

A Promise Theory Approach to Collaborative Power Reduction in a Pervasive Computing Environment

Mark Burgess and Frode Eika Sandnes

Oslo University College, Norway
`mark@iu.hio.no, frodes@iu.hio.no`

Abstract. A grid-like environment may be constructed from ad hoc processing devices, including portable battery-powered devices. Battery lifetime is a current limitation here. In this paper we propose policies for minimizing power consumption using voluntary collaboration between the autonomously controlled nodes. We exploit the quadratic relationship between processor clock-speed and power consumption to identify processing devices which can be slowed down to save energy while maintaining an overall computational performance across a collaboration of nodes.

1 Introduction

Power consumption is rapidly becoming the central focus in pervasive computing scenarios where computers are densely packed. Whether one considers a future vision of mobile and embedded devices, in smart homes and workplaces, where power is often supplied by batteries with limited capacity, or more conventional computer systems, power is a fundamental resource to be managed. Pervasive computing environments exist today in Web Hotels and at Internet Service Providers' server rooms and data centres, where power consumption costs and cooling requirements are the main threat to operation. Here the problem is not that power will run out, but that it will raise temperatures to dangerous levels in a confined space. Either way, power consumption is an enemy to efficiency and survivability.

In this paper, we consider strategies for minimizing total power consumption using a paradigm for policy based management that can easily deal with the autonomous nature of devices in a pervasive scenario. This involves voluntary sharing of resources. We use the new framework of promise theory, where the idea is to treat each processor as an autonomous agent in a voluntary cooperative grid[1, 2]. In the autonomous agent point of view, each node has complete control over its own decisions in response to events, and no authority is capable of forcing any node to perform services without its explicit consent. This scenario is clearly appropriate for a pervasive scenario, either in a mobile ad hoc network, or in a server room full of business competitors.

A node or agent relates to others in the environment by making *promises*, or declaring constraints about its behaviour, of various types. To give up its autonomy and allow for external control, a node can simply promise to follow someone else's orders. In this work, we attempt to ask the question: could rationally behaving agents share their abilities to save power using a policy that sacrifices none of their autonomy, and hence minimises the risk of exploitation?

2 Power saving

Portable, consumer electronic products are increasing in computational power with the advances in microprocessor technology. Such devices include mobile phones, portable digital assistants, MP3 players, digital cameras and notebook computers. Increasingly, the boundaries between such products are blurred as they offer overlapping functionalities. As the personal devices become more powerful, they are used to conduct a wider range of tasks, but mostly their processors are idle. It is worth asking whether one could utilize this latent processing power for a beneficial purpose.

Wireless sensor networks of autonomous embedded devices are in a similar predicament; they might also reap the benefits of migratory processing power: distributed processing can be built into the nodes for immediate analysis of data, and perhaps to save power required to transmit large amounts of data back to a base-station over a potentially noisy or insecure wireless network. Distributed facial face recognition in security sensors, intrusion detection analysis, or rule matching could bring large fluctuations in power demand suddenly to a single node amongst many. What if that load could be balanced across several devices in order to save power?

Batteries are often a limiting factor in portable systems. Either the battery is unable to provide a sufficiently long uptime, or the available power capacity is unable to support the desirable hardware components. Although a component may be functionally perfect for the task at hand, the limited power provided by a particular battery may not match the consumption needs of the component.

A vast body of research has been conducted to overcome the limits imposed by batteries such that the operational time of the device can be prolonged [3–6]. These techniques are generally labelled power management. The most obvious approach is to switch off subsystems that are not in use. It is common to see power management in action on notebook computers. The disc will spin down and go to rest if the disc has not been accessed for a specified time-interval [7]. A similar spin-up delay is experienced when the disk is first accessed again. The light emitting screens of notebook computers are often switched off just after a few minutes of user inactivity. Inter-device communication also affects power – WLAN is known to consume large amounts of power while Bluetooth is more economic in power usage. Other techniques used in power aware computing systems include disks that spin at different speeds [7], where a low speed consumes less power than high speed, power consuming memory architectures and finally power aware microprocessors.

One technique commonly cited in the literature includes frequency scaled microprocessors. Such processors allow the clock speed of the processors to be scaled either continuously or in discrete steps. A processor running at a lower clock speed consumes less power than one running at a high clock speed, hence computational power can be traded against power consumption. However, there is an interesting quadratic relationship between the clock speed and the power consumption. The power consumption increases as a quadratic function with clock speed. This physical phenomenon is exploited in a number of power aware scheduling algorithms [8–11]. The most common technique is to adjust the slack time between tasks, i.e. when there is slack time between tasks the task before or after the slack is shifted to cover the slack by slowing down the processor [11]. The net effect is that the same computation is performed in the same time-interval, however power has been saved as one or more of the processor has been running at a lower clock speed for some portion of the schedule.

3 Speed-Power consumption relationship

The quadratic relationship between the processing speed and the power consumption is the prime driving force behind power saving. Generally, the energy stored in a capacitive CMOS circuit is of the familiar form $E = \frac{1}{2}CV^2$, for applied D.C. voltage V . It is the capacitance C that makes power consumption frequency dependent. For an alternating voltage $V(t)$ with fundamental clock frequency f Hertz, one has a Fourier series $V(t) \sim V_0 \sin(2\pi ft) + \dots$, and hence from the fundamental capacitive charge law $Q = CV$, one has the current $I = \frac{dQ}{dt} = C\frac{dV}{dt}$, thus $I(t) \sim fCV(t)$. The power dissipated on releasing this current through resistive elements is, by Ohm's law ($V = IR$),

$$\overline{W} = IV = I^2 R \sim f^2 RC^2 \overline{V^2}(t). \quad (1)$$

Thus, it depends – on average – approximately on the square of the fundamental clock frequency.

The rationale behind the idea of reducing power consumption is as follows. If a processing problem admits *parallelism* then the task can either be computed using one fast processor or several slower and cheaper processors in parallel with the same computational delay. Since the power consumption is quadratic with respect to the clock speed, the total power consumed by the parallel system may be less than the power consumed by the fast processor. Consequently, the system consuming the least power is preferred.

More formally, if a compound task θ comprised of the N partially ordered tasks $T_1, T_2, \dots, T_N$ can be computed in sequential time using one processor P running at a clock speed of S_{high} Hz, then the same compound task can be performed using M processors $P_1, P_2, \dots, P_M$ running at S_{low} Hz. Then, if the sum of the power consumed by the single processor exceeds the power consumed by the parallel processor the parallel processors are used, namely:

$$W(P(S_{\text{high}})) > \sum_{i=1}^M W(P_i(S_{\text{low}})), \quad (2)$$

where $W(P)$ denotes the accumulated power consumed by processor P . The idea of using multiple processors in a single embedded system has been standard practice for over a decade in specialized digital signal processing applications [12–14], and multiple DSP processors can be used to construct such a system. There are even multiprocessor chips such as the classic Texas Instruments TMS320C80, which had four shared memory slave processors and one master processor on a single chip. Recent trends include the multiple core microprocessors which are similar in principle. The option of integrating multiple processors into the same low cost consumer electronics device is increasingly viable with current technology.

The quadratic relationship between the clock-speed and the voltage level can be utilized such that the overall battery level can be prolonged. A grid already comprises a large set of distributed processing nodes. The processing nodes are often highly heterogeneous in terms of function, processing power and processing load. A grid problem can be job farmed into reasonably sized chunks. Sometimes these chunks can be customized in terms of size such that processing elements with a history of slower performance is given smaller computation loads than processing nodes with a history of larger capacity. However, the nature of some problems makes it difficult, if not impossible, to choose a specific chunk-size. Furthermore, in other situations it might be difficult to merge results of varying size. Traditionally, in such situations the processors run until they have finished the allotted work and are then given a new chunk. To achieve synchronisation processing nodes are left idle while waiting for the last results. In this paper the idea of slowing down faster processing nodes such that their performance matches more closely those of the slower processing nodes is explored. The general benefit of slowing down nodes is that they will consume less power.

4 Common currency, cooperative payoff

We wish to turn this discussion into one about policy based management in a framework of voluntary cooperation between autonomous systems. This is representative of both independent organizations in a data centre and personal mobile devices in a public space. The central question now is what would induce the agents to cooperate in reducing the total power budget? What does a device or agent stand to gain by collaborating with its neighbours to exploit parallel processing?

- A mobile device must balance the cost of transmitting a parallelizable job to a neighbouring node against the power saving of exploiting the parallelism at lower clock rates.
- A fixed node in a data centre incurs essentially no additional power consumption for farming out tasks. The potential payoff is a saving that can reduce the total electricity bill for the data centre so that every customer can reduce their expenditure on electricity and cooling. The total share of the final bill can therefore be proportionally reduced for all.

The energy saving in a wired network is somewhat greater here, since cables channel all of the power directly along the waveguide to the receiver. A wireless signal, on the other hand, radiates in all directions and most of its power is wasted. The efficiency of wireless communication falls off as $1/r^3$ and WAN protocols step up the transmission power in fixed steps to compensate as the distance grows. The mobile situation is thus potentially more complex, and perhaps harder to imagine in practice. Nonetheless, in exploratory robotic applications, e.g. planetary exploration, where power is extremely scarce, the issues are even more essential[15].

Imagine that the number of jobs is N and the number of processing devices is P . Further, imagine for simplicity that each job has the same time and space complexity and are computationally identical. The processing delay of computing a job on processor i is given by d_i . Since the processing elements are heterogeneous and have different computational loads the computation delays follows some distribution with mean $\bar{d}$ and a maximum delay of $d_{\max}$. If slack retention is used the processing delay d_i of some task is extended from $\bar{d}$ to $d_{\max}$ by slowing down the processor such that the new mean processing delay is $\bar{d}' \simeq d_{\max}$. To achieve this, the processor speed is reduced by a factor of: $s_i = \frac{d_i}{d_{\max}}$, where a factor of 1 means no reduction, i.e. full speed, and a factor of 0 means that the processor has ceased to operate. The mean speed reduction ratio is therefore $\bar{s} = \frac{\bar{d}}{d_{\max}}$.

The power consumed using a specific processor speed ratio is given by Ks_i^2 where K is a constant. The mean power saved using slack retention per node is therefore

$$W_{\text{saved}} = W_{\text{before}} - W_{\text{after}} = Ks_{\text{before}}^2 - K\bar{s}^2 = K(1 - \bar{s}^2), \quad (3)$$

since $s_{\text{before}} = 1$ (maximum processor speed). The total power saved is therefore

$$W_{\text{total}} = (M \bmod P)K(1 - \bar{s}^2). \quad (4)$$

This holds for the situations where $N > P$, i.e. more-jobs limited-processors and $P > N$, i.e. limited-jobs multiple-processors, since W_{total} reduces to $NK(1 - \bar{s}^2)$. Clearly, the potential for reducing the power consumption in such a computing environment depends on the spread of the computation delay distribution, i.e. the mean is very different to the maximum, and the relationship between the number of jobs and the number of processing elements.

Figure 1 shows the relationship between the reductions in power consumption in relation to mean speed reduction. By reducing the speed by 10% for the case where $N < P$ then 20% of the power consumption is saved, and by reducing the speed by 30% then 50% of the power consumption is saved. However a speed reduction of 30% represents a large spread in the computation delay distribution. Furthermore, for the cases where $N > P$ then the ratio of jobs that are subjected to slack reduction serves as an upper bound for the theoretical achievable reduction in power consumption. i.e. if only 10% of the jobs are subjected to slack retention then it is obviously not possible to achieve more than 10% reduction in power consumption.

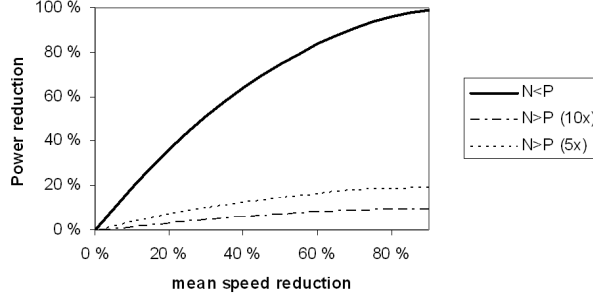


Fig. 1. Reduction in power consumption as a function of mean speed reduction.

5 Promise theory

Policy has two aspects for autonomous systems: voluntary cooperation (with attendant trust issues) and the logic of constraints. What policy rules are needed to encourage the individual agents to cooperate towards a common goal, without having to sacrifice their autonomy?

We use promise theory[1] to discuss this. Promise theory is an attempt to represent policy constraints using an atomic theory of assertions that makes no *a priori* assumptions about agents' willingness to comply with one another's wishes[16, 17]. In promise theory, each assertion of policy is made from a source node to a receiver node, in the manner of a service provided. Promises are therefore labelled links in a graph of nodes representing the autonomous devices in the system.

Three main categories of promise are known to exist. These are basic service promises π , promises to cooperate with someone else's promise (denoted $C(\pi)$), and a promise to accept and use a service provided by another agent (denoted $U(\pi)$). The promises are essentially fixed in time (over many computational epochs). Promises do not represent separate events, or messages, but rather they summarize the entire history of interaction over the interval. Their role is to guide the behaviour of the agents in response to individual events or needs that arise in some unknown manner.

In the formulation of promises, one considers both the topology of the relationships between nodes and the nature of the constraints linking the nodes. Agents can form loose coalitions of nodes, or they can be tightly coordinated depending on the number of promises made. The promises from any sender s to any receiver r we need here are of the general form:

1. s promises r to accept jobs/tasks for completion with delay d_s and return the result.
2. s promises r to cooperate with a promise r makes to a third party, i.e. will behave the same as a neighbour in respect to the party.

3. s promises r to reimburse the power outlay r made on s 's behalf, in some form of currency. This is payment, made perhaps in the form of a reduced bill for power and cooling, or perhaps in some fictitious currency used simply to balance the energy accounts.
4. Additional promises could be numerous in regard to other policies concerned with the independent functioning of the devices, e.g. promises to report alarms to a central node, or to relay network traffic in an ad hoc network, etc.

Extensive details of the specific promise model are not possible within the space allowed. We mention only some summarial remarks.

Promise 1 is the basic promise for task sharing. Without further embellishment it enables nodes to perform power sharing. However, it easily allows nodes to be exploited too. A malicious node could easily send requests that are taken on trust in order to force the receiver to use up its power[2].

Promise 2 allows nodes to organize itself into a 'club' in which groups of nodes will behave similarly. This allows regions of autonomous devices to work together voluntarily, without sacrificing their autonomy.

Promise 1 can be made conditional on reimbursement, so that one programs a tit-for-tat relation into the interaction[18, 19]. One might also make the promise contingent on a return promise that the sending node will actually use the result, having expended energy on it, but such requirements rapidly start to expend the energy budget on mistrust.

Could trust be exploited? Clearly there are several ways in which trust can be abused, both accidentally and by means of ill-conceived policy. If a sender sends a request at too high a power level, it might fool distant nodes into accepting a task that it too expensive to reply to. This would either waste their power, or they would not transmit a return signal strongly enough and the processing would be wasted. Thus a node could maliciously attack such a collective of mobile devices, forcing them to exhaust their batteries on pointless calculation. An explicit promise from all senders to all other receivers to transmit within an agreed power range would codify policy on this point. Alternatively, a reciprocal promise to fairly weight the tasks could be required by receivers that must be in place before expending significant power.

If a central node in the system has unlimited, cheap power then it might be used as an intermediary for brokering transactions between the individual agents. However, such a situation would not work for robotic probes on the Martian surface: in that case, all of the nodes must be considered as equal peers, none of which should be penalised. Control, in this case, must be distributed and based on implicit trust.

Mobile promises are the same as the fixed infrastructure ones, but they add additional issues. Again transmission power is an issue. Promises to stay in range of one another could have a significant saving effect[15, 20].

Three scenarios are shown in fig. 2.

- In figure 2a, the nodes pick a representative node (e.g. a base station) within the nodes (or perhaps externally) and promise that representative to adjust

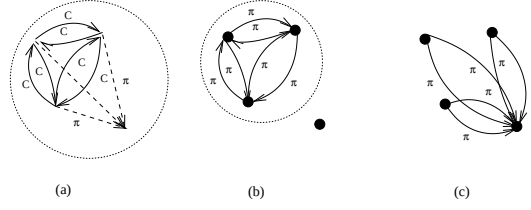


Fig. 2. Three management policy topologies.

their power according to an appropriate sharing algorithm. This binds the nodes together as a group or role in the graph. By the algebra of ref. [16], the collections of agents that make binding bilateral agreements represent a natural *role* within the graph. Typical promises between the agents are cooperative promises to one another to honour the promise to share (which is formally made to the arbiter).

- In fig.2b the agents promise to help one another, independently of an external arbiter. In doing so they place considerable trust in one another. What do they do if they fail to keep their promises to cooperate?
- In fig. 2c the nodes do not communicate directly at all, but always use an external authority to control the scheduling.

6 Promise 1 body remarks

Most literature on power aware scheduling assumes that there is one central power source. However, in a mobile setting there are strong reasons to argue for multiple power sources, where each processor has its own power source. In the context of wearable computers, or exploratory robotic agents, these processors can be distributed at different locations.

We denote these architectures according to the spirit of Flynn's taxonomy: single instruction single battery (SISB), single instruction multiple battery (SIMB), multiple instruction single battery (MISB) and multiple instruction multiple battery (MIMB).

Many-jobs many-processors (MJMP): The many-jobs, many-processors scenario is typical of the traditional grid. A pervasive grid might expect that the overall job to be computed has a lifetime that exceeds the battery life of the processing device. It is therefore natural to introduce the notion of a computational epoch, i.e. a subset or chunk of the overall computation that can be computed using the battery capacity available in the pervasive grid, and the overall goal is to maximize the amount of computation per unit of power. This scenario therefore can be reduced to a limited-jobs many processors scenario, where the limited jobs are the jobs that can fit into a computational epoch.

Reducing power consumption by slack reclamation is only applicable once the number of tasks is less than the number of processors; if the number of jobs

is much larger than the number of processors the potential power savings are moderate. Given a tight power-budget a small overall reduction in performance can result in large power savings due to the speed-energy quadratic relationship. The speed of all the processors should be adjusted proportionally to its load and capability, such that the performance scales evenly across the processing devices. This would require another collaborative promise.

Limited-jobs many processors (LJMP): In the limited-jobs many-processors scenario there is a known set of jobs and a large number of processing elements. The objective is to compute the result in an as short time as possible while at the same time maximize the uptime of the system. This is achieved by splitting the problem into as small chunks as possible for distribution. Based on the computational power and history, it is possible to estimate the finishing time from the size of the problem. According to Amdahls law the completion time of a problem is the sum of the parallel parts and the sequential parts, and clearly the overall problem cannot be solved faster than computation delay of the slowest device in use and consequently faster devices will finish the computation too early and perhaps remain idle. Instead, the clock speed of these devices can be slowed down such that the completion-time of computing the given task on the device matches that of the slowest device. The end result is that the result is computed in the same time, but the overall system has consumed less power.

Many-jobs limited-processors (MJLP): If one has more jobs than processors, each processor must execute more than one single job. Again given a variety of time-variant completion times for the various processing devices, jobs waiting to be computed are assigned processing devices that become idle. It is then only once the last task has been assigned that slack reclamation can be used. This thus becomes a many-jobs many-processors phase followed by a limited-jobs many-processors phase.

Limited-jobs limited-processors (LJLP): A limited-jobs limited-processors scenario can be reclassified as either being a limited-jobs many-processors, if there are the same or more processors than jobs, or a many-jobs limited-processors scenario, if there are more jobs than processors.

7 Conclusion

It is known that, in a regime of authoratative control over a group of devices, it is possible to make power savings to reduce the energy cost of parallel processing tasks. A group of autonomous agents can function similarly by voluntary cooperation in a pervasive computer setting without loss of autonomy.

References

1. Mark Burgess. An approach to policy based on autonomy and voluntary cooperation. *Lecture Notes on Computer Science*, 3775:97–108, 2005.
2. M. Burgess. Voluntary cooperation in a pervasive computing environment. *Proceedings of the Nineteenth Systems Administration Conference (LISA XIX) (USENIX Association: Berkeley, CA)*, page 143, 2005.

3. L. Benini, D. Bruni, A. Mach, E. Macii, and M. Poncino. Discharge current steering for battery lifetime optimization. *IEEE Transactions on Computers*, 52(8):985–995, 2001.
4. L. Benini et al. Extending lifetime of portable systems by battery scheduling. In *Proceedings of Design, Automation and Test in Europe 2001.*, pages 197 – 201.
5. L. Benini et al. Scheduling battery usage in mobile systems. *IEEE Transactions on Very Large Scale Integration (VLSI) System*, 11(6):1136–1143, 2003.
6. L. Bloom, R. Eardley, and E. Geelhoed. Investigating the relationship between battery life and user acceptance of dynamic, energy-aware interfaces on handhelds. In *Proceedings of Mobile HCI 2004*, pages 13–24.
7. S. Gurumurthy, A. Sivasubramaniam, M. Kamdemir, and H. Franke. Reducing disk power consumption in servers with drpm. *IEEE Computer*, 36(12):59–66, 2003.
8. H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez. Power-aware scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, 53:584–600, 2004.
9. J.-J Han and Q.-H Li. Dynamic power-aware scheduling algorithms for real-time task sets with fault-tolerance in parallel and distributed computing environment. In *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium, 2005*, pages 6–6.
10. D. Zhu, R. Melhem, and B. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems. In *Proceedings of 22th IEEE Real-Time Systems Symposium, 2001*, pages 84–94.
11. D. Zhu, R. Melhem, and B. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation. *IEEE Transactions on Parallel and Distributed Systems*, 2003.
12. F.E. Sandnes and O. Sinnen. A graph transformational approach to the multiprocessor scheduling of iterative computations. In *Proceedings of the Fourth International Conference of Parallel and Distributed Computing, Applications and Techniques, Chengdu, China, 2003*, pages 571–576.
13. F.E. Sandnes and O. Sinnen. A new scheduling algorithm for cyclic graphs. *International Journal of High Performance Computing and Networking*, 2004.
14. F.E. Sandnes and O. Sinnen. Stochastic dfs for multiprocessor scheduling of cyclic taskgraphs. In *Proceedings of Parallel, Distributed Computing, Applications and Technologies PDCAT 2004, Singapore, 2004*, pages 342–350.
15. J.M. Hecrickx et al. Rigidity and persistence of three and higher dimensional forms. In *Proceedings of the MARS 2005 Workshop on Multi-Agent Robotic Systems*, page 39, 2005.
16. M. Burgess and S. Fagernes. Pervasive computing management i: A model of network policy with local autonomy. *IEEE eTransactions on Network and Service Management*, page (submitted).
17. M. Burgess and S. Fagernes. Pervasive computing management ii: Voluntary co-operation. *IEEE eTransactions on Network and Service Management*, page (submitted).
18. R. Axelrod. *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration*. Princeton Studies in Complexity, Princeton, 1997.
19. R. Axelrod. *The Evolution of Co-operation*. Penguin Books, 1990 (1984).
20. J.M. Hecrickx et al. Structural persistence of three dimensional autonomous formations. In *Proceedings of the MARS 2005 Workshop on Multi-Agent Robotic Systems*, page 47, 2005.