computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

DESIGN MODELS FOR EMBEDDED SYSTEMS AND SENSOR NETWORKS *

Zdravko Karakehayov

Technical University of Sofia,
Kliment Ohridski 8, Sofia-1000, Bulgaria,
e-mail: zgk@tu-sofia.bg, www.tu-sofia.bg

Abstract: *This paper introduces a hierarchical design model for embedded systems which includes models of the application, the embedded system and the resources. The application model deals with locations to better describe distributed architectures. A location-driven method for design of distributed embedded systems is proposed as a direct extension of the design model. The method allows to benefit from fuzzy relationships between I/O variables and locations. An example shows a 29% reduction of the hardware complexity measured in number of pins. In case of wireless links the model allows sensor networks to gradually emerge as a class of embedded systems.*

Keywords: *Embedded systems, sensor networks, design models, medium access control model*

1. INTRODUCTION

Embedded computing systems have grown tremendously in recent years, not only in their production volumes, but also in their complexity. Embedded applications often involve much computation which must be done in a timely fashion. Real-time behavior is the defining characteristic of embedded computing. The physical constraints arise through the two ways that computational processes interact with the physical world: reaction to a physical environment and execution on a physical platform. Also, power consumption is an important design metric for battery-powered embedded systems. As a result, embedded systems need multiobjective design. Designers will have a great deal of freedom for system optimization if a single application is the only interface that matters. Consequently, the key to a successful product is a design model that includes both the application and the embedded system. Also, the design model must deal with resources which inevitably influence the production cost and time to market.

The design of large, complex systems requires a hierarchical model constructed by a sequence of models at different abstraction levels. Distributed embedded systems are implemented with several processors which work concurrently. The distributed architecture can improve the timing properties of the system and alleviate interfacing problems. When sensors and actuators are physically spread out in different locations of the controlled object, the distributed architecture is the only efficient approach. Distributed embedded systems require another axis along which the system is designed – location. Location-driven design may give us timing and energy properties which we would not otherwise be able to obtain.

Ad hoc networks are employed in situations where installing an infrastructure is too expensive, too vulnerable or the network is transient. The interaction between the nodes is based on wireless communication. Packets are forwarded in a multihop manner. Nodes have a limited radio footprint and when a node receives a packet it applies a routing algorithm to select a neighbor for forwarding.

There is a class ad hoc networks, sensor networks, where the requirements for lifetime and size of the nodes are driven to extremes. A wireless sensor network consists of a large number of nodes that may be randomly and densely deployed. Sensor nodes are capable of sensing many types of information such as temperature, light, humidity and radiation. Again, locations are employed to meet timing and energy constrains.

In this paper we consider design models suitable for both embedded systems and sensor networks.

* This paper was selected by Vladimir Oleshchuk and Dominique Dallet, IPC Co-Chairmen of IDAACS'09 Workshop, www.idaacs.net

2. RELATED WORK

Models for embedded systems are discussed in [1, 2, 3, 4, 5, 6]. Models of systems that interact by message exchange are introduced in [1]. The models deal with data, states, interfaces and processes. A modeling theory must address the following issues: in the data view, the data model and algebras; in the usage view, the interface model and features and component models; in the structure view, the distributed system model and composed systems; in the state view, the state machine model and state transitions; and, in the process view, the process model, events, and interaction [2]. Using structured models is a way to master complexity. Conditional process graph is used as a model for representing the behavior of the application [3]. Also, the incremental design process is an important theme in [3]. Typically, the design starts from an already existing system running a certain application and the design problem is to implement new functionality on this system [3]. Design methodologies and models of computation are discussed in [5]. An adaptive life-cycle model of hardware/software codesign is introduced in [6]. The model helps manufacturers to reduce production cost and maximize profits by modifying codesign options over the life cycle.

Techniques for minimizing and balancing I/O during functional partitioning are described in [7]. In related research, we proposed an I/O-driven design method [8].

Different MAC protocols for sensor networks are discussed in [9, 10, 11]. Energy efficiency is the primary goal of the research. Methods for low-power multihop communication are discussed in [10, 12, 13]. Energy models for wireless communication are in focus in [14, 15].

3. EMBEDDED SYSTEM DESIGN MODEL

We introduce an embedded system design model as

$$\mathbf{D} = \{\mathbf{A}, \mathbf{S}, \mathbf{R}\} \tag{1}$$

where **A** is the model of the application, **S** is the model of the embedded systems and **R** is the model of the resources needed. Next, we define the model of the application as

$$\mathbf{A} = \{\mathbf{T}, \mathbf{L}, \mathbf{I}\} \tag{2}$$

where **T** is the model of the embedded system functionality which the application dictates, **L** is the model of the physical distribution of the application and **I** is the model of the interface application – embedded system. The embedded system functionality can be implemented either in software or in hardware.

$$\mathbf{T} = \{\mathbf{T_H}, \mathbf{T_S}\} \tag{3}$$

where $\mathbf{T_H}$ is the functionality directed to hardware and $\mathbf{T_S}$ is the functionality directed to software. In addition, we need to model the partitioning of the functionality into tasks.

$$\mathbf{T_S} = \{\mathbf{T_1}, \mathbf{T_2}, \ldots \mathbf{T_{n(T_S)}}\} \tag{4}$$

Since the application is distributed over a set of locations, we describe the physical distribution as

$$\mathbf{L} = \{\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}, \mathbf{L_4}, ..\mathbf{L_{n(L)}}\} \tag{5}$$

The model of the interface can be partitioned as

$$\mathbf{I} = \{\mathbf{IN_D}, \mathbf{IN_A}, \mathbf{OUT_D}, \mathbf{OUT_A}\} \tag{6}$$

where $\mathbf{IN_D}$, $\mathbf{IN_A}$, $\mathbf{OUT_D}$ and $\mathbf{OUT_A}$ stand for the models of digital inputs, analog inputs, digital outputs and analog outputs respectively. We define these models as

$$\mathbf{IN_D} = \{\mathbf{IN_{D1}}, \mathbf{IN_{D2}}, \ldots, \mathbf{IN_{Dn(L)}}\} \tag{7}$$

$$\mathbf{IN_A} = \{\mathbf{IN_{A1}}, \mathbf{IN_{A2}}, \ldots, \mathbf{IN_{An(L)}}\} \tag{8}$$

$$\mathbf{OUT_D} = \{\mathbf{OUT_{D1}}, \mathbf{OUT_{D2}}, \ldots, \mathbf{OUT_{Dn(L)}}\} \tag{9}$$

$$\mathbf{OUT_A} = \{\mathbf{OUT_{A1}}, \mathbf{OUT_{A2}}, \ldots, \mathbf{OUT_{An(L)}}\} \tag{10}$$

where location is the axis along which the partitioning is organized.

4. LOCATION-DRIVEN DESIGN

Tasks are mapped into locations as defined

$$\mathbf{TL} = \{\mathbf{T_{L1}}, \mathbf{T_{L2}}, \ldots, \mathbf{T_{Ln(L)}}\} \tag{11}$$

where $\mathbf{T_{Li}}$ is the set of tasks mapped to location $\mathbf{L_i}$.

Each task is mapped to the location where the number of its analog inputs is maximal.

Algorithm 1

```
1:  s=0
2:  for $1 \le i \le n(\mathbf{T_S})$ do
3:      for $1 \le j \le n(\mathbf{L})$ do
4:          if $\mathbf{IN_{ATi}} > s$
5:              $s = \mathbf{IN_{ATi}}$
6:              for $1 \le k \le n(\mathbf{L})$ do
7:                  if $\mathbf{T_i} \subset \mathbf{TL}(k)$
8:                      $\mathbf{TL}(k) = \mathbf{TL}(k) \cap \overline{\mathbf{T_i}}$
9:                  end if
10:             end for
11:             $\mathbf{TL}(j) = \mathbf{TL}(j) \cup \mathbf{T_i}$
12:         end if
13:     end for
14: end for
15:     if $s$=0
16:         for $1 \le i \le n(\mathbf{T_S})$ do
```

```
17:         for $1 \le j \le n(\mathbf{L})$ do
18:             if $\mathbf{IN_{DTi}} > s$
19:                 $s = \mathbf{IN_{DTi}}$ $s = \mathbf{IN}_{\mathbf{DTi}}$
20:                 for $1 \le k \le n(\mathbf{L})$ do
21:                     if $\mathbf{T_i} \subset \mathbf{TL}(k)$
22:                         $\mathbf{TL}(k) = \mathbf{TL}(k) \cap \overline{\mathbf{T}}_\mathbf{i}$
23:                     end if
24:                 end for
25:                 $\mathbf{TL}(j) = \mathbf{TL}(j) \cup \mathbf{T_i}$
26:             end if
27:         end for
28:     end for
29: end if
```

Algorithm 1 describes the mapping procedure. If a task does not have any analog inputs, the mapping is guided by the number of digital inputs.

Allocation of identical hardware modules for the system, homogeneous architecture, can bring the price down. Also, it would be easier to scale the system. The requirements for the hardware module I/O capacity are determined by four most demanding clusters.

$$\mathbf{IN_{DH}} \ge \max_{1\le i\le n(L)}\left(\max_{1\le j\le n(K_i)} n\left(\mathbf{IN_{DKij}} \cap \mathbf{IN_{Di}}\right)\right) \tag{12}$$

where $\mathbf{IN_{DKij}}$ is the set which represents all digital inputs, cluster j related to location $\mathbf{L_i}$.

$$\mathbf{IN_{AH}} \ge \max_{1\le i\le n(L)}\left(\max_{1\le j\le n(K_i)} n\left(\mathbf{IN_{AKij}} \cap \mathbf{IN_{Ai}}\right)\right) \tag{13}$$

where $\mathbf{IN_{AKij}}$ is the set which represents all analog inputs, cluster j related to location $\mathbf{L_i}$.

$$\mathbf{OUT_{DH}} \ge \max_{1\le i\le n(L)}\left(\max_{1\le j\le n(K_i)} n\left(\mathbf{OUT_{DKij}} \cap \mathbf{OUT_{Di}}\right)\right) \tag{14}$$

where $\mathbf{OUT_{DKij}}$ is the set which represents all digital outputs, cluster j related to location $\mathbf{L_i}$.

$$\mathbf{OUT_{AH}} \ge \max_{1\le i\le n(L)}\left(\max_{1\le j\le n(K_i)} n\left(\mathbf{OUT_{AKij}} \cap \mathbf{OUT_{Ai}}\right)\right) \tag{15}$$

where $\mathbf{OUT_{AKij}}$ is the set which represents all analog outputs, cluster j related to location $\mathbf{L_i}$.

The smallest number of hardware modules in location $\mathbf{L_i}$ can be calculated as

$$n\left(M_i^{MIN}\right) = \max\left(\begin{array}{l}\left\lceil \frac{n(I_i)}{IN} \right\rceil, \left\lceil \frac{n(A_i)}{AN} \right\rceil, \left\lceil \frac{n(P_i)}{PN} \right\rceil, \\ \left\lceil \frac{n(Q_i)}{QN} \right\rceil, \left\lceil \frac{n(C_i)}{CN} \right\rceil \end{array}\right) \tag{16}$$

If the smallest number of modules is not sufficient, we gradually increase the number until all I/O requirements are met. The same procedure is applied for all locations.

All possible mappings of clusters toward hardware modules are

$$\underset{1\le i\le n(L),\, 1\le m\le \left(n\left(N_{Li}^{MIN}\right)\right)^{n(K_{Li})}}{\varphi_{\mathbf{i,m}}} : \underset{1\le j\le n(K_{Li})}{\mathbf{K_{ij}}} \rightarrow \{\ \mathbf{N_{i1}}\ \ \mathbf{N_{i2}}\ \ \mathbf{N_{i3}}\ \cdots\ \mathbf{N_{in(N_{Li})}}\} \tag{17}$$

Fig.1 shows how we search to find an acceptable mapping between clusters and hardware modules, location $\mathbf{L_i}$. The search begins with the smallest number of hardware modules, $n(\mathbf{N_{Li}^{MIN}})$. Each time a hardware module appears on a line, the corresponding cluster occupies a fraction of its capacity. A line is considered a successful mapping if all hardware modules provide sufficient capacity to meet the required I/O.

Some applications are characterized by fuzzy relationships between I/O variables and locations. For example, a certain physical variable can be measured either at $\mathbf{L_1}$ or at $\mathbf{L_3}$ [8]. This is an opportunity to optimize the system. Assume the following application.

$$\begin{array}{l}\mathbf{A} = \{\mathbf{T}, \{\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}, \mathbf{L_4}\}, \{\{\{d_1, d_2\}, \\ \{d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}, \{d_{11}, d_{12}\}, \\ \{d_{13}, d_{14}\}\}, \{\{a_1, a_2\}, \{a_3, a_4\}, \emptyset, \emptyset\}, \\ \{\emptyset, \{b_1, b_2, b_3\}, \{b_4\}, \emptyset\}, \{\{c_1\}, \emptyset, \emptyset, \emptyset\}\}\}\end{array} \tag{18}$$

In $\mathbf{L_1}$ the I/O interface includes two digital variables, d_1 and d_2. In $\mathbf{L_2}$ the embedded systems must sense eight digital variables, d_3 through d_{10}. All other requirements for system's digital inputs can be seen in (18) as well.

In $\mathbf{L_1}$ the system must input two analog variables, a_1 and a_2. In $\mathbf{L_2}$ the system must input two analog variables as well, a_3 and a_4. In the same location the system must influence the environment via three digital outputs, b_1, b_2 and b_3. Finally, in $\mathbf{L_1}$ the embedded system must possess an analog output, c_1.

The required digital I/O capacity emerges as

$$\begin{array}{l}\mathbf{IN_{DH}} + \mathbf{OUT_{DH}} \ge \max(n(\emptyset \cap \{d_1, d_2\}), n(\{d_1, d_2\} \\ \cap \{d_1, d_2\}), n(\{d_1, d_2, d_3, d_4\} \\ \cap \{d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}), \\ n(\emptyset \cap \{d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}), \\ n(\{d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\} \\ \cap \{d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}), \\ n(\{d_{11}, d_{12}, d_{13}, d_{14}\} \cap \{d_{11}, d_{12}\}) + \max(n(\emptyset \cap \emptyset\}), \\ n(\{b_1\} \cap \emptyset), n(\{b_2\} \cap \{b_1, b_2, b_3\}), n(\emptyset \cap \{b_1, b_2, b_3\}), \\ n(\{b_3\} \cap \{b_1, b_2, b_3\}), n(\{b_4\} \cap \{b_4\}) \\ = \max(0,2,2,0,8,2) + \max(0,0,1,0,1,1) = 9\end{array} \tag{19}$$

The required number of analog inputs can be calculated as

$$\begin{aligned}&\mathbf{IN_{AH}} \geq \max(n(\{a_1,a_2\} \cap \{d_1,d_2\}), n(\phi \cap \{a_1,a_2\}),\\ &n(\phi \cap \{a_3,a_4\}), n(\{a_3,a_4\} \cap \{a_3,a_4\}),\\ &n(\phi \cap \{a_3,a_4\}), n(\phi \cap \phi) = \max(2,0,0,2,0,0) = 2\end{aligned} \tag{20}$$

The required number of analog outputs can be calculated as

$$\begin{aligned}&\mathbf{OUT_{AH}} \geq \max(n(\{c_1\} \cap \{c_1\}), n(\phi \cap \{c_1\}),\\ &n(\phi \cap \phi), n(\phi \cap \phi),\\ &n(\phi \cap \phi), n(\phi \cap \phi) = \max(1,0,0,0,0,0) = 1\end{aligned} \tag{21}$$

Assume that the microcontroller with smallest I/O capacity has eight digital inputs and outputs altogether. Also, this component has four analog inputs and one analog output. Taking into account communication interface pins and power supply pins the total number of pins emerges as twenty. The next in size microcontroller has 16 digital I/Os and 28 pins. Once homogeneous architecture has been adopted for the current example, the 28-pins microcontroller must be allocated for all locations. The resulting overall number of pins becomes 112.

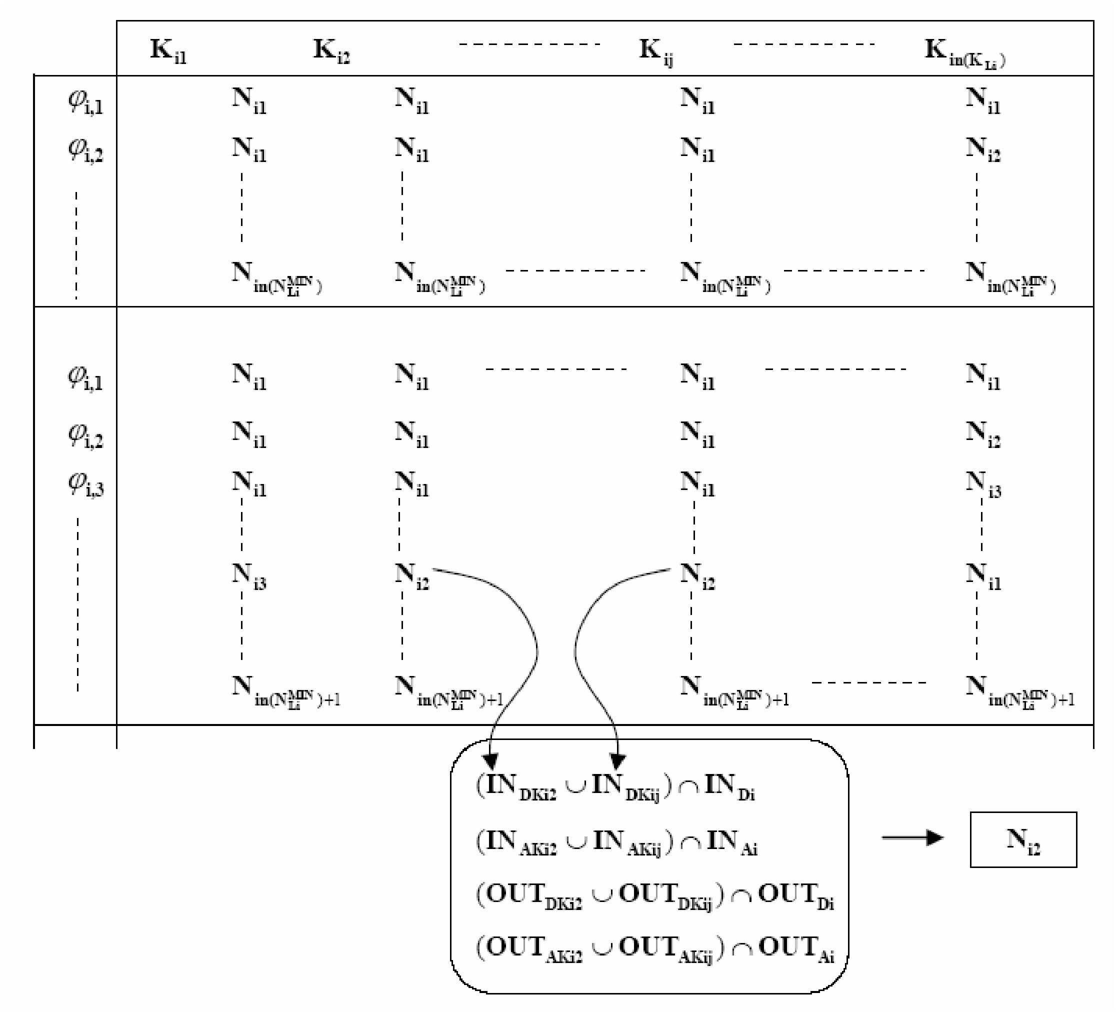
In case of fuzzy relationships for d_3, d_4 and d_5 which can be sensed either at $\mathbf{L_2}$ or at $\mathbf{L_1}$, the following description of the application

$$\begin{aligned}&\mathbf{A} = \{\mathbf{T}, \{\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}, \mathbf{L_4}\}, \{\{\{d_1,d_2,d_3,d_4,d_5\},\\ &\{d_6,d_7,d_8,d_9,d_{10}\}, \{d_{11},d_{12}\}, \{d_{13},d_{14}\}\}, \{\{a_1,a_2\},\\ &\{a_3,a_4\}, \phi, \phi\}, \{\phi, \{b_1,b_2,b_3\}, \{b_4\}, \phi\}, \{\{c_1\}, \phi, \phi, \phi\}\}\}\end{aligned} \tag{22}$$

results in

$$\begin{aligned}&\mathbf{IN_{DH}} + \mathbf{OUT_{DH}} \geq \max(n(\phi \cap \{d_1,d_2,d_3,d_4,d_5\}),\\ &n(\{d_1,d_2\} \cap \{d_1,d_2,d_3,d_4,d_5\}),\\ &n(\{d_1,d_2,d_3,d_4\} \cap \{d_6,d_7,d_8,d_9,d_{10}\}),\\ &n(\phi \cap \{d_6,d_7,d_8,d_9,d_{10}\}),\\ &n(\{d_3,d_4,d_5,d_6,d_7,d_8,d_9,d_{10}\}\\ &\cap \{d_6,d_7,d_8,d_9,d_{10}\}), n(\{d_{11},d_{12},d_{13},d_{14}\} \cap \{d_{11},d_{12}\})\\ &+ \max(n(\phi \cap \phi)), n(\{b_1\} \cap \phi), n(\{b_2\} \cap \{b_1,b_2,b_3\}),\\ &n(\phi \cap \{b_1,b_2,b_3\}), n(\{b_3\} \cap \{b_1,b_2,b_3\}), n(\{b_4\} \cap \{b_4\})\\ &= \max(0,2,0,0,5,2) + \max(0,0,1,0,1,1) = 6\end{aligned} \tag{23}$$

Consequently, the embedded systems will include four small microcontrollers, 80 pins altogether. Compared to the previous version, the number of pins is reduced by 29%.



$\mathbf{K_{i1}}$
$\mathbf{K_{i2}}$
$\mathbf{K_{ij}}$
$\mathbf{K_{in(K_{Li})}}$
$\varphi_{i,1}$
$\varphi_{i,2}$
$\varphi_{i,3}$
$\mathbf{N_{i1}}$
$\mathbf{N_{i2}}$
$\mathbf{N_{i3}}$
$\mathbf{N_{in(N_{Li}^{MIN})}}$
$\mathbf{N_{in(N_{Li}^{MIN})+1}}$
$(\mathbf{IN_{DKi2}} \cup \mathbf{IN_{DKij}}) \cap \mathbf{IN_{Di}}$
$(\mathbf{IN_{AKi2}} \cup \mathbf{IN_{AKij}}) \cap \mathbf{IN_{Ai}}$
$(\mathbf{OUT_{DKi2}} \cup \mathbf{OUT_{DKij}}) \cap \mathbf{OUT_{Di}}$
$(\mathbf{OUT_{AKi2}} \cup \mathbf{OUT_{AKij}}) \cap \mathbf{OUT_{Ai}}$
$\mathbf{N_{i2}}$

Fig. 1 – Clusters mapping

5. SENSOR NETWORKS

The model of the embedded system

$$\mathbf{S} = \{\mathbf{H}, \mathbf{N}\} \tag{24}$$

where **H** is the model of the hardware and **N** is the model of networking between hardware blocks. With the expansion of the physical environment the needs of sensing and control will require a larger number of hardware blocks.

$$\mathbf{H} = \{\mathbf{N}_1, \mathbf{N}_2, \ldots \mathbf{N}_{\mathbf{n(H)}}\} \tag{25}$$

To optimize for price the hardware should be based on cheap, homogeneous modules. In case of

$$\mathrm{n}(\mathbf{H}) = \mathrm{n}(\mathbf{L}) \tag{26}$$

and wireless links between the hardware blocks sensor networks emerge as a class embedded systems. The communication model is

$$\mathbf{N} = \{\mathbf{M}, \mathbf{E}\} \tag{27}$$

where **M** is the medium access control model and **E** is the energy model.

6. MEDIUM ACCESS CONTROL MODEL

Medium access control (MAC) mechanism has a significant impact on the energy efficiency [9, 10, 11, 12]. Currently available MAC protocols for wireless sensor networks can be broken down into two major types: contention-based and scheduled-based. While under contention-based protocols nodes compete among each other for channel access, scheduled-based schemes rely on prearranged collision-free links between nodes. There are different methods to assign collision-free links to each node. Links may be assigned as time slots, frequency bands, or spread spectrum codes. However, size and cost constrains may not permit allocating complex radio subsystems for the node architecture. Logically, TDMA scheduling is the most common scheme for the domain of wireless sensor networks. The limited communication range of network nodes provides an extra opportunity for collision-free interaction, space division access [10, 11].

In order to save energy nodes should stay in a sleeping mode as long as possible. Ideally, nodes should have prearranged collision-free links and wake up only to exchange packets. This MAC approach can be termed Assume Scheduled Links, **ASL**. The **ASL** model has two parameters: a packet length in bits, p, and a bit rate, B.

$$\mathbf{M} = \{\mathbf{ASL}, \mathrm{p}, \mathrm{B}\} \tag{28}$$

Beacon Advertise Transmit, BAT, model is a widespread MAC mechanism [9]. Beacons are employed to synchronize internode communications. A beacon period, $\mathrm{T_B}$, includes two major sections. The period begins with a traffic indication window, $\mathrm{T_A}$. During $\mathrm{T_A}$ all nodes are listening and pending packets are advertised. The nodes addressed till the end of $\mathrm{T_A}$ send acknowledgements and receive data packets. Data transmissions are followed by acknowledgement frames to confirm successful reception. Fig.2 illustrates a beacon period.

The BAT model has five parameters: $\mathrm{T_A}$, $\mathrm{T_B}$, a data packet length in bits, p, a control packet length in bits, q, and a bit rate, B.

$$\mathbf{M} = \{\mathbf{BAT}, \mathrm{T_A}, \mathrm{T_B}, \mathrm{p}, \mathrm{q}, \mathrm{B}\} \tag{29}$$

ALS-MAC is a single channel MAC protocol suitable for location-aware networks [11]. The protocol employs beacons to synchronize internode communications. Scalability and collision avoidance are achieved via contention-based advertising slots mapped to scheduled-based transmission slots. ALS-MAC has two dedicated modes of operation for data intensive traffic which assign temporary priority to nodes with many or long buffered packets. Beacons and signaling frames are combined to reduce the collision and control packet overhead.

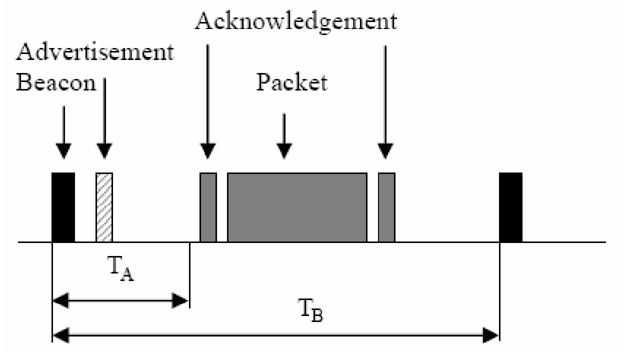


Fig. 2 – Beacon period

7. ENERGY MODEL

The energy used to send a bit over a distance *d* via radio communication may be written as

$$\mathrm{E} = ad^n + b \tag{30}$$

where *a* is a proportionality constant [14]. The radio parameter *n* is a path loss exponent that describes the rate at which the transmitted power decays with increasing distance. Typically, *n* is between 2 and 4. The *b* constant is associated with specific receivers, CPUs and computational algorithms. Thus the model emerges as

$$\mathbf{E} = \{a, n, b, \mathrm{P_R}\} \tag{31}$$

where P_R is the power consumption of a turned on receiver.

A combination of the **ASL** model for medium access control and the energy model allows to calculate the optimal distance between nodes for simple settings. Assume that equally spaced nodes are deployed over a distance *d*. A packet sent from the source will reach the destination via *h* hops. The communication energy

$$\mathrm{E} = \mathrm{ph}(a(d/h)^n + b) + \mathrm{h}(\mathrm{p/B})\mathrm{P_R} \qquad (32)$$

We get for the first derivate

$$\mathrm{E}' = (1-n)\mathrm{p}ad^n\mathrm{h}^{-n} + \mathrm{p}b + (\mathrm{p/B})\mathrm{P_R} \qquad (33)$$

$$\begin{aligned}\mathrm{E}'' &= a\mathrm{p}d^n\mathrm{n}^2\mathrm{h}^{-n-1} - a\mathrm{p}d^n\mathrm{nh}^{-n-1} = \\ &a\mathrm{p}d^n\mathrm{nh}^{-n-1}(n-1) > 0\end{aligned} \qquad (34)$$

Therefore, the energy has a minimum for

$$\mathrm{h} = (((\mathrm{n}-1)ad^n)/(b + \mathrm{B}^{-1}\mathrm{P_R}))^{1/n} \qquad (35)$$

Fig.3 shows the energy for different number of hops under the specified conditions. Simulation results suggest that for a distance of 600 m a single intermediate node is required. If the source-destination distance is 1000 m three intermediate nodes will minimize the energy.

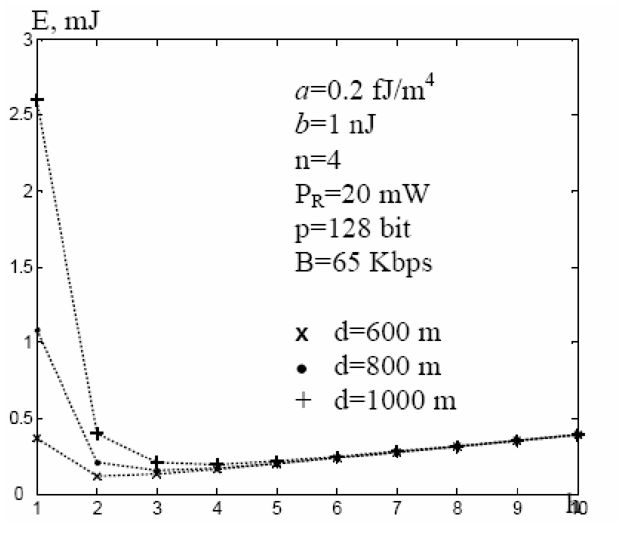


Fig. 3 – Energy for different number of hops

6. CONCLUSION

We have developed a hierarchical design model for embedded systems. A location-driven method for design of distributed embedded systems is proposed as a direct extension of the design model. The method allows to benefit from fuzzy relationships between I/O variables and locations. A direct extension of the model can be employed for communication modeling of wireless sensor networks.

8. REFERENCES

[1] M. Broy. Hierarchies of models for embedded systems. *Proc. First ACM and IEEE Conference on Formal Methods and Models for Co-Design*, 2003, pp. 183-190.

[2] M. Broy. The ‘grand challenge’ in informatics: engineering software-intensive systems, *IEEE Computer*, October 2006, pp. 72-80.

[3] P. Pop, P. Eles and Z. Peng. *Analysis and Synthesis of Distributed Real-Time Embedded Systems*, Kluwer, 2004.

[4] T. A. Henzinger and J. Sifakis. The discipline of embedded systems design. *IEEE Computer*, October 2007, pp. 32-40.

[5] W. Wolf. *High-Performance Embedded Computing*, Morgan Kaufmann, 2007.

[6] D. Lee, H. P. In, K. Lee, S. Park and M. Hinchey. A survival kit: adaptive hardware/software codesign life-cycle model, *IEEE Computer*, February 2009, pp. 100-102.

[7] F. Vahid. Techniques for minimizing and balancing I/O during functional partitioning, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, No. 1, January 1999, pp. 69-75.

[8] Z. Karakehayov and E. Saramov. A fuzzy geography approach to hardware-software co-design of distributed embedded systems. *IEEE International Workshop on Embedded Fault-Tolerant Systems*, Dallas, USA, 1996.

[9] D. Dewasurendra and A. Mishra. Design challenges in energy-efficient medium access control for wireless sensor networks. in *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, M. Ilyas and I. Mahgoub, Eds., CRC Press LLC, 2005, pp. 28-1–28-25.

[10] D. Puccinelli and M. Haenggi. Wireless sensor networks: applications and challenges of ubiquitous sensing. *IEEE Circuits and Systems Magazine*, pp. 19-29, 2005.

[11] Z. Karakehayov and N. Andersen. Energy-efficient medium access for data intensive wireless sensor networks. *In Proc. Int. Workshop on Data Intensive Sensor Networks*, Mannheim, Germany, pp. 116-120, May 2007.

[12] I. Stojmenovic and Xu Lin. Power aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed*

Systems, vol. 12, no. 11, pp. 1122-1133, November 2001.

[13] Z. Karakehayov. Low-power communication for wireless sensor-actuator networks. *In Proc. of the Fifth IASTED Int. Conference on Communication Systems and Networks*, Palma de Mallorca, Spain, ACTA Press, Aug. 2006, pp. 1-6.

[14] J. L. Gao. *Energy Efficient Routing for Wireless Sensor Networks*, Ph.D. dissertation, University of California, Los Angeles, 2000.

Zdravko Karakehayov *is an Associate Professor in the Department of Computer Systems at the Technical University of Sofia, Bulgaria. Formerly he was with the Technical University of Denmark, Lyngby and the University of Southern Denmark, Sønderborg. He is a senior member of the IEEE Computer Society. Dr. Karakehayov currently chairs the Computer Chapter, IEEE Bulgarian section. He co-authored five books in the field of embedded systems and holds eight patents. His research field includes low-power design for embedded systems, low-power and secure routing for wireless sensor networks.*