Power-Aware Systems

by

Manish Bhardwaj

B.A.Sc.(Honors), Nanyang Technological University (1997)

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

Master of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001

Author .....
Department of Electrical Engineering and Computer Science
May 11, 2001

Certified by .....
Anantha Chandrakasan
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Power-Aware Systems

by

Manish Bhardwaj

Submitted to the Department of Electrical Engineering and Computer Science
on May 11, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

In this thesis, we formalize the notion of *power-aware* systems and present a methodology to systematically enhance power-awareness. We define a power-aware system as one which scales its power consumption with changes in its operating scenario with a view to maximizing its energy efficiency. Operating scenarios are primarily characterized by five dimensions - input statistics, output quality requirements, tolerable latency (and/or throughput constraints), internal state and environmental conditions. We *quantify* the power-awareness of a system by equating it to the energy efficiency with which it can track changes along these dimensions. This is done by comparing the system's energy consumption in a scenario to that of a dedicated system constructed to execute only that scenario as energy efficiently as possible. We then propose a systematic technique that enhances the power-awareness of a system by composing ensembles of point systems. This technique is applied to multipliers, register-files, digital filters and variable-voltage processors demonstrating increases in battery-lifetimes of 60%-200%. In the second half of this thesis we apply power-awareness concepts to data-gathering wireless networks. We derive fundamental bounds on the lifetime of networks and demonstrate the tightness of these bounds using a combination of analytical arguments and simulation. Finally, we show that achieving a high degree of power-awareness in a wireless sensor network is equivalent to optimally or near-optimally solving the *role-assignment problem.* Provably optimal role assignment strategies using linear programming are presented. Hence, optimal strategies can be determined in a time that is polynomial in the number of nodes. As a result of applying power-awareness formalisms, the energy efficiency, and hence the lifetime of data gathering networks increases significantly over power-unaware schemes.

Thesis Supervisor: Anantha Chandrakasan
Title: Associate Professor

Acknowledgments

One afternoon, over the course of a half-hour trans-Atlantic phone call, Anantha infected me with his energy and enthusiasm for wireless sensor networks. I spent that evening intrigued by how computation and communication interact in such networks. Little did I realize then that it would take two years, a new education and lots of help from lots of people to gain a first understanding of these interactions!

None of this would be possible without Anantha. From that phone call to his reading of this thesis, Anantha's involvement with this work - as an advisor, teacher, mentor and friend - has been energetic, total and indispensable. I cannot thank him enough for it!

Employing my by-now-infamous habit of classifying everything, I want to thank my "Erstwhile-Core-Gang", "The Elders", "Out-of-Lab-Gang" and "Friends". Thanks to Rex Min for clocking all that time with me, actively soliciting my views on Evolutionary Psychology, believing that power-awareness was *really* different from low-power and for being my first friend at MIT; Eugene Shih for listening to my rambles on optimization, network flows and bounds, for teaching me innumerable LaTeX tricks and for his invigorating idealism (always dispensed in liberal doses!); Amit Sinha for sharing my passion for discussing the meaning of life (esp. existentialism) over Hot Vanilla Lattes, and for his enthusiastic endorsements of matrimony; Alice Wang for the Lobdell lunches, the afternoon coffee walks, for bravely bearing the mantle of being the lone female graduate student in the group this past year, and for being the true "MIT insider"; SeongHwan Cho for patiently listening to my (often half-baked) ideas, for explaining radios and their energy models and for being the comforting elder figure in the lab; Jim Goodman for making me feel at home my very first day at MIT and for being a mentor and a thorough gentleman; Wendi Heinzelman for taking the time to explain sensor networks to me and for the detailed feedback on my work and papers; Raj Rao for his friendship, for pointing me to 6.241 and for sharing his encyclopedic knowledge of what lies beyond the Harvard bridge; Ghislain Granger for being a great roommate my first year at MIT and Weichang and Hua for being

great apartment-mates! Thanks to Timothy Garnett for implementing the simulator that allowed verification of my work on bounds and to Ovidiu Gheorghioiu for his feedback on chapter eight and for working on speeding up optimal role assignment algorithms. Thanks also to the many friends who made it all worthwhile - Albert, Ali, Andy, Belinda, Ben and Mary-Kathryn, Danny, Dario, Dave Carter, Dave Wentzloff, Frank, Fred, James and Jo, Jelena, Kush, Muyiwa, Paul-Peter, Paul Herz, Piyada, Rajesh, Raul, Syed, Theodore, Travis Furrer and Travis Simpkins. Thanks to Margaret Flaherty, our administrative assistant, for keeping us well-stocked and shielding us from the paperwork!

I wish to thank Professors Dave Forney, Munther Dahleh, John Tsitsiklis and Gilbert Strang for their inspirational teaching; Prof. John Kassakian for his advice as my graduate advisor and Professors Charlie Sodini and Benjamin Premkumar (at Nanyang Technological University, Singapore) for writing fellowship recommendations. Thanks to IBM for their Research Fellowship. Thanks also to Professor Rajeev Jain and everyone at Angeles Design Systems for their generous hospitality.

In more ways than I can possibly list, this thesis would be impossible without my family - my parents Manoj and Renu Bhardwaj and my brother Ashutosh Bhardwaj. Thank you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Low power system design assuming a worst-case power dissipation scenario is being supplanted by a more comprehensive philosophy variously termed power-aware or energy-aware or energy-quality scalable design [6]. The basic idea behind these essentially identical approaches is to allow the system power to scale with changing conditions and quality requirements.

There are two main views to motivating power-aware design and its emergence as an important paradigm. The first view is to explain the importance of power-awareness as a consequence of the increasing emphasis on making systems more *scalable*. In this context, making a system scalable refers to enabling the *user* to tradeoff system performance parameters as opposed to hard-wiring them. Scalability is an important figure-of-merit since it allows the end-user to implement operational policy, which often varies significantly over the lifetime of the system. For example, consider the user of a portable multimedia terminal. At times, the user might want extremely high performance (say, high video quality) at the cost of reduced battery lifetime. At other times, the opposite might be true - i.e. the user might want bare minimum perceptual quality in return for maximizing battery lifetime. Such trade-offs can only be optimally realized if the system was designed in a power-aware manner. A related motivation for power-awareness is that a well designed system must *gracefully* degrade its quality and performance as the available energy resources are depleted [41]. Continuing our video example, this implies that as the expendable energy decreases, the

system should gracefully degrade video quality (seen by the user as increased "blockiness", for instance) instead of exhibiting a "cliff-like", all-or-none behavior (perfect video followed by no video) [41, 36].

While the view above argues for power-awareness from a user-centric and user-visible perspective, one can also motivate this paradigm in more fundamental, system-oriented terms. With burgeoning system complexity and the accompanying increase in integration, there is more diversity in the operating scenarios than ever before. Hence, design philosophies that assume the system to be in the worst-case operating state most of the time are prone to yield sub-optimal results. In other words, even if there is little explicit user intervention, there is an imperative to track operational diversity and scale power consumption accordingly. This naturally leads to the concept of power-awareness. For instance, the embedded processor that decodes the video stream in a portable multimedia terminal can display tremendous workload diversity depending on the temporal correlation of the incoming video bit-stream. Hence, even if the *user* does not change quality criteria, the processor must exploit this operational diversity by scaling its power as the workload changes.

Since low-energy and low-power are intimately linked to power-awareness, it is important and instructive to provide a first-cut delineation of these concepts even at this introductory stage. This is to convince the reader that power-awareness as a metric and a design driver *doesn't* devolve to traditional worst-case-centric low-power/low-energy design. As preliminary evidence of this, consider the system architect faced with the task of increasing the power-awareness of the portable multimedia terminal alluded to above. While the architect can claim that certain engineering reduces worst-case dissipation and/or overall energy consumption of the terminal and so on, these traditional measures still fall short of answering the related but different questions:

How *well* does the terminal scale its power with user or data or environment dictated changes? What prevents it from being arbitrarily proficient in tracking operational diversity? How can we *quantify* the *benefits* of such proficiency? How can we *systematically enhance* the system's ability to scale its power? What are the *costs*

of achieving such enhancements?

It is precisely these questions that this thesis addresses and attempts to resolve. We first build a formal framework that allows us to talk in precise terms about the notion of power-awareness and next we show how this metric can be enhanced in a systematic manner. In the latter half of the thesis, we use these concepts to attack the problem of maximizing energy efficiency of wireless sensor networks.

1.1 Contributions of this Thesis

Our work on power aware systems has resulted in the following contributions:

(a) Formalizing the Notion of Power-Awareness

To understand power-awareness in a rigorous fashion, we introduced the notion of *operating scenarios* which are characterized by the five key *awareness dimensions* (input statistics, output quality, tolerable latency, ambient environment and internal state). Next, the concept of dedicated *point systems* was introduced to bound the energy efficiency that the most power-aware system, or, *perfect system* could achieve. Finally, *scenario distributions* were introduced to describe the frequency with which the system resided in a set of scenarios. These formalisms allow system designers to discuss the challenge of increasing energy efficiency or power-awareness in precise and rigorous terms.

(b) Metrics for Quantifying Power-Awareness

The formalisms above are used to define the power-awareness of a system for a specified scenario distribution. As defined by us, power-awareness is a number between 0 and 1 and indicates the energy efficiency of the system relative to the most power-aware system. Equivalently, the number indicates the lifetime that this system would attain normalized to the lifetime of the most power-aware system. Thus, this metric not only allows one to see how far the practical system is from the ideal system but also allows an objective comparison of the power-awareness of various systems.

(c) Enhancing Power-Awareness via Ensemble Construction

If the only contribution of power-awareness theory, as developed in this thesis was to disambiguate the notion, it would have limited use. We have developed a formal technique to enhance the power-awareness of a system using a construction called *ensemble of point systems* which can lead to a significant increase in the energy efficiency of systems (for the examples that we present, it can triple the energy efficiency in some cases!).

In summary, our work on power-aware systems allows designers to talk in precise terms about the problem, quantify the power-awareness of their systems, gain an insight into how far are they from the most power-aware system and how they can get close to this optimal.

The other main contributions of this thesis are in the area of power-aware wireless sensor networks. By applying the formalisms above, we have tackled a significant challenge in wireless sensor networks - maximizing their sensing lifetime. The specific contributions in this area are,

(a) Bounds on Sensing Lifetime

To our knowledge, our work was the first to propose bounds on the active sensing lifetime that a wireless sensor network can achieve. As a result we now know the absolute physical limits of energy efficient collaboration in such networks. Our derived bounds have been verified via simulation and are either provably tight or near-tight.

(b) Optimal Collaborative Strategies

Having an energy efficient node is only half the story in wireless sensor networks. The other important half is to determine optimal collaborative strategies that allow these networks to maximize their sensing lifetime. Most previous work in this area dealt primarily with optimal routing. Our work introduces more rigorous formalisms of *role assignments* and *collaborative strategies* and we show that while routing is an important aspect, it is but one facet of the more holistic

notion of a collaborative strategy. We also pose the problem of determining the collaborative strategy that maximizes lifetime and then present polynomial time algorithms that can solve this problem.

1.2 Thesis Structure

The next four chapters deal with developing the theory and practice of power-aware systems while the last three deal with energy efficient wireless sensor networks. The next chapter reviews previous work in the area of power-aware systems. Chapter 3 deals with power-aware formalisms and the task of deriving a power-awareness metric. Enhancing power-awareness using the "ensemble of point systems" idea is the topic of chapter 4. We illustrate this idea for various systems in chapter 5. The next chapter introduces wireless sensor networks and the energy models used in the remainder of the thesis. Chapter 7 derives fundamental bounds on the lifetime of sensor networks. Discovering optimal collaborative strategies in polynomial time is dealt with in chapter 8. In the last chapter we summarize the work done and conclude.

Chapter 2

Background

It might seem implausible that with the extensive research done in the area of low power systems, there was no work that addressed the issue of power-aware systems in the manner that we propose to. While shades of the proposed framework can be seen in recent work, we believe that no published research has targeted the problem as formally and comprehensively as detailed in this thesis. In the following section we detail previous work which touched upon some of the issues that are relevant to engineering power-aware systems.

2.1 Early Beginnings: Linking Latency and Power

In 1992, the landmark paper [10] changed the very way the circuits and systems community thought about power in digital integrated circuits. While that paper has had tremendous overall impact in this area, we will focus on the power-latency trade-off that was first promulgated there. The authors demonstrated using contemporary VLSI technology that power could be decreased by close to 70% without sacrificing throughput if twice the latency could be tolerated. This demonstration explicitly linked power and tolerable latency. As an aside, the work also linked area and power, since trading latency for low power needs more area if the throughput is to be maintained. To this day, trading latency for low power is arguably the one method that has had the most significant impact on power dissipation.

The paper that further disambiguated this vital connection appeared three years later in the context of general purpose processors [19, 18]. In some sense, this paper re-iterated what [10] had established. Its message was - it makes no sense to talk about the "lowest energy circuit" till one imposed a latency bound on the circuit. While the link between tolerable latency and power/energy seems obvious today, it was not so before these two papers tied the two concepts together. As we shall see later, this link is of fundamental importance while discussing power-awareness.

2.2 Scaling Voltage with Desired Throughput

Fundamental work in operating at the lowest voltage level under given speed constraints was first described in [28]. The importance of this idea for ultra low-power digital systems was fully realized in [22]. This paper described a dynamically varying supply voltage as the workload changed - a technique that subsequently became very popular (and which recently found its way into several mainstream micro-processors, see for e.g. [30]). Scaling the voltage dynamically remains a powerful general purpose technique for achieving low energy in systems with varying throughput. This is because power varies quadratically with voltage and only linearly with other factors like operating frequency and switched capacitance. The other reason is that efficient DC-DC converters with efficiencies close to 90% have been demonstrated [14, 21, 20]. While it is a very powerful tool, fully dynamic voltage scaling is not necessarily the most energy efficient solution. First, voltage conversion has a bandwidth in the range of several tens or hundreds of kHz while digital circuits routinely run at sub-GHz speeds today. Hence, scaling voltage on a cycle by cycle basis is not possible (it is possible to increase the bandwidth somewhat at the cost of efficiency and output ripple, but this is still does not render the scheme feasible for fine grained variation). The other important reason that moving to a dynamic supply does not guarantee the most energy efficient solution is because a system that has been designed to operate for the worst-case load is not necessarily power optimal *even with* a reduced supply. It is possible, for instance, that small workloads can use totally different circuits/systems

that are more power efficient.

2.3 Scaling Power with Desired Quality

An extensive body of work exists in the area of scaling power (and by consequence energy) with the desired output quality.

One of the first non-trivial demonstrations of a dynamic voltage system was the scalable encryption chip described in [20]. A higher energy cost is incurred for a higher level of security and this translation is achieved by dynamically scaling the precision and as a result, the core voltage. This chip is an important example of a power-aware system.

Another approach to achieving power-aware computation is to scale the total switched capacitance with the desired quality. One such demonstration of scalable arithmetic was presented in [3]. The authors proposed a distributed arithmetic (DA) approach with the interesting characteristic that the system allows the smallest possible grain in trading off quality and energy.

The work on low power DCT and IDCT as described in [63, 62] combined the DA concept with the idea of exploiting DCT coefficient statistics to further reduce the amount of switched capacitance.

Another key paper concerns the design of power-scalable adaptive digital filters used in xDSL modem front-ends [42]. In adaptive filtering applications, high quality is usually desired during the "continuous adaptation" or "training" stage to ensure convergence to a small steady state error. Thereafter, the desired quality is highly dependent on several factors like the line conditions, the desired SNR, noise profile and channel characteristics etc. Hence it is important to scale the filter power with the quality needed. The authors describe a means of doing just that by reducing the amount of switched capacitance through clock gating. A related paper that adapts the filter quality in response to line conditions leading to an impressive 88% reduction in overall energy is [17].

One of the first papers in the area of energy scalable *algorithms* was [41], where

the authors discussed how signal processing algorithms could be transformed so that the quality degraded gracefully as the available energy decreased . The major impact of this work was in reducing switched capacitance at a very high level of abstraction. Unlike previous domain/application specific work, the scalability technique developed in this paper was demonstrated for an entire class of algorithms.

A more recent investigation of energy scalable software [54, 56] treats the problem even more generally. The authors argue that transformations that increase "energy scalability" or what we term power-awareness are ones that decrease the concavity, or equivalently, increase the convexity of the energy-quality curve. They demonstrate this "concave to convex" transformation for FIR filtering, polynomial evaluation and DCT/IDCT computation. This interesting contention of increasing convexity lends their approach a more formal and rigorous touch. However, the authors stop short of discussing what limits them from monotonically increases convexity. A related shortcoming is how the convexity relates to the overall energy consumption of the system and finally the problem of dealing with real-world curves that are non-convex.

2.4 Factoring Usage Patterns to Drive Low Energy Design

System architects have been factoring in typical usage patterns or benchmarks to drive design for several decades now. The most well-known example is clock gating in processors. Bus- and arithmetic coding are two well known low power/energy techniques that are based on (a mostly) heuristic analysis of usage patterns [58, 52]. An ingenious low swing technique suitable for data busses is also based on exploiting usage patterns [26]. Techniques that incorporate usage patterns more formally and rigorously include pre-computation [2] and low energy finite-state-machines as proposed in [4]. While these previous techniques have significant impact in their respective, often narrow, domains, none of them has led to a generally applicable technique for enhancing power-awareness.

In contrast, the first hints of the general technique for enhancing power-awareness that we propose in this thesis can be seen in the work on variable precision multiplication in digital speech processing [55] and to a lesser extent in variable length decoding in MPEG streams [13].

The work on variable length decoding (VLD) in MPEG streams [13] analyzed the statistical nature of the stream that was most likely to occur. This information was then used to drive the engineering of a hierarchical look-up technique that drastically reduced the energy of the average case. The main contribution of this paper was that it focussed on the average case rather than the worst case and set the tone for the work that would follow in [55] above. While the authors achieved impressive energy reduction for the case of VLD, they did not focus on generalizing the technique. Also, there are some hints of energy modelling and analytical derivation of the right hierarchy, but the final choice was guided more by heuristics than a formal analysis.

The research presented in [55] is possibly the first that explicitly incorporates the usage pattern of a multiplier to minimize the overall energy consumed. The authors treat their solution as a degenerate case of the dynamic power supply that [22] proposed. They argue that the bandwidth of the supply is not sufficient to enable a cycle by cycle change and hence propose a fixed two-supply system. One supply caters to the worst case (which is maximum precision in this case). The other supply is designed by factoring in the usage pattern of the multiplier. A complete analytical solution is presented to determine the other fixed supply once the precision distribution is known. Implicit in the above work is the acceptance that except for bandwidth, the dynamic power supply is still the best way to scale energy. Due to this, they do not address the issue whether their proposed system is in fact optimal or close to optimal in terms of energy. The other issue that remains unresolved is why the authors stopped at two sub-systems and whether a plain double latency system (as proposed in [10]) with identical throughput would do better. Note that such a double latency system would reduce energy by close to 70% which is in the same range as the reported savings using the more elaborate scheme. In any case, one can argue that the work in [55] pioneered the technique of explicitly factoring in usage

distributions.

2.5 Work on Common Case Optimization (CCO)

An important paper appeared in 1999 which once again highlighted the need to look at usage patterns [31]. The broad idea in the paper was that one should reduce the power of the most commonly occurring case by adding some Common Case (CC) specific hardware. The paper focusses extensively on how this can be done at the RTL level and hence does not focus on issues like the role of usage distributions and the degree of additional hardware one should add. However, it demonstrates how knowledge of the common case can be factored in at a very high level of design abstraction and exploited to reduce the average case power.

2.6 Conclusion: A Formal Framework for Power-Aware Design is Needed

The techniques described above have been pioneering in their recognition that energy efficiency and power-awareness are essentially identical concepts. If a system is to achieve the lowest possible energy, it must be efficient in every dissipation scenario - not just the worst case scenario. Hence, the system must dissipate only as much energy as is instantaneously needed if it is to be truly energy efficient. In other words, the more energy scalable or power-aware a system is, the more energy efficient it is.

As we pointed out at the beginning of this thesis, there is a need for a clear and unambiguous theoretical and practical framework that can capture the concept of energy efficiency just like the well known framework that exists for reducing power in digital circuits. Although the techniques above demonstrate crucial and novel scalability techniques, they stop short of generalizations that are both important and necessary to establish such a framework.

The first important concept that the previous work does not address is *quantifying* the degree of scalability or equivalently the degree of power-awareness. We have

reviewed many techniques that claim to be "energy scalable", but surely some are more scalable than the other. In making engineering decisions, it is important to have a metric that reflects on how efficiently these systems can scale. We call such a metric the "power-awareness of a system".

The second issue that is closely related to the one above is the fact that previous work does not attempt to bound the power-awareness possible which makes it difficult to judge the inherent quality of the implementation.

Another significant omission in previous work is the lack of a clear and well substantiated *procedure* that explicitly factors in energy and scenario distribution curves. Loosely speaking, a scenario corresponds to a particular operating condition the circuit or system might find itself in. An energy curve is a function that maps scenarios to the energy they consume. A scenario distribution maps a set of scenarios to their probability of occurrence. Hence, a scenario distribution is nothing but the histogram of usage patterns. One can express the total energy as the product of these two curves summed up over all scenarios. The power awareness problem is then simple one of minimizing this sum.

In summary, most of the previous work was focussed on domain-specific and strongly application dependent solutions. Although the resultant work demonstrated concepts that were occasionally broadly applicable (like scalable arithmetic) the techniques fell short of a comprehensive treatment of the power-awareness problem, a deficiency that this thesis addresses.

Chapter 3

Quantifying Power-Awareness

In this chapter we develop the basic power-awareness formalisms using a simple system - a 16x16-bit array multiplier [61] - as an example. This will allow us to elucidate the essence of our arguments without getting bogged down by detail.

3.1 Preliminaries

Consider a given system H that performs a certain set of operations F while obeying a set of constraints C. For the illustrative system, H would be the given implementation of a 16x16 bit array multiplier. While the set F would ideally contain all m-bit by n-bit multiplications, where $m, n \in [1, 16]$, we restrict F to be set of all m-bit by m-bit multiplications instead. We shall see the value of this restriction in the following discussion. Finally, the constraint might be simply one of fixed latency (i.e. H cannot take more than a given time, t, to perform F).

Given this information, we ask the following question:

The Power-Awareness Question: *How well does the energy of a system, H, scale with changing operating scenarios?*

Note that we use energy and not power in the statement above because energy allows us to seamlessly include latency constraints later on. Next, observe that our understanding of power-awareness can only be as exact as our understanding of operating "scenarios". As one might expect, these scenarios can be characterized with

arbitrarily high detail. For instance, in the case of the multiplier, we can define the scenario by the precision of the current multiplicands or the multiplicands themselves or even the current multiplicands and the *previous* multiplicands, since the power dissipation is a function of those too. In the interests of simplicity, we choose to characterize the set of scenarios S by the precision of the multiplicands. Normally, this would need a two-tuple since there are two multiplicands. But, by our choice of F only one number (the precision of the two identical bit-width multiplicands) characterizes the scenario. Hence, H can find itself in one of 16 scenarios. We denote henceforth, a scenario by s and the set of 16 scenarios by S.

Having defined scenarios, we take the first step towards characterizing the power-awareness of H by tracing its energy behavior as it moves from one scenario to the other. For a 16-bit multiplier, we would do this by executing a large number of different scenarios and measuring the energy consumed by each scenario [1]. Henceforth, we call these energy vs. scenario curves of H simply as the "energy curves" of H.

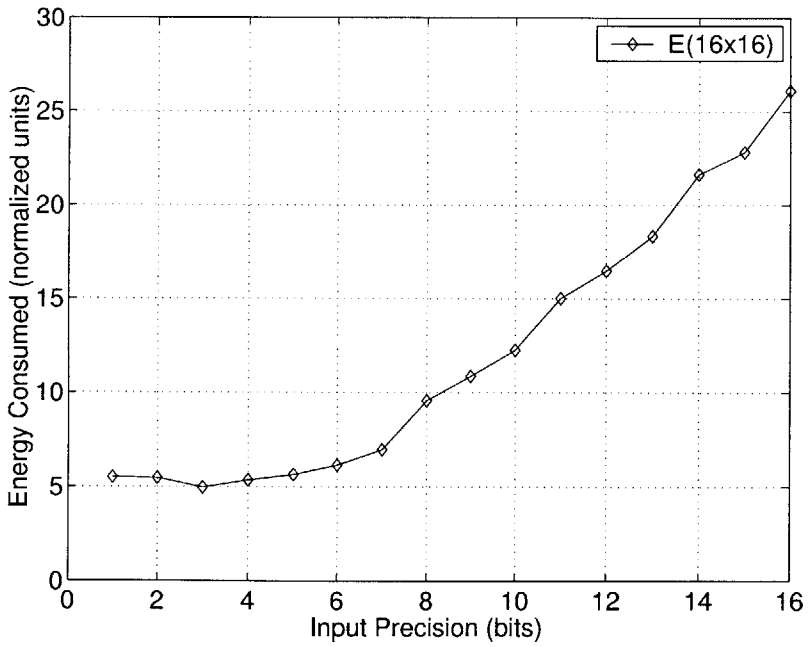


Figure 3-1: Multiplier energy as a function of input precision.

Figure 3-1 shows the energy curve of our 16x16-bit array multiplier over 16 sce-

[1] All multiplier energy curves were derived by extensive (>1000 vectors) $PowerMill^{TM}$ simulations of multiplier SPICE netlists in a 0.35μ process.

narios (which represent the precision of the multiplication). Note that the multiplier has a natural degree of power-awareness even though it was not explicitly designed for it. This is easy to understand since lower precision vectors lead to lesser switched capacitance than higher precision ones.

An energy-curve like the one in figure 3-1 is the first step to answering the power-awareness question. However, at this stage, it is difficult to answer the "how well" in the question by looking at a system by itself. Hence, instead of a single energy curve, we look at a few curves together to get a better understanding of the desirable properties of energy-curves.

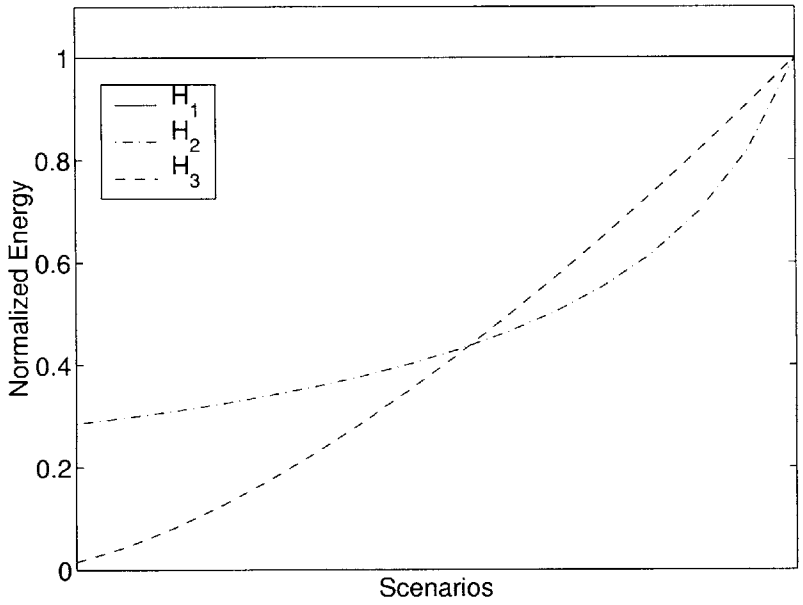


Figure 3-2: Energy curves of three different hypothetical systems.

Figure 3-2 plots the energy curves of three hypothetical systems H_1, H_2 and H_3 executing a certain, identical set of scenarios. If we had to judge the power-awareness of these systems from their energy curves, we would intuitively classify H_1 as the system that is the most *unaware* of the executing scenario. Such an undifferentiated energy curve might be expected if, for instance, these systems were implementing multiplication and H_1 was a 32-bit RISC processor (since the energy taken by the other parts would be so great that the actual precision of multiplication would have insignificant impact). H_2, on the other hand, definitely displays more energy differ-

entiation than H_1 and is intuitively "more scalable". Furthermore, since the energy of H_2 is strictly less than H_1, it seems unequivocally better. Similar arguments can be applied while comparing H_3 to H_1 and we conclude that H_3 is more scalable than H_1. However, these "intuitive" arguments break down when we try comparing H_3 with H_2. On the one hand, H_3 displays better scalability than H_2. On the other, its energy dissipation exceeds that of H_2 over a certain interval. For this reason, at this point in our development, it is unclear whether we should pick H_2 or H_3 as the more power-aware system. To help answer that question, it might help to think of the energy curve of the *most* desirable system, say $H_{perfect}$ executing the same operations under the same constraints as the three systems discussed above. In a second step, we could potentially compare the curves of H_2 and H_3 to that of $H_{perfect}$ to decide which is more power-aware. It helps to state that:

The Perfectly Power-Aware System (I) *A system $H_{perfect}$ is defined as the most power-aware system iff for every scenario in S, $H_{perfect}$ consumes only as much energy as its current scenario demands* [2].

It is clear from the above statement that we need to formally capture the concept of "only as much energy as a scenario demands". To derive this energy for a given scenario, say s_1, we consider constructing a system H_{s_1} that is designed to execute this and only this scenario. The reasoning is that we should not hope that a given system H can ever consume lesser energy in a scenario compared to H_{s_1} - a *dedicated* system which was specially designed to execute *only* that scenario. We often refer to the H_{s_i}s as "point" systems because of their focussed construction to achieve low energy for a particular scenario (or point) in the energy curve. Hence, in the context of power-awareness, the energy consumed by H_{s_1} is in a sense, the lower bound on the dissipation of H while executing scenario s_1. Generalizing this statement,

Bounds on Efficiency of Tracking Scenarios *The energy consumed by a given*

[2] *More formally, $H_{perfect}$* is the most power-aware system iff for every scenario in S, $H_{perfect}$ consumes only as much energy as demanded by its current operation $\in F$ executing in the current scenario under constraints C. *In our multiplier example, we have chosen to construct S such that it has a one-one correspondence with F and hence, it makes sense to talk about the "energy of a scenario" executing on H.*

system H while executing a scenario s_i cannot be lower than that consumed by the a dedicated system H_{s_i} constructed to execute only that scenario s_i as efficiently as possible.

This leads to our next definition of $H_{perfect}$,

The Perfectly Power-Aware System(II) *The perfect system, $H_{perfect}$, is as energy efficient as H_{s_i} while executing scenario $s_i \quad \forall \quad s_i \in S$.*

We denote the energy-curve of the perfect system by $E_{perfect}$. From a system perspective, the perfect system behaves as if it contains a collection of dedicated point systems - one for each scenario. When $H_{perfect}$ has to execute a scenario s_i, it routes the scenario to the point system H_{s_i}. After H_{s_i} is done processing, the result is routed to the common system output. This abstraction of $H_{perfect}$ as an *ensemble of point systems* is illustrated in figure 3-3.

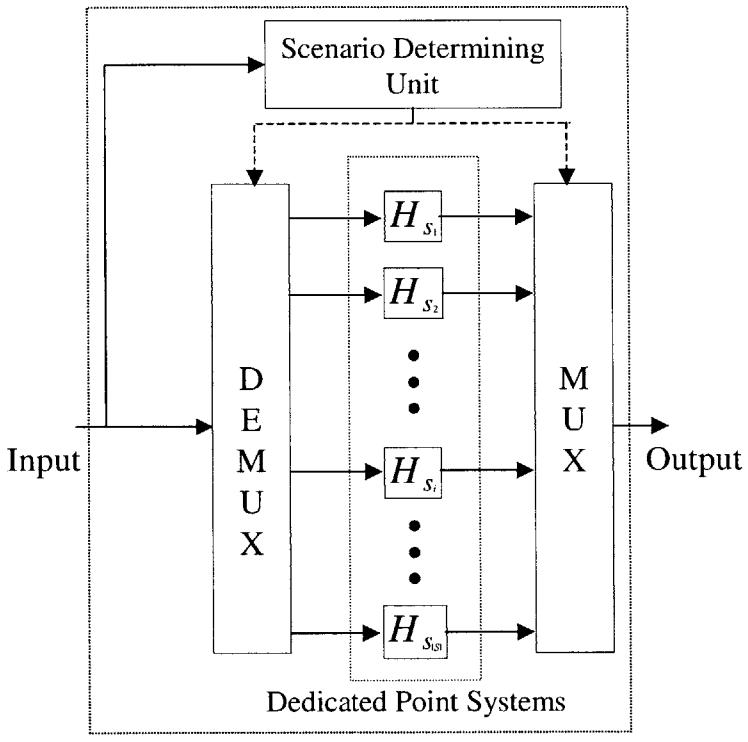


Figure 3-3: The Perfect System ($H_{perfect}$) can be viewed as an ensemble of point systems.

The task of identifying the scenario by looking at the data input is carried out by the *scenario determining* block. Once this block has identified the scenario, it

configures the *mux* and *de-mux* blocks such that data is routed to, and results routed from, the point system that corresponds to the current scenario. Note that if the energy costs of identifying the scenario, routing to and from a point system and activating the right point system are zero, then the energy consumption of $H_{perfect}$ will indeed be equal to that of H_{s_i} for every scenario s_i. Since these costs are never zero in real systems, this implies that $H_{perfect}$ is an abstraction and does not correspond to a physically realizable system. Its function is to provide a non-trivial lower bound for the energy-curve.

To construct the $E_{perfect}$ curve for our 16-bit multiplier, we emulated the ensemble of points construction outlined above. The point systems in our example were 16 dedicated point multipliers - 1x1-bit, 2x2-bit, ..., 16x16-bit - corresponding to H_{s_1} to $H_{s_{16}}$. When a pair of multiplicands with precision i came by, we diverted them to H_{s_i} (i.e. the ixi-bit multiplier). Since we are deriving $E_{perfect}$, only the energy consumed by the H_{s_i}s was taken into account. The $E_{perfect}$ curve thus derived is plotted in figure 3-4, where the energy-curve of a single 16x16 multiplier is repeated for comparison.

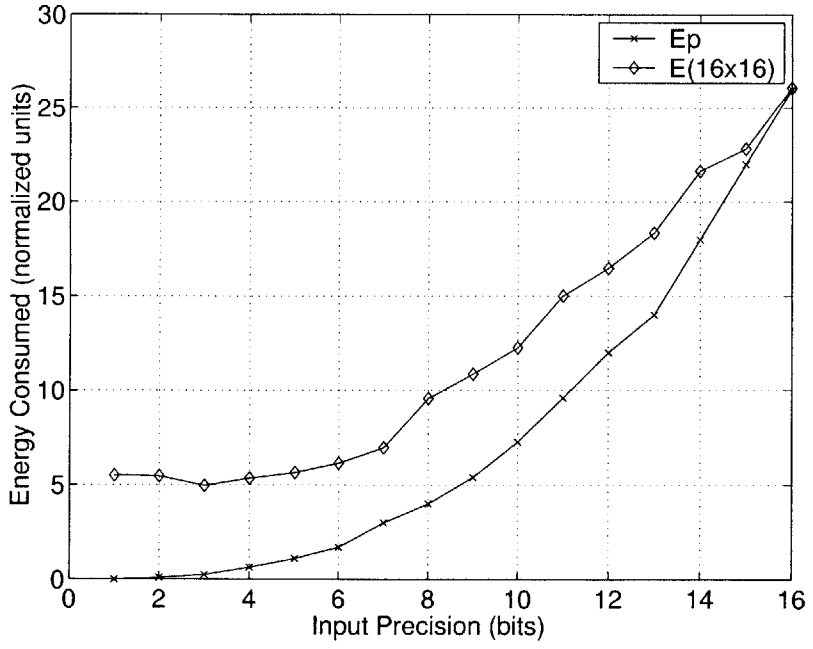


Figure 3-4: Comparing the 16x16 multiplier curve to the "perfect" curve ($E_{perfect}$) denoted by Ep in the plot.

Note that $E_{perfect}$ scales extremely well with precision since the scenarios are being executed on the best possible point systems that we could construct. Before we indulge in a more detailed comparison of the two curves, it is essential to note that the latter curve really depends on the kind of "point" systems we allow. In the case of the multiplier, we allowed any ixi−bit multiplier. The set of point systems we allow is henceforth denoted by P. This set captures the resources available to engineer a power-aware system. Like the scenario and constraint sets, it can be specified with increasing rigor and detail. This new formalism, P has two key purposes. Firstly, it gives a more fundamental basis to $E_{perfect}$. While it is not possible to talk about the "best possible energy curve", it is indeed possible to talk about the "best possible energy curve for a specified P". Secondly, P is also important when we discuss enhancing the power-awareness of H. In that context, P specifies exactly which building blocks are available to us for such an enhancement.

To quantify how power-aware our multiplier is, we plot the scenario efficiency ratio,

$$\eta_i = \frac{E(H_{perfect}, s_i)}{E(H, s_i)} \tag{3.1}$$

as i ranges over scenarios in figure 3-5. [3]

A η_i value of unity indicates that the system under consideration is as power-aware as it can be for that scenario. The smaller the η_i value, the worse the system's awareness of scenario s_i. In the case of the multiplier, note that H tracks $H_{perfect}$ fairly closely for higher precisions. This is to be expected since a 16x16 bit multiplier would be very efficient for scenarios where the operand precision is close to 16. For lower precision scenarios, H loses its ability to track as well as $H_{perfect}$ and can dissipate up to two orders more energy than $E_{perfect}$. This is a recurring theme in system design. There are energy costs to pay when a single system (H) is used over diverse operating conditions and the η curve above quantifies those costs.

[3]The notation $E(H, s_i)$ denotes the energy consumed by a system H while executing scenario s_i.

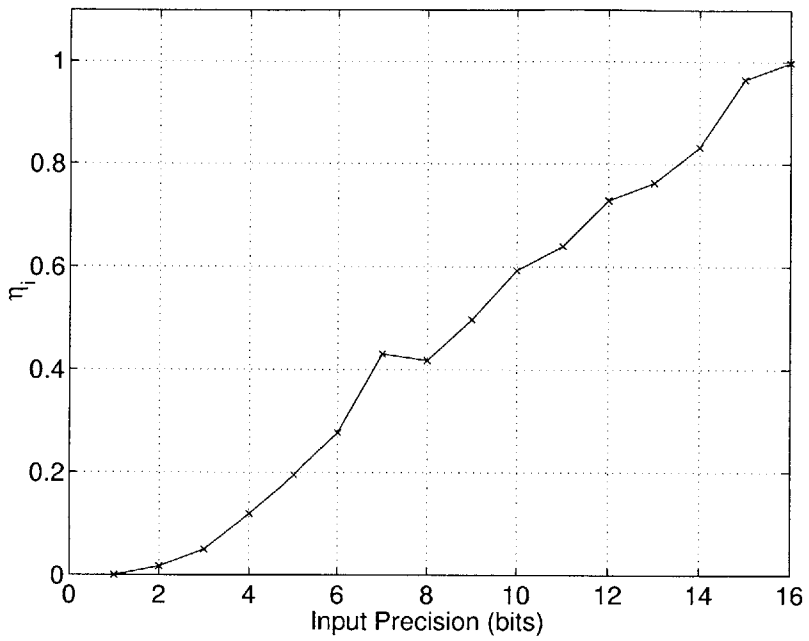


Figure 3-5: The scenario efficiency or awareness (η_i) of a 16x16 bit multiplier.

3.2 Defining Power-Awareness

In the preceding section, we quantified the power-awareness of a system on a scenario-by-scenario basis using the η function. Hence, we have partially answered the power-awareness question posed earlier. The η curve is a partial answer because it still doesn't help us resolve the other question we posed in the last section - given the energy curves of two systems, can we determine which of the two is more power-aware? It is clear that to answer this question, we need to develop a measure of power-awareness that distills the *entire* η curve. Mathematically, our problem is one of mapping a vector η to a scalar ϕ (power-awareness) by a well defined function f,

$$\phi = f(\eta)$$

Although there are infinitely many possibilities, we will describe those that have a useful system-level meaning and can be practically employed by system architects. A definition that reflects the *average-case* power-aware behavior of the system is its

expected ability to track scenario changes,

$$\phi_1 = \mathbf{E}(\eta) = \frac{\sum_{i=1}^{|S|} \eta_i}{|S|} \tag{3.2}$$

where $|S|$ is the cardinality of set S and $\mathbf{E}(.)$ is the expectation operator and should not be confused with energy. The physical interpretation of ϕ_1 is that if *all scenarios were equally likely*, H would track the scenario changes with an expected efficiency of ϕ_1. For the 16x16 multiplier, $\phi_1 = 0.501$.

Clearly, we can refine the definition of ϕ_1 to be more realistic if we had a sense of the *likelihood* that the system will reside in a particular scenario rather than just assuming all scenarios to be uniformly likely. For instance, figure 3-6 charts out the probability that a multiplier will be in a certain precision scenario when it is filtering a typical speech signal [56].

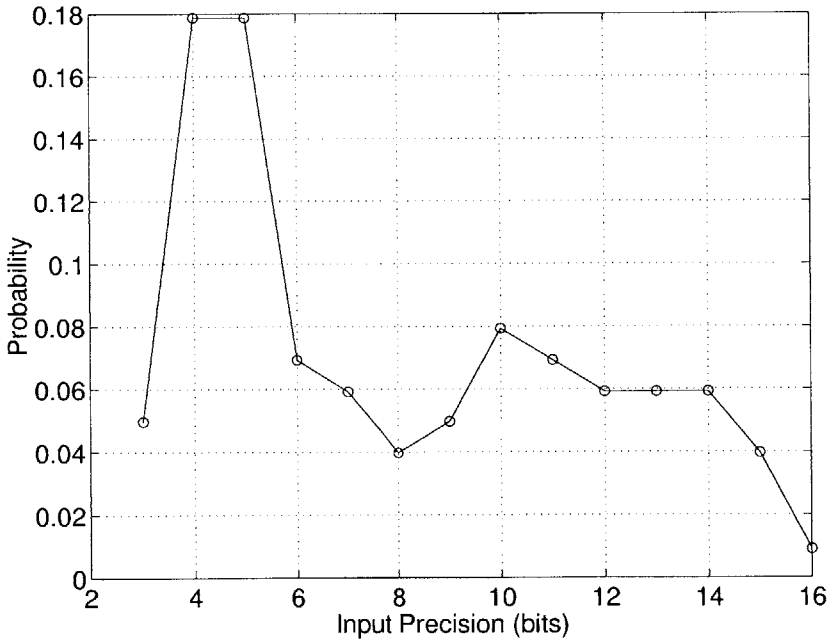


Figure 3-6: Typical multiplier usage pattern in speech filtering applications.

We call the curve in figure 3-6 a *scenario-distribution* curve (henceforth, we use d to denote scenario-distributions and d_i to denote the probability of occurrence of scenario s_i). We can now factor in d to arrive at a more reasonable value of the

expected power-awareness of H:

$$\phi_d = \mathbf{E}(\eta) = \frac{\sum_{i=1}^{|S|} \eta_i d_i}{\sum_{i=1}^{|S|} d_i} \tag{3.3}$$

For the case of the multiplier for the distribution d_{speech}, this turns out to be close to 0.42. Hence, if we were to observe the multiplier executing the speech filtering application at a randomly picked point in time, we would expect to see it dissipating about 140% more energy for a scenario compared to $H_{perfect}$.

While a scenario's frequency of occurrence is a fair indicator of its importance, its not the only one. For instance, a scenario might have a low probability of occurrence, but when it *does occur*, the architect *might want* the system to track the change well. If we plug in this importance (m), we arrive at a more generalized version of (3.3),

$$\phi_{d-m} = \frac{\sum_{i=1}^{|S|} \eta_i d_i m_i}{\sum_{i=1}^{|S|} d_i m_i} \tag{3.4}$$

A very useful application of power-awareness as defined by (3.4) is in predicting and enhancing battery lifetime of the system H. In the context of maximizing battery lifetime, the importance, m_i, of a scenario is simply the energy dissipated by that scenario,

$$m_i = E(H, s_i)$$

Plugging in this definition of importance into (3.4) and simplifying using (3.1), we get,

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

$$= \mathbf{E}(\text{Normalized Battery Lifetime}) \tag{3.5}$$

The interpretation above is important enough that we do not attach any sub-script and consider it *the* default definition of power-awareness unless specified otherwise. It is one of the most useful interpretations of power-awareness since it directly equates

the metric to the *expected battery lifetime of the system normalized to the lifetime of the perfect system.* To see why this is so, note that the denominator is a summation of the expected energy consumption per scenario and hence equal to the expected energy consumed by the system H displaying a scenario distribution d. Similarly, the numerator is the expected energy dissipation of the perfect system. Since battery lifetime is inversely proportional to the energy consumed, ϕ as defined by (3.5) represents the normalized battery lifetime of the system H. For our 16-bit multiplier, it turns out that $\phi = 0.57$, which implies that in a speech filtering application, this multiplier will have a battery lifetime that is about half of the perfectly power-aware system. Note that in equating ϕ to the expected normalized lifetime of the system, we ignore second-order effects like the dependence of the battery capacity on the discharge pattern [35].

Coming back to our original motivation - resolving which of H_2 or H_3 is more power-aware - we see that the question cannot be answered in the battery lifetime sense without specifying a scenario distribution. We can unambiguously answer which of H_2 or H_3 is more efficient *for any specified* d. Interestingly, if we lack scenario distribution information, and assume that all *scenario distributions* are equally likely, this statistical assumption is equivalent to assuming that all scenarios are equally likely. In this case d reduces to a uniform distribution.

Chapter 4

Enhancing Power Awareness

4.1 Motivation

Enhancing the power-awareness of a system is composed of two well defined steps:

1. Engineering the best possible point systems.
2. Engineering the desired system using the point systems constructed in step (1) such that power-awareness is maximized.

In the context of a power-aware multiplier, the first task is understood easily. It involves engineering 1x1, 2x2, ..., 16x16-bit multipliers that are as efficient as possible while performing 1x1, 2x2, ..., 16x16-bit multiplications respectively. The second task - that of engineering a system *using point systems* - is illustrated by the multiplier shown in figure 4-1.

Note the overall similarity between this figure and the abstraction of $H_{perfect}$ in figure 3-3. The ensemble of point systems was used as an abstract concept in the context of explaining $H_{perfect}$'s energy curve. In the present context, however, we are illustrating an *actual physical realization* of a system based on this concept. The basic idea is to detect the precision of the incoming operands using a zero detection circuit and then route them to the most suitable point system. In the case of $H_{perfect}$, the matching is done trivially - multiplier operands which need a minimum precision of i-bits are directed to a ixi-bit multiplier. Similarly, the output of the chosen multiplier is

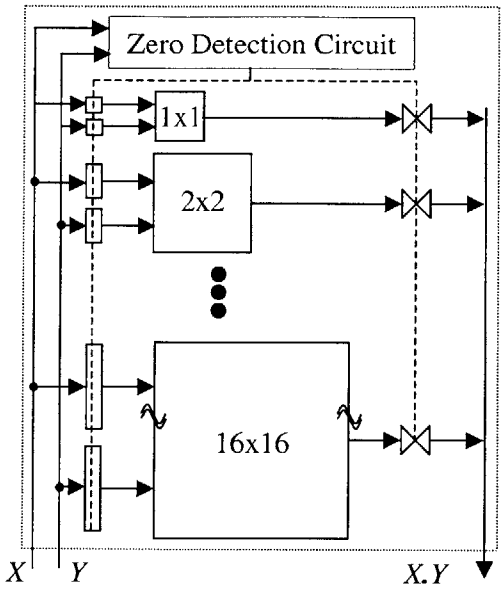


Figure 4-1: The $\hat{H}_{perfect}$ system mimics the abstract $H_{perfect}$ system by using an ensemble of 16 dedicated point multipliers and a zero-detection circuit as the scenario-detector.

multiplexed to the system output. As we might expect though, $\hat{H}_{perfect}$, has significant overheads. Even if we were to ignore the area cost of having 16 point multipliers, and focus solely on the power-awareness, the energy curve of $\hat{H}_{perfect}$ wouldn't be the same as $E_{perfect}$. This is because, while the scenario execution itself is the best possible, the energy costs of determining the scenario (the zero detection circuit), routing the multiplicands to the right point system and routing the result to the system output (the output mux) can be non-trivial.

A system that uses a less aggressive ensemble in an effort to reduce the energy overhead of assembling point systems is shown in figure 4-2.

The basic operation of this multiplier ensemble is the same. The precision requirement of the incoming multiplicand pair is determined by the zero detection circuitry. Unlike the previous 16-point ensemble, this 4-point ensemble is not complete and hence mapping scenarios to point systems is not one-one. Rather, precision requirements of[1]:

[1]Note that this is just one of the many possible mappings.

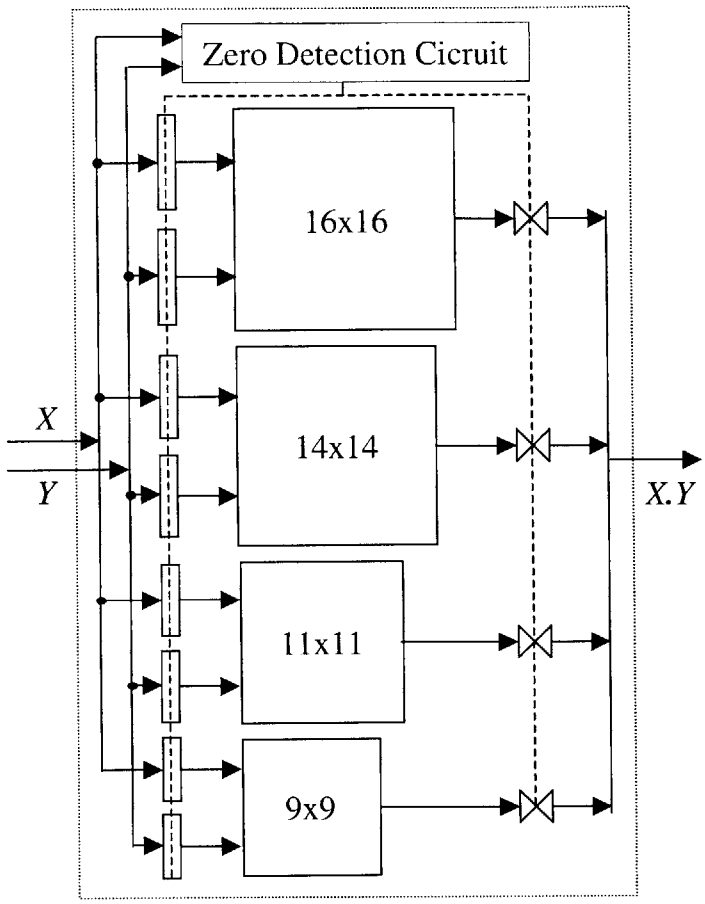


Figure 4-2: The 4-point ensemble multiplier system.

1. ≤ 9 bits are routed to the 9-point multiplier,

2. 10,11 bits are routed to the 11-point multiplier,

3. 12-14 bits are routed to the 14-point multiplier,

4. 15,16 bits are routed to the 16-point multiplier,

Similarly, the results are routed back from the activated multiplier to the system output. While scenarios are no longer executed on the best possible point systems (with the exception of 16, 14, 11 and 9 bit multiplications), this ensemble has the advantage that energy overheads of routing are significantly reduced over $\hat{H}_{perfect}$. Also, while the scenario to point system mapping of the 4-point ensemble is not as simple as the one-one mapping, it is important to realize two things. Firstly, the

energy dissipated by the extra gates needed for the slightly more involved mapping in the 4-point ensemble is low relative to that dissipated in the actual multiplication. Secondly, only 4 systems have to be *informed* of the mapping decision compared to 16 earlier. This reduction further offsets the slight increase in scenario mapping. The energy curve of the 4-point ensemble is plotted in figure 4-3 where it is compared to $E_{perfect}$ and the energy curve of a single 16x16 multiplier.

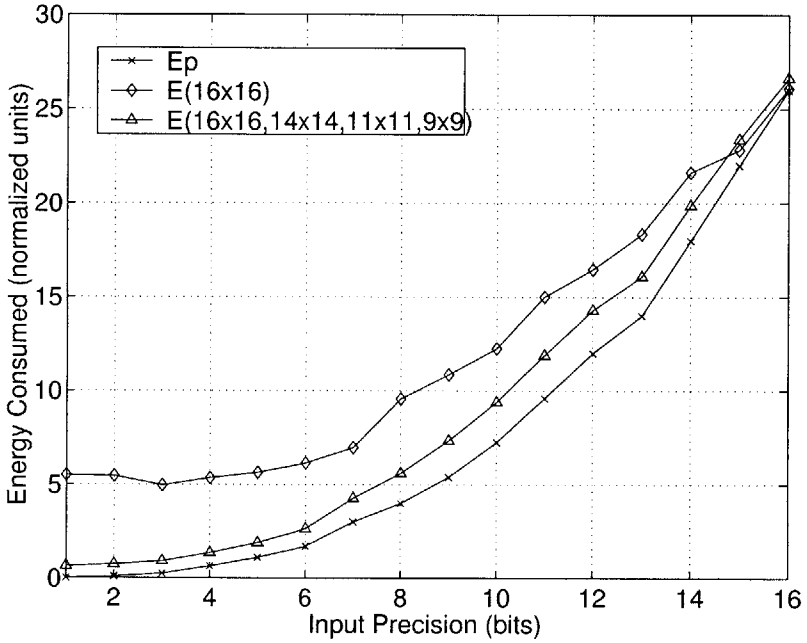


Figure 4-3: Energy curve of the 4-point multiplier system in figure 4-2 compared to the "perfect" curve and the conventional 16x16 multiplier curve.

It is not difficult to see the basic trade-off at work here. Increasing the number of point systems decreases the energy needed for the scenario execution itself but increases the energy needed to co-ordinate these point systems. Hence, it is intuitively reasonable to assume the existence of an *optimal* ensemble of point systems which strikes the right balance. Motivated by this possibility, we can now pose the problem of *enhancing* power-awareness thus:

Determining the Most Power-Aware System Practically Realizable (I) *Can we construct a system $H_{optimal}$ as an ensemble of point systems drawn from P such that $H_{optimal}$ is unconditionally more power-aware than any other such constructed system?*

Invoking property (9) discussed earlier, unconditional power-awareness only leads to partial ordering. Hence, the existence of a unique $H_{optimal}$ as defined above cannot be guaranteed. In other words, while it is possible to present a *set* of solutions that are unconditionally more power-aware than all other solutions, we cannot guarantee that this set will have only one member. In fact, this last condition is highly unlikely to occur in practice - unless routing costs are very low or very high compared to scenario execution costs (in which cases the optimal ensembles would be the complete and single-point solutions respectively). Hence, in general, it is futile to search for an "optimal" ensemble of point systems that is unconditionally better than all other ensembles. Thus, we set our ambitions lower and ask a slightly different question:

Determining the Most Power-Aware System Practically Realizable (II) *Can we construct a system $H_{optimal}$ as an ensemble of a point systems drawn from P such that $H_{optimal}$ is more power-aware than any other such constructed system for a specified scenario d_{given}?*

Since a specified scenario distribution d_{given} imposes a total ordering on the power-awareness of all possible subsets of P, it is easy to prove the *existence* of an optimal system. Note that the proof based on total ordering is non-constructive i.e. *it only tells us that $H_{optimal}$ exists but doesn't help us determine what it is.* This is unfortunate because a brute-force search of the optimal subset of P would require an exponential number of operations in $|P|$ - a strategy that takes unacceptably long even for the modestly large P.

To see if there are algorithms that can find $H_{optimal}$ in non-exponential run-times we pose the problem more formally as follows.

4.2 Formal Statement of the Power Awareness Enhancement Problem

Given:

1. F : A system function to be realized.

2. S : A set of scenarios characterized by a *scenario basis*. For example, the basis in our multiplier example was the precision of the multiplicands.

3. P : A set of point systems available to realize F. Also, we denote the power-set of P i.e. the set containing all the sub-sets of P by $\mathcal{P}(P)$.

4. d : The scenario distribution,

$$d : S \rightarrow \mathcal{R}^+ \cup \{0\}$$

that obeys the additional constraint,

$$\sum_{s_i \in S} d(s_i) = 1$$

expected of a distribution functions.

5. e : The energy function,

$$e : S \times P \rightarrow \mathcal{R}^+ \cup \{\infty\}$$

In other words, for any given pair (s_i, p_i), e gives us the energy consumed when scenario s_i is executed on point system p_i. For instance, the energy taken by a 4x4-bit multiplication is different on a 4-bit multiplier than, say, a 9-bit multiplier. If the scenario s_i cannot be executed on the point system p_i, an infinite cost is assigned to the pair [2].

6. w : The *energy overhead* cost function,

$$w : \mathcal{P}(P) \rightarrow \mathcal{R}^+$$

Hence, w maps every sub-set of P to the sum of all energy spent in co-ordinating

[2]This has the nice property of preventing any *infeasible* ensemble of point systems. For instance the ensemble {14-bit, 11-bit, 9-bit multiplier} is a subset of P, but is an infeasible solution since 15- and 16- bit multiplications cannot be executed on any of these point systems.

the points in the sub-set ensemble (routing energy, determining the scenario, mapping the scenario etc.).

Form of the Solution:

1. An ensemble of point-systems, $h \in \mathcal{P}(P)$, and,

2. A corresponding *mapping*,

$$g : S \rightarrow h$$

i.e. g maps each scenario to a point system in h. For instance, in the 4-point multiplier example above, g would specify that scenarios 1-9 execute on the 9-bit multiplier, 10 and 11 execute on the 11-bit multiplier and so on.

Measure of the Solution:

Since we are interested in the expected battery lifetime of a system, the measure of a proposed solution - h, g - is the expected energy consumption $E(h)$ given by,

$$E(h) = w(h) + \sum_{i=1}^{|S|} e(s_i, g(s_i))d_i \tag{4.1}$$

Note that like all models, the one for energy above can be made increasingly more precise. For instance, the interconnect energy will display some dependence on scenario distributions. Hence, the w function can take d as an argument and so on. However, we refrain from these refinements because our intent here is to use a realistic but simple model to analyze the complexity of finding a solution.

Problem:

Determine a solution that minimizes the measure.

It seems very likely that the problem of finding $H_{optimal}$ as stated above belongs to the class of NP-complete problems. In other words, we cannot hope to determine the construction of $H_{optimal}$ in polynomial time [16]. The proof of NP-completeness and suitable approximation algorithms to find $H_{optimal}$ are beyond the scope of this thesis. At this point, it suffices to say that we are currently working with heuristics

to determine $H_{optimal}$ and as the application examples in the next chapter show, these heuristics yield good results. For example, in the case of the multiplier system and the speech distribution d_{speech}, the 4-point system described above was constructed using a greedy incremental algorithm and achieved a power-awareness of close to 0.9 compared to about 0.57 for the single point 16x16-bit multiplier. Finally, it is important to note that the re-engineered system must not violate any constraints that the original system was expected to obey unless the constraints are relaxed explicitly for the sake of increasing power-awareness.

4.3 Reducing Area Costs Incurred in Enhancing Power-Awareness

Our focus in the preceding discussion was maximizing power-awareness without regard to implementation costs like area. While such an approach is acceptable for systems where power-awareness must be increased at all costs, it might need to be reformulated for those with area constraints. In these latter cases, the problem would be to find the most power-aware ensemble for a specified distribution *while still obeying specified area constraints.* If the area costs are significant enough, it is often beneficial to think of implementing an ensemble *temporally* rather than *spatially.* For example, instead of a spatial layout of 4 multipliers as illustrated earlier, we must imagine a temporal layout of these 4 multipliers. In other words, the same physical hardware is reconfigured to a 16, 14, 11 or 9-bit multiplier as desired. A possible solution is to selectively shut off the parts of a 16-bit multiplier and make it behave like smaller multipliers. While such a solution may or may not save any energy in the case of multipliers (due to the overhead of latches and the latch control network), *it is an important illustration of the fact that spatial mappings aren't the only means to implement ensembles.* In fact, our discussion of power-aware processors in the next chapter is a real world example of a system where a *purely temporal* ensembles increase power-awareness significantly.

If we reformulated the fitness measure of an ensemble to include its silicon real

estate costs, we can expect that the optimal ensemble might neither be totally temporal nor totally spatial, but a hybrid. Continuing our multiplier example, it might mean that we end up with, say 3, point multipliers, one or more of which are reconfigurable to differing extents. To find such an optimal, possibly hybrid, solution we must extend the spatial formulation of the problem (as stated in the last section) in two ways. Firstly, we must allow new point systems that correspond to temporally reconfigurable ensembles. In the multiplier example, this means including point systems like a nxn-bit multiplier that can be explicitly reconfigured as more efficient r_0xr_0, r_1xr_1, ..., r_kxr_k-bit multiplier where $r_i \in [1, n]$. Secondly, we must factor in the energy costs of *temporal reconfiguration*. In simple models, these costs could be factored into the scenario execution energy itself. Hence, the e function that maps (s_i, p_i) pairs to energy values would not only include the cost of executing scenario s_i on point system p_i, but also the expected energy cost of possibly reconfiguring p_i to execute scenario s_i.

Finally, it is worth noting that although we motivated temporal and hybrid ensembles to reduce area costs, such ensembles might in fact *outperform* purely spatial ones in power-awareness even if we allow unlimited area for both. In other words, one should not expect area saving temporal ensembles to be always inferior than the best possible, area-unconstrained spatial ensemble. With some thought, this should not be surprising because in moving from spatial to temporal ensembles, we *augment* our set of point systems allowing temporal ensembles a larger solution space to pick from.

Chapter 5

Practical Illustrations

It is amply clear from the previous chapter that enhancing power-awareness by constructing ensembles of point systems carefully chosen from P is a general technique that can be used not just for multipliers but other systems as well. In this chapter, we shall illustrate how this ensemble idea can be applied to enhance the power-awareness of multi-ported register files, digital filters and a dynamic voltage scaled processor. In each case, we express the problem in terms of the framework we have developed above and characterize the power-awareness of the system. Then we use an ensemble construction to enhance power-awareness. It is interesting to note that these applications cover not just spatial ensembles, but purely temporal (processor example) and spatial-temporal hybrid ensembles (register files and adaptive digital filters) as well.

5.1 Power-Aware Register Files

5.1.1 Motivation

Architecture and VLSI technology trends point in the direction of increasing energy budgets for register files [65]. The key to enhancing the power-awareness of register files is the observation that over a typical window of operation, a microprocessor accesses a small group of registers repeatedly, rather than the entire register file. This locality of access is demonstrated by the 20 benchmarks comprising the SPEC92 suite

that were run on a MIPS R3000 (fig. 5-1)[1]. More than 75% of the time, no more than 16 registers were accessed by the processor in a 60-instruction window. Equally importantly, there was strong locality from window to window. More than 85% of the time, less than 5 registers changed from window to window.

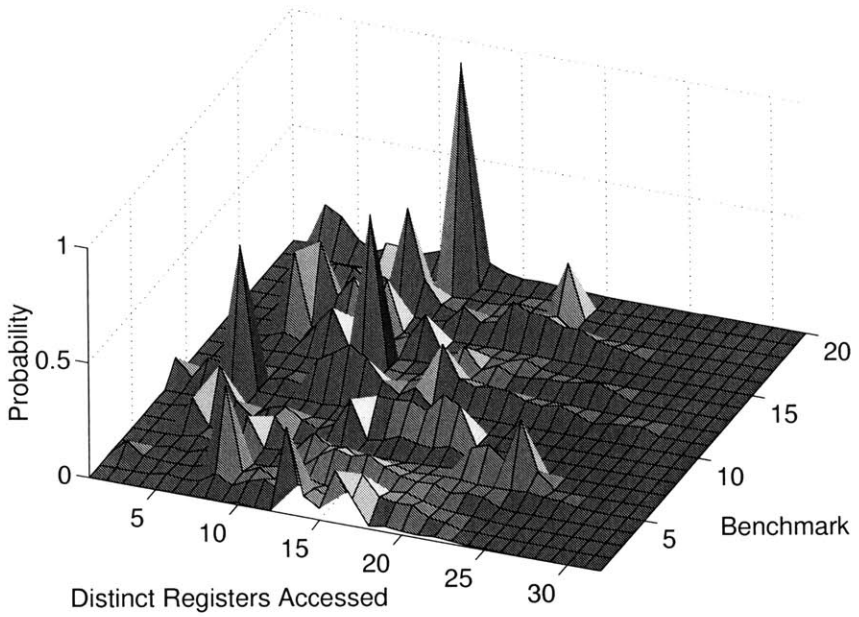


Figure 5-1: Distribution showing number of distinct registers accessed in a 60-instruction long window for the MIPS R3000 processor executing 20 benchmarks.

If we think of the number of registers the processor typically needs over a certain instruction window as a scenario, the curves in figure 5-1 are simply scenario distributions. When a processor uses n registers over a window, we would want the file to behave as if it were a n-register (i.e. n-word) file. This would lead to a register file architecture which is significantly more power-aware than one where the files always behaves as a, say, 32-register file. The reason for this of course is that smaller files have lower costs of access because the switched bit-line capacitance is lower. Hence, from a power-awareness perspective, over any instruction window, we want to use a file that is as small as possible.

[1]The author wishes to acknowledge Rex Min for gathering these statistics and also for writing the auto-generator used to produce layout for the custom register files.

5.1.2 Modeling the problem

We model the problem of increasing the power-awareness of register files using the terminology developed in section III:

1. Function to be realized (F): A 2^n-word x m-bit register file with r read ports and t write ports.

2. Set of scenarios (S): We use the number of registers accessed in an instruction window of length k to characterize scenarios. In picking k, one must remember that the longer the window, the larger the number of accessed registers, leading to lesser differentiation. A smaller window needs frequent scenario to point-system mapping changes which has energy costs too. We choose k=60.

3. Point Systems Available (P): We assume the availability of 1, 2, 4, ... 2^n word x m-bit register files with r read ports and t write ports. Hence, the number of words is the only degree of freedom allowed. While it is possible to have more exotic point systems (different read and write ports, bit-widths etc.), our choice is reasonable and works well when practically implemented.

4. Scenario Distributions (d): The twenty register access profiles in figure 5-1 are the scenario distributions.

5. Energy function (e) and overhead energy (w): All register file results were obtained by generating layouts using a custom-written program, extracting the layouts into SPICE netlists, and simulating the netlists in $PowerMill^{TM}$ with test vectors. The register files themselves were implemented using NAND-style row decoding in dynamic logic with precharged address decoding lines, and use a standard cross-coupled inverter pair for static storage. The file that we use to illustrate power-aware engineering is a 32x4 bit, 3 read, 2 write port file. We chose $m = 4$ although, as long as m is not unreasonably large, it does not affect the results in any material way. This is because the bit-line switched cap is essentially independent of m.

5.1.3 Results

A monolithic 32-word file has an awareness varying between 0.2 and 0.3 for the different distributions. Using a (16,8,4,4) ensemble as shown in figure 5-2 we increase awareness to between 0.5 and 0.8 for the different distributions.

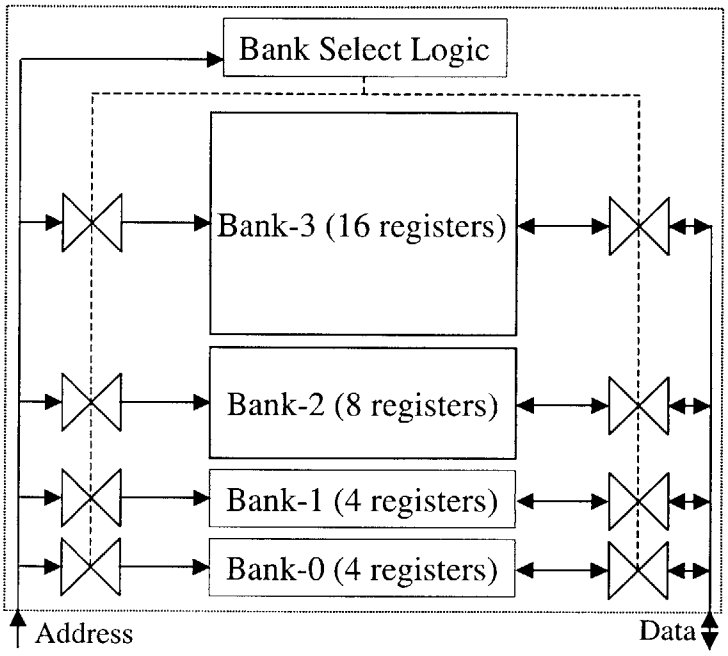


Figure 5-2: This (16,8,4,4) ensemble of 4 register files increases power-awareness by twice for the d_1 scenario distribution and about 2.5 times for the d_2 distribution.

The energy curves of the single point solution (a 32-word file) and the 4-point (16,8,4,4) ensemble are plotted in figure 5-3. Interpreted in terms of lifetime increase, the non-uniform 4-point ensemble increases lifetime by between 2 and 2.5 times for the twenty distributions used.

5.2 Power Aware Filters

5.2.1 Motivation

There are significant motivations for investigating power-aware filters. As an example, consider the adaptive equalization filters that are ubiquitous in communications

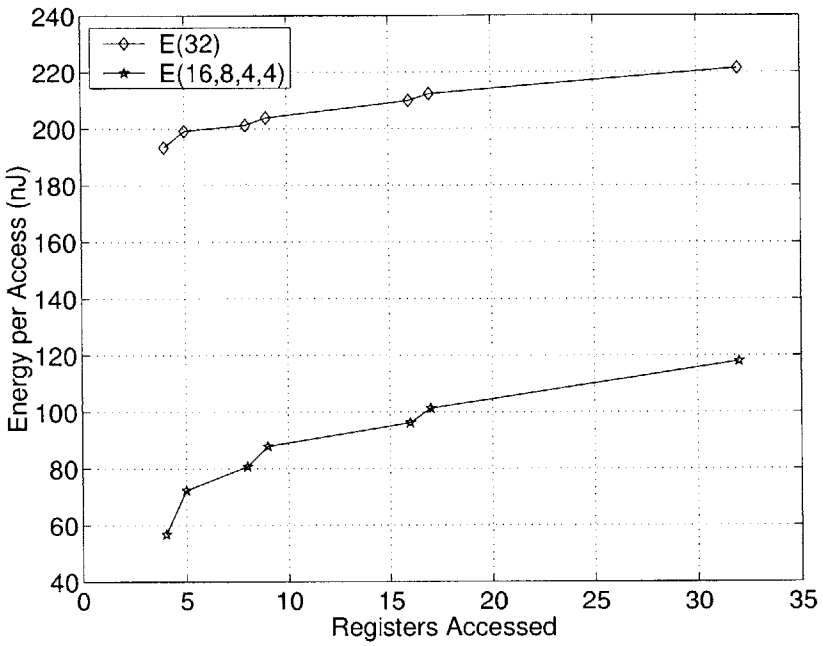


Figure 5-3: Energy curves of a monolithic 32-word file and the non-uniform 4-point ensemble (16,8,4,4).

ASICs. The filtering quality requirements depend strongly on the channel conditions (line lengths, noise and interference), the state of the system (training, continuous adaptation, freeze etc.), the standard dictated specifications and the quality of service (QoS) desired. All these considerations lead to tremendous scenario diversity which a power-aware filtering system can exploit [42].

5.2.2 Modeling the problem

1. Function to be realized (F):

$$y[n] = \sum_{k=1}^{\text{Number of Taps}} h[k]x[n-k]$$

   We have chosen a 64-tap, 24-bit filter.

2. Set of scenarios (S): We use the basis <*Number of taps, Precision*> to characterize the operational state that the system is in. The precision refers to both the data and coefficients.

3. Point Systems Available (P): We assume the availability of all possible < *Number of taps, Precision*> filters. We pick distributed arithmetic (DA) filters as described in [3] because they allow the energy to scale with both taps and desired precision. A 4-tap DA filter is shown in figure 5-4. In each step, incoming and delayed data bits with the same weights are used to access a memory which has pre-computed combinations of coefficients $h[n]$. This precomputed value is then either added to or subtracted from a partially accumulated sum $y[n]$. The number of cycles needed is the same as the precision of the multiplicands. Thus filters that have to manage precisions lower than the maximum can scale their voltages lower and still meet deadlines. Hence, this filter architecture allows extremely fine-grained control over energy dissipation and is highly aware. The problem is that since it relies on lookups, it has to resort to partitioning and hybrid schemes to remain feasible as the number of taps grows [3].

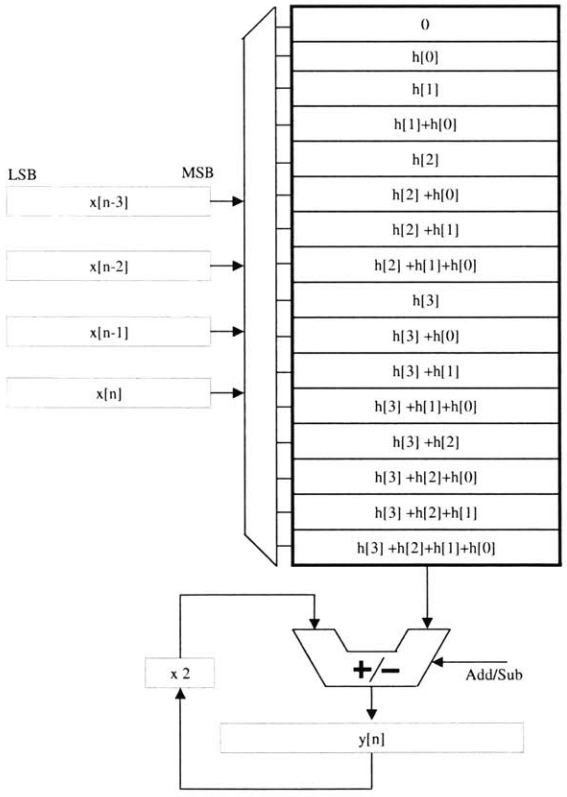


Figure 5-4: A 4-tap distributed arithmetic (DA) filter architecture.

4. Scenario Distributions (d): We model the desired filtering quality using a syn-

thetic distribution centered around a <16-taps,8-bit> scenario. Such a distribution will prevail, for instance, when the system is in the freeze mode with a high line quality and/or low SNR requirements.

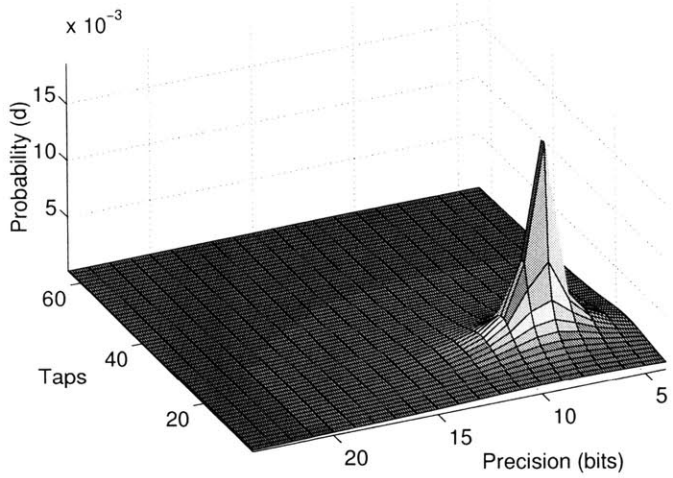


Figure 5-5: Probability distribution of anticipated adaptive filter quality needed when the system is in "freeze" mode with good line conditions and/or low SNR requirements.

5. Energy function (e) and overhead energy costs (w): We parametrically model the filters described since the nature of the DA architecture lends itself to reasonably accurate energy model [55]. The energy curve that results from this model is shown in figure 5-6. Note that while energy scales about linearly with the number of taps, it scales in a quadratic manner with precision. This is because of the fact that lower precision filters can scale their voltage.

5.2.3 Results

Before we illustrate suitable ensemble constructions that enhance power-awareness, it is instructive to look at the energy characteristics of the perfect system. Figure 5-7 plots the product of scenario energy and scenario probability for the perfect system (which would be an ensemble of 1536 (=(64)(24)) point systems). The scenario energy-probability product curve shows the energy consumed as a function of scenarios. Note that although energy consumption around the (16-tap,8-bit) scenario is

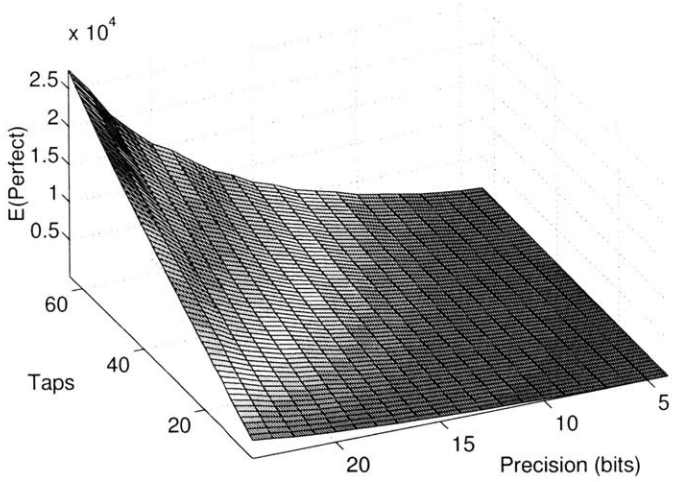


Figure 5-6: The "perfect" energy curve for 64-tap, 24-bit DA-based filtering.

clearly prominent in figure 5-7, some high-precision, high-tap scenarios also account for significant contributions to the overall energy consumed. This is easily understood because although they occur infrequently (as seen in the distribution plot in figure 5-5), they consume significant energy when they do occur (as seen in the energy plot in figure 5-6). If we used a single, 64-tap, 16-bit filter (i.e. a one point ensemble),

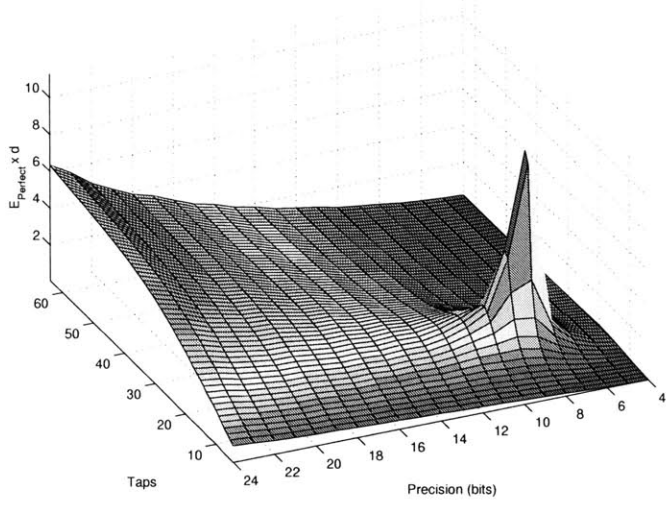


Figure 5-7: The perfect *energy-probability* curve for 64-tap, 24-bit DA-based filtering.

the resultant energy-probability product curve turns out to be the one plotted in figure 5-8. A rough comparison of the energies consumed by different scenarios in this system to that in the perfect system shows that the former is significantly non-

optimal. In fact, the power-awareness of the single point system is only 0.17. To find

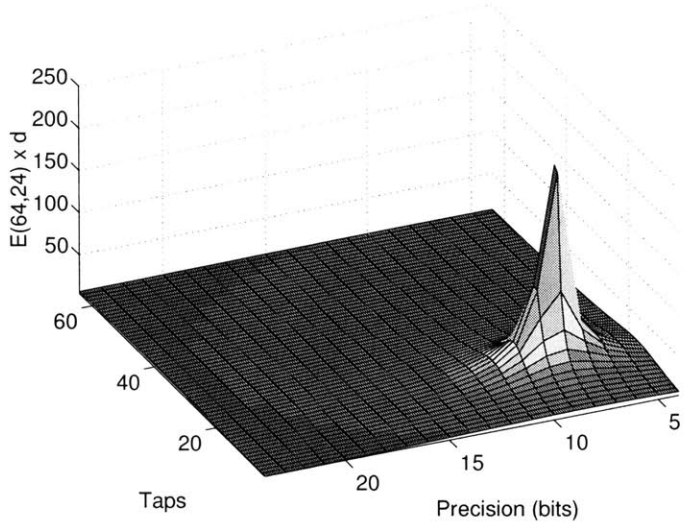


Figure 5-8: The energy-probability curve for a single 64-tap, 24-bit DA-based filter.

more optimal ensembles, we programmed a brute-force exhaustive search algorithm that could find the best 4-point ensemble. Due to exponential timing requirements, it broke down after that and a greedy heuristic took over. The optimal 4-point ensemble turns out to be $((64,24),(64,15),(58,20),(51,10))$ [2] as shown in figure 5-9. Its energy-probability curve is plotted in figure 5-10. Note that although not quite

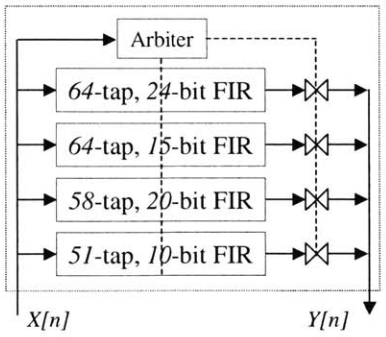


Figure 5-9: This 4-point ensemble of DA filters improves power-awareness by over three times over a single point system.

optimal, it has a power-awareness of 0.52, which is over three times better than the single point ensemble. Interestingly, our greedy heuristic revealed that if we include 4

[2](64,24) stands for 64-tap, 24-bit precision etc.

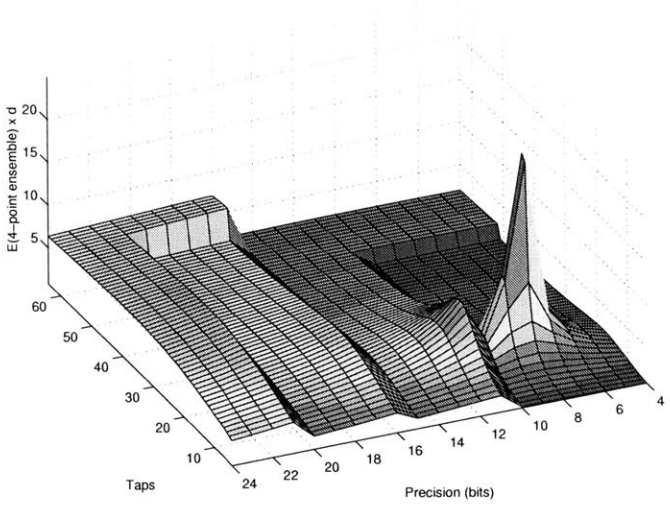


Figure 5-10: The energy-probability curve for the 4-point ensemble in figure 5-9. Note the similarity to the "perfect" curve in figure 5-7.

more points - (30,17),(43,23),(64,7),(43,13) - in the above ensemble, we can increase the power awareness to 0.64.

5.3 Power-Aware Processors

5.3.1 Motivation

Having looked at three examples of power-aware sub-systems (multipliers, register files and digital filters), we illustrate power-awareness at the next level of the system hierarchy - a power-aware processor that scales its energy with workload. Unlike previous examples, however, this one illustrates how an ensemble can be realized in a *purely temporal* rather than a spatial manner.

It is well known that processor workloads can vary significantly and it is highly desirable for the processor to scale its energy with the workload. A powerful technique that allows such power-awareness is dynamic frequency and voltage scaling [22]. The basic idea is to reduce energy in non-worst-case workloads by extending them to use all available time, rather than simply computing everything at the maximum clock speed and then going into an idle or sleep state. This is because using all available time allows one to lower the frequency of the processor which in turn allows scaling

down the voltage leading to significant energy savings [22, 39, 8].

In terms of the power-awareness framework that we have developed, a scenario would be characterized by the workload. The point systems would be processors designed to manage a specific workload. As the workload changes, we would ideally want the processor designed for the instantaneous workload to execute it. It is clear that implementing such an ensemble spatially is meaningless and must be done temporally using a dynamic voltage scaling system. Before we look at such a system, we state the problem more concisely.

5.3.2 Modeling the problem

1. Function to be realized (F): Any workload running on a given processor. In this case, the processor we use is the Intel StrongArm SA-1100. The workload variation comes from a variable tap filter running on the SA-1100 (the reader is referred to [39] for details of the actual setup).

2. Set of scenarios (S): We use the workload $\in [0, 1]$ as a basis (with 0 for no workload to 1 for a completely utilized processor). Note that the workload requirement has a one-one mapping to a frequency and voltage requirement.

3. Point Systems Available (P): A point system in this case would refer to the SA-1100 designed for a specific workload. Since we are interested in achieving power awareness through voltage scaling, this corresponds to a SA-1100 with a dedicated voltage and frequency (which are the minimum possible to achieve the workload). Also, due to an infinite number of scenarios, there are infinite number of point systems - one for every workload between 0 and 1. Equivalently, in terms of voltages, there are an infinite number of point systems between 0 and V_{dd}^{max}, the latter being the highest voltage the SA-1100 can run at, which also corresponds to its highest frequency and a workload of unity.

4. Scenario distribution (d): We assume, for simplicity, that all workloads are equally probable. As we see below, such an assumption is pessimistic and in

real applications, we can expect to see even better numbers for power-awareness.

5. Energy function (e) and energy overhead (w): The energy dissipated by the SA-1100 was physically measured.

5.3.3 Results

We now analyze an *actually constructed system* that recently demonstrated this power-awareness concept [39]. The overall setup is summarized in figure 5-11 adapted from [39]. The basic idea is that a power-aware operating system ($\mu - OS$) running on the SA-1100 determines the current workload, scales the frequency accordingly and then instructs a switched regulator supply to scale the voltage accordingly. The reader is referred to [39] for the details of the setup and the dynamic voltage circuitry etc.

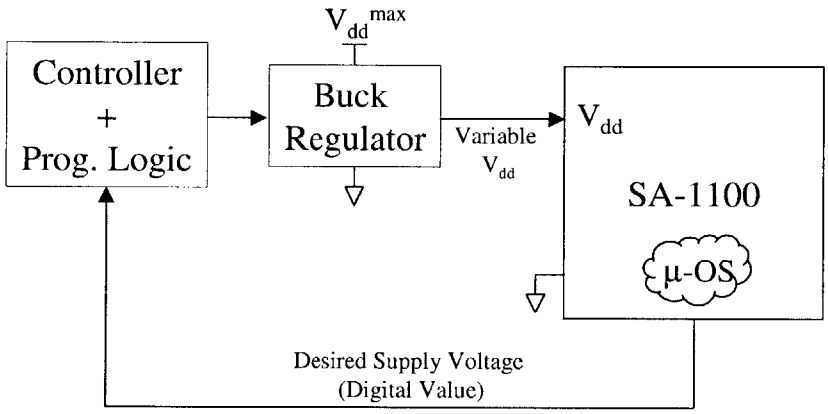


Figure 5-11: The dynamic voltage scaling (DVS) system used to enhance power-awareness.

The DVS system uses a *temporal ensemble* of 32 point systems with voltage levels uniformly distributed between 0 and V_{dd}^{max}. The energy-curves of a non-aware i.e. fixed voltage system and the implemented dynamic voltage system are plotted in figure 5-12 .

For uniform workload distributions, power-awareness improves from 0.63 for a fixed voltage system to 1.0 for the implemented dynamic voltage system. Note that although the 32-point ensemble is by no means perfect, it was chosen as a reference to

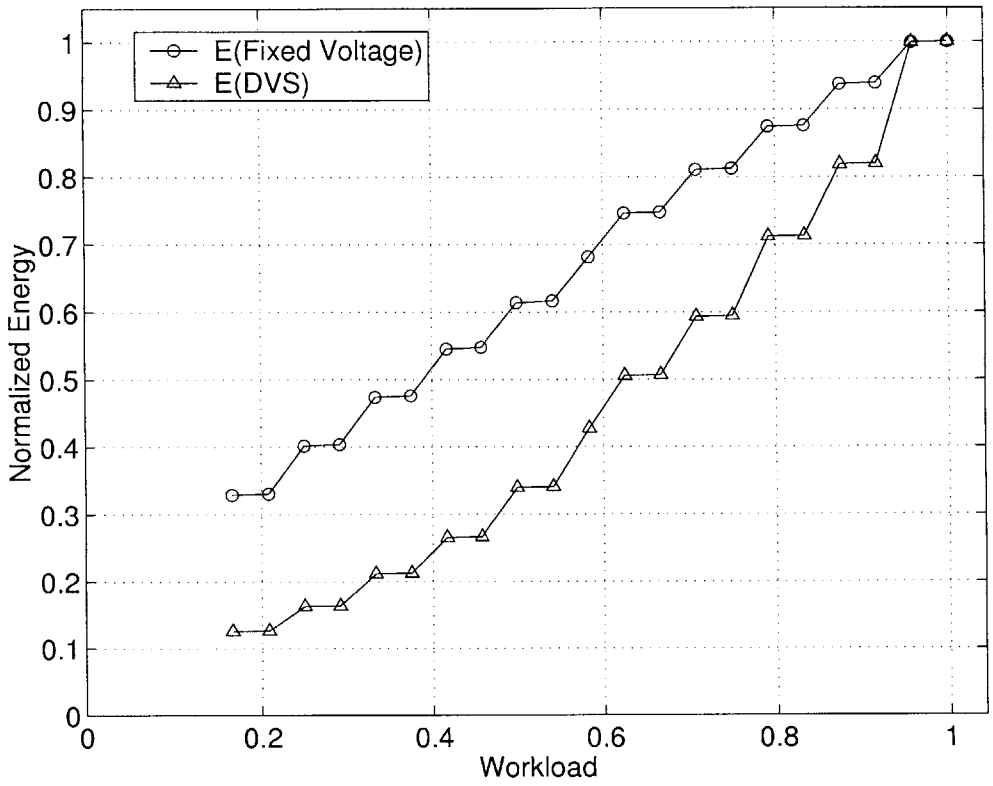


Figure 5-12: The energy curves of the fixed voltage and DVS system. Both curves have been normalized with respect to the maximum load case.

define the power-awareness (since the ratio of the power-awareness of one system to the other is independent of the perfect system). Hence, for uniform load distributions, DVS leads to battery lifetime increases of about 60%.

Chapter 6

Wireless Sensor Networks

6.1 Introduction

Increasingly integrated and lower power electronics are enabling a new sensing paradigm - wireless networks composed of tens of thousands of highly integrated sensor nodes allow sensing that is far superior, in terms of quality, robustness, economics and autonomous operation, to that offered by using a few, ultra high precision macro-sensors [46, 29]. Such sensor networks are expected to find widespread use in a variety of applications including remote monitoring (of climate, equipment etc.) and seismic, acoustic, medical and intelligence data-gathering. Since these integrated sensor nodes have highly compact form factors and are wireless, they are highly energy constrained. Furthermore, replenishing energy via replacing batteries on up to tens of thousands of nodes (in possibly harsh terrain) is infeasible. Hence, it is well accepted that the key challenge in unlocking the potential of such networks is *maximizing their post-deployment active lifetime* [23]. In what follows, we show precisely how to achieve this the power-awareness framework that we have developed earlier.

Any effort at increasing the lifetime of sensor networks is necessarily two pronged. Firstly, the node and the physical layer must be made as energy efficient as possible. Tremendous progress has been reported in this area [7, 15, 9, 45, 33, 32]. Secondly, the *collaborative strategy* i.e. the strategy that governs how nodes cooperate to perform the sensing operation, must be energy efficient as well. Previous work in this area has

dealt with different aspects of the problem. In [27], the focus was *scalable* collaboration, with energy-efficiency an important, but not the central goal. The work reported in [53] first highlighted the need for metrics other than those used in traditional networks (like number of hops) when energy is an issue. Various energy-aware routing heuristics were also proposed in this paper. A related sampling of work that proposed energy-aware routing heuristics is [44, 34, 57, 25, 24]. The first demonstration of near-optimal maximum lifetime routing in ad-hoc networks was [11, 12]. Minimum-energy, but infinite lifetime ad-hoc networks were the subject of [50, 37, 51]. The common thread in almost all previous work is its exclusive focus on energy-efficient routing[1] in ad-hoc networks. While this body of work has important implications for maximizing lifetime in sensor networks, we will repeatedly see in this thesis that routing is but *one* aspect of a collaborative strategy. In the next chapter, we will propose non-constructive bounds on the lifetime enhancing power of collaborative strategies. In chapter 8 we rigorously develop the notion of a collaborative sensing strategy from ground up and show that networks employing this more holistic concept can easily outlast optimally routed minimum-energy networks by 2-3 times.

6.2 Basic operation

The goal of a sensor network is to gather information from a specified region of observation, say $\mathcal{R}$, and relay this to an energy-unconstrained *basestation* (B) (figure 6-1). This information originates due to one or more *sources* located in $\mathcal{R}$. At any given instant, nodes in a sensor network can be classified as *live* or *dead* depending on whether they have any energy left or not. Live nodes collaborate to ensure that whenever a source resides in $\mathcal{R}$, it is sensed and the resultant data relayed to B. A concept central to sensor networks is that of *roles* which nodes assume in order to allow sensing. In the collaborative model we assume, live nodes play one or more of the following *basic roles*:

[1] Exceptions are [25, 24], where energy-aware heuristics were used to guide protocol design in networks that support certain types of collaboration.

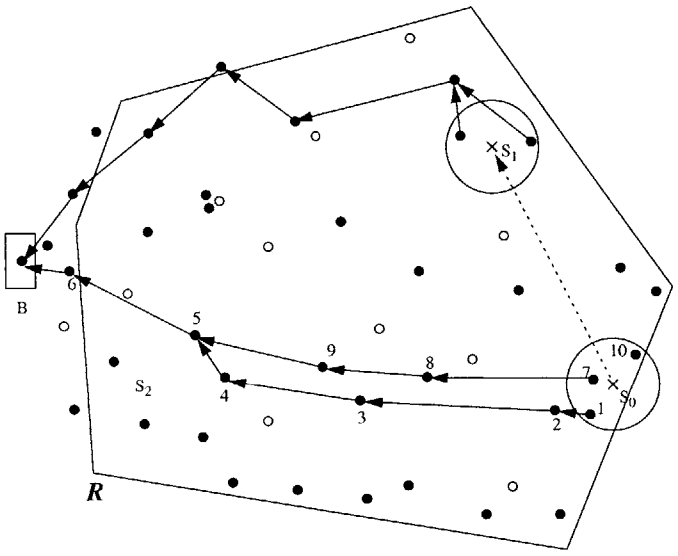


Figure 6-1: A sensor network gathering data from a circularly observable source (denoted by a $\times$) residing in the shaded region R. Live nodes are denoted by $\bullet$ and dead ones by $\circ$. The basestation is marked B. In this example we require that atleast two nodes sense the source. When the source is at S_0, nodes 1 and 7 assume the role of sensors and nodes $2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$ form the relay path for data from node 1 while nodes $7 \rightarrow 8 \rightarrow 9 \rightarrow 5 \rightarrow 6$ form the relay path for data from node 7. Data might be *aggregated* into one stream at node 5. This is not the only feasible role assignment that allows the source to be sensed. For instance, node 10 could act as the second sensor instead of node 7 and $10 \rightarrow 7 \rightarrow 8 \rightarrow 4 \rightarrow 5 \rightarrow 6$ could form the corresponding relay path. Also, node 6 might aggregate the data instead of node 5 etc. Finally, note how the sensor, aggregator and relay roles must change as the source moves from S_0 to S_1. At every instant, the following decisions must be made: which sensor(s) to use, whether to aggregate or not, where to aggregate, what fraction to aggregate, how to route data to the aggregator, how to route aggregated data and how to account for changes in source location and user desired quality. Chapter 8 of this thesis deals with demonstrating a computationally feasible methodology to arrive at these decisions (and their variation with time) such that they are globally optimal in the sense of maximizing network lifetime.

- Sensor: The node observes the source via an integrated sensor, digitizes this information, post-processes it and produces data (this is often referred to as "raw sensor data" as opposed to "aggregated data" as explained below). It is this data which needs to be relayed back to the basestation. Note that although the role is "sensor", the node must forward this stream to another node (possibly itself for aggregation - see later). Hence the sensor role is really a "sense and transmit" role and needs to be qualified by a single attribute - that of the destination of the raw sensor data. The minimum number of sensors that must observe a source is dependent on the desired quality and the topology of the network (more on this in section 7.2).

- Relay: The node simply forwards the received data onward without any processing. A relay role is qualified by two attributes - the source of the data being relayed and the destination to which data is being forwarded.

- Aggregator: The node receives two or more raw data streams and then aggregates them into a single stream. While the actual mechanism of aggregation differs depending on the application, the underlying motivation is always the same - the total volume of data to be routed to the basestation is reduced and the quality of the aggregated stream is invariably higher than that of the raw streams from which it is derived [47]. Consider a sensor network that detects tank intrusion in a specified region. Several nodes might declare a tank present with varying levels of confidence. All these "tank detected" messages may be routed to a node that aggregates them into a single message with a revised confidence measure. Aggregation here corresponds to *data-fusion*. As another example, consider a sensor network collecting acoustic data. When an acoustic event takes place, sensors gather acoustic data with varying signal-to-noise ratios (SNRs). In this case aggregation might correspond to *beamforming* these different streams to obtain a single aggregated stream with enhanced SNR [64]. The aggregation role is qualified by two attributes - the set of nodes transmitting raw data to be aggregated and the destination node receiving the aggregated

stream.

- Dormant: The node is live but is not currently participating in any of the roles above (and hence consuming negligible power). Obviously, this role needs no further qualification.

While nodes can change their roles with time (as illustrated in figure 6-1), we assume that their locations are fixed. Also, mutually compatible basic roles can manifest themselves concurrently in a node giving rise to a set of *complex roles*. For instance, although there are only 4 basic roles above, all but the dormant role is compatible with the others, leading to a total of 8 *complex roles* (the three basic ones, sensor concurrent with relay, relay concurrent with aggregation, all three concurrent and so on).

6.3 Node Composition

In spite of the multiplicity of implementations [46, 48, 38], every integrated wireless sensor node has the same overall composition illustrated in figure 6-2. The node has

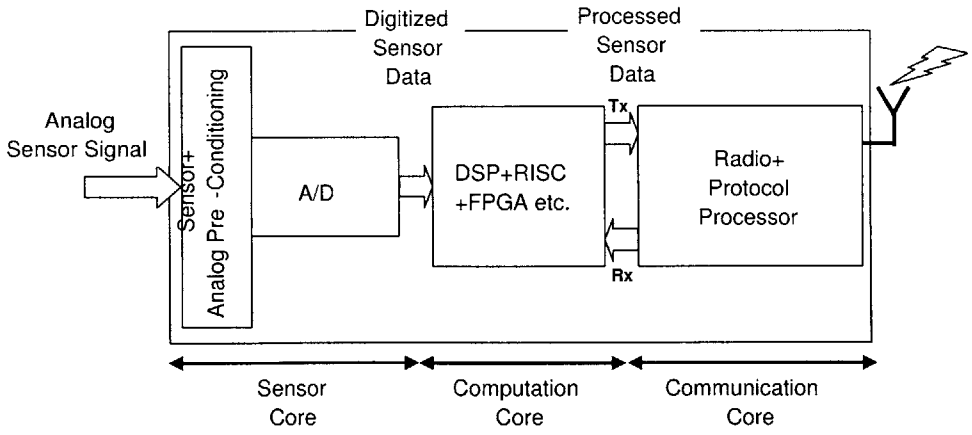


Figure 6-2: Composition of the node.

three key subsystems:

- Sensing core: This subsystem is responsible for digitizing analog sensor data. It is composed of analog pre-conditioning circuitry and a analog-to-digital converter (A/D). This subsystem is active when the node acts as sensor.

- Computation core: This subsystem processes digitized data either from its own sensor core or from a sensor core from another node (in which case that data is conveyed via the radio link). It is this unit which carries out aggregation. Even in the absence of aggregation, this unit is needed to implement compression of the sensor stream. This subsystem is active when a node acts as an aggregator.

- Communication core: This subsystem is responsible for providing a digital communications link between different sensors. It is in turn composed of a radio transceiver that handles the physical layer and a protocol processor that implements the remainder of the protocol stack. This subsystem is *always* active except during dormancy.

6.4 Energy Models

We now detail the energy consumption characteristics of each node subsystem. This combined with our knowledge of the subsystems used in a particular role will allow us to specify the energy consumption of a node when it assumes a certain role[2]. As will become clearer over the course of the next chapters, none of our developments depend on the linearity or presence of a bias or similar characteristics in the dependence of power consumption on the constituent factors. Hence, presented with a choice, we pick the simplest, reasonably realistic models to gain insight and keep the math tractable.

[2]In what follows, the energy of a role (say sensing) should not be confused with the energy of a subsystem with the same name (sensing core).

6.4.1 Sensor Core

We assume that the energy needed to sense a bit (E_{sense}) is a constant (denote by α_3) whose value is in the range of 50 nJ/bit [24]. Hence if the sensing rate is given by r bits/sec then the sensing power is given by $P_{sense} = \alpha_3 r$. As stated earlier, the linear dependence on rate can be replaced by a more general non-linear relationship and a rate-independent bias introduced to better model the sensing core.

6.4.2 Computation Core

We assume the following model for the computational energy consumed in the core during aggregation [60]:

$$P_{comp} = n_{agg}\alpha_4 r \tag{6.1}$$

where P_{comp} is the power required for aggregating n_{agg} streams present at the node into one stream and r is the rate of each of the n_{agg} streams (and also that of the output stream). A typical order of n_{agg} is 10s of nJ/bit. Note that since P_{comp} is just the energy dissipation of the computational core it does not include the energy costs of receiving the streams and that of transmitting the aggregated stream.

6.4.3 Communication Core

This core has two logical sub-units - transmit and receive. The energy needed to transmit and receive a bit is given by [49]:

$$E_{tx} = \alpha_{11} + \alpha_2 d^n \tag{6.2}$$

$$E_{rx} = \alpha_{12} \tag{6.3}$$

where E_{tx} is the transmit energy per bit, d is the distance to which the bit is being transmitted, n is the path loss index, E_{rx} is the receive energy per bit. The terms α_{11} and α_{12} capture the energy dissipated by the transmitter and receiver electronics respectively, while α_2 captures the energy radiated via the power-amp. Note that α_2 depends on n (which is greater than 2 and rarely greater than 5). Typical values

of these parameters are α_{11}=45 nJ/bit, α_{12}=135 nJ/bit, α_2=10 pJ/bit/m^2 (n=2) or 0.001 pJ/bit/m^4 (n=4) [24].

In the next chapter, we use these models to derive fundamental bounds on lifetime. In chapter 8, we show how these bounds can be actually achieved.

Chapter 7

Fundamental Bounds on Network Lifetime

Any effort to increase the network lifetime must necessarily be two-pronged. Firstly, the node itself must be made as energy efficient as possible [47, 9, 40]. Secondly, the *collaborative strategies* which govern how nodes co-operate to sense data must be energy efficient. Most work in this latter area has been directed towards energy-aware routing - both in the context of sensor and mobile ad-hoc networks (which share some of the characteristics of sensor networks). Some representative work in this area is [53, 12, 25, 50]. In this chapter, our key objective is neither proposing new energy-aware routing heuristics nor new protocols. Instead, it is to explore the fundamental limits of data gathering lifetimes that these strategies strive to increase. Our motivation for doing so is several-fold. Firstly, bounds on achievable lifetime of sensor networks allow one to calibrate the performance of collaborative strategies and protocols being proposed regularly. Unlike strategies which are mostly heuristic due to the combinatorially explosive nature of the problem, the proposed bounds are crisp and widely applicable. Secondly, in order to prove that the proposed bounds are tight or near tight, we construct real networks and simulate data gathering and show that their lifetimes often come arbitrarily close to optimal. This exercise gives an insight into near-optimal data gathering strategies if the user has some level of deployment control. Thirdly, in bounding lifetime, we expose its dependence on source behavior,

source region, basestation location, number of nodes, available initial energy, path loss and radio energy parameters. This allows us to see what factors have the most impact on lifetime and consequently where engineering effort is best expended.

7.1 Defining Lifetime

There are several possible definitions of the lifetime of a sensor network, each suitable in a different context. A network as a whole can be in different states:

1. Source present in region but network not sensing. This is termed "loss of coverage".

2. Source present and network sensing while satisfying user dictated quality and latency constraints. This state is termed "active".

3. Source present and network sensing but not satisfying user dictated quality constraints. This state is termed "quality failure".

4. No source present in the region.

In non-mission-critical applications, a reasonable definition of lifetime is the cumulative active time of the network (i.e. whenever the network is active its lifetime clock is ticking, otherwise not). In mission-critical applications, lifetime is defined as the cumulative active time of the network *up until the first loss of coverage or quality failure*. In this paper, we adopt this latter definition of lifetime. Note that active lifetime is different from *physical* lifetime. For instance a sensor network deployed to detect tank intrusion can "live on" for several hundreds of years (ignoring battery degradation etc.) in the absence of activity. But it can only actively detect, say, 1000 hours of tank intrusion.

7.2 Factors Affecting Lifetime

Factors affecting lifetime of energy limited systems can be methodically listed down by invoking the theory of energy-aware systems developed in [5, 6]. In the context of

sensor networks, lifetime depends on the following factors and their variations (with time and possibly space):

- **Inputs and/or input statistics**: This dimension includes factors like the topology of the region to be sensed, the topology of the network, number of sources and their characteristics etc. For instance, the lifetime of a network sensing 5 sources will generally be very different than the scenario where it senses just one. The lifetime of a sensor network detecting tank intrusion is dependent on the velocity constraints on the tank and a stochastic description of its location (i.e. does it breach the perimeter uniformly or is it more likely to breach certain intervals more than others?).

- **Desired output quality**: As one would expect, higher quality sensing leads to shorter lifetimes. The two-step approach we take in this thesis to quantitatively link desired quality and lifetime follows. In the first step, desired quality is used to derive the minimum stream rate needed and the minimum number of sensors that must observe the source[1]. It suffices to say that rate-distortion based information theoretic arguments provide a fundamental characterization of the quality-rate tradeoff [47]. Consider the tank intrusion application again. Once the user specifies a particular temporal and spatial resolution (i.e. locate the tank to within 2 meters every 10 seconds), information theoretic arguments provide reasonable lower bounds on the rate of the stream needed (i.e. the network must support a rate of 10 kbps for the above resolution). Linking the desired quality to the minimum number of sensors needed usually draws on array signal processing formalisms like beamforming [64]. For instance, if we assume that environmental noise is spatially uncorrelated, using three sensors and simple delay-and-sum beamforming will increase the SNR (and loosely speaking quality) by three times.

- **Tolerable latency**: In a well designed network, higher tolerable latency is generally exploited to yield higher lifetimes. The key mechanism enabling this

[1] As one would expect, these two factors are not totally unrelated.

latency-lifetime tradeoff is scaling the voltage of node electronics [39, 59]. Specifically, power consumption varies as the square of the voltage and delay varies in an inverse manner with voltage. Hence higher delays permit lower energy consumption. Another reason why increased latency allows one to reduce energy is buffering i.e. by increasing the packet sizes (at the cost of latency), one can amortize the energy costs of radio startup time. In this thesis, we do not address tolerable latency explicitly. Rather, it is factored in implicitly via the node energy models.

- **The ambient environment**: The ambient temperature, noise characteristics, behavior of the wireless channel all affect the cost of computation and communication in a sensor network. Our methodology is flexible enough to allow different costs of computation and communication for different nodes and links in the network.

- **The state of the network**: Networks with different initial states exhibit different lifetimes and different optimal collaborative behavior. Like most previous work, our framework allows specification of networks with arbitrary initial energy states.

7.3 The Lifetime Bound Problem

The Lifetime Bound Problem: Given the region of observation ($\mathcal{R}$), the source radius of observability (d_S), the node energy parameters ($\alpha_1, \alpha_1 1$, $\alpha_1 2$,α_2, α_3 and n), the number of nodes deployed (N), the initial energy in each node (E), what is the upper bound on the active lifetime (t) of *any* network established using these nodes which gathers data from a source residing in $\mathcal{R}$ with spatial location behavior $l_{source}(x, y)$.

In the following sections, we solve this problem for a variety of source behavior. We also illustrate the tightness (or near-tightness) of these bounds by analytical construction and simulation.

7.4 Characteristic Distance and Minimum Energy Relays

A recurring theme in bounding lifetimes of data gathering networks is the problem of establishing a data link with a certain rate r between a radio transmitter (at A) and a receiver (at B) separated by D meters. There are several ways of doing this. One can directly transmit from A to B or one can use several intervening nodes acting as *relays* to prevent any node from having to spend too much transmit energy (figure 7-1). This is often referred to *multi-hop routing*[2]. We refer to the scheme that

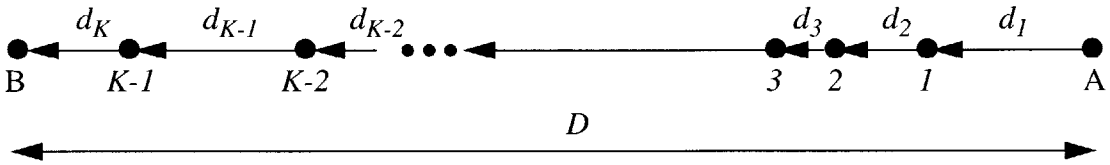


Figure 7-1: Introducing $K-1$ relay nodes between A and B to reduce energy needed to transmit a bit.

transports data between two nodes such that the *overall* rate of energy dissipation is minimized as a *minimum energy relay*. If we introduce $K-1$ relays between A and B (fig. 7-1), then the overall rate of dissipation is given by,

$$P_{link}(D) = \left(-\alpha_{12} + \sum_{i=1}^{K} P_{relay}(d_i)\right) r \tag{7.1}$$

Often we omit the rate term (r) and it is understood that the expression has been written for unit rate i.e. $r = 1$. The $-\alpha_{12}$ term accounts for the fact that the node at A need not spend any energy receiving. We disregard the receive energy needed at B because in most applications of minimum energy relays it is the basestation and hence has no energy constraints. Even if it did, the receive energy is independent of the number of intervening relays anyway and hence we can safely ignore it. We now present some simple properties of minimum energy relays.

[2]This is a term that the author is not particularly fond of. In all but the most trivial networks, data is routed via more than one node. Hence every network is a multi-hop network, rendering the adjective vacuous.

Lemma 7.1. *Minimum energy relays have all nodes collinear and no directed link has a negative projection on vector* $\overrightarrow{AB}$.

Proof. Consider an arrangement in which either of these statements does not hold. Then, carry out the following two step transformation (figure 7-2):

- Step I: Move every node to its projection on AB.
- Step II: Create a link in the direction AB between every two adjacent nodes.

We claim that the overall power dissipation in the transformed network is no more than that in the original one. This is obvious since the projection of a vector cannot exceed the length of a vector. The result follows. □

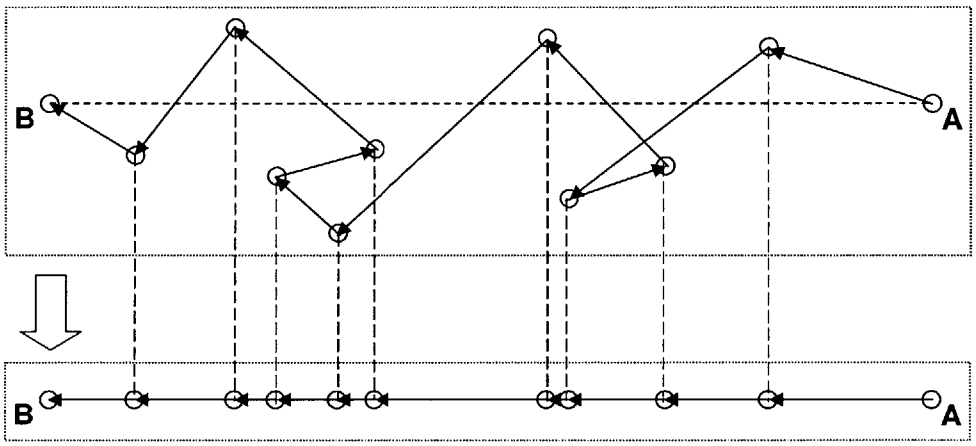


Figure 7-2: Every non-collinear network with links that have negative projections along AB can be transformed into a more efficient collinear network with links only in the AB direction.

Hence, given $K-1$ intervening nodes, the problem is merely one of placing these nodes along the line AB to minimize overall energy. The following lemma tells us how.

Lemma 7.2. *Given* D *and the number of intervening relays* $(K-1)$, $P_{link}(D)$ *is minimized when all the hop distances (i.e.* d_i*s) are made equal to* $\frac{D}{K}$. *This result holds for all radios with convex power versus distance curves i.e. whose energy per bit is a convex function of the distance over which the bit is transmitted.*

Proof. Recall that $P_{relay}(d)$ if of the form $\alpha_1 + \alpha_2 d^n$ and hence strictly convex. Next, note Jensen's inequality for convex functions,

$$\forall\{\lambda_i\}, \lambda_i \in \mathbb{R}^+ \text{ such that } \sum_i \lambda_i = 1$$
$$f\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i f(x_i) \tag{7.2}$$

with equality *iff* all the x_is are equal. Then, it follows that,

$$P_{relay}\left(\frac{d_1 + d_2 + \cdots + d_K}{K}\right) \leq \frac{P_{relay}(d_1) + P_{relay}(d_1) + \cdots + P_{relay}(d_K)}{K}$$
$$KP_{relay}\left(\frac{D}{K}\right) \leq P_{relay}(d_1) + P_{relay}(d_1) + \cdots + P_{relay}(d_K)$$
$$KP_{relay}\left(\frac{D}{K}\right) - \alpha_{12} \leq P_{link}(D) \tag{7.3}$$

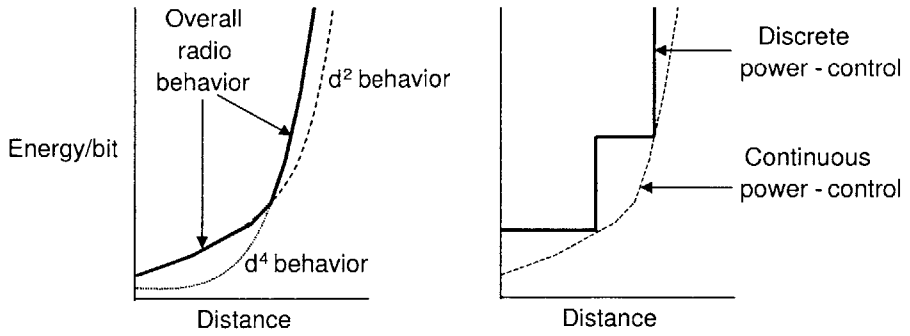
with equality *iff* all the d_is are equal (to D/K). The result follows. □

Corollary 7.3. *The minimum energy relay for a given distance D has either no intervening hops or K_{opt}* equidistant *hops where K_{opt} is completely determined by D.*

It is instructive to point out that while making hops equidistant seems like an obvious thing to do, it works only due to the convexity of the radio's power-distance curve. This is illustrated in figure 7-3. The radio in 7-3(a) does not have a uniform path loss characteristic. Like all practical radios, its path loss index n increases with distance [24]. Since a "piecewise convex function" retains its convexity, the radio curve stays convex in this case. It follows that minimum energy relays using this radio must have equidistant hops. However, if we factor in another practical concern - the granularity of power control in a radio, then we might lose convexity, as shown in the figure 7-3(b). For radios like this, equidistant spacing does *not* lead to the lowest energy solution.

We have proved that the hops must be equidistant for convex radios. We now derive the relation between the optimal number of hops K and the distance D.

Lemma 7.4. *The optimal number of hops (K_{opt}) is always one of,*



(a) A Convex Radio Curve (b) A Non-Convex Radio Curve

Figure 7-3: Two deviations from ideal radios are shown here. The radio on the left retains a convex energy-distance curve while the radio on the right does not. Hence, while equidistant hops are optimal for the radio on the left, they are rarely optimal for the radio on the right.

$$K_{opt} = \left\lfloor \frac{D}{d_{char}} \right\rfloor or \left\lceil \frac{D}{d_{char}} \right\rceil \tag{7.4}$$

where the distance d_{char}, called the characteristic distance, *is* independent of D *and is given by,*

$$d_{char} = \sqrt[n]{\frac{\alpha_1}{\alpha_2(n-1)}} \tag{7.5}$$

Proof. From (7.3), we have, for equidistant hops,

$$P_{link}(D) = KP_{relay}\left(\frac{D}{K}\right) - \alpha_{12}$$

Let us optimize $P_{link}(D)$ w.r.t K for the case when $K \in \mathbb{R}^+$. Then, by taking the derivative and setting it to zero, we get,

$$K_{opt(continuous)} = \frac{D}{d_{char}} \tag{7.6}$$

where d_{char} is defined above. Next, note that if $f(x)$ is strictly convex then, $xf(a/x)$ is convex too[3]. This implies that $KP_{relay}\left(\frac{D}{K}\right)$ is convex in K. Hence, once we calculate

[3]Quick proof: The second derivative of $xf(a/x)$ w.r.t. x is $(a^2/x^3)f''(a/x)$. But f is convex,

the minima for $K \in \mathbb{R}^+$ as given by (7.6), the minima for $K \in \mathbb{N}$ is given by (7.4). □

Corollary 7.5. *The power needed to relay a stream with unit rate over distance D can be bounded thus:*

$$P_{link}(D) \geq \alpha_1 \frac{n}{n-1} \frac{D}{d_{char}} - \alpha_{12} \tag{7.7}$$

with equality if and only if D is an integral multiple of d_{char}.

Proof. By the convexity of $KP_{relay}\left(\frac{D}{K}\right)$ and the lemma above, it follows that,

$$\begin{aligned} P_{link}(D) &\geq K_{opt(continuous)} P_{relay}\left(\frac{D}{K_{opt(continuous)}}\right) - \alpha_{12} \\ &= \frac{D}{d_{char}}\left(\alpha_1 + \alpha_2 d_{char}^n\right) - \alpha_{12} \\ &= \frac{D}{d_{char}}\left(\alpha_1 + \alpha_2 \frac{\alpha_1}{\alpha_2(n-1)}\right) - \alpha_{12} \\ &= \alpha_1 \frac{n}{n-1} \frac{D}{d_{char}} - \alpha_{12} \end{aligned}$$

with equality in the first step only if D/d_{char} is integral (i.e. when K_{opt} is the same as $K_{opt(continuous)}$). □

The corollary above makes several important points:

- For any loss index n, the energy costs of transmitting a bit can always be made *linear* with distance.

- For any given distance D, there is a certain optimal number of intervening nodes acting as relays that must be used (K_{opt}). Using more or less than this optimal number leads to energy inefficiencies.

- The most energy efficient relays result when D is an integral multiple of the characteristic distance.

---

which implies $f''(a/x)$ is positive. Hence, $(a^2/x^3)f''(a/x)$ is strictly positive for positive a, x, proving the convexity of $xf(a/x)$

7.5 Bounding Lifetime

We now derive the upper bounds on the lifetime of sensor networks for a variety of source behavior.

7.5.1 Fixed Point Source Activity

The simplest sensor network is one that harvests data from a fixed source located at a distance d_B away from the basestation (figure 7-4).

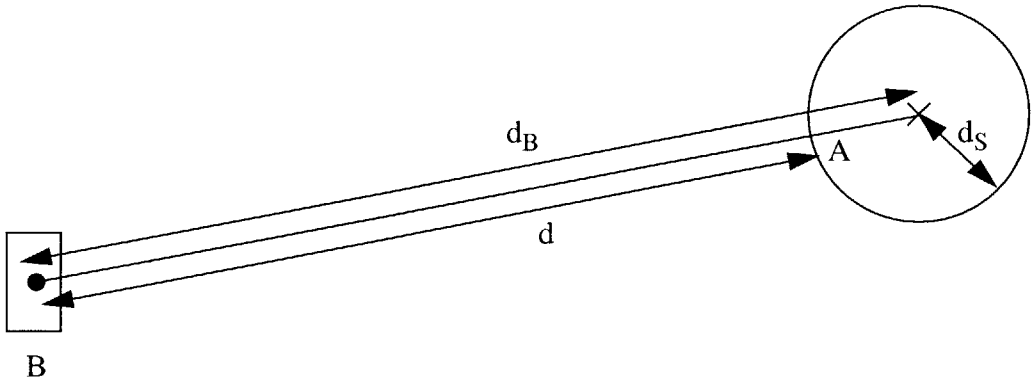


Figure 7-4: Gathering data from a circularly observable, fixed point source d_B away from the basestation (B). The source location is marked by $\times$.

It is easy to see that we have to establish a link of length equal to at least $d = d_B - d_S$ and sustain the source rate (say r) over this link. If we denote the energy dissipation in the *entire* network by $P_{network}$ then it follows from our discussion on minimum energy relays that,

$$\begin{aligned} P_{network} &\geq P_{link}(d) + P_{sensing} \\ &\geq \left(\alpha_1 \frac{n}{n-1}\frac{d}{d_{char}} - \alpha_{12}\right) r + \alpha_3 r \end{aligned} \tag{7.8}$$

Clearly, achieving an *active* lifetime of say, t_{point}, demands that the total energy consumed be no greater than the total energy available at the start, i.e.,

$$t_{point} P_{network} \leq \sum_{i=1}^{N} e_i(0)$$

which reduces to,

$$t_{point} \leq \frac{N.E}{\left(\alpha_1 \frac{n}{n-1} \frac{d}{d_{char}} - \alpha_{12} + \alpha_3\right) r} \tag{7.9}$$

for the case of a N node network with $e_i(0)$ (i.e. the energy of node i at deployment) set to E. While the bound in (7.9) is exact, we often use the following approximation noting that in most networks of practical significance, the cost of relaying data dominates,

$$t_{point} \leq t_{point_{max}} = \frac{N.E}{\alpha_1 \frac{n}{n-1} \frac{d}{d_{char}} r} \tag{7.10}$$

One can eliminate d_{char} in (7.10) to obtain,

$$t_{point_{max}} = \frac{N.E}{\frac{n}{n-1} \sqrt[n]{\alpha_1^{n-1} \alpha_2 (n-1)} (d_B - d_S) r} \tag{7.11}$$

For the simplest path loss model ($n = 2$), this simplifies to,

$$t_{point_{max}} = \frac{N.E}{2\sqrt{\alpha_1 \alpha_2} (d_B - d_S) r} \tag{7.12}$$

Proposition 7.6. *The bound in (7.10) is tight when $d = d_B - d_S$ is an integral multiple of d_{char} and N is an integral multiple of $\frac{d}{d_{char}}$ i.e. with these conditions satisfied, there exist networks whose lifetime equals the upper bound.*

Proof. We present a proof by construction. Let $d_B - d_S = d = M d_{char}$, $M \in \mathbb{N}$, and $N = PM$, $P \in \mathbb{N}$. Then form P parallel "backbones" of $M = \frac{d_B - d_S}{d_{char}}$ nodes as shown in figure 7-5 and use exactly one backbone at any time instant. Each node in the backbone is dissipating power equal to $\alpha_1 + \alpha_2 d_{char}^n$ which evaluates to $\frac{n}{n-1}\alpha_1$. Hence the lifetime achieved by a backbone is $\frac{E}{\frac{n}{n-1}\alpha_1}$. The overall lifetime is simply $P = \frac{N}{M} = \frac{N}{\frac{d_B - d_S}{d_{char}}}$ times this lifetime and thus given by $\frac{N}{\frac{d_B - d_S}{d_{char}}} \frac{E}{\frac{n}{n-1}\alpha_1}$ which is the same as the bound derived in (7.12). □

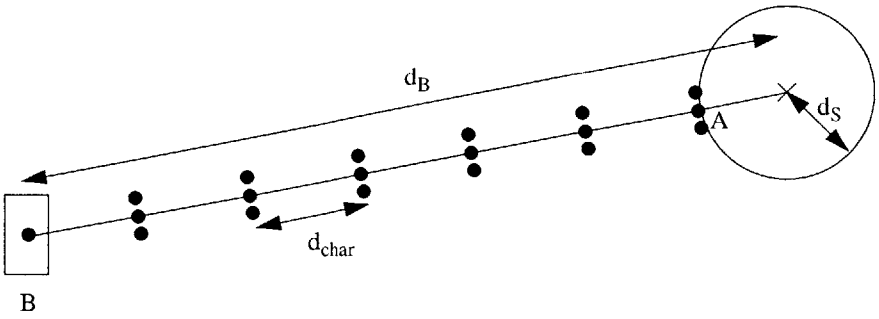


Figure 7-5: An example network that achieves the upper bound on lifetime. Here, $d_B - d_S = 6d_{char}$ i.e. $M = 6$ and $N = 18 = 3(6)$ i.e. $P = 3$. Thus there are 3 parallel minimum-energy backbones of 6 nodes each.

To experimentally validate the derived bounds, a custom network simulator was used to simulate networks that gathered data from a point source using nodes with the energy behavior described earlier[4]. Figure 7-6 charts the lifetimes achieved by thousands of networks with different values of N and d. As predicted, some networks

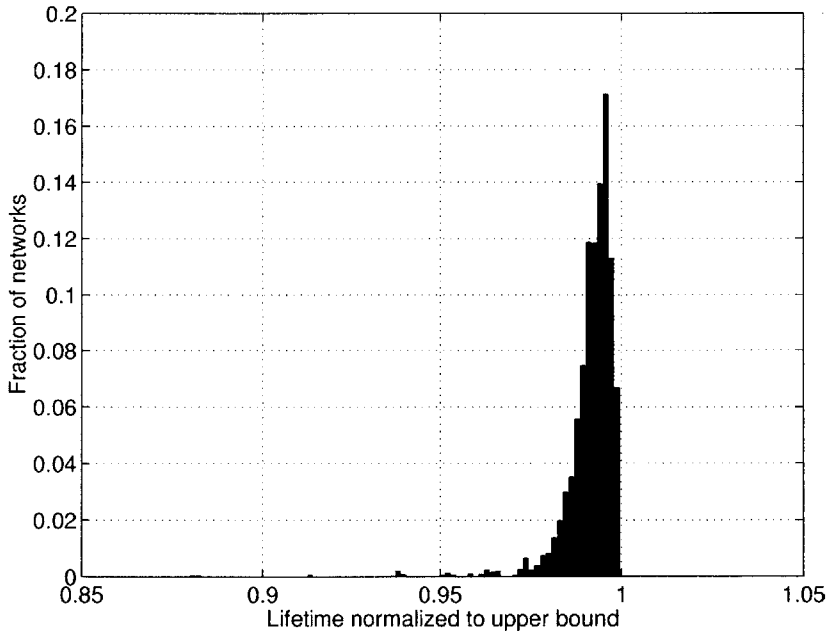


Figure 7-6: Observed lifetimes of >50,000 actually constructed networks that gather data from a fixed point source. For each network, the lifetime was determined via simulation and then normalized to the upper bound in (7.10). These networks had $400 \geq N \geq 100$ and $20d_{char} \geq d_B - d_S \geq 0.1d_{char}$.

do achieve a lifetime equal to the bound. The networks used to obtain this data did

[4]The author wishes to acknowledge Timothy Garnett for developing the simulator.

not have random topologies. Rather, the networks and the collaborative strategies were designed with the express intention of defeating the upper bounds. Hence, the tightness apparent in figure 7-6 should not be interpreted as the lifetime expected of general networks but rather as the *best* possible lifetimes that *some* networks can achieve. Note that any network can achieve an arbitrarily poor lifetime and hence the issue of worst possible lifetime is vacuous.

7.5.2 Activity Distributed Along a Line

We now consider the case of harvesting information from a source that is located along a line (S_0S_1) of length d_N as shown in figure 7-7. The minimal power for sensing the

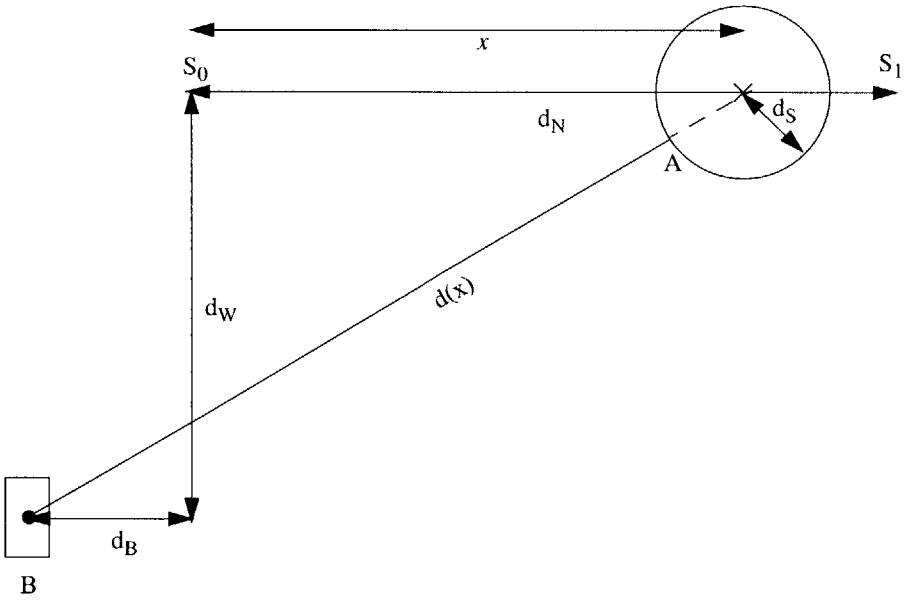


Figure 7-7: Gathering data from a source that resides on a line (S_0S_1).

source at a unit rate is[5],

$$P_{network}(x) \geq P_{link}(d(x)) \geq \frac{n}{n-1}\alpha_1 \frac{d(x)}{d_{char}} \tag{7.13}$$

[5]Recall that we are ignoring the α_3 and $-\alpha_{12}$ term.

For the case when the source is located along S_0S_1 with equal probability $\frac{1}{d_N}$, the expected overall rate of dissipation for sensing is,

$$\begin{aligned} P_{network} &\geq \int_{x=d_B}^{x=d_B+d_N} P_{network}(x) l_{source}(x) \mathrm{d}x \\ &\geq \int_{x=d_B}^{x=d_B+d_N} \frac{n}{n-1} \alpha_1 \frac{d(x)}{d_{char}} \frac{1}{d_N} \mathrm{d}x \\ &\geq \frac{n}{n-1} \alpha_1 \frac{d_{linear}}{d_{char}} \end{aligned} \tag{7.14}$$

where,

$$d_{linear} = \frac{d_1 d_2 - d_3 d_4 + d_W^2 \ln(\frac{d_1+d_2}{d_3+d_4})}{2d_N} - d_S \tag{7.15}$$

Hence, the bound on the lifetime of a network gathering data from a source that resides on a line with equal probability is,

$$t_{linear\,max} = \frac{N.E}{\frac{n}{n-1}\alpha_1 \frac{d_{linear}}{d_{char}} r} \tag{7.16}$$

When S_0S_1 passes through the basestation B i.e. $d_W = 0$ (or S_0, S_1 and B are collinear) we have,

$$t_{collinear\,max} = \frac{N.E}{\frac{n}{n-1}\alpha_1 \frac{d_B + \frac{d_N}{2} - d_S}{d_{char}} r} \tag{7.17}$$

Figure 7-8 plots the lifetime achieved by *non*-collinear networks gathering data from a source that resides along a line. For each simulated network, the lifetimes have been normalized to the upper bound in (7.16). Clearly, the bound is near tight.

Proposition 7.7. *The bound for the collinear case (7.17) is tight i.e. there exist networks that achieve a lifetime equal to the bound.*

Proof. Consider the case when,

- d_N is an integral multiple of $2d_S$ i.e. $d_N = M(2d_S), M \in \mathbb{N}$.
- $2d_S$ is an integral multiple of d_{char} i.e. $2d_S = Ld_{char}, L \in \mathbb{N}$.
- $d_B + d_S$ is an integral multiple of d_{char} i.e. $d_B + d_S = Td_{char}, T \in \mathbb{N}$.

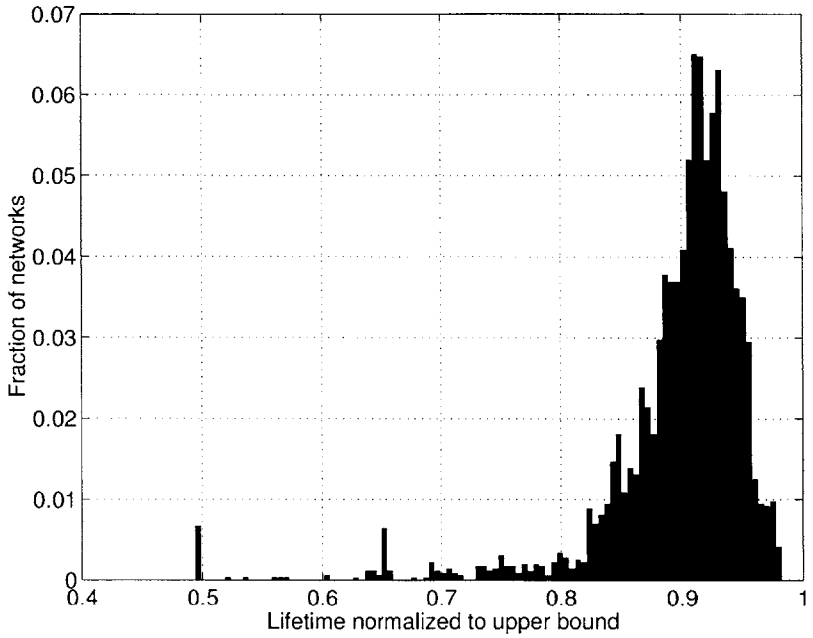


Figure 7-8: Observed lifetimes of ≈3500 networks that gather data from a source residing on a line ($400 \geq N \geq 100$, $d_B \leq 10d_{char}$, $d_W \leq 7.5d_{char}$, $11d_{char} \geq d_N \geq d_{char}$, $1.5d_{char} \geq d_S \geq 0.25d_{char}$).

Then, we can arrange nodes as shown in figure 7-9. Note that the key idea is to establish M minimum energy relays. For this we need exactly $\sum_{m=1}^{M}(L(m-1)+T) = M(L(M-1)/2+T)$ nodes. If the number of nodes available is an integral multiple of this number, then we achieve the bound with equality (the proof is an exact analogue of that in (7.6)). □

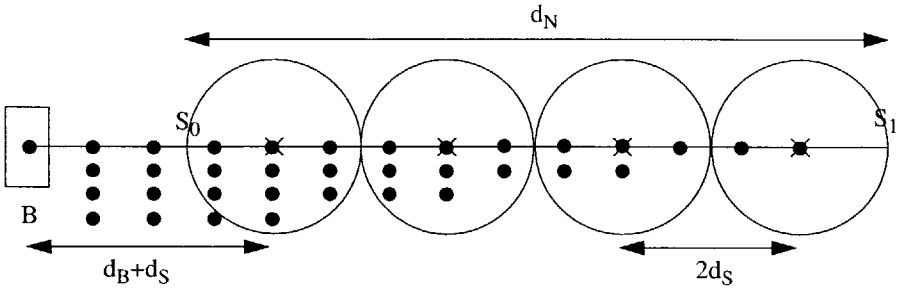


Figure 7-9: An example network that achieves the upper bound on lifetime. Here, $d_N = 4(2d_S)$ i.e. $M = 4$, $2d_S = 3d_{char}$ i.e. $L = 3$, $d_B + d_S = 4d_{char}$ i.e. $T = 4$. We establish $M = 4$ minimum energy relays consisting of 4,7,10 and 13 nodes respectively.

7.5.3 Activity Distributed Over a Rectangular Region

Consider harvesting information from a source that resides in a rectangle (fig. 7-10). Assuming $l_{source}(x, y)$ to be uniform, we have,

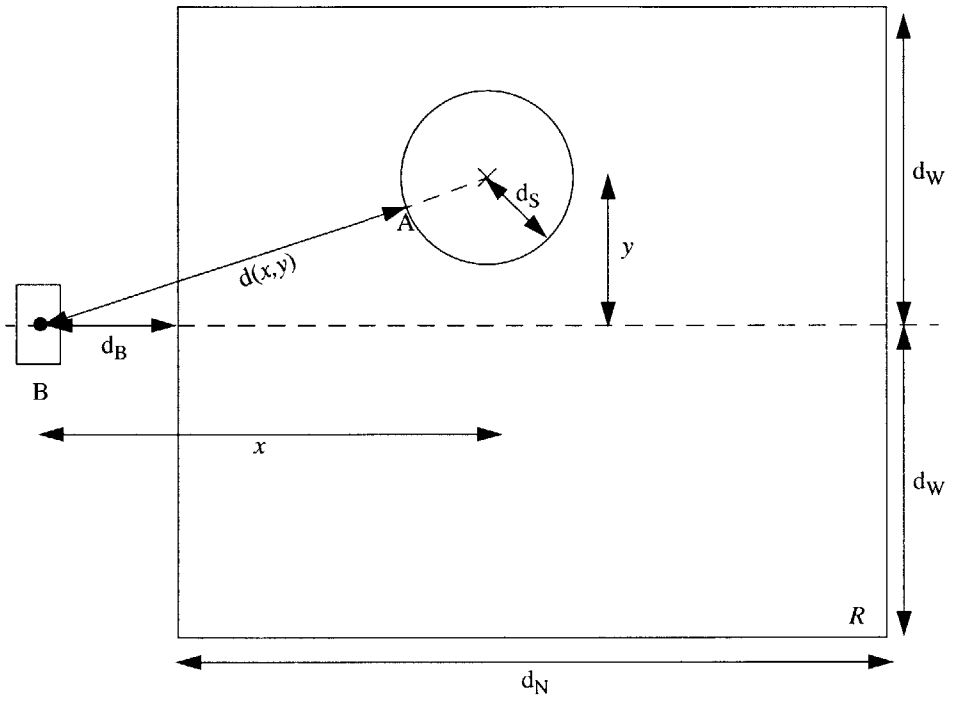


Figure 7-10: Gathering data from a source that resides in a $2d_w$ by d_N rectangle that is d_B away from the basestation (B).

$$\begin{aligned} P_{network} &= \iint_R P_{network}(x,y) l_{source}(x,y)\,\mathrm{d}x\,\mathrm{d}y \\ &\geq \int_{x=d_B}^{x=d_B+d_N} \int_{y=-d_W}^{y=d_W} P_{link}(d(x,y)) \frac{1}{2d_W d_N}\,\mathrm{d}x\,\mathrm{d}y \\ &\geq \int_{x=d_B}^{x=d_B+d_N} \int_{y=-d_W}^{y=d_W} \frac{n}{n-1}\alpha_1 r \frac{\sqrt{x^2+y^2}-d_S}{2d_W d_N d_{char}}\,\mathrm{d}x\,\mathrm{d}y \\ &\geq \frac{n}{n-1}\alpha_1 r \frac{d_{rect}}{d_{char}} \end{aligned} \tag{7.18}$$

where

$$\begin{aligned} d_{rect} = -d_S + \frac{1}{12 d_N d_W}\Big[& 4d_W(d_1 d_2 - d_3 d_4) + \cdots \\ & 2d_W^3 \ln\left(\frac{d_1+d_2}{d_3+d_4}\right) + d_3^3 \ln\left(\frac{d_4-d_W}{d_4+d_W}\right) + d_1^3 \ln\left(\frac{d_2+d_W}{d_2-d_W}\right)\Big] \end{aligned} \tag{7.19}$$

Hence, the bound on expected lifetime is,

$$t_{rectangle_{max}} = \frac{N.E}{\frac{n}{n-1}\alpha_1 r \frac{d_{rect}}{d_{char}}} \tag{7.20}$$

Figure 7-11 plots the lifetime of networks gathering data from sources that reside in rectangles. Once again, the lifetime of each network has been normalized to the bound in (7.20) and the tightness of the bound is apparent.

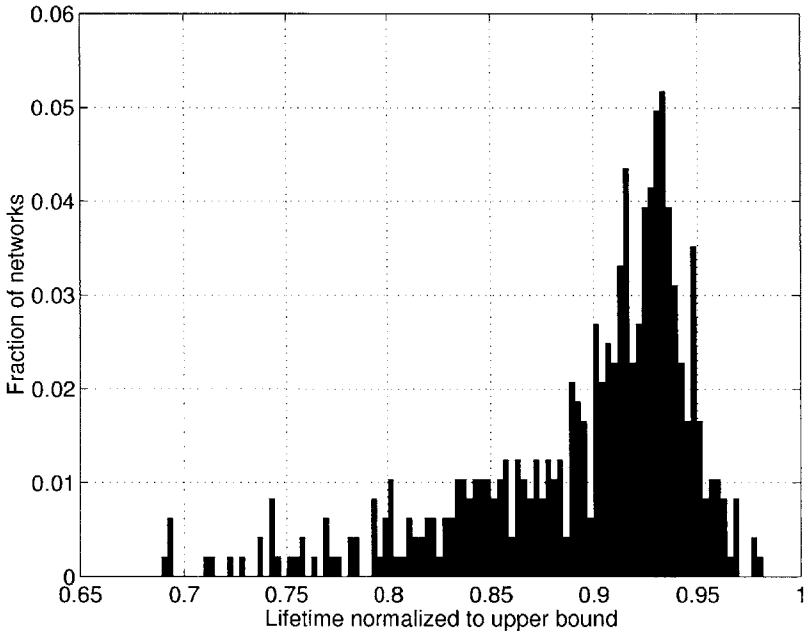


Figure 7-11: Observed lifetimes of ≈500 networks that gather data from sources residing in rectangles ($1000 \geq N \geq 100$, $d_B \leq 10d_{char}$, $6.75d_{char} \geq 2d_W \geq 0.5d_{char}$, $13.5d_{char} \geq d_N \geq d_{char}$, $1.5d_{char} \geq d_S \geq 0.25d_{char}$).

7.5.4 Activity Distributed Over a Sector

Consider a source that resides in a sector as shown in figure 7-12. In a manner analogous to earlier derivations, we can show that the expected lifetime is bounded

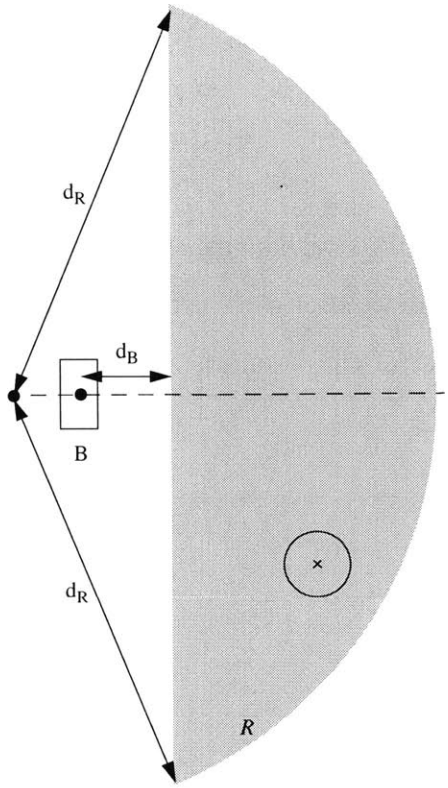


Figure 7-12: Gathering data from a source that resides in a sector (subtending angle 2θ at the centre) d_B away from the basestation (B).

by,

$$\mathcal{L}_{sector} = \frac{N.E}{\frac{n}{n-1}\alpha_1 r \frac{d_{sector}}{d_{char}}} \tag{7.21}$$

where,

$$d_{sector} = \frac{2\theta d_R^3 - d_B d_R d_W - d_B^3 \ln\left(\frac{d_R+d_W}{d_B}\right)}{3(\theta(d_B^2 + d_W^2) - d_B d_W)} - d_S$$

For the special case of a basestation at the center of a semi-circle (i.e. $d_B = 0, \theta = \pi$) we get,

$$\mathcal{L}_{semi-circle} = \frac{N.E}{\frac{n}{n-1}\alpha_1 r \frac{\frac{2}{3}d_R - d_S}{d_{char}}} \tag{7.22}$$

Figure 7-13 compares the lifetime of some semi-circular data gathering networks compared against the upper bound.

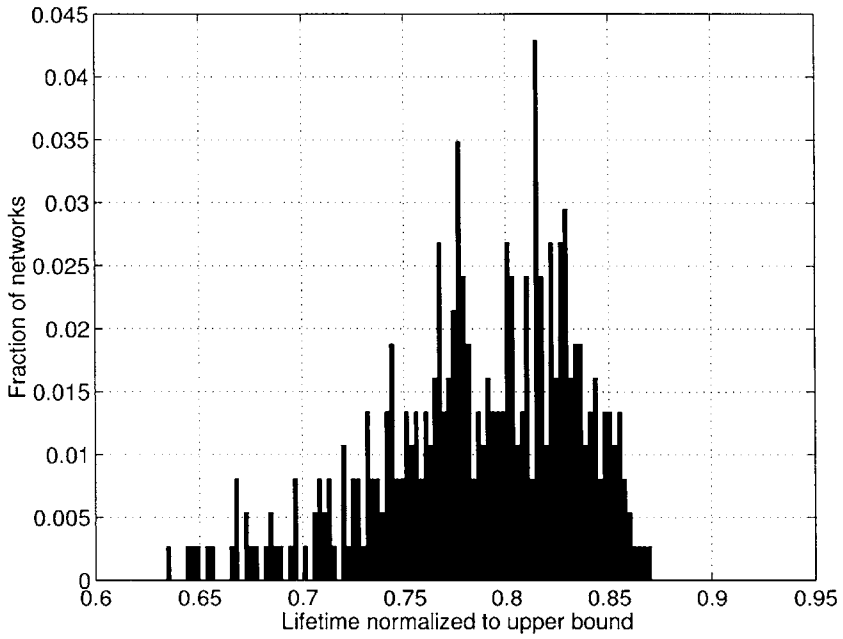


Figure 7-13: Observed lifetimes of ≈350 networks that gather data from sources residing in semi-circles ($1000 \geq N \geq 100$, $d_B = 0$, $10d_{char} \geq d_R \geq 1.25d_{char}$, $2d_{char} \geq d_S \geq 0.2d_{char}$).

7.6 Bounding Lifetime by Partitioning

The following theorem is useful in deriving lifetime bounds for source regions that can be partitioned into sub-regions for which the bounds are already known, or easier to compute.

Theorem 7.8. *The lifetime bound* per unit initial energy, $\tau(\mathcal{R})$, *of a network gathering data from a source region* $\mathcal{R}$ *that can be partitioned into* Q *disjoint regions* $\mathcal{R}_j, j \in [1, Q]$ *with their corresponding lifetime bounds (also normalized),* $\tau(\mathcal{R}_j)$ *is given by,*

$$\tau(\mathcal{R}) = \left(\sum_{j=1}^{Q} \frac{p_j}{\tau(\mathcal{R}_j)} \right)^{-1}$$

where p_j *is the probability that a source resides in region* $\mathcal{R}_j$.

Proof. First note that the lifetime bound (not normalized) is obtained by dividing the total energy available at the start by the expected rate of dissipation. Hence, we

have,

$$t(\mathcal{R}_j) = \frac{E(\mathcal{R}_j)}{P(\mathcal{R}_j)} \quad \Rightarrow \quad \tau(\mathcal{R}_j) = \frac{1}{P(\mathcal{R}_j)}$$

where $E(\mathcal{R}_j)$ denotes the initial energy in region $\mathcal{R}_j$. Our task is simply to express the bound of the entire network in terms of the available parameters, which we do as follows,

$$\begin{aligned} t(\mathcal{R}) &= \frac{E(\mathcal{R})}{P(\mathcal{R})} \\ \frac{t(\mathcal{R})}{E(\mathcal{R})} &= \left(\sum_{j=1}^{Q} p_i P(\mathcal{R}_j) \right)^{-1} \\ \tau(\mathcal{R}) &= \left(\sum_{j=1}^{Q} \frac{p_i}{\tau(\mathcal{R}_j)} \right)^{-1} \end{aligned}$$

□

As an illustration, consider a source residing equiprobably along a line with the basestation's (i.e. B's) projection on S_0S_1 lying *within* the segment rather than outside it (fig. 7-14).

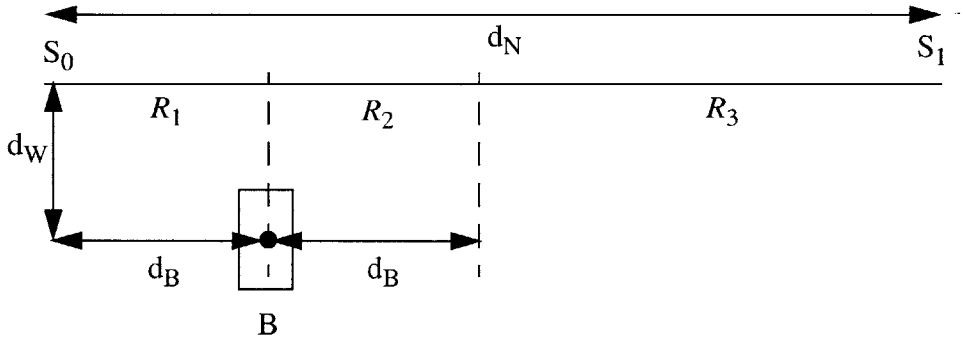


Figure 7-14: Bounding lifetime when source moves along a line, but with basestation "between" S_0S_1.

While the lifetime expression derived in section 5.B (for sources residing on a line) can't be used directly, it can be used via partitioning $\mathcal{R}$ into three regions as shown in figure 7-14 thus,

$$\frac{d_N}{\tau(\mathcal{R})} = \frac{d_B}{\tau(\mathcal{R}_1)} + \frac{d_B}{\tau(\mathcal{R}_2)} + \frac{d_N - 2d_B}{\tau(\mathcal{R}_3)} \tag{7.23}$$

Note that $\tau(\mathcal{R}_1)$ $(=\tau(\mathcal{R}_2))$ and $\tau(\mathcal{R}_3)$ can both be obtained using (7.16).

7.7 Factors Affecting Practical Tightness of Bounds

Throughout this chapter, we have emphasized the "near-tightness" of these bounds. We now discuss the validity and tightness of these bounds under the influence of various practical concerns. The first of these concerns is topology. In an effort to be fundamental, our bounds ignore the topology of the network. Hence, bounds derived in the chapter are maximized over all achievable network topologies. A randomly deployed network will seldom have the near-ideal topologies needed to approach the bound. However, if the network has high enough density and is Poisson distributed (i.e. the number of nodes in a certain region is proportional to the area of that region), then lifetime bounds for such networks can be expected to approach those derived. Next, we expect the source to obey a certain location distribution. If a real world source deviates significantly, then the achievable lifetime will be lower than the derived bounds. Finally, achieving these bounds requires global co-ordination and information, which has finite energy costs in real-world networks. This will also contribute to the gap between achievable lifetime and the derived bounds. A related issue is the energy loss incurred in accessing a multi-user channel.

For all these reasons, it is unlikely that real world ad-hoc networks can approach the derived bounds closely. Note however that none of these factors lead to an invalidation of the bounds i.e. the bounds stay valid, but tend to be optimistic. In the next chapter, we factor in topology, source movement, multiple sources and desired quality thus closing the gap between achievable lifetimes and bounds. However, the gap due to absence of global information and multiple-access issues still remains.

Chapter 8

Optimal Collaborative Strategies

In this chapter, we formalize the notion of a collaborative strategy and then present computationally efficient algorithms for determining the strategy that maximizes lifetime.

8.1 Key Formalisms

8.1.1 Role Assignments

The network we will use to illustrate the formalisms is shown in figure 8-1. Consider then, that we want to maximize the active lifetime of this network with the source in the position illustrated. Assume first that we require at least one node to sense. Note that the *set of potential sensors*, S, in this case is {1,2}. Our only means of influencing the lifetime of the network is by changing the roles that nodes assume with time. An assignment of roles to nodes is called a *role assignment*. Hence, there are M^N role assignments in a non-aggregating network with N nodes and M basic roles. When there are restrictions on the roles that certain nodes can assume, only some role assignments are *valid*. In the example above, the sensing role is restricted to nodes 1 and 2 since these are the only two nodes in the circular region of source observability. The number of *valid role assignments* is thus $4^2 3^{(15-2)}$ (since 2 nodes are allowed all roles and the remaining 13 are allowed only non-sensing roles). As one would

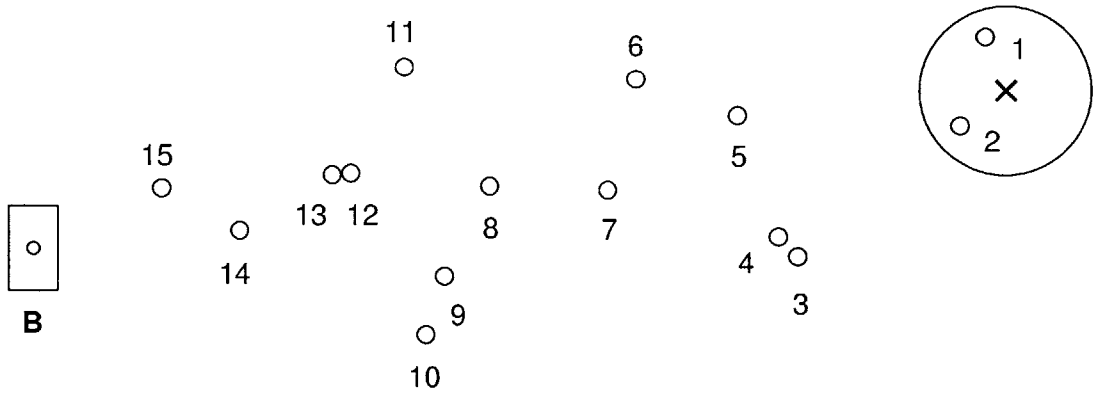


Figure 8-1: A sensor network (the source is denoted by a $\times$).

expect, not all valid role assignments lead to the network being active i.e. obeying its contractual obligation of sensing data and then relaying it back to the basestation. Figure 8-2 shows one such assignment. A valid role assignments that leads to an

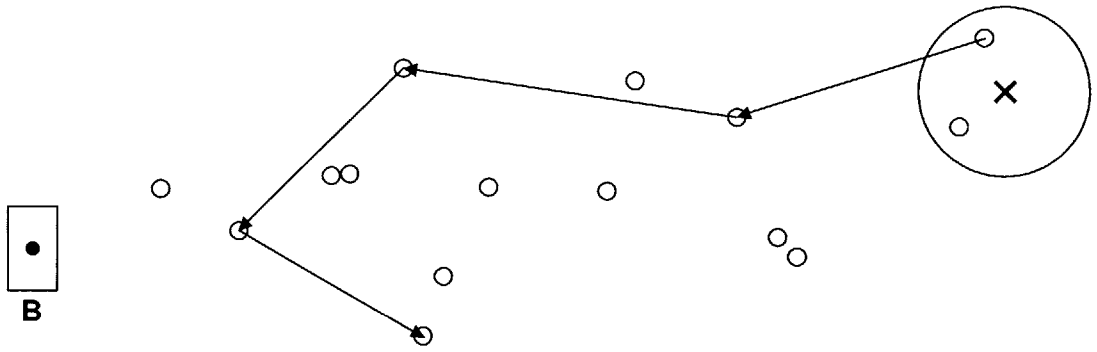


Figure 8-2: A valid but *inactive* role assignment (since sensed data is not reaching the basestation).

active network is termed an *active role assignment.* Figure 8-3 shows an active role assignment. Finally, an active role assignment is termed *feasible iff* it is non-redundant i.e. no non-sleeping node can be put to sleep without rendering the assignment inactive (note that the example active network in figure 8-3 was infeasible). Every active role assignment can be made non-redundant to yield (at least) one feasible role assignment (FRA). If a feasible assignment involves aggregation, we call it an *aggregating FRA* else we call it a *non-aggregating FRA*. Examples of FRAs are shown in figures 8-4 and 8-5. We denote by F the set of all feasible role assignments (FRAs).

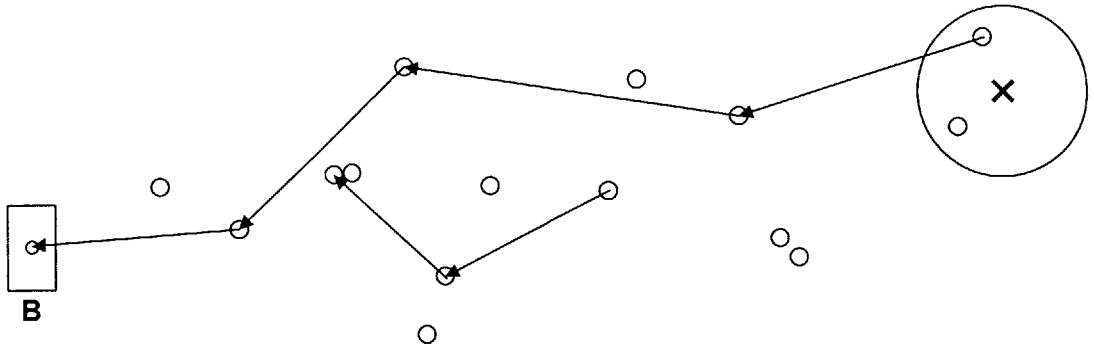


Figure 8-3: An active (but infeasible) role assignment.

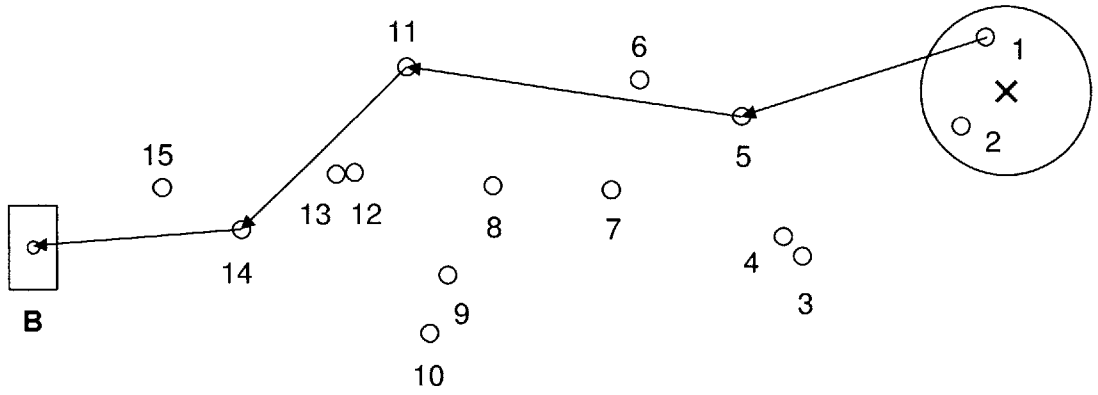


Figure 8-4: A feasible role assignment (FRA) written as $1 \rightarrow 5 \rightarrow 11 \rightarrow 14 \rightarrow B$. Note that this is a non-aggregating FRA.

The concept of FRAs is central to maximizing the lifetime of sensor networks. This is due to the following observation which follows from our definition of FRAs:

Lemma 8.1 (Completeness of F). *A sensor network that employs only role assignments present in F* can *attain a lifetime that is no less than that achieved by employing a superset of F.*

Proof. Assume the contrary i.e. using role assignments not in F allows one to achieve a lifetime strictly greater than that possible by using FRAs alone. Then the network sustained a role assignment, $r \notin F$ at some time instant. Note that r must be active, else the network would violate its contract for the time for which r was sustained. But any active r can be replaced by a FRA, say s, such that $s \in F$ and s is at least as energy efficient as r. Similarly replace all role assignments not in F by FRAs.

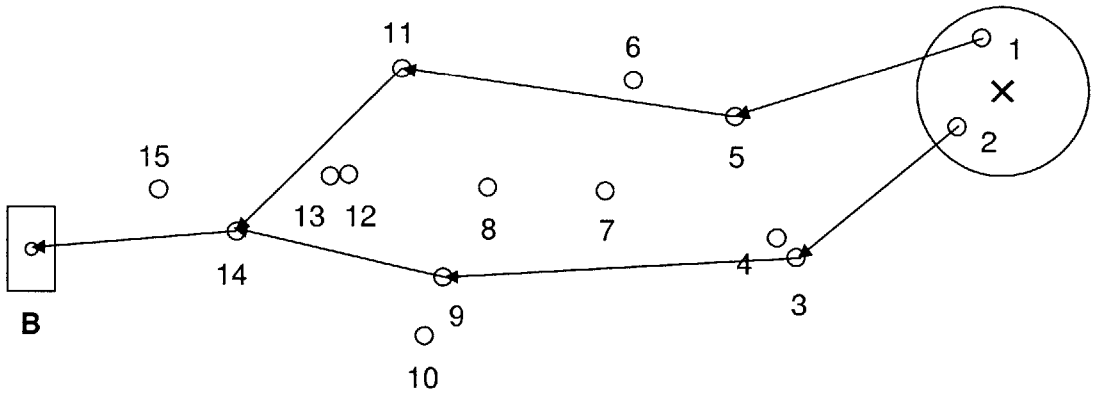


Figure 8-5: A feasible role assignment written as $1 \rightarrow 5 \rightarrow 11 \rightarrow \mathbf{14} \rightarrow B; 2 \rightarrow 3 \rightarrow 9 \rightarrow \mathbf{14} \rightarrow B$. This is an aggregating FRA with the aggregating node (here 14) in boldface.

It is clear that we can achieve a lifetime that is no less than the original while only employing FRAs, which is a contradiction. □

8.1.2 Collaborative Strategies

As mentioned earlier, the problem of maximizing lifetime is really one of finding the optimal *collaborative strategy*[1] i.e. the optimal manner for nodes to cooperate such that the sensor network fulfils its contract. We have already seen that it is sufficient to consider fulfilment of this contract using only FRAs. Hence a collaborative strategy is characterized by a sequence of FRAs each of which is sustained for some specified time. A possible collaborative strategy for the network in figure 8-1, assuming that only one sensor needs to sense is:

$$\{(1 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 12 \rightarrow 13 \rightarrow 14 \rightarrow 15 \rightarrow B, 1.923 \text{ sec}),$$
$$(1 \rightarrow 5 \rightarrow 7 \rightarrow B, 0.089 \text{ sec}),$$
$$(2 \rightarrow 5 \rightarrow 12 \rightarrow 13 \rightarrow 15 \rightarrow B, 123.5 \text{ sec})\}$$

[1]Some authors reserve the word "collaboration" to mean data aggregation. In this work, *any* cooperation between nodes is termed a collaboration.

Each of the elements is a *time-annotated FRA*. Not all collaborative strategies are practically feasible. For example node 2 might be drained after the first FRA above has been sustained for a second. Hence the first FRA cannot be sustained for 1.923 seconds due to the death of node 2. Clearly, the feasibility of a collaborative strategy is dependent on the initial energy in the nodes and the power dissipation incurred when a certain FRA is sustained. For example, in the absence of energy constraints, i.e. if nodes have infinite energy, all collaborative strategies are trivially feasible.

Definition 8.2 (Feasible Collaborative Strategy). *A collaborative strategy is* feasible *for a given initial energy state of the network* iff *at the termination of the collaborative strategy, no node has a negative residual energy. This is the same as saying that no constituent FRA in a feasible collaborative strategy employs a dead node.*

Feasible collaborative strategies have two properties that give them significant structure.

Corollary 8.3 (Invariance to Permutation). *If a collaborative strategy is feasible, it follows that any permutation of the constituent time annotated FRAs is also feasible.*

Corollary 8.4 (Linearity). *If two consecutive and identical FRAs are sustained for times t_1 and t_2 then this pair can be replaced by a single FRA with a combined time of $t_1 + t_2$ without affecting feasibility. This follows from the linear dependence of energy dissipated on time*[2].

These properties lead to the *canonical form* of a feasible collaborative strategy.

Theorem 8.5 (Canonical Collaborative Strategies). *Consider the set of all FRAs $F = \{r_0, r_1, r_2, ..., r_{|F|-1}\}$ and a given feasible collaborative strategy. Then, there exists a collaborative strategy of the form,*

$$\{(r_0, t_0), (r_1, t_1), ..., (r_{|F|-1}, t_{|F|-1})\}$$

[2] *Note that this dependence is* always *linear for time-invariant systems even when power varies in a complicated non-linear manner with factors like rate of the stream etc.*

which achieves the same lifetime. This strategy is called the canonical form of the given collaborative strategy.

Proof. First permute the given feasible collaborative strategy such that all identical FRAs are consecutive. Then replace every set of consecutive identical FRAs by a single FRA with a time that is the sum of the replaced FRAs. Next insert FRAs that are absent with zero times. Finally permute the time annotated FRAs such that they are in the order dictated by F. □

Since the optimal collaborative strategy is trivially feasible, the significance of the above result is that determining the optimal simply entails determining the t_is i.e. the time for which each FRA is sustained. We need not concern ourselves with determining the order in which FRAs are sustained or if they are split over several time intervals.

8.1.3 Formal Description of the Maximum Lifetime Problem

Given :

1. $G(V, E)$: An undirected, weighted graph that captures the topology of the network and the position of the source. The weights correspond to the Euclidean distance. The number of nodes is denoted by N. It follows that $|V| = N + 2$ (N nodes, a basestation and the source) and $|E| = \binom{|V|}{2}$.
2. S : The set of potential sensors. The cardinality of this set is denoted by m (i.e. $|S| = m$).
3. k : The minimum number of nodes that must sense. Note that $k \leq m$.
4. $e \in (\mathbb{R}^+)^N$: The vector describing the initial energy in the N nodes. We denote the initial energy of node i by e_i.
5. F: The set of feasible role assignments, $F = \{r_0, r_1, r_2, ..., r_{|F|-1}\}$.
6. p : The power function,

$$p : V \times F \rightarrow \mathbb{R}^+$$

In other words, for any given pair (n, r) where n is a node and r a FRA, p returns the power consumption of n when the network sustains r.

Form of the Solution : A *feasible* collaborative strategy,

$$C = \{(r_i, t_i) | r_i \in R, t_i \in \mathbb{R}^+\}$$

Measure of the Solution : The lifetime of the network,

$$t = \sum t_i$$

Problem : Find a solution that maximizes the measure.

8.2 Determining the Optimal Strategy Via Linear Programming

The formulation posed above naturally suggests a solution via the linear program in table 8.1. The first set of constraints is obvious - it makes no physical sense to

---

Objective :

$$\max \qquad t = \sum_{i=1}^{|F|} t_i$$

where t_i corresponds to the time for which FRA r_i is sustained.

Constraints :

$$t_j \geq 0 \quad : \quad 1 \leq j \leq |F| \tag{8.1}$$

$$\sum_{j=1}^{|F|} p(i, r_j) t_j \leq e_i \quad : \quad 1 \leq i \leq N \tag{8.2}$$

Table 8.1: Linear program for determining optimal collaborative strategy

sustain a FRA for negative time. The second set of constraints are energy conservation

constraints, one for each node, with the LHS denoting the energy consumed and the RHS the initial energy the node started out with. The solution of this linear program is then the solution of our optimal collaboration problem.

8.3 Illustrations

We now illustrate this technique using a simple example network shown in figure 8-6. This network is collinear i.e. all nodes and the basestation lies on a line. To keep the

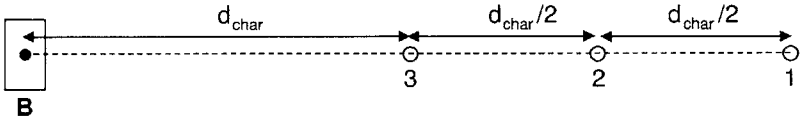


Figure 8-6: A collinear 3-node network with node 1 as the assigned sensor (d_{char} is 134 meters).

example tractable, the set of potential sensors has a single node - $\{1\}$. In other words, we do not have any choice in picking the sensor. Also, a single sensor precludes any aggregation. All nodes start out with energy equal to $2\alpha_1$ which, using the typical values introduced earlier is 180 nJ[3]. What collaborative strategy will maximize the lifetime of this network? Our first task is to characterize the set F. The following simple counting results do just that.

Lemma 8.6.

$$S_N = \sum_{k=0}^{N} \binom{N}{k} k! = \lfloor eN! \rfloor$$

[3]Typical nodes have energies in the 100s-1000s of Joules (depending on form factor). To put this in perspective, a typical size AA cell packs in about 10 kJ of energy. The reason for using such small initial energies in our examples is to keep the lifetime close to a unit i.e. a second. Hence the reader should not be alarmed at sub-second network lifetimes!

Proof.

$$\begin{aligned}
\lfloor eN! \rfloor &= \left\lfloor N! \left(\frac{1}{0!} + \frac{1}{1!} + \cdots + \frac{1}{N!} + \frac{1}{(N+1)!} + \cdots \right) \right\rfloor \\
&= \left\lfloor S_N + N! \left(\frac{1}{(N+1)!} + \frac{1}{(N+2)!} + \cdots \right) \right\rfloor \\
&= S_N + \left\lfloor N! \left(\frac{1}{(N+1)!} + \frac{1}{(N+2)!} + \cdots \right) \right\rfloor \\
&= S_N + \lfloor U_N \rfloor \\
&= S_N
\end{aligned}$$

since,

$$\begin{aligned}
U_N &= \frac{1}{(N+1)} + \frac{1}{(N+1)(N+2)} + \frac{1}{(N+1)(N+2)(N+3)} + \cdots \\
&< \frac{1}{N} + \frac{1}{N^2} + \frac{1}{N^3} + \cdots = \frac{1}{N-1}
\end{aligned}$$

□

Theorem 8.7. *A N node network with a pre-assigned sensor has $|F| = \lfloor e(N-1)! \rfloor$.*

Proof. First pick the number of intervening hops, say k. Note that $0 \leq k \leq N-1$. Now pick the k hops which can be done in $\binom{N}{k}$ ways. Finally, there are $k!$ permutations for k given hops. Hence the total number of FRAs is simply,

$$|F| = \sum_{k=0}^{N-1} \binom{N-1}{k} k! \tag{8.3}$$

which is simply $\lfloor e(N-1)! \rfloor$ by lemma 8.6. □

In the case of collinear networks, we can reduce this number significantly by replacing all "self crossing" FRAs with non-self-crossing ones as shown in figure 8-7. The rationale behind eliminating self crossing FRAs is that they can always be replaced by a more efficient non-self-crossing FRA and hence are provably inferior. It is easy to see that if we restrict ourselves to non-crossing FRAs, $|F|$ is simply 2^{N-1} since the $k!$ permutation factor in 8.3 disappears. For our 3 node network, the 4

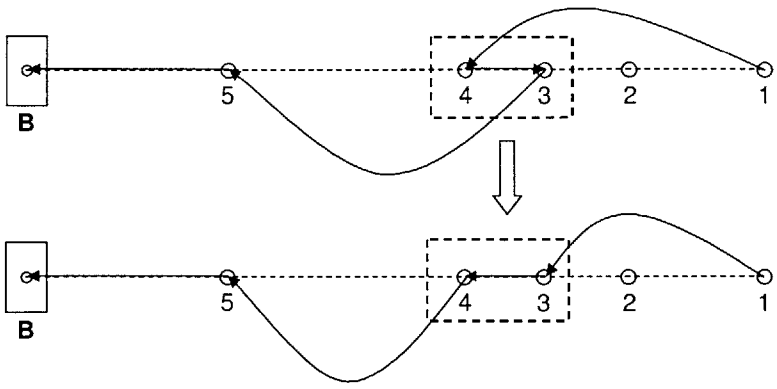


Figure 8-7: A *self-crossing* FRA can always be substituted with a more energy-efficient non-self-crossing FRA.

FRAs are given by,

$$F = \{r_1, r_2, r_3, r_4\}, \text{ where,}$$

$$r_1 : 1 \rightarrow B$$

$$r_2 : 1 \rightarrow 2 \rightarrow B$$

$$r_3 : 1 \rightarrow 3 \rightarrow B$$

$$r_4 : 1 \rightarrow 2 \rightarrow 3 \rightarrow B$$

Note the absence of self-crossing FRAs like $1 \rightarrow 3 \rightarrow 2 \rightarrow B$. To determine the optimal collaborative strategy, we simply have to determine the time for which each of these role assignments should be sustained i.e. we must determine t_1, t_2, t_3 and t_4. Plugging in the requisite values into the linear program (table 8.1) above we find that for maximum lifetime,

$$t_1 = 0 \text{ sec}$$

$$t_2 = 0.375 \text{ sec}$$

$$t_3 = 0.375 \text{ sec}$$

$$t_4 = 0.625 \text{ sec}$$

which yields a total lifetime of 1.375 sec. We hope that it is immediate to the reader

that this lifetime *cannot* be improved by, say, using any other formalism, as long as nodes are restricted to playing one of the four basic roles at any point in time.

Note that the linear programming formulation allows one to tackle arbitrarily complex sets of role assignment possibilities, as the next two examples illustrate. Consider for instance, the same network, but with $S = \{1, 2\}$ and $k = 2$ i.e. both 1 and 2 must sense and hence we have no choice in picking the sensors (our next example will eliminate this restriction). We now allow aggregation. This leads to,

$F = \{r_i | 1 \leq i \leq 12\}$, where,

$r_1 : 1 \rightarrow B; 2 \rightarrow B$

$r_2 : 1 \rightarrow 2 \rightarrow B; 2 \rightarrow B$

$r_3 : 1 \rightarrow 3 \rightarrow B; 2 \rightarrow B$

$r_4 : 1 \rightarrow 2 \rightarrow 3 \rightarrow B; 2 \rightarrow B$

$r_5 : 1 \rightarrow B; 2 \rightarrow 3 \rightarrow B$

$r_6 : 1 \rightarrow 2 \rightarrow B; 2 \rightarrow 3 \rightarrow B$

$r_7 : 1 \rightarrow 3 \rightarrow B; 2 \rightarrow 3 \rightarrow B$

$r_8 : 1 \rightarrow 2 \rightarrow 3 \rightarrow B; 2 \rightarrow 3 \rightarrow B$

} Non-aggregating FRAs

$r_9 : 1 \rightarrow \mathbf{2} \rightarrow B; \mathbf{2} \rightarrow B$

$r_{10} : 1 \rightarrow \mathbf{2} \rightarrow 3 \rightarrow B; \mathbf{2} \rightarrow 3 \rightarrow B$

$r_{11} : 1 \rightarrow \mathbf{3} \rightarrow B; 2 \rightarrow \mathbf{3} \rightarrow B$

$r_{12} : 1 \rightarrow 2 \rightarrow \mathbf{3} \rightarrow B; 2 \rightarrow \mathbf{3} \rightarrow B$

} Aggregating FRAs

The reader may wish to verify that this is an exhaustive list of all non-self-crossing FRAs that allow aggregation. The first 8 FRAs are non-aggregating. In the first FRA (r_1) for instance, the data from node 1 (which is sensing) is routed straight to the basestation and the same is true for node 2 (also sensing). Consider now an aggregating FRA, say r_{12}. Here, node 3 is the aggregator[4]. Node 1 is sending its raw sensor data to 3 via 2 while 2 is sending its data straight to 3. 3 is aggregating these two streams into one and sending the aggregated stream straight to the basestation.

[4]The aggregator has been emphasized in aggregating FRAs

What is the strategy that maximizes lifetime in this example? Again, solving the linear program, we obtain,

$$t_6 = 0.3192 \text{ sec}$$

$$t_8 = 0.8938 \text{ sec}$$

$$t_{10} = 0.3192 \text{ sec}$$

with the other t_is either zero or negligible. This yields a total lifetime of close to 1.5322 sec. We have assumed here that the energy cost of aggregation is negligible compared to the cost of communication. Note that inspite of this, the optimal strategy aggregates only about 75% of the time and not always. Note also that 2 is used as an aggregator three times as often as node 3 (because 2 has the advantage of not having to transmit its data when it acts as both sensor and aggregator).

Consider next the network shown in figure 8-8. This is similar to the network we

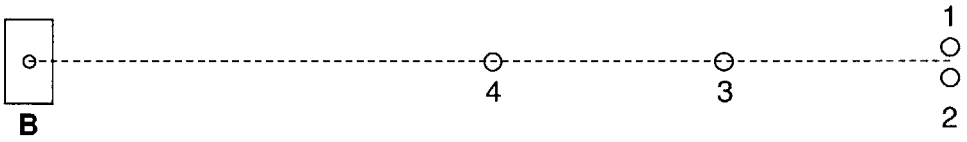


Figure 8-8: The network in figure 8-6 but with an extra node as potential sensor.

worked with in the previous examples except that we have an additional node and $S = \{1, 2\}$ with $k = 1$ i.e. we have two nodes that can act as sensors but any time,

we need only one to be active. This leads to the following FRAs,

$$
\begin{array}{ll}
r_1 : 1 \rightarrow B & \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} \text{Node 1 senses} \\
r_2 : 1 \rightarrow 2 \rightarrow B & \\
r_3 : 1 \rightarrow 3 \rightarrow B & \\
r_4 : 1 \rightarrow 4 \rightarrow B & \\
r_5 : 1 \rightarrow 2 \rightarrow 3 \rightarrow B & \\
r_6 : 1 \rightarrow 2 \rightarrow 4 \rightarrow B & \\
r_7 : 1 \rightarrow 3 \rightarrow 4 \rightarrow B & \\
r_8 : 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow B & \\
r_9 : 2 \rightarrow B & \\
r_{10} : 2 \rightarrow 1 \rightarrow B & \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} \text{Node 2 senses} \\
r_{11} : 2 \rightarrow 3 \rightarrow B & \\
r_{12} : 2 \rightarrow 4 \rightarrow B & \\
r_{13} : 2 \rightarrow 1 \rightarrow 3 \rightarrow B & \\
r_{14} : 2 \rightarrow 1 \rightarrow 4 \rightarrow B & \\
r_{15} : 2 \rightarrow 3 \rightarrow 4 \rightarrow B & \\
r_{16} : 2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow B &
\end{array}
$$

Again, solving the linear program yields the following optimum collaborative strategy,

$$
\begin{aligned}
t_1 &= 0.123 \text{ sec} \\
t_2 &= 0.3075 \text{ sec} \\
t_3 &= 0.5 \text{ sec} \\
t_9 &= 0.123 \text{ sec} \\
t_{10} &= 0.3075 \text{ sec} \\
t_{11} &= 0.5 \text{ sec}
\end{aligned}
$$

with the other t_is equal to zero. This yields a total lifetime of close to 1.816 sec. Note that the optimal solution reflects the fact that nodes 1 and 2 are topologically

indistinguishable (i.e. the FRAs used by 1 and 2 are perfect analogues of each other as one would expect from the symmetry in the network). Note also that the collaborative strategy in the two node case is significantly different from that in the one node case above although the topology of the non-sensing nodes is exactly the same.

8.4 Polynomial Time Algorithms

While the linear programming framework above works perfectly and is the most general and powerful, it suffers from the drawback that it can be computationally burdensome. Using the Ellipsoid method a linear program can be solved in a time that is polynomial in the number of constraints and the *size* of the program[5][43]. If we use the setup above to discover the optimal collaborative strategy, our linear program will take a time that is polynomial in the number of FRAs and nodes. The number of FRAs however is exponential in the number of nodes for practical problems of interest. Hence, the linear programming formulation above takes a time that is *exponential* in the number of nodes. This severely restricts the utility of the formulation. Interestingly however, we now show that for a broad class of role assignment problems, we can get arbitrarily close to the optimal collaborative strategy while in a time that is *polynomial* in the number of nodes. Our approach to computationally efficient discovery of collaborative strategies is motivated by prior work in the area of energy efficient multi-hop routing in ad-hoc networks [11, 12]. Specifically, we exploit the fact that if the role assignment problem can be transformed to a *network-flow* problem, there might exist a polynomial time approach to determining the optimal collaborative strategy.

To motivate this idea, consider again the simple, pre-assigned sensor, non-aggregating network first shown in figure 8-6. One way to view the optimal collaborative strategy is that r_2 is sustained for 0.375 sec, r_3 for 0.375 sec and r_4 for 0.625 sec. In other words, r_2 and r_3 are each responsible for shipping $\frac{3}{11}$ of the data while r_4 ships the remaining $\frac{5}{11}$. Furthermore, we can say now that link $1 \rightarrow 2$ now carries $\frac{8}{11}$ of the data

[5]The size of a linear program is the number of bits needed to represent it.

of which $\frac{3}{11}$ is due to r_2 and $\frac{5}{11}$ is due to r_4. Similarly link $1 \rightarrow 3$ is responsible for shipping $\frac{3}{11}$, all of which is due to r_3. Figure 8-9, demonstrates this transformation. We have transformed the role assignment view to the network flow view:

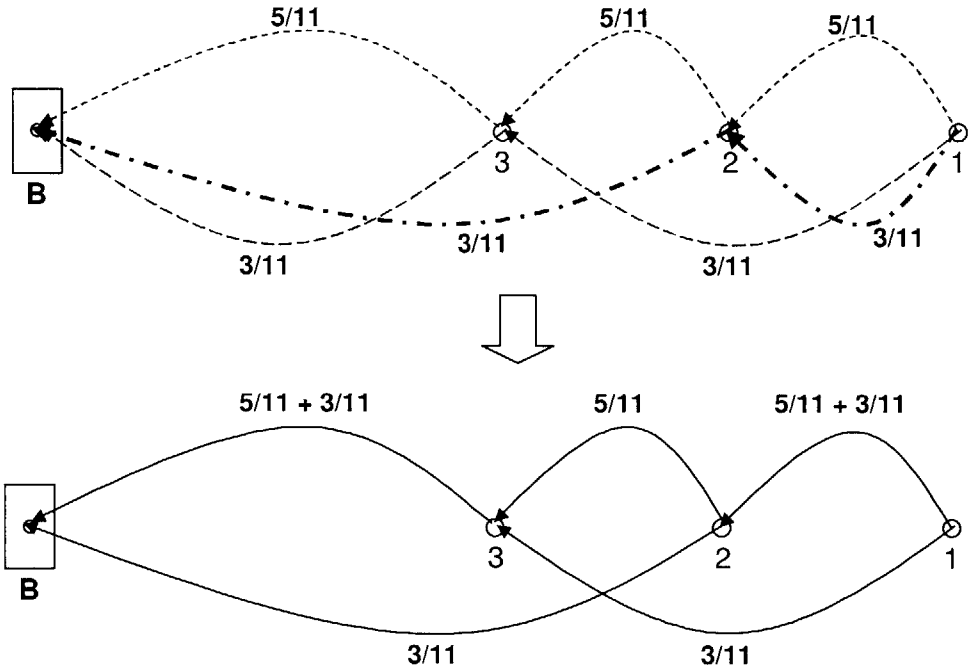


Figure 8-9: Deriving a flow view from a role assignment view.

$$\left\{ \begin{array}{l} t_1 = 0 \text{ sec} \\ t_2 = 0.375 \text{ sec} \\ t_3 = 0.375 \text{ sec} \\ t_4 = 0.625 \text{ sec} \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} f_{12} = \frac{8}{11} \\ f_{13} = \frac{3}{11} \\ f_{1B} = 0 \\ f_{23} = \frac{3}{11} \\ f_{2B} = \frac{5}{11} \\ f_{3B} = \frac{6}{11} \end{array} \right\}$$

where f_{ij} is the flow from node i to node j. Note that we are justified in calling the above view a flow because it satisfies the following properties expected of any valid flow:

1. **Non-negativity** of flow.

2. **Conservation** of flows at all nodes but the sensor. In other words the total

flow out of a node is the same as the total flow into a node.

It is fairly straightforward to see that for the class of networks with an assigned sensor (as in the flow construction example above), every flow view that is constructed from a collaborative strategy will have these properties i.e. is valid. We now ask the reverse question - can one always derive a role assignment from a flow[6]?

Theorem 8.8. *Consider the class of sensor networks with a pre-assigned sensing node. Then, for every valid flow, there exists an equivalent role assignment. In other words, if we constructed a flow view starting with this role assignment, we would get back the given flow.*

Proof. We draw on a well known result in flow-theory - every flow can be expressed as a sum of cycles and paths with non-negative weights [1]. Furthermore all the paths are from the source to the sink, which in our case is node 1 and B respectively. It is easy to see that these paths correspond to FRAs and their flows are proportional to the time for which the FRA is sustained[7]. □

Corollary 8.9 (Cardinality of the Optimal Collaborative Strategy). *Although there are an exponential number of FRAs, only a polynomial number of them need to be sustained for non-zero time to achieve maximum lifetime.*

Corollary 8.10. *A network flow can be converted to an equivalent collaborative strategy in polynomial time.*

Corollary 8.11. *If the optimal network flow can be derived in a time that is polynomial in the number of nodes then the optimal collaborative strategy can also be determined in polynomial time.*

In the present context, the corollary applies only to networks with a single assigned sensor. We will soon show that it holds for much more complex scenarios. But first - can we construct the optimal network flow in polynomial time? This is indeed possible. Consider the program in table 8.2. The first two constraints simply state

[6]From now on, when we use the word flow, the fact that the three properties are obeyed will be assumed.

[7]The constant of proportionality is determined if the lifetime achieved is known.

Objective :

$$\max \quad t$$

Constraints :

Non-negativity of flow:

$$f_{ij} \geq 0 \tag{8.4}$$

Conservation of flow:

$$\sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si} = \sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id} : \quad i \in [2,N] \tag{8.5}$$

Total sensor flow:

$$\sum_{d\in[2,N+1]} f_{1d} - \sum_{s\in[2,N+1]} f_{s1} = 1 \tag{8.6}$$

Energy constraints:

$$t\Bigg(\sum_{\substack{d\in[1,N+1]\\ d\neq i}} p_{tx}(i,d) f_{id} + \sum_{\substack{s\in[1,N+1]\\ s\neq i}} p_{rx} f_{si} + \underbrace{p_{sense}}_{\textit{For node 1 only}} \Bigg) \leq e_i : \quad i \in [1,N] \tag{8.7}$$

Table 8.2: Program for calculating optimal network flow for non-aggregating networks with a pre-assigned sensor.Without any loss of generality the pre-assigned sensor is labelled 1. Also, the basestation is always labelled $N+1$.

that this is a valid flow. The third constraint normalizes the flow out of the sensor to 1. This ensures that if the flows above can be sustained for time t, then t is simply the lifetime of the network. The last condition states that the total energy drained over the lifetime of the network be no greater than the initial energy present. Note that we use p_{tx} and p_{rx} rather than p for the power function. This program is *not* linear since an unknown (lifetime t) is multiplying another (flows f_{ij}). Consider however a binary search based strategy to discover the optimal lifetime. We invoke the program with a guess t_{guess}. With t fixed to t_{guess} the program is linear and can be solved in a time that is polynomial in the number of nodes. Depending on the success or failure of the program, we revise t_{guess} and continue using the binary search technique. Note that our earlier development on the upper bound gives us an initial

guess to start with. Using the binary search technique, in l trials, the error between the optimal lifetime and our guess as a fraction of the upper bound is reduced to 2^{-l}. To resolve the optimum lifetime to within l bits, we need a time that is proportional to l and polynomial in the number of nodes. In reality l is a small constant - say 12. Increasing l greater than this is usually meaningless since the error in energy models is the limiting factor. Thus, we have managed to develop a computational strategy that can discover a network flow that is arbitrarily close to optimal in a time that is polynomial in the number of nodes.

We now present increasingly more sophisticated classes of networks and show how careful manipulation of flows allows similar results.

8.5 Generalization: Set of Potential Sensors

While strategies with nodes pre-assigned as sensors are good as illustrative examples, they can be significantly removed from the optimal solution. The more practically useful formulation is one where there is a *set* of potential sensors S and the user specifies that k sensors must be active at any time (with $1 \leq k \leq m = |S|$). In most cases, the set S is automatically determined by the location of the source and its observability radius. However, the user may explicitly specify this set. We have already seen an example of this kind earlier where S was a 2 node set and k was 1. There we used the original linear program (table 8.1), to illustrate how this class of problems may be solved. The problem, as we have mentioned earlier, is the exponential run-time. Two modifications to the program in table 8.2 which handles only one sensor will allow us to capture this more general case of sensor sets using the following program:

Objective :

$$\max \quad t \tag{8.8}$$

Constraints :

Non-negativity of flow:

$$f_{ij} \geq 0 \tag{8.9}$$

Flow conservation:

$$\sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si} = \sum_{\substack{d\in[1,N+1]\\ d\neq j}} f_{id} : \quad i \in [1,N], i \notin S \tag{8.10}$$

Overall flow from sensor set (S):

$$\sum_{i\in S} \left(\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si} \right) = k \tag{8.11}$$

Non-consumption of flow in sensors:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si} \geq 0 : \quad i \in S \tag{8.12}$$

Limit on total flow from any single sensor:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si} \leq 1 : \quad i \in S \tag{8.13}$$

Energy constraints:

$$t\left\{ \sum_{\substack{d\in[1,N+1]\\ d\neq i}} p_{tx}(i,d) f_{id} + \sum_{\substack{s\in[1,N+1]\\ s\neq i}} p_{rx}(s,i) f_{si} + \underbrace{p_{sense} \left(\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si} \right)}_{\textit{Term to be included for potential sensors only}} \right\} \leq e_i : \quad i \in [1,N] \tag{8.14}$$

The first modification is obvious - instead of unit from one sensor, we now desire the net sensor flow out of S to be k. Hence 8.6 has been modified to 8.11. Simply

equating the *total* flow out of S to be k does not guarantee that the resulting flow solution will have a meaningful equivalent collaborative strategy. There are two issues. The first is - what if a sensor is actually *consuming* flow? Recall that we have imposed flow conservation only on non-sensors and have imposed no condition on the net flow of any *individual* sensor. This concern is easily resolved by imposing constraint 8.12. While this precludes the menace of a sensor that is consuming, the reverse situation - that of a sensor generating too much volume is also a problem. Consider, for instance, a case where $m = |S|$ is 5 and k is 2. Solving the program above might yield a solution where one of the nodes in S accounts for a flow of 2. Clearly, such a flow cannot be translated to a collaborative strategy where at least two nodes sense at all given times. The problem is that one node has *monopolized* the flow. To prevent monopolies, we restrict the total sensor flow from any node in S to be 1 (constraint 8.13). While this is clearly necessary, is it sufficient to guarantee the existence of an equivalent strategy? In other words, given a set of m flows, all of which are less than 1 and add up to k, can we guarantee that there exists a strategy where exactly k of m sensors are active at any time? We now prove that this is indeed possible.

Lemma 8.12 (The k of m Coloring Problem). *Given $m, k \in \mathbb{N}$ with $k \in [1, m]$. There are k line segments of unit length aligned one below the other, in parallel, as illustrated in figure 8-10. We have m colors available to color these segments with*

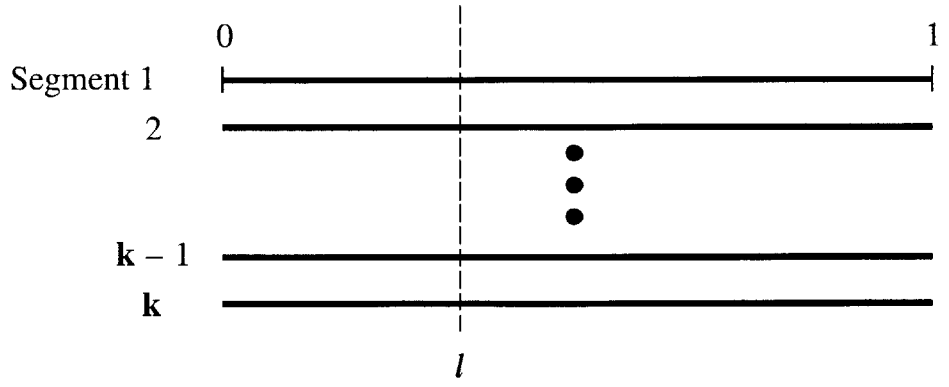


Figure 8-10: The k segments coloring problem. We want to color these segments using certain specified quantities of $m \geq k$ colors such that any line l perpendicular to them does not intersect the same color twice.

color j available in a non-negative quantity x_j, with $x_j \leq 1$. The total *quantity of color available is k. Then we can always color these segments such that any line drawn perpendicular to and through these segments always intersects unique colors.*

Proof. We present a simple constructive proof. The notation $j : [a, b]$ will refer to the segment $[a, b]$ on line segment $j \in [1, k]$ with $j : l$ referring to the point $j : [l, l]$. Then, consider the following coloring strategy:

Step I Initialize variable t to 1 and current_loc to $1 : 0$.

Step II Fill [current_loc, current_loc+x_t] with color t while rolling over properly to the next segment if needed.

Step III Update current_loc to current_loc+x_t and t to $t + 1$.

Step IV Stop if t is $m + 1$ else go to **Step II**.

This algorithm clearly terminates after precisely m iterations. Since $\sum x_j = t$ all the segments have been colored with no color left when it terminates. To prove that any perpendicular cannot intersect the same color, assume otherwise i.e. there exists a perpendicular that intersects color j. This clearly implies $x_j > 1$, which is a contradiction. □

Corollary 8.13 (Existence of Equivalent Strategy for Program 8.8-8.14). *There exists a schedule with exactly k of m sensors active at any given instant if and only if the flows satisfy constraints 8.11 and 8.13.*

Proof. The *only if* part is trivially proved by contradiction: if the total sensor flow is not k then k sensors could not have been active throughout the lifetime. Again, if any sensor had a flow greater than 1 and it is possible to find precisely k nodes sensing over the lifetime, then the total flow must be greater than k which we have just shown to be impossible. The more interesting *if* part follows directly from lemma 8.12. □

Corollary 8.14. *Given m sensors, of which k must be active at any time, allows $\binom{m}{k}$ different ways of satisfying the requirement. While $\binom{m}{k}$ is in general* exponential in m,

there always exists a collaborative strategy which requires at most $m + 1$ *possibilities, which is* linear in m.

Proof. Lemma 8.12 shows a construction where no more than m colors are employed. Consequently, if we sweep line l from the left to the right, we will encounter no more than $m + 1$ color changes. □

8.6 Generalization: Aggregation

We have considered non-aggregating networks so far. We now introduce aggregation (while continuing to support specification of arbitrary sensor sets). Once more, we stress the fact that the linear program in table 8.1 is capable of dealing with this class of strategies although the resulting, exponential, computational complexity is unacceptable. Hence, we develop a transformation to network flows again. In non-aggregating network, every sensed bit undergoes only one operation - routing - before it lands at the basestation. Hence, at any given point, there is only one kind of data or, to use flow jargon, a *single commodity* flowing through the network. Consider an aggregating network with, say, three, potential aggregators (specified by the user) - nodes 4, 6 and 7. There are several kinds of bits floating in the network:

1. Raw sensor bits which will never be aggregated.
2. Raw sensor bits *destined* for aggregation at:
   (a) Node 4.
   (b) Node 6.
   (c) Node 7.
3. Bits produced as a *result* of aggregation at:
   (a) Node 4.
   (b) Node 6.
   (c) Node 7.

As will become clearer in a while, the first and last class of bits need not be classified as different commodities since neither is destined for aggregation (recall that an aggregated stream is never aggregated again). We lump these classes into a single class called "not destined for aggregation" or *unagg* for brevity. The other class is "destined for aggregation" or *agg* for short and we distinguish it further into three commodities - one each for the three potential aggregators. Thus we now have a total number of 4 commodities. In the general case where the user specifies a set of potential aggregators P, we have $1 + |P|$ commodities. We now have on our hands, a *multi-commodity flow* problem[8]. In the program for this case, which follows, flow $f_{ij,z}$ indicates flow on the link $i \rightarrow j$ carrying commodity z i.e. destined for aggregation at node z. We use $z = 0$ for the case when the flow will not be aggregated. In other words, $z = 0$ is used for the unagg commodity. Given a node, any flow originating from or terminating in it is said to be *related* to the node if the commodity it carries is destined for the node. Hence, $f_{ij,z}$ is related to node i *iff* $z = i$ and to node j *iff* $z = j$. A flow that is not related to a node is said to be *unrelated* to it. Note that we reserve the use of related or unrelated for agg commodities only. It follows from this definition that all flows into and out of a node that is not a potential aggregator, i.e. not in P, are unrelated to it.

Objective :

$$\max \quad t \tag{8.15}$$

General Constraints :

Non-negativity of flow:

$$f_{ij,z} \geq 0: \quad i \in [1, N+1], j \in [1, N+1], z \in 0 \cup P \tag{8.16}$$

[8]When no aggregators are specified (i.e. $P = \phi$, $|P| = 0$), this simply reduces to a single commodity problem which we have already solved.

Absence of related aggregated flow in output:

$$f_{id,i} = 0 : \qquad i \in P, d \in [1, N+1], d \neq i \tag{8.17}$$

Overall flow from the sensor set S:

$$\sum_{i \in S} \sum_{\substack{z \in \{0\} \cup P \\ z \neq i}} \left(\sum_{\substack{d \in [1,N+1] \\ d \neq j}} f_{id,z} - \sum_{\substack{s \in [1,N+1] \\ s \neq i}} f_{si,z} \right) = k \tag{8.18}$$

Energy constraints:

$$t \Bigg\{ \sum_{\substack{d \in [1,N+1] \\ d \neq i}} p_{tx}(i,d) \sum_{z \in \{0\} \cup P} f_{id,z} + \sum_{\substack{s \in [1,N+1] \\ s \neq i}} p_{rx}(s,i) \sum_{z \in \{0\} \cup P} f_{si,z} + \underbrace{p_{agg} \sum_{\substack{s \in [1,N+1] \\ s \neq i}} f_{si,i}}_{\text{Term to be included for potential aggregators only}} + \underbrace{p_{sense} \sum_{\substack{z \in \{0\} \cup P \\ z \neq i}} \left(\sum_{\substack{d \in [1,N+1] \\ d \neq j}} f_{id,z} - \sum_{\substack{s \in [1,N+1] \\ s \neq i}} f_{si,z} \right)}_{\text{Term to be included for potential sensors only}} \Bigg\} \leq e_i : \; i \in [1, N] \tag{8.19}$$

Conservation Constraints :

Conservation of agg commodities in the basestation:

$$\sum_{s \in [1,N]} f_{s(N+1),z} = \sum_{d \in [1,N]} f_{(N+1)d,z} : z \in P \tag{8.20}$$

Conservation of flow in nodes that neither sense nor aggregate:

$$\sum_{\substack{s \in [1,N+1] \\ s \neq i}} f_{si,z} = \sum_{\substack{d \in [1,N+1] \\ d \neq j}} f_{id,z} : \quad i \in [1, N], i \notin S \cup P, z \in \{0\} \cup P \tag{8.21}$$

Conservation of *unrelated* flow in aggregators that do not sense:

$$\sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z} = \sum_{\substack{d\in[1,N+1]\\ d\neq j}} f_{id,z} : \quad i \in P - S, z \in P, z \neq i \tag{8.22}$$

Aggregation Constraints : Compression of *related* flow in aggregators that do not sense:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,0} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,0} = \frac{1}{k} \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,i} : \quad i \in P - S \tag{8.23}$$

Non-consumption of flow in sensors :

Non-consumption of unrelated flows in sensors:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq j}} f_{id,z} \geq \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z} : \quad i \in S, z \in P, z \neq i \tag{8.24}$$

Non-consumption of the unagg commodity in sensors that are not aggregators:

$$\sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,0} \geq \sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,0} : \quad i \in S - P \tag{8.25}$$

Non-consumption in sensors that are aggregators:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,0} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,0} \geq \frac{1}{k-1} \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,i} : \quad i \in S \cap P \tag{8.26}$$

Limits on sensor flows :

Limits on total flow from any single sensor:

$$\sum_{z\in\{0\}\cup P, z\neq i} \left(\sum_{\substack{d\in[1,N+1]\\ d\neq j}} f_{id,z} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z} \right) \leq 1 : \quad i \in S \tag{8.27}$$

Limits on sensor flow destined for aggregation:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq j}} f_{id,z} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z} \leq \frac{1}{k} \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{sz,z} : \quad i \in S, z \in P, z \neq i \qquad (8.28)$$

Non-negativity of flows is necessary for the flow to be valid. The second constraint (8.17) states that if a flow originates from an aggregator, it must not carry any commodity that is destined for that aggregator.

The third constraint (8.18) is essentially the same as (8.11) with the only difference that since we have multiple commodities here, we must sum over all of them. The summation index $z = \{0\} \cup P$ runs over all commodities ($\{0\}$ is the unagg and P has the $|P|$ agg commodities) with the exception of $z = i$ for node i. For a sensor that is not an aggregator, this exclusion is vacuous i.e. the case $z = i$ never arises. For a sensor that is an aggregator, we exclude the related flows from the total sensor volume computation. As we have seen in the previous constraints, related outgoing flows are zero anyway (8.17). Related incoming flows are destined for aggregation and hence not directly related to the volume of the sensing flow. Note that a sensor that is also an aggregator has an implicit internal flow which must be included in the summation. This is taken care of via the unagg commodity. Once a stream is aggregated at a node, the result is sent out via the unagg flow. Hence subtracting the unagg inflow from the unagg outflow includes the volume of the aggregation output, which is the same as the volume of the internal implicit flow.

The energy constraint (8.19) is again an extension of (8.14). As before, the only change is that we sum up over all commodities. Also, instead of four different constraints depending on whether a node is in S or P or both or none, we have simply marked the terms that should be included for potential sensors and aggregators.

Next we have conservation constraints. The basestation acts as the sink for all unaggregated flow and hence there is no conservation constraint for the unagg commodity. The agg commodities reaching the basestation are however conserved (8.20). In nodes that neither sense nor aggregate i.e. which only route, these constraints are

the same as before - every commodity is conserved as expected (8.21). In the case of an aggregating node that does not sense, unrelated flows must be conserved (8.22). Again, for these kinds of nodes, the net unagg flow is augmented by an amount equal to the volume of the aggregated stream. This volume in turn is simply $1/k^{\text{th}}$ the volume of the total inflow destined for aggregation (8.23).

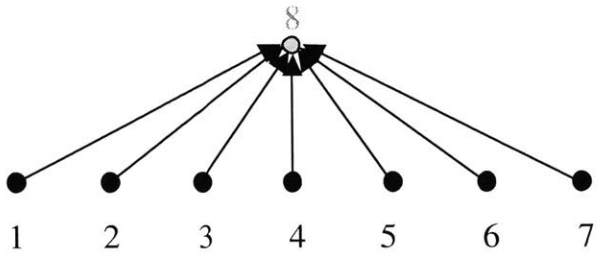
While sensors were governed by a single non-consumption constraint (8.12) in our non-aggregating program, they are governed by three constraints here. Consider agg commodities first. If these are related to the sensor, then they are already governed by the aggregation constraint in (8.23). For unrelated agg commodities, the non-consumption constraint is given by (8.24) which is conceptually identical to constraint (8.12) presented earlier. Next, we consider the unagg commodity. For sensors that are not aggregators, the constraint on the unagg commodity is straightforward as can be seen in (8.25). For sensors that are also aggregators, the unagg output must be not only greater than the unagg input, but greater by the volume of the aggregated stream produced at the node (8.26). Note that we use $1/(k-1)$ and not $1/k$ since one stream destined for aggregation comes from the node itself.

The final constraints are the limits on sensor flows, which were motivated in the last section. The first of these (8.27) is the same as (8.13) and prevents any sensor from monopolizing the output from the sensor group. We also need to prevent a sensor from monopolizing the sensor flow *to a particular aggregator* which is achieved by (8.28).

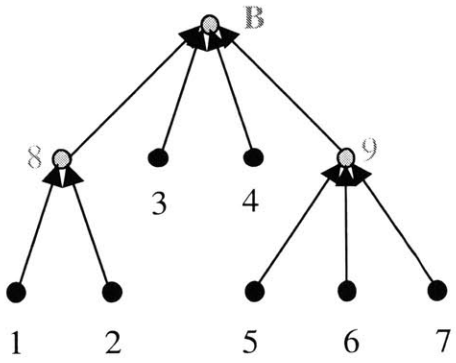
8.7 Generalization: Hierarchical Aggregation

Aggregating networks come in two main flavors - *non-hierarchical* and *hierarchical.* In the last section we dealt with non-hierarchical aggregation, where all the streams are aggregated at a single node. In this section we introduce a powerful generalization - hierarchical aggregation - where streams can be fused over several nodes in a hierarchical fashion (figure 8-11).

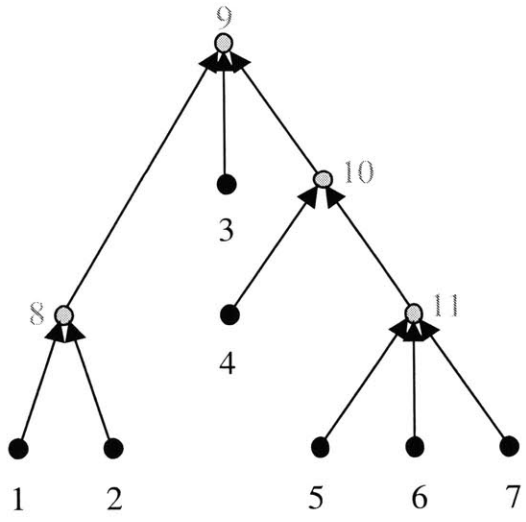
It turns out that completely arbitrary hierarchical aggregation seems to take a



(a) Non-hierarchical Aggregation



(b) Depth-2 Hierarchical Aggregation



(c) Unconstrained Hierarchical Aggregation

Figure 8-11: Three flavors of aggregation illustrated. Dark nodes are sensors and light ones aggregators. Potential relay nodes between sensors and aggregators are not illustrated.

time that is exponential in the number of potential sensors (m) and hence computationally hard unless m is a small constant[9]. It turns out that by suitably constraining hierarchical aggregation, we derive a strain that is considerably more flexible than non-hierarchical aggregation while being computationally tractable. In this flavor, aggregation of aggregated streams is only permitted at the basestation. Since this restriction gives rise to a tree of at most depth 2, we call this *depth-2 hierarchical aggregation* (figure 8-11(b)). The great advantage of depth-2 aggregation is that all

[9]This is obviously an opinion and not a proof that generalized hierarchical aggregation is indeed computationally hard!

the sensor data is not forced to go to a single aggregator, which steals some benefits of aggregation.

Devising a polynomial time algorithm for depth-2 aggregation turns out to be a fairly involved, but very interesting, exercise. The two key ideas are stated below:

Idea I : Reducing aggregation possibilities from 2^m to $\binom{m}{2}$

Idea II : "Linearizing" a non-linear aggregation cost function

We now present a series of lemmas which are used to devise a suitable flow for depth-2 hierarchical aggregation[10].

Definition 8.15 (Aggregation Strategy). *Consider j sensors directing flow $\{f_1, f_2, \cdots, f_j\}$ to a node for aggregation. There are 2^j ways of aggregating data*[11]*, which we denote by $\mathcal{W} = \{W_1, W_2, \cdots, W_{2^j}\}$. Hence, the aggregation strategy is completely defined by a vector $\mathbf{a} \in (\mathbb{R}^+)^{2^j}$ where a_j corresponds to the amount of flow being aggregated via way W_j.*

Definition 8.16 (Greedy Aggregation). *Given a set of flows $\{f_1, f_2, \cdots, f_j\}$, the greedy strategy is defined as follows:*

Initialize *: $\mathbf{a}$ to all zeros, and F to $\{f_1, f_2, \cdots, f_j\}$.*

Extract flow *: Let $f_{\min} = \min F$. Let W_d be the way corresponding to F. Assign $a_d = f_{\min}$.*

Update *: Subtract $f_{\min}$ from all the flows in F. Delete all elements that are reduced to zero as a result of this operation.*

Iterate *: If F has one or zero elements terminate else goto **Extract Flow**.*

The resultant vector $\mathbf{a}$ corresponds to a greedy aggregation strategy. Note that the algorithm can potentially terminate with a single flow in F. This flow is not aggregated anywhere.

[10] Henceforth, we simply term such aggregation hierarchical aggregation, with the depth-2 constraint implied.

[11] *First decide how many streams to aggregate, say, $h \leq j$. Then decide which of $\binom{j}{h}$ combinations to pick.*

Lemma 8.17 (Lower Bound on Total Aggregated Flow). *Given a set of flows* $\mathcal{F} = \{f_1, f_2, \cdots, f_j\}$, *no aggregation strategy can produce a total aggregated flow of less than* $\max \mathcal{F}$.

Proof. Without loss of generality, let $f_1 = \max \mathcal{F}$. Note that there might be several maxima, but we only need to pick one for this proof. Let f_1 be partitioned thus:

$$f_1 = f_{1,1} + f_{1,2} + \cdots + f_{1,2^j}$$

with flow $f_{1,i}$ being aggregated via way W_i. Clearly, the total volume of the aggregated output is $f_{1,1} + f_{1,2} + \cdots + f_{1,2^j}$ which is simply $\max \mathcal{F}$. □

Theorem 8.18 (Optimality of the Greedy Strategy). *Given a set of flows* $\mathcal{F} = \{f_1, f_2, \cdots, f_j\}$ *being aggregated on the node, no aggregation strategy can achieve a lifetime longer than the greedy strategy.*

Proof. The energy impact of an aggregation at a node is threefold:

1. Transmission of flows $\mathcal{F}$ to the aggregator.

2. Energy costs of aggregation.

3. Energy costs of transmitting the aggregated stream.

The cost of transmitting flows from the sensors to the aggregating node are independent of the aggregation strategy. The energy cost of aggregation is proportional to the total volume of the input flow and hence also does not depend on the aggregation strategy, as long as all flows are getting completely aggregated. The greedy algorithm does as well as any strategy, if not better, since it can have a residual, unaggregated flow leading to a lower energy cost of aggregation. This leaves us with the last item above. Under a greedy strategy, the total aggregation flow is simply equal to $\max \mathcal{F}$. By lemma 8.17 no other strategy can achieve a total aggregated flow less than this. Hence, the greedy strategy is optimal. □

Corollary 8.19. *If a set of flows $\mathcal{F} = \{f_1, f_2, \cdots, f_j\}$ is being optimally aggregated at a node, then the maximum flow $\max \mathcal{F}$ occurs with a multiplicity of at least 2 i.e. there are at least two flows equal to the maximum.*

Proof. Suppose not. Using the greedy strategy, we can *reduce* the energy required to aggregate while keeping all energy costs the same (by the optimality proved above) which contradicts the optimality of the aggregation strategy. □

Henceforth, it is assumed that at least two flows are equal to the maximum.

In the development above, we have tackled the first part of the problem. We have shown that although the aggregation strategy at a node is characterized by a vector with an exponential number of elements in j, the optimal amongst these can be characterized in a much simpler manner - via the greedy strategy.

The next challenge is modelling the constraints developed above. While seemingly innocuous, these constraints turn out to be non-linear and threaten to shift the linear program into the domain of non-linear or integer programs, to which computationally efficient strategies like the Ellipsoid method cannot be applied. First, note that the max operator is non-linear. To see this, define $T[.]$ as an operator on n-tuples such that $T[(a_0, a_1, \cdots, a_{n-1})] = \max a_i$, then $T[(a_0, a_1, \cdots, a_{n-1})] + T[(b_0, b_1, \cdots, b_{n-1})] \neq T[(a_0 + b_0, a_1 + b_1, \cdots, a_{n-1} + b_{n-1})]$. Stating the simple constraint that the total aggregated flow is equal to $\max \mathcal{F}$ is thus not possible in a linear program. Stating the more complex constraint that the two largest flows be equal is even tougher.

We now discuss the scheme that allows us to circumvent this problem. Consider first the case where the sensor set is given and all sensors must be used (i.e. $k = m$). We now have k sensor sending data to a given aggregator. We create $\binom{k}{2}$ *aggregation centers* in the aggregating node, one for each possible combination of the two largest flows. Thus, for example, center 1 is meant for that part of the flow where sensors s_0 and s_1 have maximum flows, center 2 is for the case when sensors s_0 and s_2 have the maximum flow and so on. For a given center, it is straightforward to impose the flow constraints. For center 1, we set the flows from s_0 and s_1 to be equal and set the flows from other nodes to be strictly less than both these. It is clear that such an

arrangement allows the network to assume an optimal flow while keeping everything linear. The price we have paid for linearity is that the number of commodities has increased by $\binom{k}{2}$ times.

This scheme of aggregation centers generalizes easily to the important case of sensor sets where $k \leq m$ rather than $k = m$. From the point of view of the set of potential sensors, S, the aggregating node can still be viewed as a monolithic entity. Hence the constraint to prevent monopolies (8.28) is as applicable as before. As long as sensors satisfy this, a sensing schedule can be recovered from a flow. Also, as is clear from the greedy aggregation, any set of flows (with the maximum two equal) to an aggregation center are meaningful.

The new program follows. Most constraints are essentially repeated with an additional summation term that covers all aggregation centers. We use the notation $f_{ij,z,c}$ to denote the flow from node i to node j carrying commodity destined for aggregation at center c in aggregating node z. We use C to denote the set of all aggregation centers. Hence the tuple $(z, c), z \in P, c \in C$ completely identifies an aggregation center - z tells us which aggregator is being used and c tells us which aggregation center is being used. Since $z = 0$ denotes the unagg commodity, we do not qualify it with any additional aggregation center information. Thus, $f_{23,0}$ is the unagg commodity flowing from node 2 to node 3. For all other commodities, an aggregation center qualification is mandatory. Hence $f_{23,9}$ is absurd in the current context. Also, instead of making flows from sensors to aggregators implicit when they are the same node, we now use the class of flows $f_{ii,i,c}$ for this purpose. These "internal" flows are only meaningful when $i \in S \cap P$ i.e. i is both a sensor and an aggregator. For all other i we force these flows to zero (8.31).

Constraint (8.33) essentially mimics (8.18) with some technicalities. Firstly, we have to sum over all aggregation centers. Secondly, we have to take care of the case when a sensor is also an aggregator. In this case, we use the $f_{ii,i,c}$ commodity and sum it over all aggregation centers and all sensor-aggregator nodes[12].

[12]Since these flows are set to zero for all nodes but sensor aggregators, summing over all nodes achieves the same effect as a more restricted summation over $S \cap P$.

The energy constraint (8.34) also follows (8.19) very closely. The only difference in the transmit and receive energy terms is the summation over all aggregation centers. In the aggregation term, there is one additional difference - we have eliminated the usual $s \neq i$ constraint to allow the inclusion of internal flows in aggregation volume. The sensing energy term has the same summation kernel as the overall sensor flow constraint discussed above and the relevant explanations apply here.

Conservation constraints in routers (the basestation, nodes that neither sense nor aggregate and lastly unrelated flows in aggregators) are straightforward as before (8.35-8.37) with the only change that we are using z, c instead of simply z to qualify the flows.

In aggregators that do not sense, the net outflow of the unagg commodity is incremented by the total volume of the streams that result from the aggregation centers (8.38). We use the notation $C^{-1}(.)$ to denote an operator which returns the set of all (node, aggregation center) pairs present in node i. Also note that in the summation kernel, we do not exclude $d = j$ which takes care of internal flows. For the same reason, it is important to exclude $s = j$ in the second summation term (else what we add via the first term is cancelled out).

The next set of constraints are the non-consumption constraints on sensors. All agg flows out of a sensor node i except those destined for aggregation at the node itself are handled by constraint (8.39). The LHS minus the RHS in this inequality is what sensor i contributes to the aggregation center z, c. The constraint for unagg flow in sensors that are not aggregators follows the same template (8.40). Finally, in sensors that are aggregators, the unagg commodity follows the same constraint as (8.38) but with the equality replaced by a strict inequality (8.41).

In the last program, we had two limits on sensor flows. The first was simply that no sensor contribute more than unit flow over the lifetime of the network (8.27)and the second that no sensor monopolize any aggregator (8.28). We have these same constraints again, but modified to use the language of aggregation centers. The LHS in (8.42) simply evaluates to the total flow a sensor contributes. If it is a pure sensor, then the second term in the LHS evaluates to zero and the $z \neq i$ constraint in the

first summation term is moot. If this is a sensor-aggregator, then we sum up the net contribution of this sensor to all aggregation centers that are not on the node itself (i.e. $z \neq i$) using the first summation term. The second summation term takes care of the internal flows. The constraint that prevents aggregator monopolies seems tedious but barring technicalities is straightforward. On the left is the total contribution of sensor i to aggregator z. This summation works even if z is in fact the same as i. On the right is the total contribution to aggregator z *from all sensors.* We are simply stating that the overall contribution from a given sensor to an aggregator not exceed $1/k^{th}$ of the agg flow destined for this aggregator.

As discussed in the flow formulation of hierarchical aggregation, we need to ensure that the top two flows to an aggregation center are equal and that all other flows are less than these. Constraint (8.44) ensures the first constraint. We use $n_1(.,.)$ and $n_2(.,.)$ to indicate operators that act on an specified aggregation center and return the index of the two nodes respectively that have the maximum flows. On the left is the total flow from node $j = n_1(i,c)$ destined for aggregation center (i,c). On the right is the total flow from node $k = n_2(i,c)$ destined for the same aggregation center. Note that by allowing d to be equal to j, we take care of internal flows. Constraint (8.44) uses the same summation kernels to ensure that contributions from all other sensors to this aggregation center are less than the two flows just equated above. This is the reason k on the RHS runs over all of S with the exclusion of $n_1(i,c)$ and $n_2(i,c)$.

Objective :

$$\max \quad t \tag{8.29}$$

General Constraints :

Non-negativity of flow:

$$f_{ij,z,c} \geq 0: \quad i \in [1, N+1], j \in [1, N+1], z \in 0 \cup P, c \in C \tag{8.30}$$

Absence of self flows in nodes that aren't sensor-aggregators:

$$f_{ii,z,c} = 0 : \quad i \in [1, N+1] - (S \cap P), z \in 0 \cup P, c \in C \tag{8.31}$$

Absence of related aggregated flow in output:

$$f_{id,i,c} = 0 : \qquad i \in P, d \in [1, N+1], d \neq i, c \in C \tag{8.32}$$

Overall flow from the sensor set S:

$$\sum_{c \in C} \sum_{i \in S} \sum_{\substack{z \in \{0\} \cup P \\ z \neq i}} \left(\sum_{\substack{d \in [1,N+1] \\ d \neq j}} f_{id,z,c} - \sum_{\substack{s \in [1,N+1] \\ s \neq i}} f_{si,z,c} \right) + \sum_{i \in S} \sum_{c \in C} f_{ii,i,c} = k \tag{8.33}$$

Energy constraints:

$$t \sum_{\substack{d \in [1,N+1] \\ d \neq i}} p_{tx}(i,d) \sum_{c \in C} \sum_{z \in \{0\} \cup P} f_{id,z,c} + \sum_{\substack{s \in [1,N+1] \\ s \neq i}} p_{rx}(s,i) \sum_{c \in C} \sum_{z \in \{0\} \cup P} f_{si,z,c}$$

$$+ \underbrace{tp_{agg} \sum_{s \in [1,N+1]} \sum_{c \in C} f_{si,i,c}}_{\substack{\textit{Term to be included for} \\ \textit{potential aggregators only}}}$$

$$+ \underbrace{tp_{sense} \left\{ \sum_{c \in C} \sum_{z \in \{0\} \cup P, z \neq i} \left(\sum_{\substack{d \in [1,N+1] \\ d \neq i}} f_{id,z,c} - \sum_{\substack{s \in [1,N+1] \\ s \neq i}} f_{si,z,c} \right) + \sum_{c \in C} f_{ii,i,c} \right\}}_{\substack{\textit{Term to be included for} \\ \textit{potential sensors only}}}$$

$$\leq e_i : \; i \in [1, N] \tag{8.34}$$

Conservation Constraints :

Conservation of agg commodities in the basestation:

$$\sum_{s \in [1,N]} f_{s(N+1),z,c} = \sum_{d \in [1,N]} f_{(N+1)d,z,c} : z \in P, c \in C \tag{8.35}$$

Conservation of flow in nodes that neither sense nor aggregate:

$$\sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z,c} = \sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,z,c} : \quad i \in [1,N], i \notin S \cup P, z \in \{0\} \cup P, c \in C \quad (8.36)$$

Conservation of *unrelated* flow in aggregators that do not sense:

$$\sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z,c} = \sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,z,c} : \quad i \in P - S, z \in P, z \neq i, c \in C \quad (8.37)$$

Aggregation Constraints :

Compression of *related* flow in aggregators that do not sense:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,0} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,0} = \sum_{(j,c)\in C^{-1}(i)} \left(\sum_{d\in[1,N+1]} f_{jd,i,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq j}} f_{sj,i,c} \right) \quad (8.38)$$

Non-consumption of flow in sensors :

Non-consumption of unrelated flows in sensors:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq j}} f_{id,z,c} \geq \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z,c} : \quad i \in S, z \in P, z \neq i, c \in C \quad (8.39)$$

Non-consumption of the unagg commodity in sensors that are not aggregators:

$$\sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,0} \geq \sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,0} : \quad i \in S - P \quad (8.40)$$

Non-consumption in sensors that are aggregators:

$$\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,0} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,0} \geq \sum_{(j,c)\in C^{-1}(i)} \left(\sum_{d\in[1,N+1]} f_{jd,i,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{sj,i,c} \right) \quad (8.41)$$

Limits on sensor flows :

Limits on total flow from any single sensor:

$$\sum_{c\in C}\sum_{z\in\{0\}\cup P, z\neq i}\left(\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,z,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z,c}\right) + \sum_{c\in C} f_{ii,i,c} \leq 1 : i \in S \quad (8.42)$$

Limits on sensor flow destined for aggregation:

$$\sum_{\substack{c\in C\\ z\neq i}}\left(\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,z,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z,c}\right) + \sum_{c\in C} f_{ii,i,c} \leq$$

$$\frac{1}{k}\sum_{\substack{i\in S\\ z\neq i}}\sum_{c\in C}\left(\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,z,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z,c}\right) + \frac{1}{k}\sum_{i\in S}\sum_{c\in C} f_{ii,i,c} : i \in S, z \in P \quad (8.43)$$

Equality of top two flows to an aggregation center:

$$\sum_{d\in[1,N+1]} f_{jd,i,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq j}} f_{sj,i,c} = \sum_{d\in[1,N+1]} f_{kd,i,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq k}} f_{sk,i,c}$$

$$: i \in P, c \in C, j = n_1(i,c), k = n_2(i,c) \quad (8.44)$$

Ordering of flows to an aggregation center:

$$\sum_{d\in[1,N+1]} f_{jd,i,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq j}} f_{sj,i,c} \geq \sum_{d\in[1,N+1]} f_{kd,i,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq k}} f_{sk,i,c}$$

$$: i \in P, c \in C, j = n_1(i,c), k \in S - n_1(i,c) - n_2(i,c) \quad (8.45)$$

8.8 Generalization: Moving/Multiple Sources/Time-Varying Quality

Unlike the previous sections where we dealt with generalizations that allow greater freedom in picking collaborative strategies this section focusses on generalizations that enable a richer description of the environment and the user's expectation of network performance. Specifically we deal with three closely related generalizations - multiple sources, moving sources and time-varying quality. The reason we treat them together is that the modelling and computational machinery needed to tackle these three seemingly unrelated generalizations is virtually identical.

8.8.1 Moving Sources

Perhaps the most interesting of these three generalizations is that of moving sources. Determining optimal collaborative strategies under source movement is perhaps the climax, the *ne plus utra* of strategies that maximize lifetime. It is for this reason, that the simplicity with which this generalization can be accommodated comes as a surprise. One can formulate several flavors of the moving source problem:

- Given a source trajectory characterized by location and absolute time tuples,

$$\mathcal{T} = \{(\vec{l_i}, \tau_i) | 0 \leq i \leq L-1\}$$

  what is the maximum lifetime that can be achieved in the *first time to failure sense*?

- Given a source trajectory as above, what is the maximum lifetime that can be achieved in the *cumulative sense*? In the formulation above, the network must obey its contract continuously. In this formulation, the network can choose to ignore certain source locations which it deems as too costly for sensing and focus on the overall cumulative lifetime.

- Given a source trajectory characterized by location and *relative* time tuples,

what is the maximum lifetime in the first time to failure sense? In other words, the time annotations of the locations are interpreted as proportions rather than absolute time. The idea is to then maximize the constant of proportionality without sacrificing any location.

The first two flavors are deterministic versions of the moving source problem while the last is a more probabilistic formulation. Recall from our discussion of the lifetime bounds section that the exact source behavior is seldom known beforehand. The fact that a prime application of sensor networks is source location points to the fact that what is available beforehand is at best a stochastic description. It is for this reason that the last flavor of the problem is the most interesting and the one we tackle in this section.

We captured this stochastic description in the chapter on bounds using a source location pdf. A straightforward generalization of that pdf is a joint source-location, desired-quality pdf. This pdf not only tells us what the probability of finding the source in a certain location is, it also tells us what the probability of the user demanding that k sensors listen is. By discretizing this joint pdf on a fine enough mesh, and taking averages over the mesh elements, we can convert it to a form like table 8.3 which is best understood by referring to the example network in figure 8-12[13].

Location (w)	S_w	k_w	Probability (η_w)
a.	{1,2}	1	0.20
b.	{3,4,5}	2	0.10
c.	{5,6,7}	2	0.25
d.	{8,9,12,13}	4	0.20
e.	{12,13,14}	2	0.15
f.	{1,2}	2	0.10

Table 8.3: Stochastic description of source movement.

[13]While the techniques we demonstrate run in a time that is polynomial in the number of locations (L), the natural question that arises is - how is L usually related to the number of nodes N? It is not hard to show that in the *worst case*, $L = O(2^{\rho d})$, where ρ is the source observation radius and d is the density of the network in nodes/m^2. In most networks, ρ and d are design constants that do not scale with N. Hence, L stays polynomial in N. If however $\rho d = \theta(m)$, then L will be exponential in N.

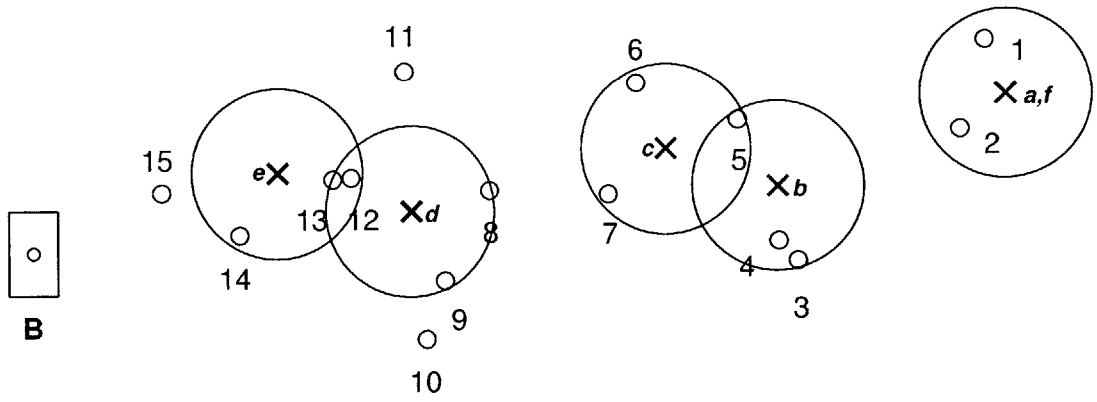


Figure 8-12: Figure illustrating source movement.

As before we need to indulge in some "flowsmithing" to attack this problem. The central idea is the moving sources problem can be converted to a multiple sources problem. Consider a source with the relative time trajectory -

$$\mathcal{T} = \{(\vec{l_i}, \tau_i) | 0 \leq i \leq L-1\}$$

from which we can derive the time-varying set of potential sensors, desired number of sensors and the proportional time spent at the location -

$$\{(S_i, k_i, \eta_i) | 0 \leq i \leq L-1\}$$

as we have shown in the table earlier. Consider next two views to approach the problem.

Definition 8.20 (Source Moving in Infinitesimals). *Given a source that moved around spending infinitesimal time at each of these L locations in the proportions indicated in the table above (η_w), what collaborative strategy would maximize lifetime?*

Definition 8.21 (Multiple Sources with Proportional Flow). *Consider a network with L sources at the locations specified by the trajectory above. At each location i, we require the total volume out of the sensor set corresponding to this location (S_i) to be $k_i\eta_i$. Again, we ask the question - what collaborative strategy would ensure maximum lifetime?*

Proposition 8.22. *The two problems above have identical answers i.e. if we can design a collaborative strategy that allows the network to achieve a certain lifetime t with infinitesimal source movement, then there exists a strategy that allows the network to achieve a lifetime no less than t in the multiple source flavor (and* vice-versa*).*

Proof. We present a proof sketch that if we can find a strategy for the multiple source version then it automatically allows us to achieve the same lifetime for the infinitesimally moving source problem. Consider that we have a collaborative strategy,

$$C = \{(r_i, t_i) | r_i \in R, t_i \in \mathbb{R}^+\}$$

for the multiple sources version which allows us to achieve a lifetime of t. Take each time-annotated FRA in C and split it into infinitely many FRAs each sustained for an infinitesimally small unit of time dt. Also note that since we have imposed a total flow constraint of $k_i\eta_i$ from the sensor set S_i while demanding that exactly k_i sensors be active at any time, this sensor set will account for η_i of the lifetime. Hence, the infinitesimals corresponding to location i will sum up to η_i of the total lifetime. In the final step, we devise a new collaborative strategy by interleaving these infinitesimals in a repetitive cycle of period L corresponding to the L locations. Clearly, we now have a collaborative strategy for the infinitesimal source movement case. □

We now have at our hand a problem with multiple sources, rather than moving sources. Solving this multiple sources problem is a relatively simple, multi-commodity extension of what we have already seen thus far. The main conceptual extension is the generalization of the "k of m Coloring Problem" we first studied in the context of aggregation. In that problem we showed that as long as the quantity of m colors satisfied two simple constraints, we could always color k line segments with them such that no color was ever underneath the other. We now present the generalization in context of networks with multiple sources.

Definition 8.23 (The Generalized k of m Coloring Problem). *We pose the problem as follows:*

Given *We are given L sets of line segments, say $\mathcal{L}_1$ through $\mathcal{L}_L$, arranged as shown in figure 8-13. There are precisely k_i segments of length η_i in set $\mathcal{L}_i$. Also, these segments can be colored using a specified set of colors, say $\mathcal{C}_i$ whose cardinality we denote by m_i. The sets $\mathcal{C}_i$ are in general intersecting i.e. several sets may allow the use of the same color(s). The set of all colors is denoted by $\mathcal{C}$ (hence $\mathcal{C} = \cup_i \mathcal{C}_i$). The number of distinct colors is hence $|\mathcal{C}|$ and denoted by m.*

Valid Coloring Instance *A given coloring of these line segments obeying the constraints above such that no segment is without color corresponds to a coloring instance. If, for any line l perpendicular to the segments (in any set $\mathcal{L}_i$), l does not intersect the same color, we say that the coloring instance is* valid. *The total amount of a certain color c_j used in a valid coloring instance is denoted by x_j.*

Question *For a given problem instance (described by $\mathcal{L}_i$,k_i,η_i and $\mathcal{C}_i$) consider the set of all valid coloring instances (which we denote by $\mathcal{V}$). Can this set be characterized by a compact set of constraints (say $\mathcal{S}$) on the set of colors $\{x_j | 0 \leq j \leq m\}$? In other words, the set characterized by $\mathcal{S}$ is equal to the set $\mathcal{V}$.*

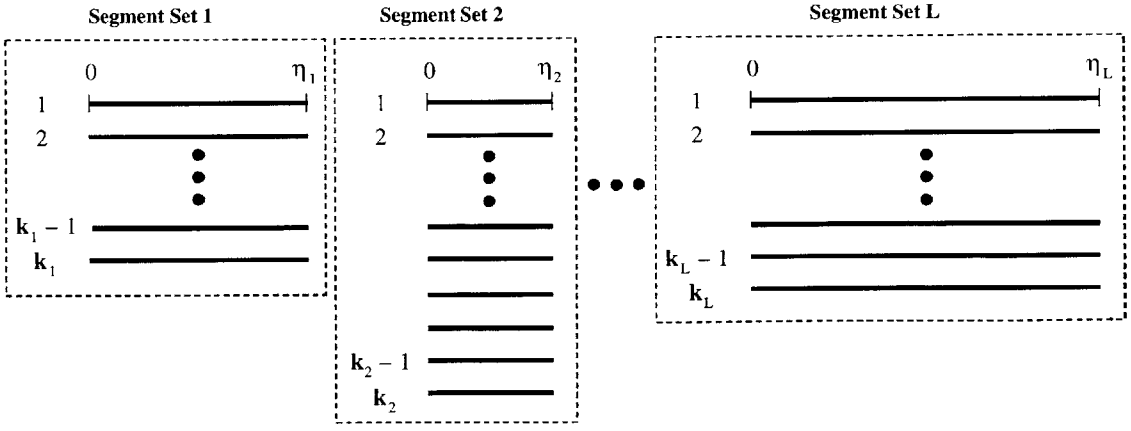


Figure 8-13: The generalized k segments coloring problem. We want to color these segments using certain specified quantities of $m \geq \max\{k_i\}$ colors such that any line l perpendicular to them does not intersect the same color twice.

The reason for a compact characterization $\mathcal{S}$ is clear - we want to ensure that every set of flows corresponds to a valid role assignment. It is not difficult to add constraints to S. Here are a few classes of constraints that all valid coloring instances obey:

- Positivity constraints: All colors can only be used in positive quantities:

$$x_j \geq 0 : 0 \leq j \leq m$$

- Total volume: The sum of the m colors must be equal to the cumulative length of the segments. Hence,

$$\sum_{i=1}^{m} x_j = \sum_{i=1}^{L} k_i \eta_i$$

- Limit on any single color: Let color c_j be allowed for use in a total of t_j sets $\mathcal{L}_{a_1}, \mathcal{L}_{a_2}, \cdots, \mathcal{L}_{a_{t_j}}$. Then the total volume of color c_j is governed by,

$$x_j \leq \sum_{i=1}^{t_j} \eta_{a_i}$$

- Exclusion constraints: Consider the segment set $\mathcal{L}_i$. These segments must be colored using the m_i colors in $\mathcal{C}_i$. Consider excluding a subset of these colors, say $\mathcal{C}_E$. Then we must have,

$$\sum_{c_j \in \mathcal{C}_i - \mathcal{C}_E} x_j \geq (k_i - |\mathcal{C}_E|)\eta_i$$

- Inclusion constraints: Pick j sets from the L coloring sets, $\{\mathcal{C}_i\}$ say $\mathcal{C}_{a_1}$ through $\mathcal{C}_{a_j}$. Then the following constraint holds:

$$\sum_{c_j \in \cup_i \mathcal{C}_{a_i}} x_j \geq \sum_i k_{a_i} \eta_{a_i}$$

It seems highly likely that the constraints above are an accurate characterization of admissible coloring tuples $\{x_j\}$. However, there are two problems. The first is that the proof of this conjecture (if it is indeed true) seems non-trivial. The bigger problem is that the number of constraints above is exponential in m. This by itself might not be a problem in a linear program - indeed it is well known that programs with an exponential number of constraints can be tackled in polynomial time using efficient separation oracles [1]. However, it turns out that we can use a simple trick that neatly sidesteps all this.

Let us denote by $x_{j,i}$ the amount of color j used in the i^{th} segment set $\mathcal{L}_i$. Instead of characterizing the set of valid coloring instances by x_js, we are going to characterize it using $x_{j,i}$s using the following set of constraints (which we call $\mathcal{S}$):

- Total volume of a color:

$$x_j = \sum_{i=1}^{L} x_{j,i} : j \in [1, m] \tag{8.46}$$

- Non-negativity:

$$x_{j,i} \geq 0 : j \in [1, m], i \in [1, L] \tag{8.47}$$

- Absence of unrelated colors:

$$c_j \notin \mathcal{C}_i \Rightarrow x_{j,i} = 0 \tag{8.48}$$

- Upper limits:

$$x_{j,i} \leq \eta_i : j \in [1, m], i \in [1, L] \tag{8.49}$$

- Constraints relating to a specific segment set $\mathcal{L}_i$:

$$\sum_{j=1}^{m} x_{j,i} = k_i \eta_i : i \in [1, L] \tag{8.50}$$

We claim that the characterization above is exactly what we need!

Lemma 8.24. *If $\mathcal{V}$ is the set of all valid coloring instances then $\mathcal{V} \subseteq \mathcal{S}$ i.e. every valid coloring instance can be expressed using a set of $x_{j,i}$s that obey $\mathcal{S}$.*

Proof. Pick a specific coloring instance say v from V. Associate a variable x_j with every distinct color c_j in v. Go through each of the segment sets and determine the quantities $x_{j,i}$s. Then we claim that the set of x_js and $x_{j,i}$s obeys the constraints in S:

- Constraints (8.46)-(8.48) hold from the definition.
- Since the coloring is valid, no perpendicular line l can meet the same color twice, as a result of which (8.49) holds.
- Consider any segment set $\mathcal{L}_i$. Then the colors that can be used for coloring $\mathcal{L}_i$ must add up to the total length. Hence (8.50) holds.

□

Lemma 8.25. *If $\mathcal{V}$ is the set of all valid coloring instances then $\mathcal{S} \subseteq \mathcal{V}$ i.e. any set of colors $\{x_j\}$ that obey the program $\mathcal{S}$ above correspond to a valid coloring instance.*

Proof. Let us call the $x_{j,i}$s *subcolors*[14]. Sets of allowable colors $\mathcal{C}_i$were allowed to intersect. But the set of allowable subcolors do not intersect. This makes it straightforward to generate a valid coloring instance from a set of specified x_js and $x_{j,i}$s. Pick the first segment set $\mathcal{L}_1$. Now pick its subcolors. They add up to the total length of the segment and none of them exceeds η_1. By the simple k of m coloring lemma (8.12), these subcolors can be used to color $\mathcal{L}_1$ such that no line l can touch the same color twice. Since every segment uses a set of subcolors disjoint from others, there is no danger of affecting the properties of any particular subcolor set and hence this procedure can be repeated until we have colored all segment subsets. □

Theorem 8.26. *The set described by the program above is the same as the set of all valid coloring instances i.e. $\mathcal{S} = \mathcal{V}$.*

[14]This is to distinguish them from x_js which are *colors*. Of course, strictly speaking, we should say "quantities of subcolors" and "quantities of colors" etc.!

Proof. The proof follows from the two lemmas above. □

The new program is simply the one we presented in the last section namely (8.29)-(8.45), along with the additional constraints (8.46)-(8.50) posed appropriately as follows.

Subflow constraints :

$$\sum_{c\in C}\sum_{z\in P, z\neq i}\left(\sum_{\substack{d\in[1,N+1]\\ d\neq i}} f_{id,z,c} - \sum_{\substack{s\in[1,N+1]\\ s\neq i}} f_{si,z,c}\right) + \sum_{c\in C} f_{ii,i,c} = \sum_{j=1}^{L} x_{i,j} : i \in S \tag{8.51}$$

$$x_{i,j} \geq 0 : \qquad i \in S, j \in [1, L] \tag{8.52}$$

$$\text{Node } i \notin \mathcal{S}_j \Rightarrow x_{i,j} = 0 : \qquad i \in S, j \in [1, L] \tag{8.53}$$

$$x_{i,j} \leq \eta_j : \qquad i \in S, j \in [1, L] \tag{8.54}$$

$$\sum_{i\in S} x_{i,j} = k_j \eta_j : \qquad j \in [1, L] \tag{8.55}$$

8.8.2 Multiple Sources and Multiple Moving Sources

In the last section we dealt with the probabilistic version of the moving sources problem and saw how it could be interpreted as having multiple sources present simultaneously. From this formulation, two generalizations follow. First, if we have Q sources present at fixed locations with the S_i and k_i defined analogously, then we can simply run the program above by fixing all η_is to $1/Q$. We must remember to divide the lifetime hence achieved by Q. Second, if we have Q sources each of which has its own L locations where it resides in a probabilistic sense. Hence, we are

provided with sets $S_{i,j}$, $k_{i,j}$ and $\eta_{i,j}$ which correspond to the sensor set, the required number of sensors and the probability corresponding to source i residing at location j. We can again reuse the framework developed for moving sources by renormalizing,

$$\eta_{i,j} \rightarrow \frac{\eta_{i,j}}{Q}$$

and treating $S_{i,j}$ and $k_{i,j}$ as before.

8.8.3 Time-varying Quality

An important subclass of problems subsumed in the above discussion is that of time-varying quality i.e. time-varying k with a fixed sensor-set S. Clearly, the framework developed for moving sources can trivially handle this case.

In this chapter we started with extremely simple collaborative strategies - simple multi-hop routing in collinear networks and have ended up with extremely sophisticated ones that can handle hierarchical aggregation in the face of multiple moving sources and time-varying quality. All this was possible using polynomial time algorithms by transforming the problem to one of finding optimal network flows and then transforming the solution back into a valid collaborative strategy.

Chapter 9

Conclusions

In this thesis, our objective was two-fold - to *quantify* the increasingly important notion of power-awareness of systems and having done that, to propose a systematic technique to *enhance* this quality. The first step in quantifying the power-awareness of a general system was to develop the notion of a perfectly power-aware system ($H_{perfect}$). The awareness of this system was shown to be an upper bound on practically achievable power-awareness. In the next step, we proposed a power-awareness metric whose physical interpretation is the expected battery lifetime of a system normalized to the lifetime of the perfect system. Next, the problem of enhancing power-awareness was treated formally using the concept of *ensembles of point systems.* We showed that constructing systems by intelligently putting together dedicated point systems could significantly enhance power-awareness. The basic factor that limited a monotonic increase in power-awareness as more and more point systems were put together was the increasingly amount of energy spent in co-ordinating these point systems. Hence, the problem of finding an optimal subset of point systems that struck the right balance was formally proposed. While it is unlikely that this optimal subset can be found using polynomial time algorithms, greedy heuristics were seen to work reasonably well. The technique of ensemble construction was illustrated using four different applications - multipliers, register files, digital filters and dynamic voltage processors. Significant power awareness improvements leading to system battery lifetime improvements in the range of 60% to 200% were seen.

Next, we addressed a key challenge facing the ubiquitous deployment of sensor networks - maximizing their active sensing lifetime.

We started out by deriving fundamental upper bounds on the lifetime of data gathering sensor networks for a variety of scenarios assuming node energy models based on $1/d^n$ path loss behavior. Using both analytical arguments and extensive network simulations, the bounds were shown to be tight for some scenarios and near-tight (better than 95%) for the rest. Lastly, we presented a technique that allows bounding lifetime by partitioning the problem into sub-problems for which the bounds are already known or easier to derive.

Next, we tackled the challenge of finding collaborative strategies that maximize lifetime. Unlike most previous work in this area which focussed exclusively on maximizing lifetime via minimum-energy routing, we treated the problem as one of determining optimal role assignments i.e. how should one assign roles to nodes in a network and how must these assignments be changed dynamically in an optimal manner. We showed that a simple, linear programming formulation can be used to derive optimal role assignments. However, this approach takes time that is exponential in the number of nodes and hence computationally infeasible. To combat this, an increasingly sophisticated class of role assignment problems were transformed to network flow problems, solved (near-)optimally and then transformed back - all in polynomial time.

We started with non-aggregating networks where the sensors were fixed beforehand (which essentially boils down to the maximum lifetime routing problem tackled in previous work). We then moved to non-aggregating networks where a set of potential sensors is specified, a minimum number of which are required to be sensing at any time. The equivalent flow program was formulated and solved for a network instance. Aggregation was introduced next and the potential for significant increase in lifetime demonstrated. Next we solved the more involved problem of optimal hierarchical aggregation. The last class of networks added on moving/multiple sources and the possibility of varying the minimum number of active sensors with time. We are confident that this last class is powerful enough to realistically capture most real

world scenarios.

It is our sincere hope that the power-awareness formalisms proposed here will be used to quantify this important aspect of systems and that the proposed framework will be employed by system architects to engineer systems that scale their power and energy requirements with changing operating scenarios leading to significant improvements in overall battery lifetimes in systems as varied as multipliers and massively distributed sensor networks.

Bibliography

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1 edition, February 1993.

[2] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou. Precomputation-based sequential logic optimization for low power. In *Proceedings of the International Conference on Computer-Aided Design*, pages 74–81, November 1994.

[3] R. Amirtharajah, T. Xanthopoulos, and A.P. Chandrakasan. Power scalable processing using distributed arithmetic. In *Proceedings of the 1999 International Symposium on Low Power Electronics and Design*, pages 170–175, 1999.

[4] L. Benini and G. De Micheli. Transformation and synthesis of fsms for low-power gated-clock implementations. In *Proceedings of the International Symposium on Low Power Design*, pages 21–26, 1995.

[5] M. Bhardwaj, R. Min, and A. Chandrakasan. Power-aware systems. In *Proceedings of the 34th Asilomar Conference on Signals, Systems, and Computers*, November 2000.

[6] M. Bhardwaj, R. Min, and A. Chandrakasan. Quantifying and Enhancing Power-Awareness in VLSI Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems (To appear)*, 2001.

[7] K. Bult et al. Low power systems for wireless microsensors. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'96)*, pages 17–21, August 1996.

[8] T. Burd, T. Pering, A. Stratakos, and R. Brodersen. A dynamic voltage scaled microprocessor system. In *Proceedings of the ISSCC*, pages 294–295, 2000.

[9] A. Chandrakasan et al. Design considerations for distributed microsensor systems. In *Custom Integrated Circuits Conference (CICC)*, pages 279–286, May 1999.

[10] A. P. Chandrakasan, S. Sheng, and R.W. Brodersen. Low power cmos digital design. *IEEE Journal of Solid State Circuits*, 27(4):473–484, April 1992.

[11] Jae-Hwan Chang and L. Tassiulas. Routing for maximum system lifetime in wireless ad-hoc networks. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, pages 1191–1200, September 1999.

[12] Jae-Hwan Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of IEEE INFOCOM 2000*, volume 1, pages 22–31, 2000.

[13] S. Cho, T. Xanthopoulos, and A. P. Chandrakasan. An ultra low power variable length decoder for mpeg-2 exploiting codeword distribution. In *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*, pages 177–180, 1998.

[14] A. P. Dancy and A. Chandrakasan. Ultra low-power control circuits for pwm converters. In A. Chandrakasan and R. W. Brodersen, editors, *Low-Power CMOS Design*, pages 137–142. IEEE Press, 1998.

[15] M. Dong, K. Yung, and W. Kaiser. Low power signal processing architectures for network microsensors. In *Proceedings International Symposium on Low Power Electronics and Design (ISLPED'97)*, pages 173–177, August 1997.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Company, New York, NY, 1979.

[17] M. Goel and N. R. Shanbhag. Low power equalizers for 51.84 mb/s very high speed digital subscriber loop (vdsl) modems. In *In Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS'98)*, pages 317–326, 1998.

[18] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose processors. In *IEEE Symposium on Low Power Electronics*, pages 12–13, 1995.

[19] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose processors. *IEEE Journal of Solid-State Circuits*, 31(9):1277 1284, September 1996.

[20] J. Goodman, A. P. Dancy, and A. P. Chandrakasan. An energy/security scalable encryption processor using an embedded variable voltage dc/dc converter,. *IEEE Journal of Solid-state Circuits*, 33(11):1799–1809, November 1998.

[21] V. Gutnik and A. P. Chandrakasan. An efficient controller for variable supply-voltage low power processing. In *Digest of Technical Papers, Symposium on VLSI Circuits*, pages 158–159, 1996.

[22] V. Gutnik and A. P. Chandrakasan. Embedded power supply for low-power dsp. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(4):425–435, December 1997.

[23] Z. J. Haas. A communication infrastructure for smart environments: a position article. *IEEE Personal Communications*, 7(5):54–58, October 2000.

[24] W. Heinzelman. *Application-Specific Protocol Architectures for Wireless Networks.* PhD thesis, Massachusetts Institute of Technology, 2000.

[25] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Routing Protocols for Wireless Microsensor Networks. In *Proc. 33rd Hawaii International Conference on System Sciences (HICSS '00)*, January 2000.

[26] M. Hiraki, H. Kojima, H. Misawa, T. Akazawa, and Y. Hatano. Data dependent logic swing internal bus architecture for ultralow-power lsis. *IEEE Journal of Solid State Circuits*, pages 397–401, April 1995.

[27] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (Mobicom'00)*, pages 56–67, 2000.

[28] V. Von Kaenel, P. Macken, and M. G. R. Degrauwe. A voltage reduction technique for battery operated systems. *IEEE Journal of Solid State Circuits*, pages 1136–1140, October 1990.

[29] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for smart dust. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*, August 1999.

[30] A. Klaiber. The technology behind crusoe processors: Low power x86-compatible processors implemented with code morphingtm software. http://www.transmeta.com/crusoe/download/pdf/crusoetechwp.pdf, January 2000.

[31] G. Lakshminarayana, A. Raghunathan, K. S. Khouri, N. K. Jha, and S. Dey. Common-case computation: a high-level technique for power and performance optimization. In *Proceedings of the 36th Design Automation Conference*, pages 56–61, 1999.

[32] P. Lettieri, C. Fragouli, and M. B. Srivastava. Low power error control for wireless links. In *Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'97)*, pages 139–150, 1997.

[33] P. Lettieri, C. Schurgers, and M. Srivastava. Adaptive link layer strategies for energy efficient wireless networking. In *Wireless Networks*, volume 5, pages 339–355, October 1999.

[34] X. Lin and I. Stojmenovic. Power-aware routing in ad hoc wireless networks. In *SITE, University of Ottawa, TR-98-11*, December 1998.

[35] T. Martin and D. Siewiorek. A power metric for mobile systems. In *Proceedings of the 1996 International Symposium on Lower Power Electronics and Design*, pages 37–42, 1996.

[36] L. McMillan and L. A. Westover. A forward-mapping realization of the inverse discrete cosine transform. In *Proceedings of the Data Compression Conference*, pages 219–228, March 1992.

[37] T. H. Meng and V. Rodoplu. Distributed network protocols for wireless communication. In *Proceedings of the 1998 IEEE International Conf. on Circuits and Systems (ISCAS'98)*, volume 4, pages 600–603, 1998.

[38] R. Min et al. An architecture for a power-aware distributed microsensor network. In *Proceedings of the IEEE Workshop on Signal Processing Systems:Design and Implementation (SiPS'00)*, 2000.

[39] R. Min, T. Furrer, and A. Chandrakasan. Dynamic voltage scaling techniques for distributed microsensor networks. In *Proceedings of the IEEE Computer Society Workshop on VLSI*, pages 43–46, 2000.

[40] MIT μ-AMPS Project. http://www-mtl.mit.edu/research/icsystems/uamps.html, 1999.

[41] S. H. Nawab et al. Approximate signal processing. *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 15(1/2):177–200, January 1997.

[42] C. J. Nicol et al. A low power 128-tap digital adaptive equalizer for broadband modems. *IEEE Journal of Solid State Circuits*, 32(11):1777–1789, November 1997.

[43] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization : Algorithms and Complexity.* Dover, 1 edition, 1982.

[44] S. Park and M. Srivastava. Power aware routing in sensor networks using dynamic source routing. In *ACM MONET Special Issue on Energy Conserving Protocols in Wireless Networks*, 1999.

[45] A.-S. Porret, T. Melly, C. C. Enz, and E. A. Vittoz. A low-power low-voltage transceiver architecture suitable for wireless distributed sensors network. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS 2000)*, volume 1, pages 56–59, Geneva, 2000.

[46] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[47] G. J. Pottie. Hierarchical information processing in distributed sensor networks. In *International Symposium on Information Theory*, page 163, 1998.

[48] J. M. Rabaey, M. J. Ammer, J. L. da Silva, Jr., D. Patel, and S. Roundy. Picoradio supports ad hoc ultra-low power wireless networking. *Computer*, 33(7):42–48, July 2000.

[49] T. Rappaport. *Wireless Communications: Principles & Practice.* Prentice-Hall, Inc., New Jersey, 1996.

[50] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. In *Proceedings of IEEE International Conference on Communications (ICC'98)*, volume 3, pages 1633–1639, 1998.

[51] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1333–1344, August 1999.

[52] J. Sacha and M. J. Irwin. Input recoding for reducing power in distributed arithmetic. In *Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS'98)*, pages 599–608, 1998.

[53] S. Singh, M. Woo, and C.S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proceedings of the Fourth Annual ACM/IEEE International*

Conference on Mobile Computing and Networking (MobiCom'98), pages 181–190, October 1998.

[54] A. Sinha. Energy aware software. Master's thesis, Massachusetts Institute of Technology, January 2000.

[55] A. Sinha and A. P. Chandrakasan. Energy efficient filtering using adaptive precision and variable voltage. In *Proceedings of the 12th Annual IEEE International ASIC/SOC Conference*, pages 327–331, September 1999.

[56] A. Sinha, A. Wang, and A. P. Chandrakasan. Algorithmic transforms for efficient energy scalable computation. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pages 31–36, 2000.

[57] K. Sohrabi and G.J. Pottie. Performance of a novel self-organization protocol for wireless ad-hoc sensor networks. In *Proc. Vehicular Technology Conference*, volume 2, pages 1222–1226, May 1999.

[58] M. R. Stan and W. P. Burleson. Bus-invert coding for low power i/o. *IEEE Transactions on VLSI Systems*, pages 49–58, March 1995.

[59] A. Wang and A. Chandrakasan. Energy efficient system partitioning for distributed wireless sensor networks. In *Proceedings of the 2001 International Conference on Acoustics, Speech and Signal Processing (ICASSP'01), (To appear)*, 2001.

[60] A. Wang, W.R. Heinzelman, and A.P. Chandrakasan. Energy-scalable protocols for battery-operated microsensor networks. In *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS'99)*, pages 483–492, 1999.

[61] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: A System Perspective*. Addison-Wesley Publishing Company, 1994.

[62] T. Xanthopoulos and A. P. Chandrakasan. A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization. In *Digest of Technical Papers, Symposium on VLSI Circuits*, pages 11–12, 1999.

[63] T. Xanthopoulos and A. P. Chandrakasan. A low-power IDCT macrocell for MPEG-2 MP@ML exploiting data distribution properties for minimal activity. *IEEE Journal of Solid-State Circuits*, 34(5):693–703, May 1999.

[64] K. Yao, C. Reed, R.E. Hudson, and F. Lorenzelli. Beamforming performance of a randomly distributed sensor array system. In *Workshop on Signal Processing Systems (SIPS 97 - Design and Implementation)*, pages 438–447, 1997.

[65] V. Zyuban and P. Kogge. The energy complexity of register files. In *Proceedings of the 1998 International Symposium on Low Power Electronics and Design*, pages 305–310, 1998.