Computation Hierarchy for In-network Processing

Ram Kumar Vlasios Tsiatsis Mani B. Srivastava
Networked & Embedded Systems Lab
Electrical Engineering Dept., UCLA
LA, CA 90095 USA
{ram, tsiatsis, mbs}@ee.ucla.edu

ABSTRACT

In this paper, we explore the network level architecture of distributed sensor systems that perform in-network processing. We propose a system with heterogeneous nodes that organizes into a hierarchal structure dictated by the computational capabilities. The presence of high-performance nodes amongst a sea of resource constrained nodes exposes new tradeoffs in the efficient implementation of network-wide applications. The introduction of hierarchy enables partitioning of the application into sub-tasks that can be mapped onto the heterogeneous nodes in the network in multiple ways. We analyze the tradeoffs between the execution time of the application, accuracy of the output produced and the overall energy consumption of the network for the different mapping of the sub-tasks onto the heterogeneous nodes in the network. We evaluate the performance and energy consumption of a typical sensor network application of target tracking via beamforming and line of bearing calculations on the different nodes. Our experiments show that more than 95% of time on average, the hierarchical network outperforms a homogeneous network for approximately the same energy budget.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design; C.3 [**Computer Systems Organization**]: Special Purpose and Application-based Systems;

General Terms

Performance, Design, Experimentation

Keywords

In-network processing, Hierarchical architecture, Computation Offloading

1. INTRODUCTION

1.1 Motivation

Advances in VLSI, MEMS and wireless networking have enabled device miniaturization and power efficiency [1]. These advances pushed forward the emergence of a new class of networked embedded systems, namely sensor networks [2][3][4]. Sensor networks present the challenge of building large scale distributed systems that are tightly coupled with the physical world in an extremely resource constrained environment and need to function for a long time providing a desirable level of performance.

Some of the common applications envisioned for sensor networks range from monitoring of habitats for specific birds and animals [13][7], detection of contaminants and pollutants in fluids [14], detection of intrusion and tracking of targets [15][16]. All these applications require processing of the raw data collected by the distributed sensors in the field. The communication computation energy tradeoffs [2][5] have shown that *in-network processing of the sensed data is more energy efficient* in comparison to the centralized server model where all the nodes simply collect and forward data to a powerful user node.

Another interesting observation is that the *complexity of the processing varies significantly from one application to another and also within an application.* For example, the calculation of the maximum value of the temperature observed over a region does not demand excessive computation power. However, in habitat monitoring applications, the identification of a bird based on the recorded audio waveform requires the computation of the correlation of the spectrogram of the input waveform with a pre-stored spectrogram. This is a very demanding signal processing computation [7] for even the most powerful sensor nodes available.

There is a rich diversity of sensor node platforms that are currently available [9][10][38][39]. Later in the paper we present two different platforms and their characteristics. The platforms cover a large range of MIPS, which is a metric that measures the rate of instruction execution in processors [17]. Also, some of them have specialized architectures (for e.g. custom H/W on FPGA) which make them efficient for a certain class of applications. However, a single node platform alone is not efficiently scalable to the large dynamic range of the computational complexity of the sensor network applications. The processors with higher MIPS have higher clock rates which directly translate into higher power consumption. Therefore, such processors are inefficient for performing computations that are not very demanding. Conversely, the processors with low MIPS are not suitable for demanding applications.

The cost of the sensor network is an important factor to be considered during design time. It is a norm for the sensor networks to be comprised of a few hundreds of nodes. Therefore, the cost of an individual node should be kept at a minimum to reduce the overall system cost. Multi-processor node architectures can be made to scale to the current computational load desired of them, but their overall components, design and fabrication cost would be very high. Instead of making every node in the network scalable, economically, the more efficient solution is to have a heterogeneous network comprising of nodes of varying levels of computational capabilities. Such a network would be scalable at a lower average cost per node.

1.2 Hierarchical Network Architecture

Instead of choosing a single hardware platform that makes a particular set of trade-offs, we believe an effective design is one that uses a *hierarchical architecture* consisting of heterogeneous hardware platforms with varying computational capability. Larger nodes with abundant resources and higher power consumption (henceforth referred to as *macro-nodes*) can be sparsely deployed across the network to assist the smaller, resource constrained nodes (henceforth referred to as *micro-nodes*) which would be densely deployed. The concept is analogous to caches in the memory sub-system of modern computer architecture [17]. Caches are faster, smaller and expensive memories which exploit the spatial and temporal locality of memory references. The caches are backed up by cheaper slower memories with larger storage capacity. Thereby, the entire memory system appears to be as fast as the cache but as vast as the secondary storage. Similarly, the hierarchical network would have a high coverage and low energy consumption that would resemble a sea of micro-nodes but yet appears to have a computational capability and performance comparable to the macro-nodes. We exploit the spatial locality of the algorithms by creating clusters in the network comprising of at least one macro-node that would execute the computationally intensive atomic units of the application.

The design challenges in instrumenting such a network are numerous. First and foremost is the partitioning of the application into sub-tasks that are executed at different nodes in the network. The sub-tasks are mapped onto nodes that are most efficient for executing them. The essential criteria dictating the mapping is the architecture of the macro-nodes and micro-nodes and the performance and the energy consumption of the sub-tasks on them.

Secondly, we need to determine the composition of the network in terms of the number of nodes of every kind. This decision is a tradeoff between the performance of the system and its overall cost. The latency of the data transfer between the sub-tasks mapped onto different nodes has a significant impact on performance. This latency can be minimized by reducing the average hop count from a micro node to a macro node which in turn implies increasing the number of macro nodes in the network. The overall system cost increases due to the increase in the number of macro nodes.

Lastly, hierarchical architecture of the networked system essentially permits computation offloading by the micro-nodes onto the macro-nodes in the network. However, the offloading requires a self-configuration algorithm wherein the network is divided into clusters of micro-nodes headed by a macro-node. The micro-nodes in the cluster would offload their computation onto the macro-node which forms the cluster head. Research effort is required in developing efficient and robust clustering algorithms.

1.3 GALORE: A Hierarchical Network

An example of a hierarchical network is the system developed in the context of the GALORE project [35]. The architecture is hierarchical, comprising of random and dense deployment of MICA motes [8] [9] at the lower level. The higher level of the hierarchy consists of a sparse and random deployment of computationally capable nodes like the iPAQ [10] and the IQinvision [11] camera nodes. The IQinvision cameras are networked and they can share their computational resources [12]. The iPAQ can communicate with the MICA motes through the MoteNIC interface [24]. The iPAQ can also communicate with the camera through the serial interface or a wired ethernet connection. The communication links between the different layers in the hierarchy is illustrated in Figure 1. The iPAQ is currently serving only as a proxy for a smaller form-factor node, Cerfcube[40]. The architecture of both the nodes is similar except that the iPAQ has a LCD screen which makes the debugging easier.

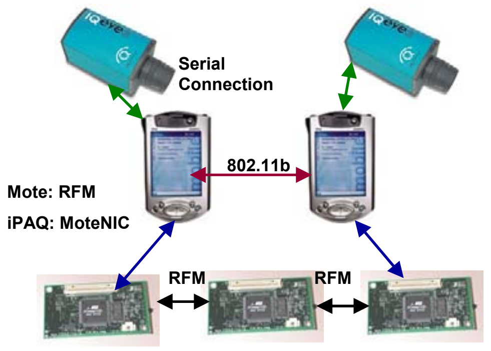


Figure 1: GALORE Hierarchical Network

The objective of the GALORE system is to perform unsupervised surveillance with multiple sensing modalities. In order to support comprehensive surveillance and situational awareness, imagers are required in some form. Important features of the target such as markings and occupants can be ascertained only by the use of imagers. However, since the imagers have a higher power consumption and lower utilization in comparison to the other system components, it is impractical for them to operate continuously. Therefore, some low-power sensing modality must be used for cueing the image sensors. We propose to implement an acoustic target tracking system on MICA motes and Compaq iPAQs using beamforming and line of bearing calculation. Upon detection and localizing a target, the acoustic sensors cue the image sensors to take over.

In the next section, we describe the acoustic tracking application via beamforming and describe the computational complexity of the operations. In section 3, we compare the architectural features of the two commonly used sensor node platforms and classify them. The beamforming algorithm was implemented on all the sensor node platforms discussed and its measured performance and energy consumption is reported in section 4. In section 5, we illustrate the performance improvements that can be realized at the same energy budget by mapping the sub-tasks onto different nodes in the network. We also comment on the deployment densities of the different heterogeneous nodes comprising the network. We conclude with a discussion on future work.

2. ACOUSTIC TARGET TRACKING

The applications that would benefit from the introduction of hierarchy are those which involve a significant amount of in-network processing. Such applications require either extensive resources like memory which is not available on the micro-nodes or are computationally intensive that it takes prohibitively large computation time and energy on the micro-nodes. Typically the execution time of such applications on the micro-nodes is much more than the communication time required in sending the inputs to the macro-nodes. There are numerous examples of such applications like target identification [7], tracking [16] etc. We have chosen acoustic target tracking as our driver application to illustrate the tradeoffs introduced by the presence of computational hierarchy in the network. The target tracking via time domain delay and sum beamforming requires significant in-network processing of the sampled acoustic data and therefore is a good representative for the class of applications that would benefit from computational hierarchy.

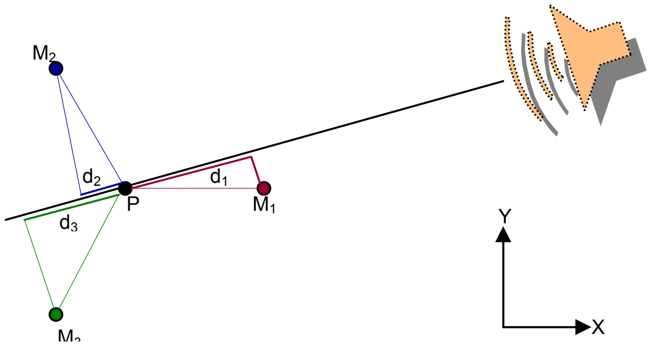


Figure 2: Beamforming on sensor nodes

2.1 Delay and sum beamforming

Acoustic beamforming algorithms estimate the line of bearing to distant acoustic emitters by time-shifting signals from microphones at known relative locations to form beams from selected directions [6] [18].

A set of three nodes M_1, M_2 and M_3 (refer to Figure 2) organize themselves into an array for sampling the acoustic waveform. The line of bearing is computed at the point P (refer to Figure 2) which is the center of the circle passing through the three nodes. The algorithm initially iterates over the possible angles of arrival Φ. For each angle, the distances d_1, d_2 and d_3 are computed. These distances are the projections of the radius vector onto the possible direction of arrival. The distances d_1, d_2 and d_3 are converted into sample delays N_1, N_2 and N_3 according to (1) where C represents the speed of sound in the medium and S represents the sampling rate of the acoustic signals.

$$N_i(\Phi) = \left(\frac{d_i(\Phi)}{C} \right) S \quad \forall\, i \in \{1,2,3\},\ \forall\, \Phi \in \textit{Search Angles}$$

(1)

At every node M_i, the sample delays N_i are computed for every possible angle of arrival and stored in array indexed by the angle of arrival Φ. Therefore $N_i(\Phi)$ represents the number of sample delays for node M_i when the direction of the arrival of the acoustic signal is Φ.

Beamforming is accomplished by time-shifting the signals from an array of microphones to a common position with time delays prescribed by (2), for an assumed LOB Φ. Beams are formed for a number of LOB search angles. The gain of the beam is computed according to the following set of equations:

*S: Sampling Rate, T: Sampling Duration, K = T * S = Number of samples*

Each microphone samples from n = 0 to n = K samples.

$S_i(n)$ = Samples recorded at microphone i

$$A(n,\Phi) = S_1(n - N_1(\Phi)) + S_2(n - N_2(\Phi)) + S_3(n - N_3(\Phi)) \quad (2)$$

Number of common samples = K + Min(N_1, N_2, N_3) – Max(N_1, N_2, N_3) = P

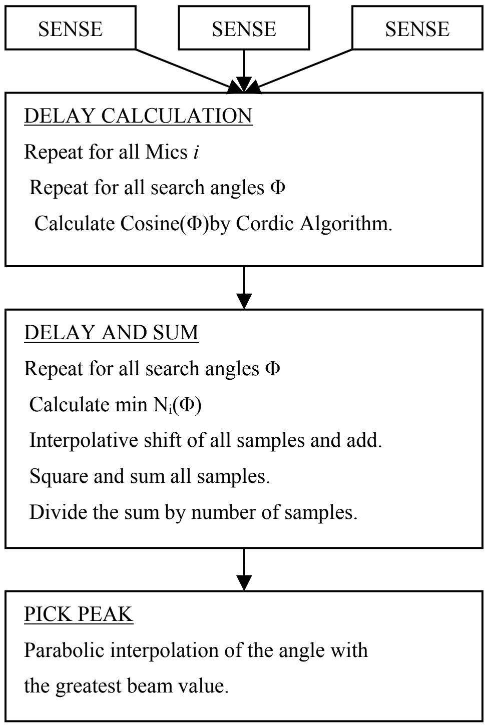
$$G(\Phi) = \sum_{n=Max(N_1,N_2,N_3)}^{K+Min(N_1,N_2,N_3)} \left| A(n,\Phi) \right|^2 / P \quad (3)$$

As the search angle approaches the correct LOB of the emitter, the signals add up in the correct phase. The search angle with the maximum power in the delay-summed signal is selected as the estimated LOB. The LOB estimate is refined by a parabolic interpolation over the adjacent search angles. The RMS error of the LOB estimate reduces and hence the tracking accuracy increases with the increase in the number of search angles in the algorithm [6].

2.2 Computation Complexity Analysis

The beamforming application has been implemented using fixed-point arithmetic routines [6]. The floating point version of the application was not used because the sensor node platforms comprise of integer units only and all floating point operations will be emulated in software. Therefore, the performance of the application depends only on the architectural features of the nodes and is independent of any software library implementation. The flow graph representing the different steps involved in the application is denoted in Figure 3.

Sensing operation involves periodic sampling of the signals received from the microphone and performing an A/D conversion to store the digitized sample in a buffer. The sampled data bit-width is 10 bits but it is further scaled to 8 bits in order to filter out low amplitude noise in the signal. The data is sampled at a rate of 1 KHz and a total of 64 samples are collected in an interval of 62.5 ms. The sampling rate is determined by the frequency of the acoustic signal produced by the target.



SENSE
SENSE
SENSE
DELAY CALCULATION
Repeat for all Mics i
Repeat for all search angles Φ
Calculate Cosine(Φ)by Cordic Algorithm.
DELAY AND SUM
Repeat for all search angles Φ
Calculate min Ni(Φ)
Interpolative shift of all samples and add.
Square and sum all samples.
Divide the sum by number of samples.
PICK PEAK
Parabolic interpolation of the angle with
the greatest beam value.

Figure 3: Task-graph of Beamforming Application

We analyzed the operations in all the computationally intensive steps of the algorithm.

Delay Calculation - Cordic Algorithm: The first step of the computation process is to calculate the delay of the three microphones at the central point in terms of the number of samples. The delay calculation involves computation of the cosine of an angle. Due to the absence of a floating point unit, the cosine is implemented through a cordic routine. In order to improve the precision, the delay is computed and stored as 32 bit fixed point integer. This computation is performed over all possible search angles for all the microphones. The delay calculation is a one-time computation for a cluster of three microphones. Once computed, the delay values can be re-used for the subsequent iterations of the application with the input data from the same cluster.

Interpolative shifting - 32 Bit Integer Multiplication: The shifting by a fraction of time between samples is achieved by linear interpolation of the samples. Interpolative shifting requires a 32-bit multiplication between the difference of two adjacent sample values and the fractional part of the delay. This operation is performed over all search angles, all microphones and all data-samples. It constitutes the bulk of the execution time.

Sum and squares – 32 Bit Integer Multiplication: The calculation of the gain requires square and sum of all the samples in the signal. The computation of the square requires a 32-bit multiplication. This operation is performed for all angles and for all samples.

Gain Calculation – Division: The computed sum and squares needs to be divided by the total number of samples in order to calculate the gain for every angle. This requires a 32 bit integer division operation. This operation is performed once for every angle.

3. SENSOR NODE ARCHITECTURES

The computation offloading enabled due to the introduction of hierarchy will be beneficial only if the macro-nodes have superior performance and abundant resources than micro-nodes. In this section, we analyze the architectural features that distinguish the different classes of nodes. We do so by comparing the architecture of the commonly used sensor node platform, Mica Mote and a convenient computation platform, Compaq iPAQ.

A sensor node platform comprises of three interacting subsystems:

Compute Subsystem: The primary task of the computation subsystem is the execution of the applications. The operating system in the sensor nodes is at the heart of the computation sub-system. It is responsible for scheduling operations and managing resources.

Communication Subsystem: The communication subsystem manages the data transfer and signaling between the sensor nodes. It maintains the radio state and executes the network protocols.

Sensing Subsystem: The sensing subsystem is responsible for managing the state of the multiple sensors hosted by the node. The state management implies powering the sensor on or off and maintaining a data transfer channel between the sensor and the memory to store the samples collected by the sensors.

The sensor nodes differ primarily in the implementation of the three sub-systems. The following sub-sections contain an overview of the common sensing platforms.

3.1 Micro-node: Berkeley Mica Motes

The MICA motes [9] from Berkeley are the current generation low-power sensor nodes. It is a COTS node with very limited resources and low energy consumption. The block diagram of the MICA node is shown in Figure 4. The key components constituting the node are a main board hosting Atmel's ATMEGA128L (AVR) [19] microcontroller, RF Monolithics TR1000 radio [20] and a sensor board hosting a large array of sensors.

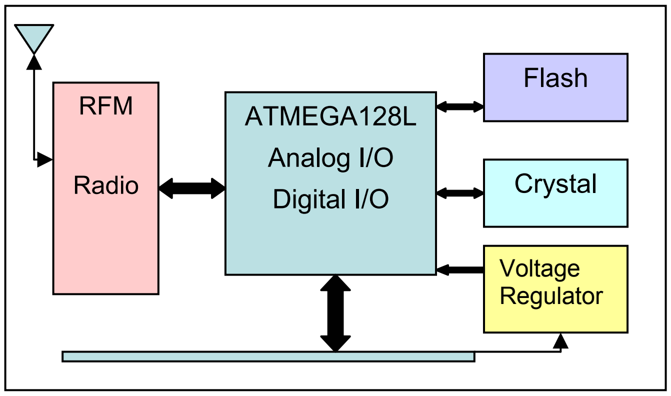


Figure 4: Block diagram of the MICA Mote

The ATMEGA128L is a RISC CPU that delivers 4 MIPS at 4 MHz [19]. The on-chip memory on the ATMEGA128L is limited comprising of only 4 Kbytes of SRAM (used as data-memory) and 128 Kbytes of flash memory (used for program storage). The micro-controller hosts a large number of peripheral devices such as timers, ADC (Analog to Digital Converter), SPI and UART etc. that enhance the functionality of the CPU and provide means for communicating with the other components on the board. The ATMEGA128L supports multiple power modes with varying current consumption. The TR1000 radio [20] from RF Monolithics is the core of the communication subsystem. It is a short range radio operating at 916.7 MHz that supports data transmission rates up to 115.2 kbps. The radio supports a low power consuming sleep mode which makes it suitable for sensor node. The sensor board hosts a photocell, thermistor, magnetometer, accelerometer, microphone and a sounder [9]. The sensor board is connected to the main board through a 51-pin connector. The power supply to each sensor is individually controlled using power switches. The sensor board also contains analog circuits to amplify the signals received from the different sensors. The analog signals are fed to the on chip ADC of the ATMEGA128L.

3.2 Macro-node: Compaq iPAQ

The iPAQ H3600 [10] is a small form-factor PDA with versatile expansion capabilities. The hardware organization of the iPAQ is proprietary and hence the block diagram is not available. However, we present the block diagram of Itsy-v2 PocketPC [21] by Compaq. The Itsy-v2 has a very similar system architecture and organization to the iPAQ and therefore is a good reference.

The iPAQ is driven by the StrongArm SA-1110 [22] microprocessor. The SA-1110 is a 32-bit RISC CPU delivering 235 Dhrystone 2.1 MIPS [23] at 206 MHz. The processor has an on-chip I-cache of 16 Kbytes and a D-cache of 8 Kbytes. There is an integrated dual-slot PCMCIA controller built into the chip which enables the addition of extra peripheral devices to the chip.

SA-1110 has advanced power-management logic. The processor has multiple sleep and idle states and also supports dynamic voltage scaling.
The wireless communication in the iPAQ is through the PCMCIA channels. There are currently two possible setups for enabling the radio link in the iPAQs. The first method is to use the 802.11b standard based WLAN cards. The alternate method is to connect a MICA mote with the RFM radio to the iPAQ through a RS-232 serial link. The RS-232 serial link card plugs into the PCMCIA slot of the iPAQ. This arrangement is known as the MoteNIC [24].

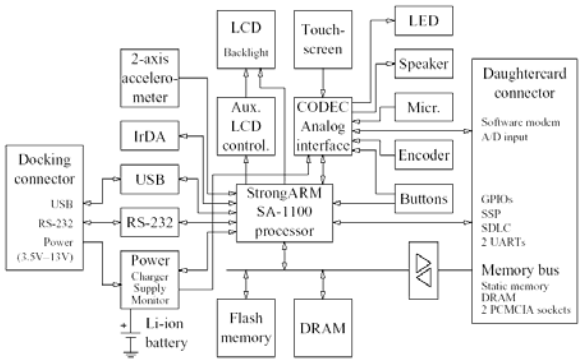


Figure 5: Itsy v2 Architecture

The iPAQ can be used as a sensing platform by attaching a host of sensors to the serial port of the device. For example, the magnetic sensor HM2300 from Honeywell [25] has a serial interface through which it can communicate with the iPAQ. In addition, the iPAQ contains an audio-system that can act as an acoustic sensor. The audio-system comprises of a Philips UDA1341 [26] codec chip that provides A/D and D/A functionality. The codec chip is connected to an on-board microphone that is used for sensing the acoustic signals. The codec has a programmable sampling rate. The acoustic samples upon sensing are streamed to the main CPU through a serial bus interface.

3.3 Platform Comparisons

Having presented the key components of the architecture of all the two platforms, we now compare them along the axes of performance, memory resources, power consumption and cost.
The overall performance of the system is dependent on the processing core of the architecture. The capability of the cores is summarized in Table 1 [19] [22]. The SA-1110 has a double advantage over ATMEGA128L. In addition to a higher clock rate and hence a higher MIPS, the ALU on the SA-1110 is 32 bits wide. Therefore, operations on 32-bit integers require fewer cycles to execute on SA-1110. Signal processing algorithms employing fixed point representation of the data benefit from the wider ALU significantly.

Table 1: Performance comparison of processing core

Platform	Core	MIPS	MHz	Data Path
MICA	ATMEGA128L	4	4	8 bits
iPAQ	SA-1110	235	206	32 bits

The memory is a very crucial resource in embedded systems. The total available data memory on the MICA motes is limited to only 4 Kb. Therefore, they are not suited for signal processing applications that require large buffers. The iPAQ has abundant memory resources in the form of on-chip memories and on-board SRAM and FLASH chips. The 32-bit memory bus increases the processor memory bandwidth.

Table 2: Computation core power consumption

Platform	Power
MICA, ATMEGA128L, 4 MHz, 3 V	15 mW
iPAQ, 133 MHz, 1.3 V	246.4 mW
iPAQ, 206 MHz, 1.5 V	387.3 mW

Since energy consumption is crucial to sensor networks, we also present the power consumption for the two platforms in Table 2 [19] [22]. The higher capability of the SA-1110 processor comes at the cost of higher power consumption.

Table 3: Acoustic sampling power consumption

Platform	Power
MICA, microphone + AVR ADC + Amplifier	27 mW
iPAQ, UDA 1341TS + SA-1110 (206 MHz)	445.8 mW

The power consumption of the acoustic sub-system is summarized in Table 3. This value includes the power consumed by the microphone, ADC and the computation core during the process of acoustic sampling. The values for the MICA mote were obtained by measuring the current consumption of the node while sampling acoustic signals at 1 KHz. The iPAQ values were obtained from the data-sheet current consumption numbers of the components [22] [26]. The iPAQ sampling rate was set to 4 KHz.
Based upon the numbers obtained in Table 1, Table 2 and Table 3, we can conclude that operations with low computation requirement like thresholding of a signal etc. are more suited to be performed on MICA motes due to its lower power consumption for sensing and computation. But compute intensive operations are better suited for iPAQ class platforms. Even though these platforms have higher power consumption, the performance improvement is more dominating due to resource-rich architecture.

4. IMPLEMENTATION

4.1 Performance Measurements

The performance measurements of beamforming application for the individual platforms are given below.

4.1.1 MICA Mote

The beamforming application was compiled using the NesC-1.1 compiler [27] targeted towards the ATMEGA128L micro-controller. The entire application was implemented as a set of tasks in the TinyOS-1.x [28]. The execution time for the application was measured by capturing the trace of an IO signal from the ATMEGA128L which was asserted every time the algorithm was initiated and was de-asserted upon the completion of the algorithm. The averages of the observed values are plotted in the graph shown in Figure 6. The execution time varies linearly with the number of search angles. This is expected because the computationally intensive operations of the application depend linearly upon the number of search angles. The computation takes 2098 ms for 90 search angles. Such a high value makes MICA mote an inefficient platform for performing the beamforming application.

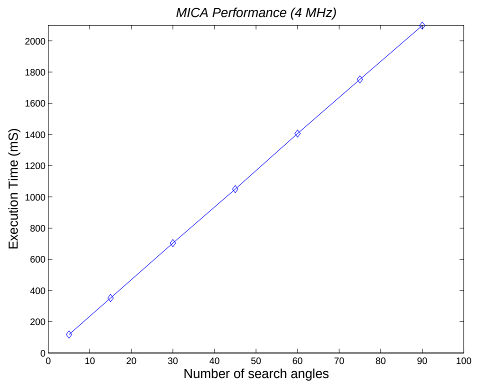


Figure 6: Mica Mote Performance

4.1.2 Compaq iPAQ

The application was compiled using the arm-linux-gcc compiler for the SA-1110 processor. The system was running the FAMILIAR Linux version [29] operating system. The execution time of the application was measured using the *gettimeofday()* function of the linux system library for a clock frequency of 206 MHz. The computation takes 3.2 ms for 90 search angles. The execution time of the application for the other clock frequencies was measured using JouleTrack [30], a performance and power analysis tools for the StrongARM SA-1110 processor. The execution times are plotted in Figure 7.

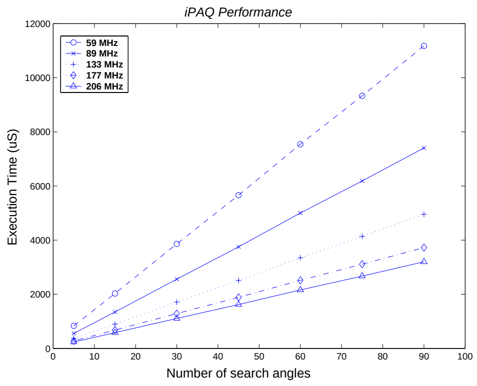


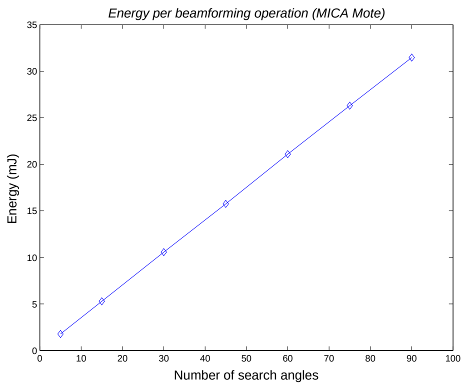
Figure 7: iPAQ performance

4.2 Energy Measurements

Energy measurements were carried out for the computation of LOB, acoustic sampling and the data transfer over the radio link.

4.2.1 Beamforming Algorithm

The energy consumption for the MICA platform was obtained by sampling the current drawn by the AVR core for the duration of the computation. The averaged drawn current was multiplied with the supply voltage and the computation time to obtain the total energy consumed. The iPAQ energy consumption was measured using Jouletrack [30]. The operating voltage and the frequency was varied for the iPAQ. The energy measurements were carried out for different number of search angles in the Beamforming algorithm. The obtained results are summarized in Figure 8.



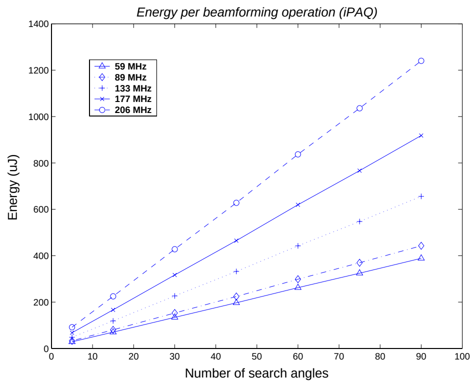


Figure 8: Beamforming energy consumption

4.2.2 Acoustic Sampling

The target tracking application requires the collection of 64 samples of the audio signal sampled at the rate of 1 KHz. Table 4 summarizes the energy required for the acoustic sampling task.

Table 4: Energy and latency of acoustic sampling

Operation	Energy	Time
MICA Mote: 8 bit audio sampling	1.688 mJ	62.5 ms
iPAQ: 16-bit audio sampling	27.844 mJ	62.5 ms

The sampling energy for iPAQ is much higher than for the MICA motes due to the much higher power consumption of the SA-1110 CPU and the audio codec. This is primarily because the sampling operation in the iPAQ is implemented as blocking read function by the audio port driver. The processor is active during the entire duration of sampling and therefore consumes much higher energy. On the MICA mote, the processor draws much lower power during sampling. Also, the audio sensing sub-system in the iPAQ consumes much higher power due to the presence of the UDA-1341TS audio codec.

4.2.3 Wireless Data Transfer

The data transfer for the beamforming application requires transmitting a 64 byte buffer from the sensing node to the node performing the beamforming operation. The data transfer is done by splitting the buffer into three packets. This is because the physical layer packet size payload is constrained to be only 29 bytes for reliable packet transmission in the Berkeley Comm Stack in TinyOS-1.x. The effective data-rate obtained over a link was 13.3 Kbps. The energy measurements were carried out by sampling the current drawn during the transmit and the receive of the buffer. From this, the energy of transfer of the buffer over a single hop was obtained to be 6.2 mJ. The values of the operations are compared in Figure 9.

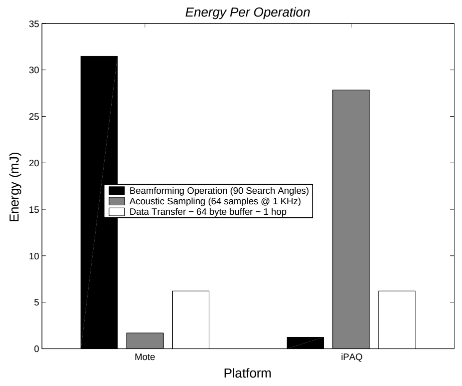


Figure 9: Energy per operation for acoustic target tracking

5. ANALYSIS AND RESULTS

5.1 Task Mapping: Energy Latency Trade-off

Task mapping is the process of assigning tasks to the macro-nodes and micro-nodes in the network. It determines the set of operations that are to be performed locally at the micro-nodes and the set of operations that are to be performed at the macro-nodes.

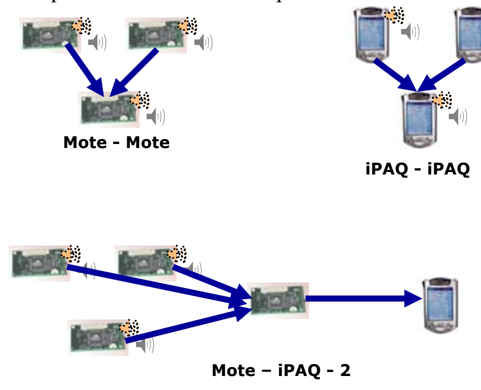


Figure 10: Network Topologies

We map the three sub-tasks of acoustic target tracking onto a network comprising of MICA motes and iPAQs. The three mapping scenarios considered are:

Mote-Mote: Acoustic sampling on MICA motes and local beamforming on one of the three MICA motes that collected the samples.

iPAQ-iPAQ: Acoustic sampling on the iPAQs and local beamforming on one of the three iPAQs that collects the samples.

Mote-iPAQ-x: Acoustic sampling on the MICA motes and beamforming on a iPAQ that is one or more hops away from the motes collecting the samples. The suffix x denotes the number of hops between the audio sampling motes and the iPAQ.

The topologies chosen for evaluation are as shown in the Figure 10.

For each scenario, we calculate the total energy consumed for calculating one beam angle and the total latency of the computation. The total energy consumed is the sum of acoustic sampling energy, acoustic buffer transfer energy and the beamforming computation energy. The total latency of the application is obtained by summing up the computation latency and the network latency for wireless data transfer. The results are show in Figure 11.

The energy measurements in Figure 9 and Figure 11 indicate that for very small number of search angles, it is most energy efficient to map the sampling and the beamforming sub-tasks to the motes. However, increasing the number of angles increases the computation energy on the motes drastically, so much so that it becomes cheaper to transfer the sensed data over the network to an iPAQ. Hence, the mapping of the sampling sub-task to the motes and the beamforming sub-task to the iPAQ (which is one or two hops away) is more energy efficient even with the higher communication energy cost. But when the number of hops increases to three, the communication energy begins to dominate and the initial configuration with all the sub-tasks mapped onto the motes becomes energy efficient. The configuration with both the sampling and the beamforming sub-tasks mapped to the iPAQs has the highest energy consumption. This is primarily due to the high cost of performing acoustic signal sampling on the iPAQs.

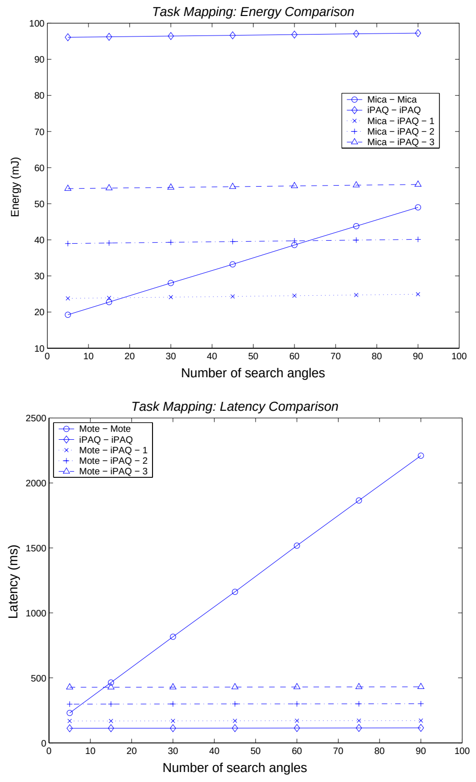


Figure 11: Mapping latency and energy

The latency has two components, the computation latency and the network latency. The computation latency is obtained from the performance measurements of the beamforming application. The network latency is calculated assuming global time synchronization between the nodes [34]. This requirement is not very strict for the current application because we require micro-second accuracy time synchronization for the acoustic sampling operation. Also there is no other cross-traffic in the network. The latency over a link for a node is computed using the packet size and the data-rate of the radio. The three sampling micro-nodes coordinate amongst themselves to decide the order of data-transfer to the macro-node. We have implemented an ordering based upon the node identities. The iPAQ-iPAQ mapping of both sensing and beamforming sub-tasks on the iPAQs has the minimum latency. The mote-iPAQ-1 mapping of sensing on motes and beamforming on iPAQs has a higher latency due to the greater overhead of communication, as in this case, three buffers of 64 bytes need to be transferred over one hop versus only two buffers in the previous case. Similary, mappings mote-iPAQ-2 and mote-iPAQ-3 have higher latencies. The mote-mote mapping of both sensing and beamforming on the motes has the highest latency due to the poor performance of the ATMEGA128L in performing the beamforming sub-task. By observing carefully, it can be easily concluded that the computation latency on iPAQ is negligible in comparison to the network latency. Therefore, there is not much impact of increasing the number of search angles. However, the computation latency constitutes the bulk of the overall latency on the MICA motes

5.2 Macro-node Density

From the results presented in the previous sub-section it is clear that the overall energy consumption of computation offloading at three hops and beyond is more than with no offloading. The number of hops from a micro-node to the closest macro-node is determined by the density of the macro-nodes in the network relative to the micro-nodes and their deployment. We consider only the random deployment of iPAQs and motes in a uniformly distributed manner over the network terrain.

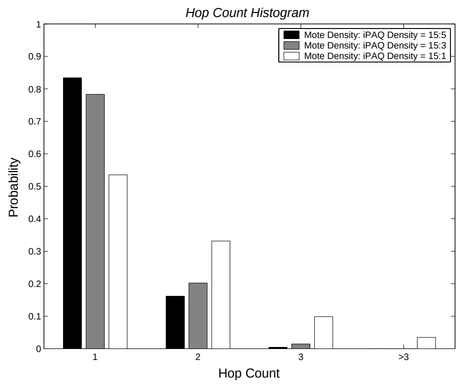


Figure 12: Hop count histogram

We simulated a hierarchical network on NESLsim [36], a PARSEC based sensor network simulator. In our simulations, we kept the density of the micro-nodes in the network to be a constant. The density was chosen to ensure connectivity of the micro-node network with a very high probability. A total of 120 micro-nodes were deployed randomly with a uniform distribution over a square terrain of dimension L (= 100). We varied the density of the macro-nodes in the network. Events were generated randomly throughout the network and a cluster of three micro-nodes that was closest to the event was chosen to sample it. The hop-count is equal to the length of the shortest path to the closest macro-node from the cluster of three micro-nodes. The simulations were averaged over 1000 networks and 10 events were generated for each network. A histogram of the observed hop-counts for varying densities of the macro-nodes is show in Figure 12.

The hop count is the significant determinant of the overall network latency. The delay due to medium access would be much lower than the transmission times due to two factors. First, the three micro-nodes that sample are time synchronized and they transmit the data in a coordinated manner to the macro-node. Second, there is no other cross traffic in the network to pose any contention for the wireless channel. Retransmission in the case of packets being dropped is not required. The macro-node simply flushes the incomplete buffers and does not compute the LOB. From the histogram, we can infer that in a hierarchical network with only one iPAQ for a cluster of 15 motes, the hop count to the closest iPAQ is more than 3 hops only about 5% of the times. Barring these cases, the performance and the energy consumption of a hierarchical network would be superior to all Motes or an all iPAQ network (about 95% of the instances).

6. RELATED WORK

Computation offloading has been studied in the context of low-power handheld systems. [31] uses profiling information during computation time and data sharing at the level of the procedure calls to construct a cost graph for a given application program. A partitioning scheme is then applied to statically divide the program into client tasks (running on the hand held) and server tasks (running on workstation) such that the energy consumed by the program is minimized. The method of task mapping that we have employed is also somewhat similar as we partition and map the functionality based upon the overall energy dissipation. The major difference between the two scenarios is that in distributed sensor systems, we also need to consider the energy consumption of the macro-nodes as even they have a limited energy supply. However, in the proposed cost graph approach, the available energy for the workstation is infinite. Also, overall performance of the application is not an optimization issue in the cost graph approach. Network portable terminals like the Infopad [32] have also been used in the wireless environments. Such systems have only a network I/O device with no computational power, relying on network servers to run major processes. However, such systems are not self configuring and they require extensive infrastructure support.

Computation hierarchy in the context of sensor network was proposed by [13] for a habitat monitoring application. They pointed out to the notion of using heterogeneous hardware for performing different kinds of computations. However, they did not propose any system composed of the heterogeneous nodes. The tiered system in [7] is the closest to the notion of hierarchy proposed in this paper. But the paper focused on the collaborative strategies between the macro-nodes and the micro-nodes to reduce the communication energy costs. The proposed strategies were limited to the application of habitat monitoring. They did not focus on any of the aspects of system design and the performance of the application. [33] proposed a tiered sensor network for habitat monitoring on the Great Duck Island. They use a tiered architecture solely for the purposes of communication. The computation hierarchy has not been considered.

7. CONCLUSION

In this paper, we have introduced the notion of computational hierarchy in networked embedded systems. The need for hierarchy stems from the fact that a single architecture cannot be optimal for the entire range of operations that are required to be performed in a typical sensor network application. We validated this claim through a case study of the acoustic beamforming application. We demonstrated that the sampling operation is more efficient on MICA motes while the computationally intensive beamforming routine is more efficient on the iPAQs.

Therefore, to have a scalable system, we advocate a hierarchical network level architecture comprising of a few macro-nodes in a sea of micro-nodes. Through our experiments, we verified that such architecture gives superior performance and lower power consumption than a homogeneous network of either all macro-nodes or all micro-nodes. Our experiments show that for the acoustic tracking application, a 15:1 ratio of the number of iPAQs to the number of MICA motes in the network performs better than a homogeneous network 95% times on an average.

8. FUTURE WORK

Currently, the research in hierarchical network architecture is still in its nascent stages. We have explored only a two-level hierarchy comprised of iPAQs and MICA motes. We have ported the beamforming application to the Atmel FPSLIC platform [37]. This is a unique architecture composed of a Micro-controller and a FPGA (Field Programmable Gate Array) on the same die. This architecture offers hardware on demand to the applications that seek more computation power. It would be interesting to study the trade-offs in a multi-level organization composed of three classes of nodes. Issues related to resource discovery and management needs to be resolved. Currently, we assume that only one application is being executed by the network. In the presence of multiple applications, it is possible that a macro-node close to a micro-node may be utilized to its full capacity. In such a situation, the micro-nodes need to have a mechanism to discover alternate resources in the network to off-load their computation onto. We envision a network wide distributed resource manager that is responsible for allocating computation resources to the micro-nodes on demand.

Another interesting area of future research is to explore the sensing hierarchy. Currently, the quality of acoustic sampling on the iPAQ is far superior to the Mica motes. In this scenario, it is possible to devise a collaborative strategy wherein the iPAQs are turned off all the time and can be woken up only by the micro-node. Therefore, the micro-nodes perform low-power coarse sensing mainly to detect events and then hand off to the superior nodes. This can be further extended to multiple sensing modalities like vision etc. that can be triggered by the low power sensing modalities. We are currently exploring this issue in the GALORE project.

9. ACKNOWLEDGEMENTS

This paper is based in part on research performed under the DARPA PAC/C and DARPA NEST programs. The views expressed in this paper are those of the authors and do not necessarily express those of the funding agencies.

10. REFERENCES

[1] Deborah Estrin, Chair, et al, "Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers", *Computer Science and Telecommunications Board (CSTB) Report.(http://www.cstb.org/web/pub_embedded*

[2] K. Sohrabi, J. Gao, V. Ailawadhi, G. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Communications Magazine*, Vol.7, No.5, pp. 16-27, Oct. 2000.

[3] L. Clare, G. Pottie, J. Agre, "Self-Organizing Distributed Sensor Networks," *SPIE - The International Society for Optical Engineering*, Orlando, FL, pp. 229-237, April 1999.

[4] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", *ACM Mobicom Conference*, Seattle, WA, August 1999.

[5] G. J. Pottie, W. J. Kaiser, "Wireless Integrated Network Sensors", *Communications of the ACM*, Vol. 43, no 5, May 2000.

[6] Ronald Riley, Brian Schott, Joseph Czarnaski and Sohil Thakkar, "Power aware acoustic processing.", *Second international workshop, IPSN 2003,* Palo-Alto, CA, USA, April 22-23, 2003

[7] Hanbiao Wang, Deborah Estrin and Lewis Girod, "Preprocessing in a Tiered Sensor Network for Habitat Monitoring", *EURASIP JASP special issue of sensor networks*, vol. 2003, no. 4, pp. 392-401, March 15, 2003.

[8] J. Hill, R. Szewcyk, A. Woo, D. Culler, S. Hollar and K. Pister, "System Architecture Directions for Networked Sensors", *ASPLOS 2000*

[9] MICA Sensor Node, http://www.xbow.com

[10] Compaq iPAQ, http://www.compaq.com/products/iPAQ

[11] IQinVISION: IQeye3 Camera, http:///www.iqinvision.com

[12] Ram Kumar, Soheil Ghiasi and Mani Srivastava, "Dynamic Adaptation of Networked Reconfigurable Systems", *Workshop on Software Support for Reconfigurable Systems (SSRS)*, February 2003.

[13] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton and Jerry Zhao , "Habitat monitoring: Application driver for wireless communications technology", *2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean,* Costa Rica, April 2001.

[14] Kim, J-Y., Bendikov, T. Park, Y. and Harmon, T.C., "Networked Sensing in Support of Real-Time Transport Model Parameter Estimation", *Proceedings of the European Geological Society-American Geophysical Union-European Union of Geosciences Joint Assembly*, April 6-11, 2003, Nice, France.

[15] Chen, J. C.; Yao, K.; Hudson, R. E.; "Source Localization and Beamforming", IEEE Signal Processing Magazine, March 2002.

[16] Zhao, F.; Shin, J.; Reich, J.; "Information-Driven Dynamic Sensor Collaboration for Tracking Applications, IEEE Signal Processing Magazine, pp. 61-72, March 2002.

[17] D.A.Patterson and J.L.Hennessy, "Computer Architecture: A quantitative approach", *Morgan Kaufmann Publishers Inc.*, San Francisco,1996.

[18] J.C. Chen, L. Yip, J. Elson, H. Wang, D. Maniezzo, R.E. Hudson, K. Yao, and D. Estrin, "Coherent Acoustic Array Processing and Localization on Wireless Sensor Network", to apprear in *IEEE Proceedings*, Mid 2003.

[19] Atmel ATMEGA128L Datasheets, http://www.atmel.com/avr

[20] RFM TR1000 radio datasheet, http://www.rfm.com/products/data/tr1000.pdf

[21] ITSY PocketPC Compaq, http://research.compaq.com/wrl/projects/itsy/

[22] Intel StrongARM SA-1110 Microprocessor Brief Datasheet, http://developer.intel.com/design/strong

[23] EEMBC, Embedded Micro-processor Benchmark Consortium, http://www.eembc.com

[24] MoteNIC Overview, http://lecs.cs.ucla.edu/Noteworthy/quadcharts/thanos_lecs.ppt

[25] Honeywell Magnetic Sensors, http://www.ssec.honeywell.com/magnetic/

[26] Philips UDA1341TS Datasheet, http://www-us.semiconductors.philips.com/pip/UDA1341TS.html

[27] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer and D. Culler., "The nesC Language: A Holistic Approach to Network Embedded Systems.", *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, June 2003

[28] TinyOS Operating System, http://webs.cs.berkeley.edu/tos

[29] Faimilar Linux, http://familiar.handhelds.org

[30] A. Sinha and A. Chandrakasan, "JouleTrack – A Web Based Tool for Software Energy Profiling", *Proceedings of the 38th Design Automation Conference*, Las Vegas, June 2001

[31] Z Li, C. Wang and R. Xu, "Computation offloading to save energy on handheld devices: A partition scheme," *Proc. of International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, Nov. 2001, Atlanta, Georgia, pp. 238--246, ACM Press.

[32] S. Narayaswamy and et al. Application and network support for infopad. *IEEE personal Communications*, 3(2):4–17, April 1996.

[33] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson,"Wireless sensor networks for habitat monitoring," in *1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, Atlanta, Ga, USA, September 2002.

[34] Jeremy Elson, Lewis Girod and Deborah Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts",
In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA. December 2002.

[35] Galore Project: http://galore.cs.ucla.edu

[36] NESLSim: A Parsec Event Simulator: http://www.ee.ucla.edu/~saurabh/NESLsim

[37] Atmel FPSLIC Datasheets: http://www.atmel.com/products/FPSLIC

[38] A. Savvides and M. B. Srivastava, "A Distributed Computation Platform for Wireless Embedded Sensing", *Proceedings of International Conference on Computer Design 2002*, Freiburg, Germany.

[39] Sensoria Corporation: http://www.sensoria.com

[40] Cerfcube – Intrinsyc Corporation: http://www.intrinsyc.com/products/cerfcube/