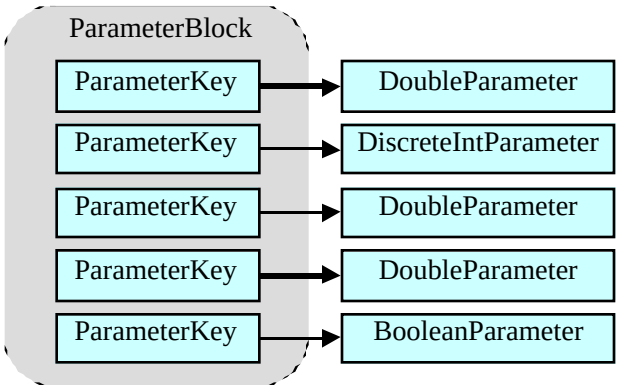


Figure 6.1: ParameterBlock objects within each node maintain a map between ParameterKeys and Parameter objects.

mine how data is transformed and moved; hardware determines how quickly and with how much energy. An algorithm can be implemented on different hardware fabrics, such as microprocessors, FPGAs, and custom integrated circuits. The choice of hardware implementation impacts energy and delay by orders of magnitude.

Quantitative models of energy and performance are conceptually separate from behavioral models (Section 6.3.3) that describe the operation of protocols and software. Hence, these two types of models are treated independently in the simulation framework. This enables any behavioral model to utilize freely any quantitative model.

Quantitative models are basically equations that solve for numerical quantities. Quantitative models express relations such as the energy consumed by the radio transmitter and receiver, the delay required for FEC decoding, and the relationship between distance and received power. Quantitative models are expressed as (usually anonymous singleton) subclasses of the EquationModel class. Each model contains methods for solving each relevant variable in the equation. A *compute* method takes a ParameterBlock containing variable values for the equation and the ParameterKey that names the parameter to be solved, and calls the appropriate method to solve the variable represented by the ParameterKey.

Given a sufficient system of equations and parameter values, the system can be solved for any variable without explicitly calling a sequence of equations. The EquationEngine class provides this functionality. EquationModels are placed into an EquationEngine, and the EquationEngine is called with a ParameterBlock of values and a ParameterKey to be

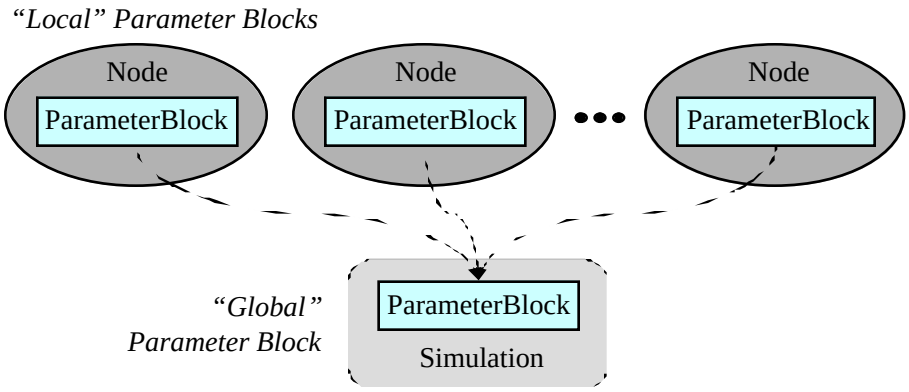


Figure 6.2: ParameterBlock objects are hierarchical, allowing blocks local to each node that descend from a global, simulation-wide block.

solved. Given sufficient input parameters, EquationEngine will call the appropriate equations required to solve for any specified output parameter. Figure 6.3 summarizes the operation of EquationModel and EquationEngine objects.

The simulation tool implements EquationModel objects for most of the equations discussed in Chapters 3-4.

6.3.3 Behavioral Models

Behavioral models describe the operation of protocols, node hardware, and packet dissemination over the wireless channel. These models communicate with each other using method calls that represent simulated events such as the arrival or departure of packets from each behavioral model. Behavioral models interact with EquationModels for timing

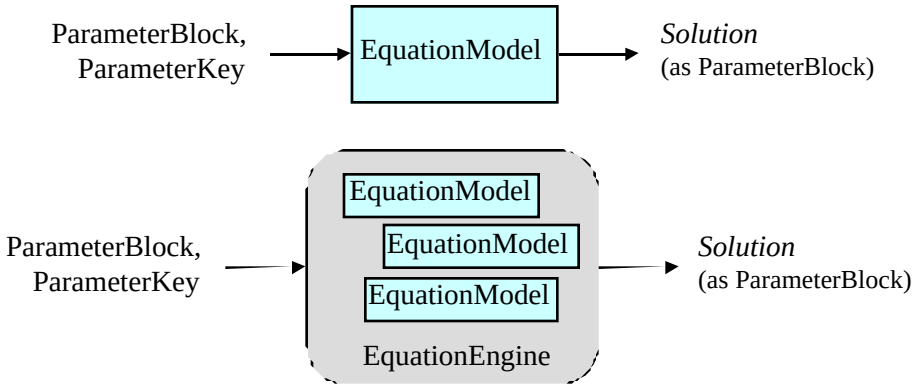


Figure 6.3: Single EquationModels can be solved for any supported ParameterKey. Systems of equations can be solved for a single ParameterKey with the assistance of an EquationEngine, which calls the appropriate equations in sequence.

and energy information, and with ParameterBlocks to receive simulation parameters. The relationships among these classes is illustrated in Figure 6.4.

The wireless channel model implemented by Ether accurately calculates inter-packet interference based on the duration, location, and strength of transmissions, and provides a carrier sense function that returns the total signal strength at any location. To reiterate the earlier point of separation between behavioral and quantitative models, the actual computations of attenuation and signal strength are handled by quantitative models that represent path loss and fading. The quantitative models may be changed dynamically without altering the core behavior of Ether.

Protocols and node hardware are described by the Protocol interface and the Node class, respectively[1]. These behavioral models are typically written as state machines, with methods that act as event handlers for incoming events of interest. Internal state variables influence how each event is are handled. For instance, a transmission incident to the radio receiver (the arriving event) will produce a behavior that depends on whether the radio is off, starting up, listening, or transmitting another packet (the internal radio state). The packet will be passed to the Protocol only if the radio is in an active listening mode.

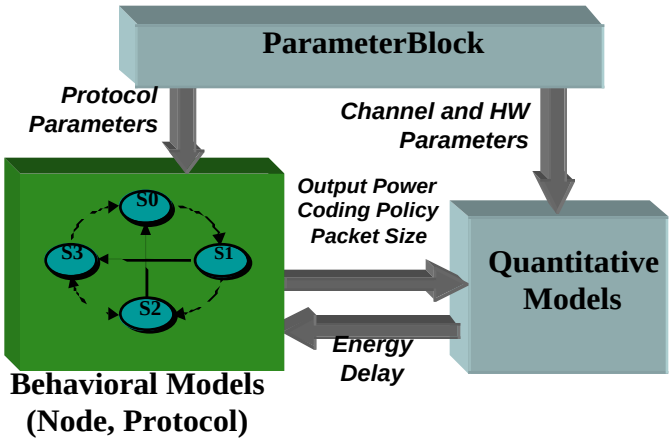


Figure 6.4: Behavioral models receive performance information from quantitative models and parameter values from the local ParameterBlock.

1. In the current implementation of the simulator, the Node class models the sensor hardware, radio hardware, and media access layer. Prior iterations of the simulator separated these behaviors into three classes, and the interaction among these state machines proved a rich source of complexity and debugging challenges. In this particular case, the battle between one monolithic FSM and smaller, interacting FSMs was won decisively by the former approach.

6.3.4 Event-Driven Simulation Core

The simulation core maintains a list of queued actions and fires events in temporal order. The simulator core maintains a “time wheel” of queued events using a TreeMap and provides functionality for adding, removing, and peeking at events on the queue. The core also includes high-quality utilities such as a strong random number generator (superior to Java’s built-in algorithm) implemented by CERN’s jet package [35].

The popularity of timewheel-based, event-driven simulation makes this component a strong candidate for code reuse from existing simulation tools. Unfortunately, a review of existing Java-based network simulators revealed that a custom implementation was the best approach. For instance, the *simjava* simulator [36] uses a basic linked list for the event queue, which is inefficient for queueing and dequeueing events for thousands of nodes. While SSFnet [65] was the most promising solution, all implementations of this framework were commercial and licensed. Moreover, SSFnet’s focus on distributed, parallel simulation is not necessarily an ideal match for this project, which places more value on interactive operation.

Simulation events are implemented as subclasses of SimAction, which implements a queueable action. The *run()* method of SimAction is overridden with the desired deferred action, which is usually a single method call. All delayed simulation events are implemented in this manner.

6.3.5 Graphical User Interface

The simulation GUI enables real-time interaction between the user and simulation. The GUI illustrates the locations and states of individual nodes, provides a panel for viewing and editing Parameter values, and allows real-time graphs of any Parameter. The GUI

was specified by the author and developed by Nidhi Sharma. Figure 6.5 is a snapshot of the GUI.

The parameter editor can access any ParameterBlock in the simulation, and changes to parameters are reflected in the simulation in real time. Graphs are presented in a pane within the main window or in a separate window, and can display any parameter or combination of parameters. Graphs may be exported to a graphics file, Matlab file, or printer. The node viewer currently displays node state through its color and a short configurable string (typically a number). Future versions of the GUI will allow interaction with this display pane. Clicking on a node will display its properties; dragging a node will update its position in the simulation, enabling interactive mobility experiments.

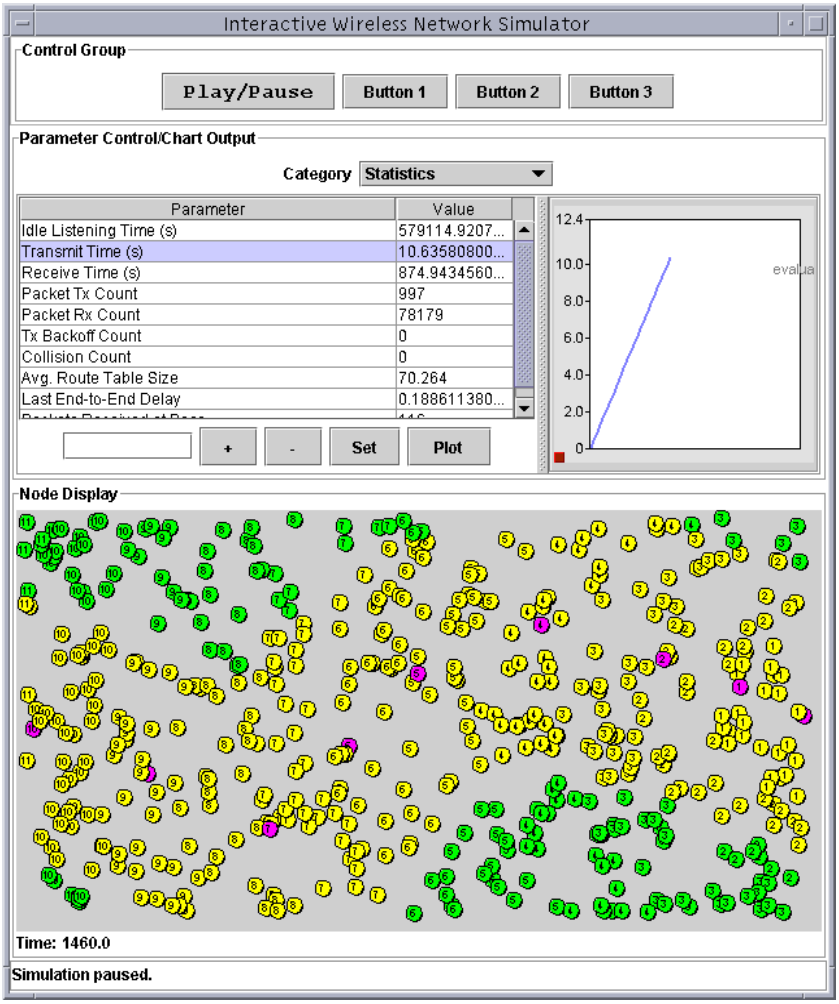


Figure 6.5: Screenshot of the simulator GUI.

6.4 Tutorial Example

This section presents a basic example of protocol programming using the simulation tool. The code, presented in Section 6.4.1 below, is followed by a detailed description of its behavior in Section 6.4.2.

6.4.1 Sample Simulation Code

The listing for the file *SimpleMultiHop.java* below implements a simple multi-hop protocol. All nodes receiving a packet forward the packet to node determined by the variable *nextHop*.

```
1     public class SimpleMultiHop extends Protocol {
2
3         Node nextHop;
4         int nodeNum;
5
6         public static final String GUI_MULTIHOP = "Simple MultiHop";
7         public static final ParameterKey RX_PACKET_COUNT =
8           new ParameterKey("Packets Received", "packets accepted by
9           protocol layer", GUI_MULTIHOP);
10        public static final ParameterKey TX_PACKET_COUNT =
11          new ParameterKey("Packets Transmitted", "packets sent from
12          protocol layer", GUI_MULTIHOP);
13
14        public SimpleMultiHop(Node myNode, Node nh, int nn) {
15            super(myNode, "SimpleMultiHop Protocol");
16            nextHop = nh; nodeNum = nn;
17
18            params.addParameterRecursive(RX_PACKET_COUNT,
19                new IntParameter(0));
20            params.addParameterRecursive(TX_PACKET_COUNT,
21                new IntParameter(0));
22        }
23
24        public String getDisplayString() {
25                return Integer.toString(nodeNum); }
26
27        public void inboundPacket(RxPacket rxp) {
28            Packet p = (Packet)rxp.pkt;
29            if (!p.getRecipients().contains(myNode)) return;
30
31            SimData sd = (SimData)rxp.pkt.getAppData();
32            params.incrementRecursive(RX_PACKET_COUNT);
33            if (nextHop == null)
34                System.out.println("Sink node received data " + sd +
35                      " at time " + sim.time() );
```

```
36              else
37                  outboundData(sd);
38          }
39
40          public void outboundData(SimData sd) {
41              System.out.println(myNode + " sending packet " + sd);
42              params.incrementRecursive(TX_PACKET_COUNT);
43              Packet pOut = new Packet(sim.time(), myNode, 10000,
44                                       null, sd, null);
45              pOut.setPower(100 /*mWatts*/);
46              pOut.setSendTime(sim.time() +
47                  sim.randomize(0.01 /*seconds*/,0.5,1));
48              pOut.setExpireTime(sim.time() + 0.500);
49              pOut.setBackoff(0.01);
50              pOut.setRecipient(nextHop);
51            myNode.outboundPacket(pOut);
52          }
53      }
```

The listing for *SimpleSimulation.java* below instantiates a line of five nodes that implement the *SimpleMultiHop* protocol. Node 1 generates packets every *packetPeriod* seconds, and the packets are forwaded to Node 5 via all intermediate nodes.

```
54      public class SimpleSimulation extends Simulation {
55          Node sendingNode;
56          Node nextHop;
57          final int NUMBER_OF_NODES = 5;
58          final double packetPeriod = 6.0;
59
60          SimAction packet_generator = new SimAction() {
61              public void run() {
62                  SimData generatedData =
63                        new SimData(sendingNode, 1000, 0.0, time());
64                  sendingNode.proto.outboundData(generatedData);
65                  events.enqueue(this, null, packetPeriod);
66              }
67          };
68
69          public void run() {
70              Node n = null;
71              for (int i = NUMBER_OF_NODES; i >= 1; i--) {
72                  if (i == NUMBER_OF_NODES) nextHop = null;
73                  else nextHop = n;
74                  n = new Node(this, "Node "+i, 100, 25*i, 10, false);
75                  n.proto = new SimpleMultiHop(n, nextHop, i);
76                  if (i == 1) sendingNode = n;
```

```
77                this.nodes.add(n);
78            }
79
80            display = new SimFrame(this, globalParams);
81            events.enqueue(packet_generator, null, packetPeriod);
82            runUntil(100.0);
83        }
84    }
```

The simulation is started by instantiating *SimpleSimulation* and calling its *run()* method.

6.4.2 Sample Code: Step by Step

A line-by-line inspection of the sample code illustrates the majority of the features offered by the network simulation. The listing is repeated below and interspersed with tutorial remarks about working with the simulation tool.

```
1     public class SimpleMultiHop extends Protocol {
2
3         Node nextHop;
4         int nodeNum;
```

Protocol is the base class that provides a basic API for packet processing and useful class variables such as the current Simulation *sim* and the node associated with this particular instantiation of the protocol object *myNode*. The class variable *nextHop* represents the node to which all incoming packets will be forwarded by this protocol. *nodeNum* simply identifies this node in numerical form.

```
5
6         public static final String GUI_MULTIHOP = "Simple MultiHop";
7         public static final ParameterKey RX_PACKET_COUNT =
8           new ParameterKey("Packets Received", "packets accepted by
9           protocol layer", GUI_MULTIHOP);
10        public static final ParameterKey TX_PACKET_COUNT =
11          new ParameterKey("Packets Transmitted", "packets sent from
12          protocol layer", GUI_MULTIHOP);
```

As discussed in Section 6.3.1, *ParameterKey* objects describe simulation parameters that may be viewed and/or changed while the simulation is running. *ParameterKey*s are instan-

tiated with a String that will be displayed in the parameter display box, a descriptive phrase, and the parameter category for GUI display purposes.

```
13
14        public SimpleMultiHop(Node myNode, Node nh, int nn) {
15            super(myNode, "SimpleMultiHop Protocol");
16            nextHop = nh; nodeNum = nn;
```

The *Protocol* superclass constructor requires the owning Node and a descriptive name as parameters. *myNode* is set to a class variable of the same name that is accessible to *SimpleMultiHop*.

```
17
18        params.addParameterRecursive(RX_PACKET_COUNT,
19                new IntParameter(0));
20            params.addParameterRecursive(TX_PACKET_COUNT,
21                new IntParameter(0));
22        }
```

params, a class variable provided by *Protocol*, is the owning node's local parameter block. The *addParameterRecursive()* call adds the (ParameterKey, Parameter) pair to the local parameter block *and* the simulation's global ParameterBlock. Parameters generally offer a rich set of constructors; the simplest instantiation of *IntParameter* is presented above.

```
23
24        public String getDisplayString() {
25                return Integer.toString(nodeNum); }
```

getDisplayString() is called by *NodeDisplay*, the GUI component that renders the nodes. The string returned is displayed within each node.

```
26
27        public void inboundPacket(RxPacket rxp) {
28            Packet p = (Packet)rxp.pkt;
29            if (!p.getRecipients().contains(myNode)) return;
```

inboundPacket() is specified by *Protocol*. This method is called by *Node* when a packet is received by the radio and medium access layers. The *RxPacket* object encapsulates statistics about the incoming packet, such as its strength, received time, and *pkt*—the packet itself. Without explicitly checking the code, there is no guarantee that lower layers have

filtered incoming packets by their intended destinations, so this check is done in line 29. *getRecipients()* returns a *Collection*.

```
30
31            SimData sd = (SimData)rxp.pkt.getAppData();
32            params.incrementRecursive(RX_PACKET_COUNT);
33            if (nextHop == null)
34                System.out.println("Sink node received data " + sd +
35                      " at time " + sim.time() );
36            else
37                outboundData(sd);
38        }
```

Once certain that the packet is indeed intended for this node, the application-level data is extracted from the packet with *getAppData()*. Only this portion is preserved and forwarded to the next node in the multihop chain as specified by *nextHop*. (*Packet* objects are created anew for each transmission.)

incrementRecursive(key) causes the *Parameter* keyed by *ParameterKey* to be incremented in both the local and global *ParameterBlock*s. Line 32 alone therefore implements a local, within-node count and a global, simulation-wide count.

```
39
40        public void outboundData(SimData sd) {
41            System.out.println(myNode + " sending packet " + sd);
42            params.incrementRecursive(TX_PACKET_COUNT);
43            Packet pOut = new Packet(sim.time(), myNode, 10000,
44                                     null, sd, null);
```

outboundData() is specified by *Protocol*. It is called by the application layer to introduce data into the network. The default call assumes that the destination of all packets is a known sink or basestation; this behavior can be overridden with additional method calls.

New *Packet*s are instantiated with six arguments that are immutable once initialized: the creation time, source node, size in bits at the protocol layer, packet type (to distinguish control and data packets if so desired), application-layer data, and additional protocol-layer data.

```
45            pOut.setPower(100 /*mWatts*/);
46            pOut.setSendTime(sim.time() +
47                sim.randomize(0.01 /*seconds*/,0.5,1));
48            pOut.setExpireTime(sim.time() + 0.500);
49            pOut.setBackoff(0.01);
```

```
50            pOut.setRecipient(nextHop);
51          myNode.outboundPacket(pOut);
52        }
53    }
```

Once the packet has been constructed, additional arguments are set. Unlike the arguments with which the packet was instantiated, these arguments are mutable until the packet is transmitted. In the above example, the following arguments are set: the power, specified in milliwatts; the time that the packet should be presented to the MAC layer; the time at which the packet expires and should no longer be transmitted; the mean of the backoff period when the channel is busy; and the intended recipient (or *null* for multicast). The call to the Simulation's utility function *randomize*() in line 47 returns a uniform random variable drawn between (0.5)(0.01) and (0.5+1)(0.01) seconds. Line 51 enqueues the packet, where it will be received by the MAC layer at the time specified by *setExpireTime()*. The default MAC is a carrier-sense multiple access implementation.

Simulations are customized by subclassing *Simulation* and instantiating the component *Node* and *Protocol* objects.

```
54    public class SimpleSimulation extends Simulation {
55        Node sendingNode;
56        Node nextHop;
57        final int NUMBER_OF_NODES = 5;
58        final double packetPeriod = 6.0;
```

This example will create a line of *NUMBER_OF_NODES* nodes. The first node will generate packets every *packetPeriod* seconds.

```
59
60        SimAction packet_generator = new SimAction() {
61            public void run() {
```

Queued actions are implemented by subclassing *SimAction* and are conveniently implemented as anonymous inner classes. The *run()* method is called by the simulation queue to fire the event. *SimAction* defines an Object *argument* that may be utilized within *run()*. This is not illustrated here.

```
62    SimData generatedData =
63                        new SimData(sendingNode, 8000, 0.0, time());
64                sendingNode.proto.outboundData(generatedData);
```

```
65                  events.enqueue(this, null, packetPeriod);
66              }
67          };
```

As the name implies, *packet_generator* is a periodic packet generator. At every call, a new application-level *SimData* object is constructed with the source node, size in bits, strength (as in received strength for a sensor application), and creation time. *SimData* may be sub-classed for additional applcations beyond wireless microsensing. Finally, *packet_generator* requeues itself to be fired *packetPeriod* seconds later. *events* is the event queue handler associated with this *Simulation.* The second argument of *enqueue* is an optional argument that is installed into the *SimAction* at fire time.

```
68
69          public void run() {
70              Node n = null;
71              for (int i = NUMBER_OF_NODES; i >= 1; i--) {
72                  if (i == NUMBER_OF_NODES) nextHop = null;
73                  else nextHop = n;
74                  n = new Node(this, "Node "+i, 100, 25*i, 10, false);
75                  n.proto = new SimpleMultiHop(n, nextHop, i);
76                  if (i == 1) sendingNode = n;
77                  this.nodes.add(n);
78              }
```

One of the first tasks in setting up a simulation is the instantiation and placement of nodes. This loop creates *NUMBER_OF_NODES Node* objects and associated *Protocol* objects. The *SimpleMultiHop Protocol* objects are initialized such that Node *i* forwards to Node *i+1*, with the last node set as the sink with a null *nextHop* argument. Nodes are instantiated with six arguments: the controlling *Simulation* object associated with this node, a String description of the node, the *x* and *y* coordinates of the node in simulation space, the initial energy in Joules, and a flag representing whether this node is a base station.

```
79
80              display = new SimFrame(this, globalParams);
81              events.enqueue(packet_generator, null, packetPeriod);
82              runUntil(100.0);
83          }
84      }
```

Finally, the display is instantiated with two arguments: the controlling *Simulation* object

and this simulation's global parameter block. *globalParams* is a class variable provided by *Simulation* and is utilized in most cases. The *packet_generator* is added to the event queue, to be fired in *packetPeriod* seconds, and the simulation is started with an end time of 100.0 seconds.

6.4.3 Output of Sample Simulation

Figure 6.6 is a screenshot of the simulator running the *SimpleSimulation* code above. The positions of the five instantiated nodes are automatically displayed, and the "Simple MultiHop" category has been added to the parameter category selection menu. Selecting this category displays the two parameters added by the example code and their values from the global *ParameterBlock*.

Output from the *println* statements appears as follows:

```
Node 1 sending packet SimData 6
Node 2 sending packet SimData 6
Node 3 sending packet SimData 6
```

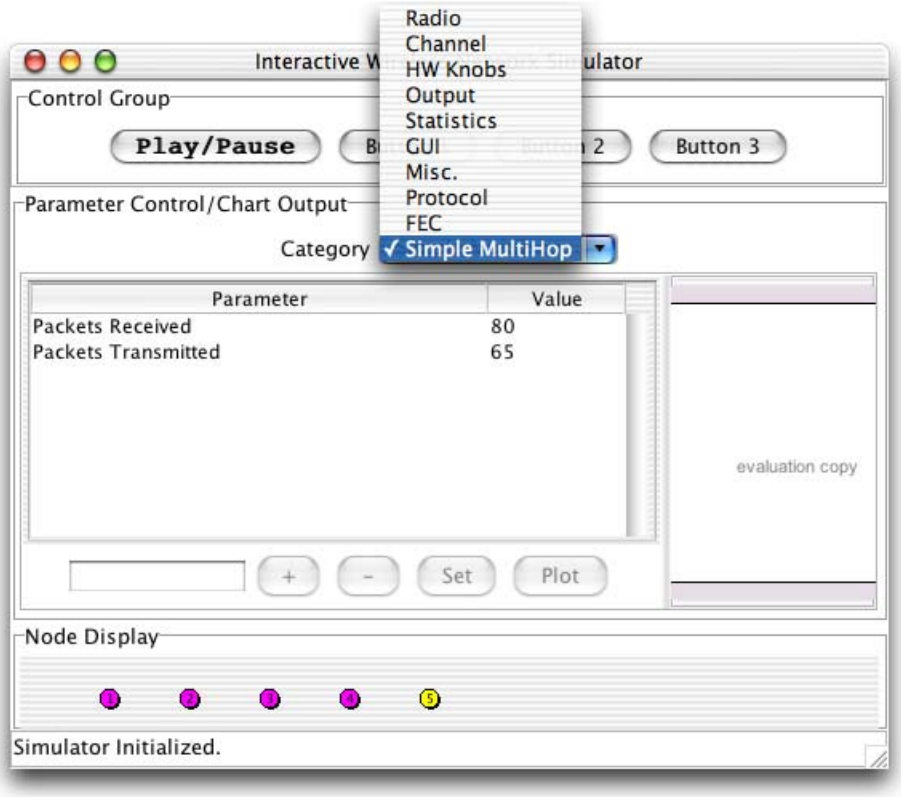


Figure 6.6: Screenshot of the simulator GUI (Mac OS X platform) resulting from the tutorial example. Note the addition of the Simple MultiHop parameter category.

```
Node 4 sending packet SimData 6
Sink node received data SimData 6 at time 6.045909717581789
Node 1 sending packet SimData 12
Node 2 sending packet SimData 12
Node 3 sending packet SimData 12
Node 4 sending packet SimData 12
Sink node received data SimData 12 at time 12.040823392696298
Node 1 sending packet SimData 18
...
```

Variations in end-to-end packet delay result from the randomized delay of line 47, as well as other random variables in the implementation of the MAC layer within *Node.*

6.5 Performance

The simulation speed for the simple multihop example above, as well as the addressed and unaddressed forwarding scenarios described in Chapter 5, was measured on a Sun Blade 2000 workstation running Java 1.4.1. Measurements of user time, as reported by the UNIX *time* function, are listed in Table 6.1. (System time—the processing time required by the core operating system for I/O, etc., was negligible in comparison.) As a reference for comparison, the performance of a standard three-node wireless multihop test on the *ns-2* simulator is also included[1]. In both cases, graphics were disabled and an ideal channel was simulated. Packet counts refer only to the number of packets *initiated* by source nodes. The actual number of relayed packets in a 500-node network is significantly higher.

Table 6.1: Performance comparison of *ns-2* vs. custom Java simulation, and of various protocols under Java simulation

	Number of Sourced Packets and User Time (s) on Sun Blade 2000				
Simulation:	1	10	100	1000	10,000
ns-2, 3 node multihop	0.40	0.41	0.48	1.6	12.0
Java, 5 node multi-hop (Section 6.4)	0.35	0.40	0.65	1.2	5.0
Java, unaddressed forwarding 500 nodes	2.02	2.53	6.93	52.16	612.3
Java, addressed forwarding 500 nodes	13.28	13.56	15.85	38.83	267.7

1. This is the *wireless-test.tcl* file provided with the ns-2.1b8a distribution.

The simple multihop routing tests provide a reasonable comparison between the two simulators, and Table 6.1 indicates that the performance of the Java simulator compares favorably with *ns-2*. Comparable performance for low packet counts suggests that the initialization time of the Java virtual machine is competitive with the *tcl*-based initialization for a simple *ns-2* simulation. Java's superior performance for larger packet lengths is most likely due to the use of 802.11b and DSR as default MAC and routing protocols, rather than a simple multihop protocol.

The number of nodes and the protocol under simulation have a substantial impact on simulation speed. In terms of simulation speed, addressed forwarding has a higher startup cost than unaddressed forwarding due to the metric announcement messages necessary to initialize routing tables, but a lower per-packet cost since receiving nodes may ignore all packets not explicitly addressed to them. It is somewhat unusual that a 10,000-packet simulation of unaddressed forwarding requires over an order of magnitude more simulation time than a 1,000-packet simulation. The most probable explanation is the repeated invocation of Java's garbage collection mechanism for the longer simulation.

6.6 Summary and Impact

A highly interactive simulation tool for high density networks has been developed using the Java language. The tool facilitates the analysis of power aware protocols with real-time displays and graphs, a flexible parameter architecture, and a powerful model engine.

This tool was utilized to generate all results in Chapter 5, and the convenience and time savings provided by the tool's interactive features proved invaluable. For instance, extensive simulations of parameter sweeps were performed only after interactive explorations using the real-time parameter editing and graphing tools. Protocol models were debugged swiftly by observing the visual node display and varying appropriate parameters. In many ways, the simulation tool took on a dual role as a visual protocol debugger.

Upon this tool's public release, it is the author's hope that the interactive features, modeling capabilities, and extensibility of the simulator will provide a welcome contribution to the networking community.

Chapter 7

Discussion and Conclusion

The shrinking size and increasing density of next-generation wireless devices imply reduced battery capacities, meaning that emerging wireless systems must be more energy-efficient than ever before. This exploration of wireless communication for high-density wireless networks has revealed two general conclusions regarding energy-efficient communication. First, it is crucial that energy-efficient communication software layers be based upon sound models of the hardware on which they will operate. Incomplete or inaccurate energy models lead to surprising discrepancies between designers' expectations and system realities. Second, two power management techniques used widely by hardware designers hold great promise for higher layers of the communication subsystem: application-specific design and energy-quality scalability.

This dissertation has presented a model framework for analyzing the energy and quality scalability of wireless communication in terms of hardware parameters and communication performance. The specific hardware parameter of dynamic voltage scaling has been explored in depth, and an early implementation of DVS on a commercial microprocessor has been incorporated into a prototype microsensor node. A power aware middleware layer have been proposed to link energy scalable "knobs" in hardware to high-level metrics of communication performance, and examples of optimal middleware policies have been presented for a real-world microsensor node with both simulations and information-theoretic bounds on coding performance. Energy-quality scalability has been shown to alleviate the need for multihop routing, and when multihop *is* required over very large sensor fields, an unusual unaddressed forwarding scheme has shown great promise as an application-specific solution.

7.1 Summary: Impact on System Lifetime

The ultimate goal of energy-efficient design in battery-powered systems is the extension of

the system's useful lifetime. This section summarizes the impact on lifetime of the techniques discussed in this dissertation.

Table 7.1: Energy scalability of μAMPS-1 hardware knobs [39].

Knob	Min. Power	Max. Power	Scalability
Base	136 mW	136 mW	1x
Processor	72 mW	564 mW	7.8x
Radio	348 mW	1224 mW	3.5x

Table 7.1 summarizes the energy scalability of the μAMPS-1 processor and radio. The "base" power represents the power consumed by sensors, memories, and other components unaffected by energy scalability. DVS enables nearly a factor of eight scalability, and a reduction of (564 - 72) = 492 mW from a peak operating power from (136+564+1224) = 1924 mW enables a 26% reduction in peak operating power. Under the rough assumption of peak power during periods of activity and zero power during idle periods, this 26% reduction translates into a system lifetime extension of 34%. Variable radio output power enables over a factor of three in energy scalability and a power reduction of (1224 - 348 mW) = 876 mW, a 45% reduction in peak power, or a lifetime extension of 82% under the above assumptions. Combining both techniques would reduce peak power to (136+72+348) = 556 mW for a reduction of 71% and a 345% extension in system lifetime. The value of multi-dimensional trade-offs is therefore clear. While the assumption of peak operating power is inaccurate in practice, the above results serve as useful bounds for interpreting the impact of upper-layer knobs.

The knob of variable-strength coding often does *not* save energy. More often, coding imparts a higher energy cost but enables a node to transmit with a greater range or reliability than otherwise possible with variable output power alone. In systems where the per-bit decoding energy is significantly lower than the per-bit transmission energy (as in Section 3.6.2), coding may allow the use of lower output power levels, enabling a lifetime extension up to the limits provided by *radio* power scaling as discussed above.

In high-density multihop networks, the knobs of radio output power and coding scheme become fixed to their minimum meter/Joule setting. The near-guarantee of relay

nodes at all distances effectively *reduces* the operational diversity of the system, allowing the output power and coding knobs to remain fixed. The protocol then takes paramount importance, and Figure 5.13 has illustrated that an unaddressed protocol coupled with receiver shutdown can extend the system's lifetime by a factor of five. However, this lifetime extension is orthogonal to any gains provided by DVS, which would provide a multiplicative lifetime extension determined by the *processor* power scaling as discussed above.

The subtle distinction between energy and lifetime, which has been ignored to this point, is worthy of note. Equating energy savings with lifetime extension assumes that energy is consumed evenly distributed across the network. This is rarely the case. For instance, microsensor nodes within transmission range of a base station would be burdened with frequent relay transmissions. Hence, lifetime computations for large networks must consider the "weakest links" in the transmission chain and ensure that transmissions are rotated to ensure an even spatial distribution of energy consumption to the greatest extent possible. Bhardwaj [5] derives policies for evening spatial energy consumption for low-density relay nodes placed at suboptimal distances.

7.2 The Top Five Myths about the Energy Consumption of Wireless Communication

To summarize this dissertation with high spirits and in tribute to D. Letterman [46], we summarize several key points emerging from this work by identifying and dispelling the "Top Five Myths" about energy-efficient wireless communication.

5. "Communication energy scales as d^n.

While the path loss of radiated radio energy is commonly modeled as d^n, this loss does not dominate the energy consumption of real-world wireless communication. A more complete model of communication energy is $\alpha + \beta d^n$. The α term, representing the *distance-independent* and constant power dissipated by the radio and digital electronics, exceeds the distance-dependent power of the output power amplifier.

4. “Multihop saves energy.”

Due to the dominance of distance-independent energy terms for short-range wireless communication, reducing the distance through intermediate hops does not save energy.

3. “Wireless performance can’t scale gracefully.”

The correct reception of bits is indeed an all-or-nothing proposition, but the performance of wireless communication can be parameterized and gracefully degraded. This chapter has enumerated range, reliability, delay, and throughput as scalable performance parameters.

2. “Abstractions lead to inefficiency.”

Abstraction is all too often viewed as a trade-off between simplicity and efficiency. Our power aware abstractions, however, offer a bridge between energy-scalable hardware and performance-scalable protocols. An API encourage the protocol layer to expose precise performance demands, allowing the energy-agile hardware to scale back performance in exchange for energy savings. This is an abstraction that encourages the efficient use of energy.

And the number one myth about the energy consumption of wireless communication:

1. “The protocol is independent of the application and hardware.”

The traditional view of isolated protocol layers is giving way to the realization that cross-layer collaboration is essential for efficient communication. This is the overarching theme that unites each of the topics presented in this chapter.

As novel applications of high density wireless networks begin to emerge, we foresee the emergence of application-specific protocols for each class of application, and increasing cooperation between hardware and protocols to enable energy- and quality-scalable communication. These trends, guided by accurate characterizations of the energy usage of hardware and applications, will ensure that next-generation wireless devices deliver the battery life required by the most demanding users.

7.3 Future Directions

As the primary goal of this work was to build bridges across traditionally distinct disciplines, it is the hope of the author that this work will suggest a number of exciting possibilities for extended exploration.

For hardware designers, a significant challenge is the design and real-world implementation of additional "knobs" for energy scalability. While DVS is now accepted and widely used, many knobs that have been published and analyzed have yet to be implemented. Scalable modulation (binary *vs.* *M*-ary) is one such example—the challenge of synchronizing the sender and receiver to the same modulation scheme is perhaps the greatest engineering challenge.

At the middleware layer, one notable open problem is the design of efficient algorithms for continuously optimizing energy consumption over multiple dimensions of hardware knobs and performance metrics. One-dimensional optimization is common and is already implemented in power control for 802.11b and IS-95 CDMA, but this work has shown the potential for significant energy savings through multi-dimensional trade-offs. Moreover, the identification of high-level communication quality metrics in this work dovetails with the designs of wireless QoS frameworks. The integration of energy-scalable communication with contemporary notions of QoS would enhance the understanding and acceptance of both concepts.

On the protocol front, the future will herald the dominance of application-specific protocols for emerging high-density, energy-constrained wireless applications. Microsensors and digital multimedia have very different operational demands, and it is difficult to accept that both can operate efficiency using the same protocols. The development of application-specific protocols would accelerate the emergence of novel wireless applications.

Finally, two specific improvements to the simulation tool are of note. Numerous optimizations for speed are possible, and the simulation landscape may be enriched with additional features such as interference sources, moving stimuli for microsensors, and topological features. Enhancements to the simulation tool are planned, and the research community will be encouraged to adopt and extend the tool freely.

Bibliography

[1] G. Asada et al., "Wireless Integrated Network Sensors: low power systems on a chip," Proc. 1998 ESSCIRC, Sept. 1998.

[2] A. Azevedo *et al*. "Profile-based dynamic voltage scheduling using program checkpoints in the COPPER framework," *Proc. of Design, Automation and Test in Europe Conference (DATE)*, March 2002.

[3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R.H. Katz. "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Trans. on Networking*, 5(6), December 1997.

[4] Berkeley Wireless Research Center, "Pico Radio TestBed Photos," *http://bwrc.eecs.berkeley.edu/Research/Pico_Radio/Test_Bed/photos/Default_.html*

[5] M. Bhardwaj, "Power-Aware Systems," S.M. Thesis, Department of EECS, Massachusetts Institute of Technology, June 2001.

[6] M. Bhardwaj, R. Min, and A. Chandrakasan, "Quantifying and Enhancing Power Awareness of VLSI Systems," *IEEE Transactions on VLSI Systems*, vol. 9, no. 6, December 2001, pp. 757-772.

[7] J. Broch et al., "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, *Proc. ACM/IEEE Int'l Conf. on Mobile Computing and Networking*, Oct. 1998.

[8] K. Bult et al., "Low Power Systems for Wireless Microsensors," *Proc. ISLPED 1996*, pp. 17-21.

[9] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," *Proc. ISSCC 2000*, pp. 294-295.

[10] M. Campione *et al.*, *The Java™ Tutorial: A Short Course on the Basics*, Boston: Addison-Wesley (3rd Edition), 2000.

[11] M. Campione *et al*., *The Java Tutorial Continued: The Rest of the JDK*, Boston: Addison-Wesley, 1998.

[12] A. Chandrakasan et al., "Design Considerations for Distributed Microsensor Systems," *Proc. CICC 1999*, pp. 279-286.

[13] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," to appear in *ACM Wireless Networks Journal*, vol. 8, no. 1, 2002.

[14] CMU Monarch Project, "The CMU Monarch Project's Wireless and Mobility Extensions to *ns*, August 5, 1999, *http://www.monarch.cs.cmu.edu/*

[15] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* Cambridge, MA: MIT Press, 2001, pp. 588-591.

[16] S. Das, C. Perkins, and E. Royer. "Performance comparison of two on-demand routing protocols for ad hoc networks," *Proc. INFOCOM 2000*, March 2000.

[17] V. De and S. Borkar, "Technology and design challenges for low power and high performance," Proc. ISLPED 1999, Aug. 1999, pp. 163-168.

[18] S. Dhar and D. Maksimovic. "Switching regulator with dynamically adjustable supply voltage for low power VLSI", *Proc. 27th IEEE Conference of the Industrial Electronics Society*, vol. 3, 2001, pp. 1874 -1879.

[19] DIGITAL Semiconductor, *DIGITAL Semiconductor SA-1100 Microprocessor Technical Reference Manual*, 1998.

[20] A. Dudani, F. Mueller, and Y. Zhu. "Energy-conserving feedback edf scheduling for embedded systems with real-time constraints." In *Proc. ACM SIGPLAN Joint Conference Languages, Compilers, and Tools for Embedded Systems (LCTES'02) and Software and Compilers for Embedded Systems (SCOPES'02)*, June 2002.

[21] D. Estrin et al., "Next Century Challenges: Scalable Coordination in Sensor Networks," *Proc. Mobicom 1999*, Aug. 1999, pp. 263-270.

[22] "eCos Documentation," *http://sourceware.cygnus.com/ecos/docs.html*

[23] K. Fall and K. Varadhan, Eds., "*ns* manual/*ns* Notes and Documentation," *http://www.isi.edu/nsnam*

[24] L. Feeney and M. Nilsson. "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," in Proc. *IEEE INFOCOM 2001*, Anchorage, AK, 2001.

[25] S. Floyd et al, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *IEEE/ACM Trans. on Networking*, Nov. 1996.

[26] R. Gonzalez, B.M. Gordon, and M.A. Horowitz, "Supply and Threshold Voltage Scaling for Low Power CMOS," *IEEE Journal of Solid State Circuits*, vol. 32, no. 8, pp. 1210-1216, August 1997.

[27] J. Goodman, A. Dancy, and A. Chandrakasan, "An Energy/Security Scalable Encryption Processor using an Embedded Variable Voltage DC/DC Converter," *Journal of Solid State Circuits*, Vol. 33, No. 11, Nov. 1998, pp. 1799-1809.

[28] D. Griswold. "The Java HotSpot Virtual Machine Architecture," March 1998. Sun Microsystems Whitepaper.

[29] M. Gruteser *et al.* "Exploiting Physical Layer Power Control Mechanisms in IEEE 802.11b Network Interfaces," University of Colorado at Boulder Technical Report CU-CS-924-01, December 2001.

[30] R. Hahn and H. Reichl, "Batteries and power supplies for wearable and ubiquitous computing," *3rd Int'l Symposium on Wearable Computers*, 1999, pp.168 -169.

[31] S. Haykin, J. Litva, T.J. Shepherd, *Radar Array Processing*, Springer-Verlag, 1993.

[32] R. Hedge and N. Shanbhag, "A low-power digital filter IC via soft DSP." *Proc. CICC 2001*, pp. 309-312.

[33] C. Hedrick, "RFC-1058: Routing Information Protocol." Request For Comments, June 1988. *http://www.ietf.org/rfc/rfc1058.txt*.

[34] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. HICSS 2000*, Jan. 2000.

[35] W. Hoschek, "The Colt Distribution," November 2002, *http://hoschek.home.cern.ch/hoschek/colt/*

[36] F. Howell and R. McNab, "simjava: a discrete event simulation package for Java with applications in computer systems modelling, " in *Proc. First International Conference*

on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation, Jan 1998.

[37] C.-H. Hsu and U. Kremer. “Compiler-Directed Dynamic Voltage Scaling Based on Program Regions.” Technical Report DCS-TR-461, Department of Computer Science, Rutgers University, November 2001.

[38] N. Hutchinson and L. Peterson, “The x-Kernel: an architecture for implementing network protocols,” *IEEE Transactions on Software Engineering*, vol. 17, no. 1 , Jan. 1991, pp. 64 -76.

[39] N. Ickes *et al.*, “A Power Aware Wireless Microsensor Node,” design contest entry for *Int’l Symposium on Low Power Electronics and Design (ISLPED’02)*, Monterey, CA, August 2002.

[40] M. Jeruchim, P. Balaban, and K.S. Shanmugan. *Simulation of Communication Systems: Modeling, Methodology, and Techniques.* 2nd edition, 2000. New York: Kluwer Academic Publishers, p. 578.

[41] E.-S. Jung and N. Vaidya, “A Power Control MAC Protocol for Ad Hoc Networks,” *Proc. ACM MobiCom 2002*, September 2002, pp. 36-47.

[42] P. Karn, “MACA — A New Channel Access Method for Packet Radio,” *Proc. ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, September 1990.

[43] L. Kleinrock, "On Giant Stepping in Packet Radio Networks," UCLA, Packet Radio Temporary Note #5, PRT 136 , March 1975.

[44] B. Leibowitz, B.E. Boser, and K. Pister, “CMOS ‘smart pixel’ for free-space optical communication,” *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 4306A (Electronic Imaging '01), San Jose, CA, 21-26 January 2001.

[45] S. Leibson. “XScale (StrongArm -2) Muscles In.” *Microprocessor Report* , 14(9):7-12, Sept. 2000.

[46] D. Letterman and S. O’Donnell, *David Letterman's Book of Top Ten Lists and Zesty Lo-Cal Chicken Recipes.* New York: Bantam Books, November 1995.

[47] P. Lettieri and M. B. Srivastava, “Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency,” *Proc. INFOCOM ‘98*, pp. 564-571.

[48] S. McCanne and S. Floyd. *ns* Network Simulator. *http://www.isi.edu/nsnam/ns/*

[49] R. McEliece, *The Theory of Information and Coding*, in *Encyclopedia of Mathematics and its Applications*, ed. G.-C. Rota, vol. 86, p. 127.

[50] N. McKeown, “CS244a: An Introduction to Computer Networks, Handout 5: Internetworking and Routing”, CS244a Course Lecture Slides, Winter 2003, Stanford University, pp. 8-9. *http://www.stanford.edu/class/cs244a/*

[51] *Ibid.* p. 72

[52] S. Miller, “Digital Communication over Fading Channels,” Course notes for Wireless Communication Systems, Texas A&M University, 2001. *http://ee.tamu.edu/~smiller/Wireless/Unit4A.pdf*

[53] R. Min *et al.*, “Energy-Centric Enabling Technologies for Wireless Sensor Networks,” *IEEE Wireless Communications* (formerly *IEEE Personal Communications*), vol. 9, no. 4, August 2002, pp. 28-39.

[54] R. Min *et al.*, "Low-Power Wireless Sensor Networks," *Proc. 14th Int'l Conf. on VLSI Design*, Bangalore, India, Jan. 2001, pp. 205-210.

[55] R. Min and A. Chandrakasan, "A Framework for Energy-Scalable Communication in High-Density Wireless Networks", *Proc. ISLPED '02*, Monterey, CA, pp. 36-41.

[56] R. Min and A. Chandrakasan, "Top Five Myths about the Energy Consumption of Wireless Communication," *ACM Sigmobile Mobile Communication and Communications Review (MC2R)*, vol. 6, no. 4.

[57] R. Min and A. Chandrakasan, "Energy-Efficient Communication for High Density Networks," submitted for publication to *Ambient Intelligence: Impact on Embedded-System Design*, eds. T. Basten, M. Geilen, H. de Groot. The Netherlands: Kluwer Academic Publishing.

[58] J. P. Monks et al. "A Power Controlled Multiple Access Protocol for Wireless Packet Networks," *Proc. IEEE INFOCOM 2001*, April 2002.

[59] J. Nagle, "RFC-896: Congestion Control in IP/TCP Internetworks." Request For Comments, January 1984. *http://www.ietf.org/rfc/rfc896.txt*.

[60] National Semiconductor Corporation, LMX3162 Evaluation Notes and Datasheet, April 1999.

[61] S. Narayanaswamy et al. "Power Control in Ad-Hoc Networks: Theory, Architcture, Algorithm and Implementation of the COMPOW Protocol," *European Wireless 2002*, February 2002.

[62] B. Narendran et al., "Evaluation of an Adaptive Power and Error Control Algorithm for Wireless Systems," *Proc, ICC* '97, vol. 1, pp. 349-355.

[63] S. H. Nawab et al., "Approximate Signal Processing," *J. of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, Vol. 15, No. 1/2, Jan. 1997, pp. 177-200.

[64] M. Neuberg, *REALBasic: The Definitive Guide, 2nd Edition.* Sebastopol, CA: O'Reilly & Associates; 2nd edition, October 15, 2001.

[65] A. Ogielski, "SSFnet," Presentation at the DARPA Next Generation Internet Conference, Arlington, VA, Dec. 15 - 17, 1999. *http://www.ssfnet.org/Papers/ngi-dec99.pdf*

[66] Ohio State University, "DRCL JavaSim," *http://javasim.cs.uiuc.edu/*

[67] Ohio State University, "Evaluation of JavaSim," *http://javasim.cs.uiuc.edu/comparison.html*

[68] F. Op't Eynde et al., "A fully-integrated single-chip SOC for Bluetooth," Proc. ISSCC 2001, Feb. 2001, pp. 196-197, 446.

[69] S. Park, A. Savvides and M. B. Srivastava, "SensorSim: A Simulation Framework for Sensor Networks," *Proc. MSWiM 2000*, August 11, 2000

[70] T. Pering,. et al., "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms," *Proc. ISLPED 1998*, pp.76-81.

[71] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", *Computer Communications Review*, October 1994, pp. 234-244.

[72] Peter Menzel Photography, "Smart Dust," *http://www-bsac.eecs.berkeley.edu/~warneke/SmartDust/*

[73] P. Pillai and K. G. Shin. “Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems.” In *Proc. of the 18th ACM Symp. on Operating Systems Principles*, 2001.

[74] R. Poor. “Gradient Routing in Ad Hoc Networks,” *www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf*

[75] J. Pouwelse, K. Langendoen, and H. Sips, “Application-Directed Voltage Scaling,” *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, September 2002.

[76] R. Powers, “Advances and trends in primary and small secondary batteries,” IEEE Aerospace and Electronics Systems Magazine, vol. 9, no. 4, April 1994 pp. 32-36.

[77] W. H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, “Incomplete Beta Function, Student's Distribution, F-Distribution, Cumulative Binomial Distribution.” §6.2 in *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd ed. Cambridge, England: Cambridge University Press, pp. 219-223, 1992.

[78] J. Proakis. *Digital Communications.* 4th edition. New York: McGraw-Hill, 1983.

[79] J. Rabaey et al., “PicoRadios for Wireless Sensor Networks: The Next Challenge in Ultra-Low Power Design,” *Proc. ISSCC 2002*, pp. 200-201, February 2002.

[80] J. Rabaey et al., “PicoRadio supports ad hoc ultra-low power wireless networking,” Computer, vol. 33, no. 7, July 2000, pp. 42-48.

[81] V. Raghunathan *et al.* “E2WFQ: An Energy Efficient Fair Scheduling Policy for Wireless Systems,” *Proc. ISLPED’02*, August 2002, pp. 30-35.

[82] Jim Reich, personal conversation on September 4, 2002 in Pasadena, CA.

[83] RF Monolithics, Inc. “TR1000, 916.50 MHz Hybrid Transceiver” (product datasheet), *www.rfm.com/products/data/tr1000.pdf*

[84] E. Royer and C-K.Toh, “A review of current routing protocols for ad hoc mobile wireless networks,” *IEEE Personal Communications*, vol. 6, no. 2 (April 1999), pp. 46-55.

[85] C. Schurgers, O. Aberthorne, M.B. Srivastava, “Modulation Scaling for Energy Aware Communication Systems,” *Proc. International Symposium on Low Power Electronics and Design (ISLPED'01)*, Huntington Beach, CA, pp. 96-99, August 6-7, 2001.

[86] E. Shih, P. Bahl, and M. Sinclair, “Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Systems,” *Proc. MobiCom 2002*, Atlanta, GA, September 2002, pp. 160-171.

[87] E. Shih et al, “Physical Layer Driven Algorithm and Protocol Design for Energy-Efficient Wireless Sensor Networks,” *Proc. MOBICOM 2001*, July 2001.

[88] A. Sinha, “Energy-Scalable Software,” S.M. Thesis, Department of EECS, Massachusetts Institute of Technology, February 2000.

[89] A. Sinha and A. Chandrakasan, “Energy Aware Software,” *Proc. Thirteenth International Conference on VLSI Design*, Jan. 2000, pp. 50-55.

[90] A. Sinha and A. Chandrakasan, “Energy efficient filtering using adaptive precision and variable voltage,” *12th Annual IEEE Asic Conference*, Sept. 1999, pp. 327-331.

[91] A. Sinha, A. Wang, and A. Chandrakasan, “Algorithmic Transforms for Efficient Energy Scalable Computation,” Accepted for presentation at the *International Symposium on Low Power Electronics and Design (ISLPED)*, 2000.

[92] Krishnamurthy Soumyanath, response to a post-talk question at the MTL VLSI Seminar Series, Massachusetts Institute of Technology, September 10, 2003.

[93] D. Su *et al.*, "A 5GHz CMOS Transceiver for IEEE 802.11a Wireless LAN," *Proc. ISSCC 2002*, vol. 45, January 2002, pp. 92-93,449.

[94] S.-L. Su and S.-S. Shieh, "Reverse-link power control strategies for CDMA cellular network," *IEEE Int'l Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'95)*, vol. 2 , 27-29 Sep 1995, pp. 461-465 vol.2.

[95] M. Sudan, Course notes for *6.897 Algorithmic Introduction to Coding Theory.* Fall 2001, Lecture 5. Available at *http://theory.lcs.mit.edu/~madhu/FT01/*

[96] Transmeta Corporation. "LongRun® Power Management Technology." *http://www.transmeta.com/technology/architecture/longrun.html*

[97] K. Turner and I. Robin, "An Interactive Visual Protocol Simulator," *Computer Standards and Interfaces*, 23:279-310, Elsevier Science: Amsterdam, The Netherlands, October 2001.

[98] Alice Wang, A. Chandrakasan, S.V. Kosonocky, "Optimal Supply and Threshold Scaling for Subthreshold CMOS Circuits", *IEEE International Symposium on VLSI (ISVLSI)*, Apr 2002, pp. 7-11.

[99] Alice Wang, W. Heinzelman, and A. Chandrakasan, "Energy-Scalable Protocols for Battery-Operated Microsensor Networks," *Proc. SiPS '99*, Oct. 1999, pp. 483-492.

[100] Andy Wang, S.-H. Cho, C. Sodini, A. Chandrakasan, "Energy efficient modulation and MAC for asymmetric RF microsensor systems," *Proc. ISLPED 2001*, August 2001.

[101] B. Warneke, B. Atwood, K.S.J. Pister, "Smart dust mote forerunners," Proc. 14th IEEE Int'l Conference on MEMS, Jan. 2001, pp. 357-360.

[102] G. Wei and M. Horowitz, "A Low Power Switching Supply for Self-Clocked Systems," *Proc. ISLPED 1996*, pp. 313-317.

[103] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," in *Low Power CMOS Design*, A. Chandrakasan and R. Brodersen, eds., 1998, pp. 177-187.

[104] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*, 2nd edition, Reading, Mass.: Addison-Wesley, 1993, p. 236.

[105] S.-L. Wu et al. "A Multi-Channel MAC Protocol with Power Control for Multi-Hop Mobile Ad Hoc Networks," *The Computer Journal (SCI)*, vol. 45, no. 1, 2002, pp. 101-110.

[106] Y. Xu, J. Heidemann, and D. Estrin, "Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks," USC/ISI Research Report 527, October 12, 2000.

[107] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed Energy Conservation for Ad-hoc Routing," *Proc. IEEE MobiCom '01*, July 2001, pp. 70-84.

[108] K. Yao et. al., "Blind Beamforming on a Randomly Distributed Sensor Array System," *IEEE J. on Selected Topics in Communication*, Vol. 16, No. 8, Oct. 1998, pp. 1555-1567.

[109] F. Ye, S.Lu, and L. Zhang, "GRAdient Broadcast : A Robust, Long-lived Sensor Network," UCLA CS IRL Technical Report, September 2001

Appendix A

Additional Simulation Results for *Ad Hoc* Wireless Protocols

The need for communication protocols resilient to receiver shutdown is motivated by considering two popular protocols for *ad hoc* wireless LAN, AODV and DSR. On-demand protocols suit a low data rate sensing application and behave relatively well under dynamic changes in topology [16]. This exploration is presented separately of Chapter 5 as the results were obtained using a somewhat different simulation methodology.

A.1 Simulation Setup and Testbed

Figure 1.1 illustrates the test scenario and network topology. The simulation scene is fixed as a two dimensional, 500 m x 100 m landscape with all nodes distributed randomly (via uniform random variables) within in the field. Ten fixed nodes at one end act as sensors that generate data at a constant rate. The remaining nodes relay this data to a base station located at the opposite end of the field.

The parameters considered in the simulations are duty cycle, node density, and radio shutdown methodology. The two shutdown techniques deployed are *slotted* and *unslotted*,

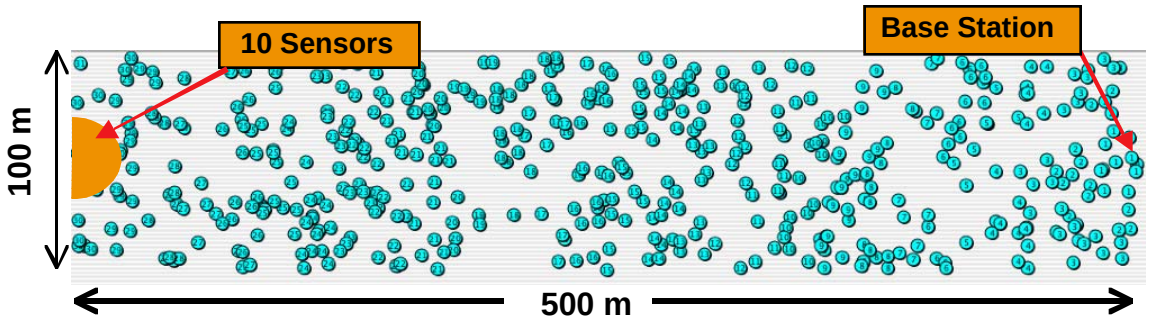


Figure 1.1: This simulation scenario consists of 500 randomly distributed nodes over a 500m x 100m field. Ten nodes (left edge) send packets destined to the base station (right edge) at constant bit rate.

as illustrated in Figure 1.2. In slotted shutdown, receivers change state at periodic intervals, while an unslotted shutdown policy allows individual nodes to shut down or awaken at any time. Duty cycles are varied by adjusting the probability that a node's receiver is on.

Simulations are run using *ns-2* [23] with the built-in CMU Monarch [14] implementations of AODV and DSR. These implementations feature a full TCP/IP protocol stack with 802.11b media access. Receive shutdown is simulated through appropriate procedure-calls to the *ns-2* energy model. Scenarios representing node location, radio receive states and packet sources are generated in advance. Mobility is disabled to simulate the static nodes that are characteristic of microsensor-type applications. Since stale data is of little use in latency-critical sensing networks, a packet is considered received only if it arrives at the base station within two seconds of its initial generation. The measured energy consumption and performance of μAMPS-1 sensor node form the basis of the simulations' energy model. This *ns-2* testbed was implemented by Piyada Phanaphat.

A.2 Simulation Results

Figures 1.1 through 1.3 summarize the results from *ns-2*. With a slotted shutdown technique (Figure 1.1a), routing with DSR and AODV both result in relatively flat and high throughputs when the duty cycle is raised above 20 percent in a 500-node scenario (1 node per 100 square meters). At a duty cycle below 20 percent, the network suffers from link breaks as shown by near-zero throughputs. Note also that at high duty cycle, the

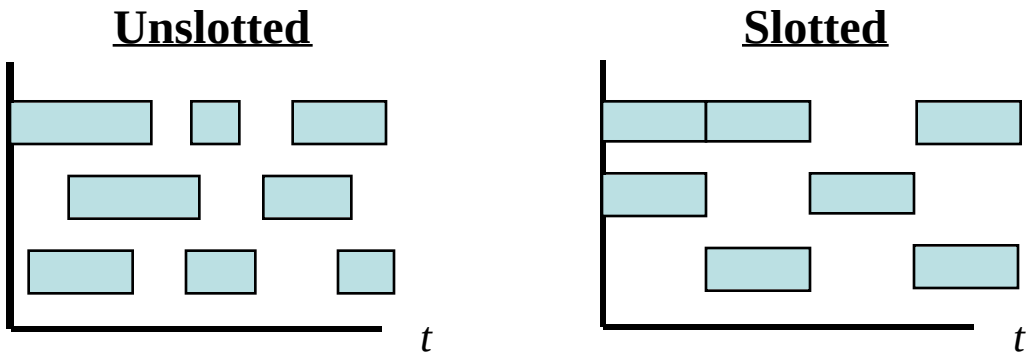
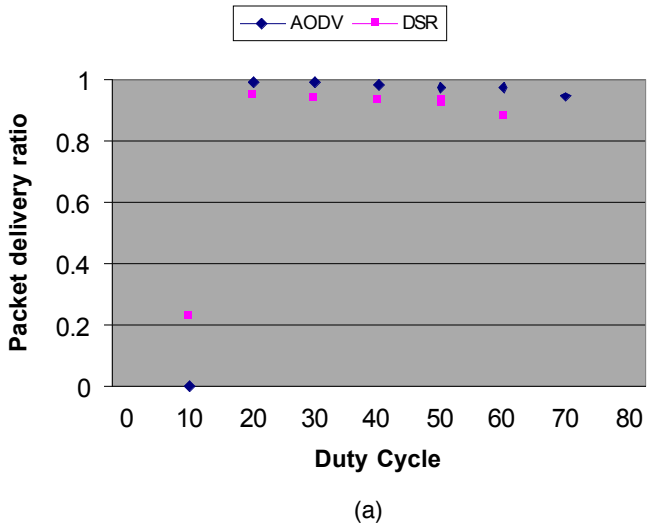
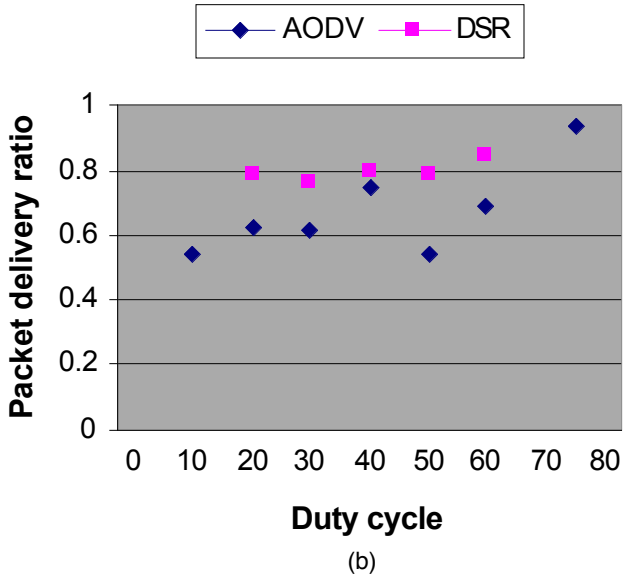


Figure 1.2: In *unslotted* receiver shutdown, receivers sleep and wake without coordination. *Slotted* shutdown utilizes a coordinated time reference across the network.

throughput of both protocols slightly declines. The likely cause is that the network becomes burdened with protocol and MAC control data as shown in Figure 1.2. At high duty cycles, the proportion of control data packets dramatically increases with respect to the overall packets sent in the network.



(a)



(b)

Figure 1.1: Throughput for slotted-shutdown (a) and unslotted-shutdown (b) of packets received at the base station.

Figure 1.1b suggests that both AODV and DSR networks fare more poorly with unslotted shutdown than with slotted shutdown. The performance degradation is due to the frequent changes in network topology brought about by unslotted receiver shutdown.

The energy per received data packet scales with duty cycle as illustrated in Figure 1.3. As low-data rate networks such as sensor networks become receive energy dominated, the total network receive energy scales linearly with duty cycle when the network is overpro-

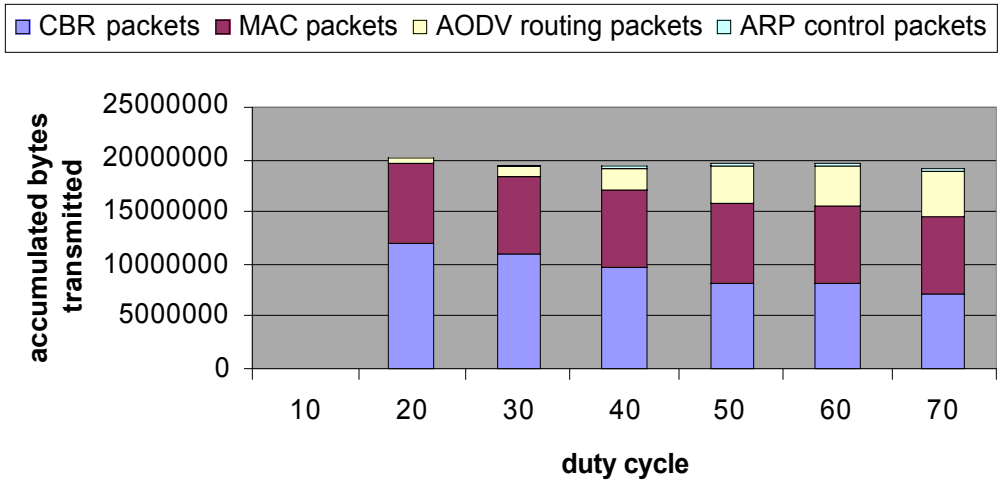


Figure 1.2: Total number of bytes transmitted in the network categorized by packet type, for AODV slotted shutdown. Low duty cycle networks exhibit retransmissions while high duty cycle networks are burdened by control packet overhead.

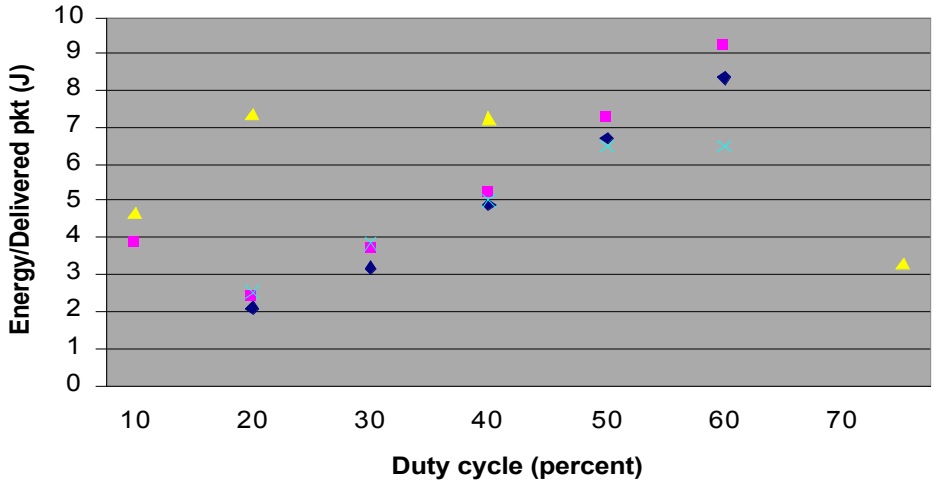


Figure 1.3: Energy per packet received at the base station for slotted and unslotted shutdown.

visioned with receivers. The erratic results from unslotted AODV are simply a consequence of the erratic throughput evident in Figure 1.1b.

It is clear that these protocols prefer infrequent but large changes in network topology, as brought about by slotted shutdown, over the small but frequent changes brought about by unslotted shutdown. AODV, in particular, persistently attempts to send data even in the case of frequent route changes and link breaks, leading to congestion. In sum, frequent shutdowns effect routing performance in a similar way as mobility, for the above results are consistent with those from mobility experiments with these protocols [7].

The moral of the preceding exploration is that protocols designed for any particular purpose (in this case, routing among high-energy nodes) can be ill-suited for others (low-power transmission to a base station). A primary concern of Chapter 5 is the transmission of data from densely populated, energy-constrained nodes to a fixed base station. It therefore behooves of us to create an *application-specific protocol* for this purpose.