In Proceedings of the International Workshop on Parallel and Distributed Real-Time Systems,
Miami, Florida, April 2008.

Design and Optimization of a Distributed, Embedded Speech Recognition System

Chung-Ching Shen, William Plishker, and Shuvra S. Bhattacharyya
Dept. of Electrical and Computer Engineering, and Institute for Advanced Computer Studies
University of Maryland at College Park, USA
{ccshen, plishker, ssb}@umd.edu

Abstract

In this paper, we present the design and implementation of a distributed sensor network application for embedded, isolated-word, real-time speech recognition. In our system design, we adopt a parameterized-dataflow-based modeling approach to model the functionalities associated with sensing and processing of acoustic data, and we implement the associated embedded software on an off-the-shelf sensor node platform that is equipped with an acoustic sensor. The topology of the sensor network deployed in this work involves a clustered network hierarchy. A customized time division multiple access protocol is developed to manage the wireless channel. We analyze the distribution of the overall computation workload across the network to improve energy efficiency. In our experiments, we demonstrate the recognition accuracy for our speech recognition system to verify its functionality and utility. We also evaluate improvements in network lifetime to demonstrate the effectiveness of our energy-aware optimization techniques.

1. Introduction

Speech recognition involves converting acoustic signals, captured by a microphone or an acoustic sensor, to a set of words. Then, these words are compared with some pre-defined words and some sort of indication is given if there is a match. The recognized words can then be analyzed and used for back-end applications such as command and control, commercial information retrieval, and linguistic processing for speech understanding. Figure 1 presents a design flow for basic speech recognition systems.

From an embedded system design aspect, a major design challenge for speech recognition is to assure the processing of large amounts of data in real-time. Various prior efforts on embedded speech recognition, e.g. [1, 12, 13], focused on implementing various speech recognition algorithms on embedded platforms, such as programmable digital signal processors (PDSPs) and comparing their performance. These efforts typically have not explored further optimization for real-time operations and energy usage beyond what is already available through a standard PDSP-based design flow. These existing design approaches therefore are not fully suited for heavily resource-limited, distributed systems, such as wireless sensor networks.

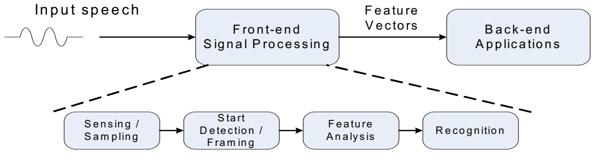


Figure 1. Basic design flow for automatic speech recognition systems.

A wireless (distributed) sensor network (WSN) system is composed of resource-limited sensor nodes, which consist of components for sensing, data processing, and wireless communication (e.g., see [7]). WSN systems have a variety of potential applications [9], such as environmental monitoring and intrusion detection. Sensor nodes are often deployed in inaccessible or, in the case of certain military and security-related applications, dangerous areas and communicate with each other through self-organizing protocols. To maximize the useful life of these systems, power consumption must carefully be considered during sensor node design.

Integrating speech recognition into a WSN system enables a new class of applications for speech recognition that have distributed configurations. We refer to such speech-recognition-equipped WSN systems as distributed automatic speech recognition (DASR) systems.

The DASR system developed in this paper is an isolated-word and speaker-dependent speech processing system, where templates of extracted coefficients of words have to be created and stored at a central node. The system functionality is to have all sensor nodes collect speech data within their sensing ranges, and transmit this data periodically — in the form of recognized words (or simple indicators for the absence of any words) — to the central node. Any application-specific analysis and usage of the recognized words is handled as back-end processing on the central node.

Based on different requirements on recognition accuracy, we describe two practical application scenarios in which our developed DASR system can be applied. The first scenario involves using a DASR system as a speech-based command and control system in

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

a battlefield environment. Since the DASR system is speaker-dependent, its word templates need to be trained by the person who will be using the system and, as we will show in our experiments, it is capable of achieving a high accuracy for word recognition in this context. When we apply the system in a battlefield for recognizing command words, the speaker-dependent property provides a benefit by rejecting command words that are spoken by enemies and author unauthorized people.

Another application scenario is to use a DASR system as a surveillance system for collecting large amounts of speech data with similar patterns from arbitrary speakers. Since sensor nodes are usually designed to be small, they can be hidden in a battlefield environment. Therefore, acoustic signals from the enemy can be secretly sensed, collected, and translated into useful data on the sensor nodes. Through a well-designed communication protocol, this data can then be transmitted to a central node for further processing and back-end analysis. The recognized words, for example, can be used to distinguish among a diversity of languages or to survey specific words in a crowd for special monitoring and detection purposes.

We observe, however, that it is difficult to apply speech recognition techniques for secretive speech monitoring (e.g., for security- or defense-related applications) if the recognition is constrained to be performed on a single, stand-alone embedded platform. This is because the sensing range of individual sensor nodes is limited, and sensors are often deployed in difficult-to-reach areas, where their placement cannot be fine-tuned. However, a distributed WSN configuration can help make such applications feasible by distributing the computation across a multitude of embedded platforms (i.e., sensor nodes), and then collecting and consolidating large amounts of monitored information in a systematic way.

2. Related Work

Pauwels [11] gives an overview of recent WSN developments for ambient intelligence applications, including speech recognition. In these applications, information provided by sensors is used to drive systems that automatically analyze human behavior.

As shown in Figure 1, the main component in the front-end of speech processing algorithms is feature extraction. Several popular feature extraction techniques, such as Mel-Frequency Cepstral Coefficients (MFCC), Weighted OverLap Add (WOLA), and Noise-Robust Auditory Features (NRAF) have been used extensively in speech recognition technology. Ahadi [1] has demonstrated that both MFCC and WOLA approaches can achieve high (reasonable) recognition accuracy for clean (noisy) speech. Ravindran [13] presents a comparison between their proposed NRAF approach and the MFCC approach and claims that their proposed NRAF approach can be implemented for low-power sensor networks.

A few research efforts have integrated speech recognition front-ends into WSN systems and demonstrated overall system feasibility. Phadke [12] presents a hardware/software co-design solution to implementing an embedded speech recognition system with the use of a modified MFCC approach for feature extraction, and a dynamic time warping (DTW) technique for template alignment and matching. In this paper, we build on Phadke's design flow for the embedded software development of our front-end speech recognition processing.

Delaney [6] investigates both computation and communication energy consumption of distributed speech recognition on a targeted embedded system and proposes optimization techniques for energy reduction in the application and network layers. However, Delaney's design and optimization approach are developed for general wireless mobile devices and sophisticated wireless networks such as 802.11b and Bluetooth.

In contrast, our objective in this paper is to customize DASR systems for heavily resource-limited sensor nodes, and to employ an application-specific, point-to-point protocol configuration for this purpose. Compared to the efforts of Phadke and Delaney, we target a very different region of the DASR system design space involving potentially higher cost, due to the use of more specialized and streamlined sensing devices, but also involving greater potential for miniaturization and longer-lifetime operation. The latter objectives are useful for our targeted class of defense- and security-related speech recognition applications, where it is important to have sensor nodes that are small enough so that they are not easily detected, and that can be deployed for long periods of time without maintenance in remote, difficult-to-access geographical areas.

Another distinguishing aspect of our approach in this paper is the application of structured design methodologies based on our recently-introduced approach to energy-driven WSN application partitioning [14], and based on high-level application dataflow modeling for optimization of the targeted embedded software. These characteristics enhance the generality of our developed solutions and the applicability to other types of DASR applications. Our work in this paper differs from [14] in that we present in this paper a complete application development case study for the DASR system, and our application study, experimental results, and analysis for energy consumption minimization are based on measurements from actual hardware implementation of the DASR system, in contrast to the simulation-based experimental context of [14]. Furthermore, we go beyond the scope of [14] by pointing out an advanced development of workload redistribution for adaptively optimizing system energy efficiency.

3. Distributed Embedded System for Speech Recognition

3.1. Master-slave network topology

In this paper, our developed DASR system is based on a hierarchical network organization, where such a clustered, hierarchical organization is applied for efficient lifetime management (e.g., see [8]). Within each cluster of the hierarchy, we refer to the designated central node as the *master node*, and we refer to all other nodes in the same cluster as *slave nodes*. In our experiments, which are based on the off-the-shelf wireless transceiver described in [4] as the communication device for each sensor node, the distance between the master node and all slave nodes should be within approximately 30 meters to ensure reliable communication. We define a parameter to indicate the number of nodes (i.e., the initial network size) being used to set up the overall system. After the system has been initially established (i.e., all nodes have joined the network) based on this number of nodes, the network size can be dynamically changed as nodes are added to or removed from the system. We use the updated value of the network size as an input to our workload redistribution scheme.

In summary, the network developed for our DASR system is a clustered, hierarchical network, where each cluster forms a master-slave network topology. The analysis and experimental results throughout the remainder of the paper are based on such a network organization.

3.2. Customized TDMA protocol

We have designed a simple but robust time division multiple access (TDMA) based protocol for our DASR system. Figure 2 illustrates the communication pattern in the protocol between the master node and slave nodes. In this protocol design, we assume that each node in the system has a unique identifier (ID). The master node uses some pre-defined number of time slots to establish a time frame. These time slots will be requested by slave nodes as the slave nodes join the network sequentially. We create several special packets, as shown in Figure 2, which are used to progressively synchronize the master node with the slave nodes that are joining the network. After synchronization, each slave node occupies a single time slot in the periodic TDMA schedule for the network cluster, and each slave node sends data to the master node at its reserved time slot.

For example, as shown in Figure 2, the master node broadcasts a packet initially with control information in TDMA slot 0, while all other slots are in an idle state. When the first slave node, S1, is turned on, it stays in the receiving mode until it receives a broadcasted packet from the master node. Then S1 sends a packet to the master node requesting that TDMA time slot 0 be reserved for node S1. The master node subsequently sends an acknowledgement packet to S1 confirming this reservation for slot 0. Now, time slot 0 of the master node is changed to the active state and is reserved exclusively for S1. Henceforth, node S1 will transmit data packets to the master node at time slot 0 in each TDMA time frame.

Next, the master node broadcasts a control packet at slot 1 in the next time frame. When the second node S2 joins the network, it follows the same procedure to synchronize with the master node as described above for S1. This process of adding nodes one-by-one to the network continues until all slave nodes within communication range have been added to the network. After all nodes have been added, the network enters a data processing state in which TDMA time frames operate periodically, and in each such frame, each slave node transmits its newly collected data during its corresponding time slot.

Our targeted DASR system is designed for surveillance applications with recognition response times in the range of 20-30 seconds, and the system collects speech data continuously in a neutral manner without targeting any specific sound object. The real-time constraint for the master node to complete a single recognition task is relatively loose in this scenario. Therefore, in our design, it is reasonable to allow the computation and communication tasks on the master and slave nodes to be completed within their associated time slots in each TDMA frame. During periods of inactivity within a TDMA frame, each slave node transforms itself into a low power idle state for maximum energy efficiency.

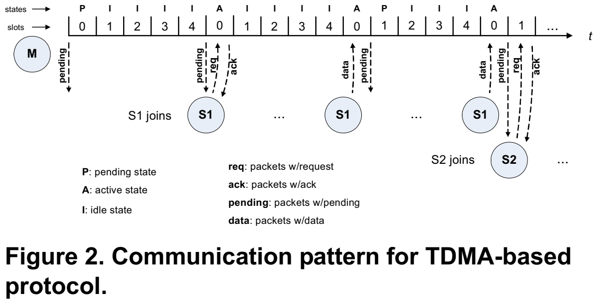
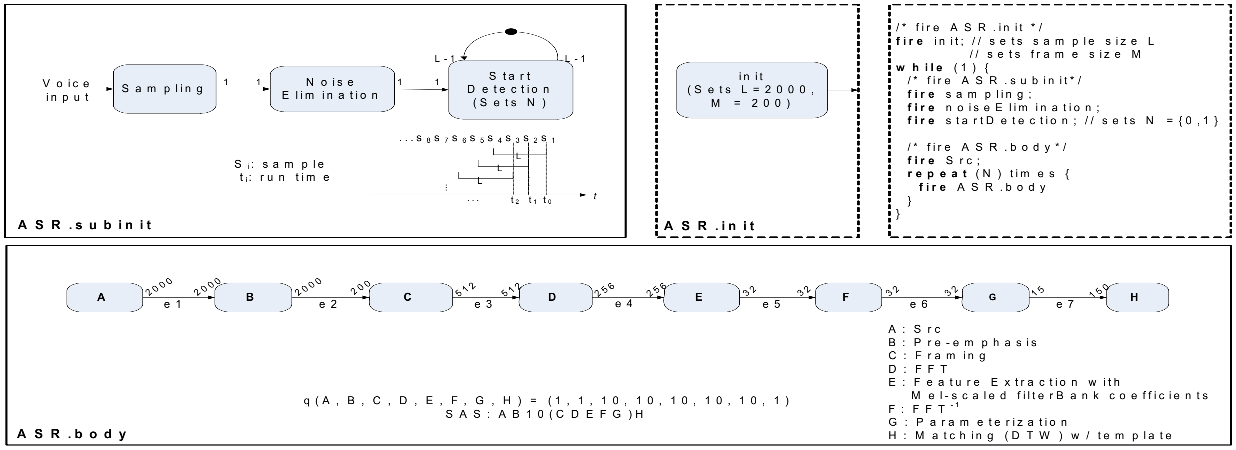


Figure 2. Communication pattern for TDMA-based protocol.



```
/* fire ASR.init */
fire init; // sets sample size L
           // sets frame size M
while (1) {
  /* fire ASR.subinit*/
  fire sampling;
  fire noiseElimination;
  fire startDetection; // sets N ={0,1}

  /* fire ASR.body*/
  fire Src;
  repeat (N) times {
    fire ASR.body
  }
}
```

Figure 3. PSDF modeling and associated quasi-static schedule for the DASR system.

3.3. Real-time acoustic sensing

In our developed system, we employ a sampling frequency of 8 KHz for human speech; therefore, a 125μs timer is set up for the microcontroller [4] to enable an 8-bit analog-to-digital converter (ADC) for sampling and converting sensed signals from an acoustic sensor. Since the sampling and conversion time of the selected ADC [4] takes around 20μs, acoustic signals can be sensed accurately at the desired 8 KHz rate. Moreover, in order to use the limited memory capacity of the sensor nodes efficiently, we select a word duration of 0.25s in our experiments, so that the total number of samples at 8 KHz is 2000, where each sample is stored as an 8 bit fixed-point value.

After the DASR system is initialized and the network is synchronized, slave sensor nodes keep capturing samples from background noise and calculating the average values of these samples. The average noise value is used to monitor signal values as they are being captured. In this process of monitoring, threshold-exceeding and zero-crossing schemes are used to detect the start of an utterance in the presence of background noise. Typically, ambient noise generates very few zero crossings compared to normal speech, so that increasing the rate of detected zero crossings can be used as an indicator for the beginning of a speech utterance [12]. Similarly, since the energy level of noise is typically low compared to that for detected speech, a threshold-based scheme can be used to detect signal levels that are likely to correspond to speech.

Once a possible utterance is detected using one or both of the threshold- and zero-crossing-based schemes described above, 2000 consecutive signal samples are captured and stored in the memory of the associated sensor node for further processing. The processing steps for speech recognition are applied to such blocks of 2000 collected samples.

3.4. Parameterized dataflow modeling

We model and implement the targeted real-time sensing and speech recognition processing behavior on slave nodes using the parameterized synchronous dataflow (PSDF) model of computation [2]. According to PSDF semantics, the sensing inputs and start detection schemes are modeled using *init* and *subinit* graphs, and the speech recognition processing algorithm is modeled using a *body* graph. Based on this modeling approach, and the scheduling features provided by PSDF, low-overhead, “quasi-static” schedules can be generated for hand-coded implementation or software synthesis [2]. Here by a quasi-static schedule, we mean an ordering of execution for the functional modules (dataflow actors) whose structure is largely fixed at compile time, with a small number of decision points or symbolic adjustments made at run-time based on the values of relevant input data.

Figure 3 illustrates our PSDF application model and an associated quasi-static schedule. Here, *ASR.init* sets the length of a circular window for the past $(L-1)$ samples at each point of time, and the frame size M for each data frame. *ASR.subinit* reads sample inputs and maintains a circular window of $(L-1)$ samples for real-time processing. In our experiments, we set $L = 2000$, $M = 200$, and use synchronous dataflow (SDF) [10] to model the core speech recognition processing functionality, which is integrated into the enclosing PSDF body graph (i.e., *ASR.body* in Figure 3) of our overall PSDF-based modeling framework.

If a valid speech token is detected from the acoustic input stream, a dynamic parameter N is configured to enable speech recognition processing in the body graph ASR.body. This parameter is configured based on the number of speech tokens that should be processed in the newly-initiated recognition step. Otherwise, the value of the parameter N is set to zero, which effectively disables speech recognition processing for the current application iteration.

3.5. Speech data processing and recognition

As described in Figure 3, speech recognition processing is modeled as an SDF graph — i.e., the ASR.body graph — that is integrated into an enclosing PSDF framework. Once ASR.body is triggered for execution on a block of 2000 collected samples, the block of samples is first partitioned into 10 overlapping frames, where each frame consists of 256 samples with padding zeros. On each frame, a 256-point FFT is applied. The result of each frame-wise FFT is processed by a 16-tap Mel-scaled filter bank to implement the feature extraction function for the frame. The feature extraction function is applied to identify the speaker's vocal tract (formants) in the speech, and in this process, 15 coefficients are obtained from an inverse discrete cosine transform to represent the parameters for each 200-sample frame. Then, a dynamic timing warping (DTW) technique is used to consume 150 parameters for a spoken word, and to find a match between the spoken word and the set of template words.

Regardless of which partitioning result of workload distribution is applied, spoken word recognition is always executed at the master node under our WSN configuration. Thus, we unconditionally store a set of parameters for template words in the master node; these parameters are trained and stored a priori through a stand-alone speech recognition process. The set of template words is chosen based on the back-end application.

In the master node, the dynamic time warping (DTW) [5] technique is used to find the best match (i.e., the smallest DTW distance) between the parameters of the spoken word and each of the template words. Here, the smallest DTW distance represents the best match for the spoken word from the inventory of template words. We refer the reader to [12] for more details on the embedded software design flow for the speech data processing and word recognition.

3.6. Memory management and real-time constraints

For embedded system design, memory management is important for real-time operation, especially if resource-limited platforms are used. We analyze buffer (i.e. memory required during the computation process) and latency requirements based on the discussed dataflow structure for modeling and implementing the DASR system.

By applying a looped, single appearance schedule (a compact form of dataflow graph schedule in which looping constructs are organized carefully so that each dataflow actor appears only once in the schedule) [5] to execute the ASR.body graph for the DASR system, the maximum data memory requirement for slave nodes ($buf(G_s)$) can be bounded as $2000 \le buf(G_s) \le 2982$, in terms of bytes (each sample size is 8 bits). This upper bound of 2982 should be within the maximum data memory size in the targeted hardware platform (e.g., 8KB for the platform of [4]).

That is, as shown in Figure 3, if the slave nodes only capture input signals and transmit them directly without any pre-processing, 2000 bytes of memory are needed to buffer the signal samples. In the other extreme, if the master node is designed to only execute the recognition operation, and the slave nodes perform the entire signal processing chain before transmitting their data, then the slave nodes require $2000 + 512 + 256 + 32 + 32 + 150 = 2982$ bytes of memory to buffer data values throughout the computation process. However, in this case, the slave nodes need to transmit only 150 bytes of data to the master node, since the volume of data is reduced significantly through the signal processing that is performed on the slave nodes.

The memory requirement in the master node is determined in terms of the amount of data required for storing the received data from the slave nodes, the required buffer size for signal processing operations, and the size of the template word inventory. By analyzing the dataflow structure in the system design of Figure 3, the memory requirement in the master node ($buf(G_m)$) can be bounded as

$$150 + buf_T \le buf(G_s) \le 2982 + buf_T \text{ bytes,}$$

where

$$buf_T = (\text{\# of template words}) \cdot (\text{\# of parameters per word}) .$$

In order to characterize the real time sensing and processing capabilities for our developed DASR application, we define a minimum duration T_d required by the slave nodes between consecutive spoken words. We apply this duration to the experiment so that the real-time constraint on each slave node can be satisfied. In terms of the application dataflow structure, T_d must be larger than the time needed to execute the ASR.subinit and ASR.body graphs, plus the time needed to transmit the required data to the master node.

For example, our target platform [4] has a 32 MHz processing speed and 250 kbps transceiver data rate. If all sensing and processing tasks are handled by slave nodes except for the recognition task, then T_d is approximately 13.675s (0.27s sensing and detection time plus 13.4s processing time and 4.8ms transmitter operation time). On the other hand, if the slave nodes only execute sensing tasks, T_d is approximately 0.334s (0.27s sensing and detection time plus 64ms transmitter operation time).

Therefore, across the different types of workload distributions that we consider, T_d is bounded by $0.334s \le T_d \le 13.675s$ when the target platform of [4] is used. This kind of analysis, which is based on the dataflow-based application model together with relevant details of the target platform, can be used to develop real-time characterizations and enforce associated constraints for the implemented system. The corresponding latency measurements for the execution of individual actors in the system are examined in the next section. Note that T_d is the real-time constraint that is applied to the slave nodes of the DASR system for satisfying the requirements of sensing and processing speech signals. To consider latency constraints in conjunction with partitioning trade-offs across the master and slave nodes, we refer the reader to [14].

3.7. Dynamic topology management

As we have shown in [14], a well-developed energy-driven partitioning (EDP) scheme can help us to find a more energy-efficient computation distribution for distributed nodes in a targeted master-slave topology. By applying the PSDF model in conjunction with this EDP scheme, we provide an advanced form of workload redistribution that can be used for dynamically-changing network sizes — i.e., for scenarios in which sensor nodes can enter or exit the system at run time. Such dynamics may arise, for example, due to incremental node deployments and exhausted batteries, respectively.

Our advanced workload redistribution scheme is based on the EDP approach described in [14] and a layer of PSDF modeling that is used to take into account the numbers of sensor nodes that are dynamically added to and removed from network. Based on this approach, whenever the network size changes, the associated EDP configuration will generally be adjusted so that the workload distribution is efficiently adapted to the new network structure.

Based on our PSDF modeling approach, the number of network nodes is characterized by a dynamic parameter that represents the number of slave nodes that are in the system, This parameter is maintained and broadcasted by the master node. From the customized protocol described in Section 3.2, the master node can determine the number of active nodes existing in the system based on the requests that it receives from slave nodes and the usage of time slots in the TDMA schedule. Thus, soon after any change in network size, all active slave nodes are informed about the change by the master node, and furthermore, EDP results are re-calculated, and re-deployed accordingly.

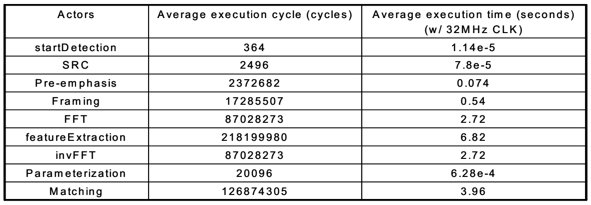
4. Experiments

4.1. Experimental setup

For demonstrating the DASR system and analyzing its performance in a complete implementation, we use the Texas Instruments/Chipcon CC2430 [4] system-on-chip (SoC) device as the main processor and transceiver on all sensor nodes. This device is therefore used on each node for executing all processing and communication tasks. In addition, each slave node platform is equipped with an acoustic sensor. We use the profiling tool in the IAR Embedded Workbench to derive a task-level timing estimate for each actor's execution in the DASR system. These profiling results are shown in Figure 4.

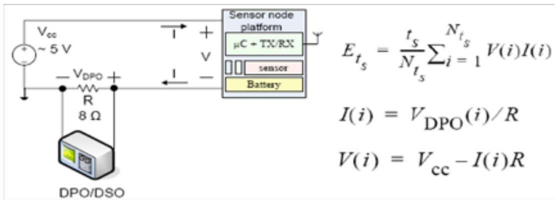
We implement the DASR system, measure voltage variations, and calculate the associated current, power, and energy consumption across multiple CC2430 platforms. The voltage variations are measured through the Tektronix 4GHz digital phosphor oscilloscope (TDS 7404). Figure 5 shows this experimental setup for energy consumption measurement and the associated equations to estimate overall energy consumption of a sensor node platform within a period of time t_s, where N_{t_s} represents the total number of points within t_s that are sampled by the oscilloscope.

We choose some common words as experimental examples and compare the recognition accuracies for the different words after the master node receives its required information from the slave nodes and finishes its recognition task. Figure 6(a) shows the captured signal samples for some spoken word examples. Figure



Actors	Average execution cycle (cycles)	Average execution time (seconds) (w/ 32MHz CLK)
startDetection	364	1.14e-5
SRC	2496	7.8e-5
Pre-emphasis	2372682	0.074
Framing	17285507	0.54
FFT	87028273	2.72
featureExtraction	218199980	6.82
invFFT	87028273	2.72
Parameterization	20096	6.28e-4
Matching	126874305	3.96

Figure 4. Task-level timing estimation for the DASR system implementation.



$$E_{t_s} = \frac{t_s}{N_{t_s}} \sum_{i=1}^{N_{t_s}} V(i) I(i)$$

$$I(i) = V_{\mathrm{DPO}}(i) / R$$

$$V(i) = V_{\mathrm{cc}} - I(i) R$$

Figure 5. Experimental setup for energy consumption measurement.

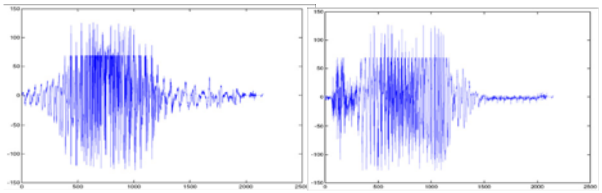
6(b) shows a comparison of recognition accuracy for the example words. Note that our experiments are established in a relatively clean (non-noisy) environment. This helps us to achieve the relatively high recognition accuracies shown in Figure 6(b).

Ahadi [1] has demonstrated that by applying different filter bank techniques, recognition accuracy can be further improved even in noisy environments. Elaboration on this method is beyond the scope of this paper. We refer the reader to [1] for more details on different filter bank techniques.

4.2. Lifetime comparison

According to the TDMA-based communication protocol discussed in Section 3.2, a sensor node in the DASR system transmits and receives data at its reserved time slot after joining the network. Based on this regular communication pattern, we estimate the lifetime for sensor nodes in terms of the workload distribution for computation and communication activities. To this end, we first measure the energy consumption when the master node and slave nodes are at their reserved time slots for executing their assigned tasks. Then, this measurement is translated into a lifetime analysis estimate by considering the battery usage on each sensor node. Figure 7 shows the results from our measurements and analysis.

In Figure 7, we show voltage variation measurements when two different partitioning schemes are applied to distribute computations across the master and slave node platforms. That is, Figure 7(a-d) shows the voltage variation on a TDMA active slot for (a) slave nodes that transmit raw data (i.e., 2000 samples); (b) slave nodes that execute the full signal processing chain and only transmit necessary parameters; (c) the master node configuration that receives all raw data and executes a full computation including the recognition task; and (d) the master node configuration that only receives necessary parameters and executes the recognition task.



(a)

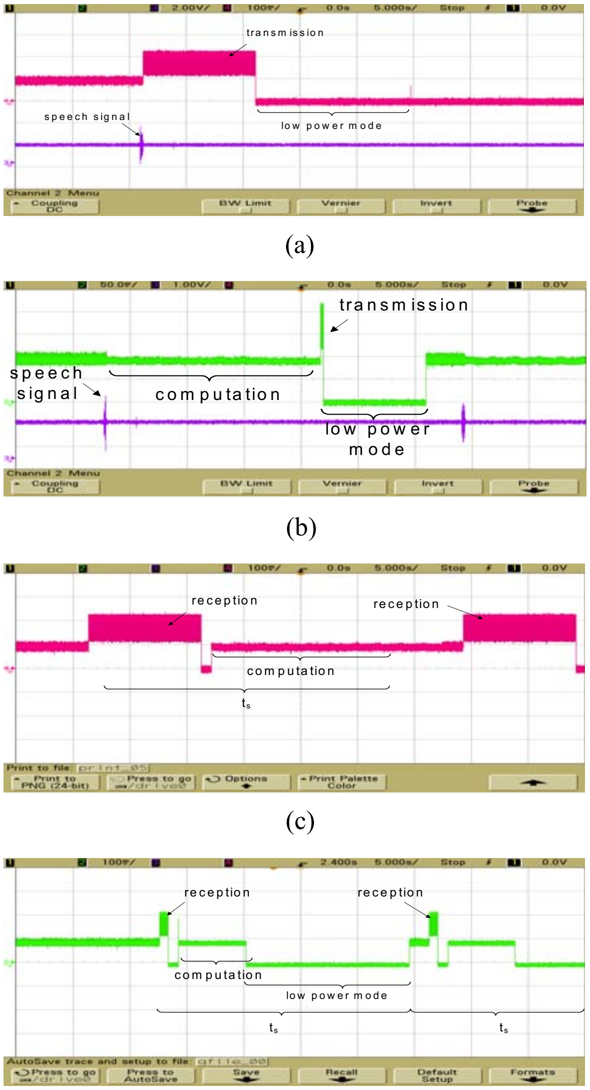
Words	'one'	'two'	'eight'	'hello'	'bomb'	'shoot'
Accuracy %	95.3	90.1	96.4	86.2	93.5	79.6

(b)

Figure 6. (a) Captured signal samples for example words: "one" and "two". (b) Recognition accuracy.

By applying the equations in Figure 5 to the measured voltage variation presented above, an estimate of the corresponding energy consumption of an active TDMA time slot is obtained.

In this experimental setup for the DASR system, the overall system consists of one master node and three slave nodes; thus, we set $N_s = 4$ and $t_s = 25$ seconds so that a TDMA time frame consists of four time slots and the whole computation on a sensor node can be completed within an active time slot. For example, as we see in Figure 7(a), a slave node takes around 10 sec-



(a)

(b)

(c)

(d)

Node type / Partition type	slave	master
partition 0	157.84 hours	32.55 hours
partition 1	161.52 hours	147 hours

(e)

Figure 7. Energy consumption measurement and lifetime comparison.

onds to transmit its raw data (i.e., 2000 samples) to the master node and stays in the idle state (i.e., switches to its low power mode) for the remaining 15 seconds of an active slot. Note that a sensor node may also stay in an idle state whenever its designated slot is not active.

Furthermore, by considering the use of a $3V \cdot 650mAh$ Lithium battery on each sensor node, lifetime analysis results for sensor nodes in the DASR system are shown in Figure 7(e). Here, we compare sensor node lifetime (i.e., sensor nodes continuously execute their tasks frame-by-frame under the given TDMA protocol until they run out of energy) in terms of hours between two different workload distributions. In Figure 7(e), the label “partition 0” represents the conventional configuration of having maximal data processing performed on the master node — i.e., the partition is “cut” on $e1$ of *ASR.body* in Figure 3. On the other hand, the label “partition 1” represents a balanced workload distribution after EDP analysis — i.e., the partition is cut on $e7$ of *ASR.body* in Figure 3.

We observe from Figure 7(e) that the partitioning of computation and communication affects the lifetime of sensor nodes significantly. It can also be observed that if the system lifetime is defined as the time that the first node in the network runs out of energy, then a DASR system whose workload distribution is based on the EDP result has a system lifetime improvement of approximately a factor of four compared to a system that uses a conventional workload configuration (i.e., the configuration in which slave nodes transmit raw data only).

5. Conclusion

In this paper, we have demonstrated a distributed automatic speech recognition (DASR) system implementation by integrating embedded processing for speech recognition with a wireless sensor network system. We have discussed several promising applications for which the proposed system can be further customized. In our system implementation, we have adopted a parameterized-dataflow-based modeling approach for structuring a well-known algorithm for embedded speech recognition. This model-based design facilitates latency and memory analysis, and helps to structure the embedded software for efficient implementation.

Through a detailed case study, we have presented our key design steps, including the definition of the network topology, protocol design, implementation of the embedded speech recognition algorithm, and distribution of computation and communication with careful consideration of energy usage. Measurement results on recognition accuracy and energy consumption demonstrate the functionality and efficiency of our DASR system implementation. We have also presented a workload redistribution scheme for improving system energy efficiency adaptively at run time.

6. References

[1] S. M. Ahadi, H. Sheikhzadeh, R. L. Brennan, and G. H. Freeman. An efficient front-end for automatic speech recognition. In *Proceedings of the IEEE International Conference on Electronics, Circuits and Systems*, Sharjah, United Arab Emirates, December 2003.

[2] B. Bhattacharya and S. S. Bhattacharyya. Parameterized dataflow modeling for DSP systems. *IEEE Transactions on Signal Processing*, 49(10):2408-2421, October 2001.

[3] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee. *Software synthesis from dataflow graphs*. Kluwer Academic, 1996.

[4] CC2430 Data Sheet. SWRS036F. Texas Instruments.

[5] M. Brown and L. Rabiner, Dynamic time warping for isolated word recognition based on ordered graph searching techniques. In *Proceedings of the International Conference on Acoustic, Speech, Signal Processing*, vol. 7, pp. 1255-1258, May 1982.

[6] B. Delaney, T. Simunic, and N. Jayant. Energy aware distributed speech recognition for wireless mobile devices. *HP Labs Technical Reports: HPL-2004-106*, June 2004.

[7] M. Kohvakka, M. Hannikanen, and T. Hamalainen. Wireless sensor prototype platform. In *Proceedings of the IEEE International Conference for Industrial Electronics, Control and Instrumentation*, pp. 1499–1504, November 2003.

[8] R. Kumar, V. Tsiatsis, and M. B. Srivastava. Computation hierarchy for in-network processing. In *Proceedings of the ACM International Conference on Wireless Sensor Networks and Application*s, pages 68-77, 2003.

[9] M. Kuorilehto, M. Hannikainen, and T. D. Hamalainen. A survey of application distribution in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, pages 774-788, 2005.

[10] E. A. Lee and D. G. Messerschmitt. Synchronous dataflow. In *Proceedings of the IEEE*, 75(9):1235-1245, September 1987.

[11] E. Pauwels, A. A. Salah, and R. Tavenard. Sensor networks for ambient intelligence. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, Chania, Crete, Greece, October 2007.

[12] S. Phadke, R. Limaye, S. Verma, and K. Subramanian. On design and implementation of an embedded automatic speech recognition system. In *Proceedings of the 17th International Conference on VLSI Design*, pages 127-132, 2004.

[13] S. Ravindran, D. Anderson, and M. Slaney. Low-power audio classification for ubiquitous sensor networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, May 2004.

[14] C. Shen, W. Plishker, S. S. Bhattacharyya, and N. Goldsman. An energy-driven design methodology for distributing DSP applications across wireless sensor networks. In *Proceedings of the 28th IEEE Real-Time Systems Symposium*, pages 214-223, Tucson, Arizona, December 2007.