Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
89(4) 435–450
© The Author(s) 2011
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0037549711400778
sim.sagepub.com

Linking information and energy—activity-based energy-aware information processing

Xiaolin Hu[1] and Bernard P Zeigler[2,3]

Abstract

We present an activity-based framework that links information and energy. The activity-based framework uses a quantization-based approach for modeling information processing and defines weighted activity to model the energy consumption of information processing. We provide a formal description of this framework and use simulation to show how it enables one to study the interaction between information and energy in energy-aware information processing. An existing discrete event system specification (DEVS)-based simulation environment, DEVS-FIRE, is employed to model wireless sensor nodes for detecting and monitoring wildfires. Simulation experimental results confirm the utility of the activity-based framework to support the analysis and design of energy-aware information processing systems.

Keywords

activity, energy-aware information processing, discrete event system specification, sensor node, wildfire

1. Introduction

Energy is the general concept that represents the physical cost of action in the real world. Information is the general concept that enables us to model how systems decide on, manage, and control their actions. As in Figure 1(a), information and energy are two key concepts whose interaction is well understood in the following common sense manner: on one hand, information processing takes energy; on the other hand, getting that energy requires information processing to find and consume energy-bearing resources. Systems that sustain themselves in the real world must somehow balance these quantities, but without a more rigorous formulation of this relationship it is difficult to study this balance in a general way. We need a more formal concept of activity (Figure 1(b)) to enable us to link energy and information.

Activity is a measure of change in system behavior – when it is divided by a quantum it gives the least number of events required to simulate the behavior with that quantum size.[1] One of the unique properties of discrete event system specification (DEVS)[2] (a brief description of DEVS is given in Section 2) is the intrinsic ability of the simulator to be aware of and, therefore, count internal and external state transitions in the model components. Let us measure information processing in a model by such state-to-state transition counts over some time interval, and call this the *activity* measure. Intuitively, components with higher counts over this interval are more actively involved in the information processing than those with low counts. This makes the connection between activity and information. To make the connection with energy, we need to link transition counts with the actual cost of information processing in terms of energy.

It would be nice to postulate that every state transition consumes the same amount of energy, since then the number of transitions relates directly to energy consumed. However, for reasons we discuss later, this is not a practical option. Instead, we allow the modeler

[1]Department of Computer Science, Georgia State University, USA.
[2]Arizona Center for Integrative Modeling and Simulation (ACIMS), C4I Center, George Mason University, USA.
[3]RTSync Corp., USA.

Corresponding author:
Xiaolin Hu, Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA.
Email: xhu@cs.gsu.edu

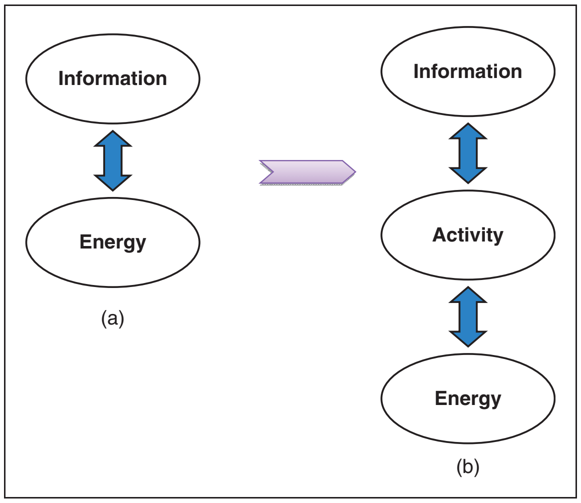

Figure 1. Activity as the concept linking information and energy.

to define a weighted transition mapping and consider this definition to be part of the model itself. In other words, the declaration concerning relative weights of state transitions is a property of a model. It is this property that enables a modeler to abstract the underlying details of energy consumption and directly link energy consumption to information processing. As is usual in modeling methodology, the level of detail required to adequately describe the transition weighting depends on the modeler's objectives. For example, if the objective is to manage energy at coarse levels, such as low, medium, and high, then fairly coarse representation of the transition weights may well be meaningful. 'Activity' in this sense is an abstraction intended to help describe the energy consumption of complex and disparate information processes using a common approach.

Linking information and energy using the concept of activity makes it possible to study the relationship and interaction between information and energy under a formal and integrated framework. On one hand, by modeling the activity we can monitor, in real time, the energy consumption of information processing. On the other hand, the energy-consumption measurement can be used as a controlling parameter for steering the information processing to support energy-aware management. An application of this activity-based framework is to support the analysis and design of energy-aware information processing systems. In this paper, we provide a formal description of this framework and show how it can support the design and analysis of energy-aware wireless sensor nodes for detecting and monitoring wildfires.

2. The activity-based framework to link information and energy

The activity-based framework that links information and energy is based on DEVS where information processing is modeled by state transitions of DEVS models. DEVS is a formalism derived from generic dynamic systems theory.[2] A basic DEVS component (an atomic model) is described by a structure $<X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta>$, where the sets X, Y, and S are the input, output, and state sets, respectively. The functions δ_{int}, δ_{ext}, and δ_{con} are the internal, external, and confluent state transition functions, respectively. The internal transition function $\delta_{int}: S \rightarrow S$ specifies the system's state change due to internal time events. The external transition function $\delta_{ext}: Q \times X \rightarrow S$ specifies the system's state change in response to external inputs, where $Q \in \{(s,e) \mid s \in S, 0 \leq e \leq ta(s)\}$ is the *total state* set, e is the *time elapsed* since last transition, and ta is the time advance function. The confluent transition function describes the evolution of the system state when internal and external events coincide. The function λ is the output function, and ta is the time advance function, which is used to schedule output and internal events. More details of DEVS and the DEVS formalism can be found in Zeigler et al.[2]

Building on the DEVS formalism, this section provides a formal description of the activity-based framework. We first establish the link between activity and energy by defining *weighted activity* as weighted state transitions, which allow modelers to specify the energy consumption (the weights) associated with the transitions of states. Then we present the link between information and activity. This is achieved through a quantization-based approach for modeling information processing as state transitions defined by a quantizer. In the theory of quantization,[2] a quantizer is a significant event detector that monitors its input and uses a logical condition to decide when a significant change occurs. A *quantum* is measure of how big a change must be to be considered significant. We then describe the two aspects of a quantizer, the information aspect (measured by activity) and the energy aspect (measured by weighted activity), to complete the link between information and energy. Finally, we present an activity-based energy-aware information processing framework that utilizes the feedback from weighted activity to the quantizer for supporting energy-aware information processing.

2.1. *Linking activity and energy – measuring energy using weighted activity*

Given a DEVS model that models information processing as state transitions, the energy consumption of information processing can be computed from individual state transition energy consumptions, which are specified by the modeler using a *weighting function* associated with the state transitions. In this paper, we define *weighted activity* as the sum of weighted state transitions over some time interval. This weighted activity is a direct measure of energy consumption based on the weighting functions specified by the modeler. Note that we use the term weighted activity to differentiate it from the previous definition of *activity*, which was defined for continuous time functions[1,3,4] (see Section 2.2). The weighted activity is measured by the weighted transition counts. As will be shown later, the weighted activity and activity is directly related.

Corresponding to the internal transition function δ_{int} and external transition function δ_{ext} of a DEVS model, the internal transition weighting function wt_{int} and external transition weighting function wt_{ext} need to be specified (the weighting function for confluent transitions can also be specified if needed). Then the internal transition weighted activity n_{int} and external transition weighted activity n_{ext} can be computed. Mathematically, the semantics of weighting is formalized as follows.

The internal transition weighting function:

$$wt_{\text{int}}: S \rightarrow \text{Integer}$$

whenever

$$s \rightarrow \delta_{\text{int}}(s) \Rightarrow n_{\text{int}} = n_{\text{int}} + wt_{\text{int}}(s)$$

The average weighted *activity* accumulated over an interval $(t, t^{'})$ from internal transitions:

$$A_{\text{int}}(t,t^{'}) = \frac{n_{\text{int}}}{t^{'} - t}$$

The external transition weighting function:

$$\begin{aligned}&wt_{\text{ext}}: Q \times \text{X} \rightarrow \text{Integer}\\&\text{whenever}\\&s \rightarrow \delta_{\text{ext}}(s,\ e,\ x)\\&\ \Rightarrow\\&n_{\text{ext}} = n_{\text{ext}} + wt_{\text{ext}}(s,\ e,\ x)\end{aligned}$$

The average weighted activity accumulated over an interval $(t, t^{'})$ from external transitions:

$$A_{\text{ext}}(t,t^{'}) = \frac{n_{\text{int}}}{t^{'} - t}$$

The average weighted activity accumulated over an interval $(t, t^{'})$ from all transitions:

$$A(t,t^{'}) = A_{\text{int}}(t,t^{'}) + A_{\text{ext}}(t,t^{'})$$

Default definitions set the weighting functions to unity:

$$wt_{\text{int}}(s) = wt_{\text{ext}}(s,\ e,\ x) = 1$$

Under these conditions, transitions are counted over a period of time and the average weighted activity over the interval is the number of transitions divided by the interval length.

Another common possibility is to set

$$\begin{aligned}wt_{\text{int}}(s) &= intTime(ta(s))\\wt_{\text{ext}}(s,\ e,\ x) &= intTime(e)\end{aligned}$$

where the weight of a transition is proportional to the time spent in the state before the transition. The *intTime*() is a rounding function or other means of turning a real number into an integer. For example, if a processor remains in a processing state for time 10.1, then the number of transitions is incremented by $intTime(10.1) = 10$ when it leaves the state. Likewise if it is interrupted by an external event after time e while processing, then the number is incremented by $intTime(e)$. To get this same effect, we could redefine the model to make actual transitions proportional to the time spent in processing. However, this would incur unnecessary inefficiency that contradicts the very basis of discrete event modeling.

Note that we could model processing in a more detailed manner, for example, by including a description of the job being performed and thereby obtain a more refined estimate of the weighted activity involved. However, there is no natural place to stop such refinement – other than letting our objectives guide such termination, as discussed above. Finally, if we are not interested in observing the weighted activity of some components, we can selectively set their weighting functions identically zero.

The weighting functions described above make it possible to calculate the weighted activity as a measurement of energy consumption for a given model with discrete transitions. This makes the connection between activity and energy. To make the connection between activity and information, one needs to model information processing as a discrete event model from which weighted activity can be calculated. Developing a discrete event model is straightforward if the information processing is discrete by nature, for example, the change of system state from *active* to *passive* and then to *sleep*, and vice versa. However, for systems that have continuous components, this straightforward approach is not possible. One approach is to measure the discrete activity for a discrete event approximation and ascribe that to the original continuous model. However, with myriad approximations to consider, there is no guarantee that there will be a consistent result. Fortunately, for differential equation models there is concept of activity that provides an intrinsic measure that relates to the number of transitions of an approximating discrete event model through quantization. The quantization-based approach is presented below.

2.2. Linking information and activity – quantization-based information processing

Activity is a measure of change in system behavior. The following is the definition of activity for a continuous segment (the description is adapted from Muzy et al.[3]). It provides a precise measure of the computational effort required by an ideal quantizer. In fact, we will show that it is a reasonable estimate of the computational effort required by implementations of these devices.

In Figure 2, $\Phi(t)$ is a continuous function of time; D corresponds to the quantum (the minimum threshold for change below which no processing occurs), and m_i corresponds to the maxima and minima of the curve, where the first and last m_i are the values of the function at the initial and final times.

The activity in an interval [0, T] can be calculated by summing the differences between the adjacent maxima and minima, that is

$$A(T) = \sum_i |m_{i+1} - m_i| \tag{1}$$

The average activity in an interval [0, T] is given by

$$AvgActivity(T) = \frac{A}{T}$$

The following fact is important because it relates the number of threshold crossings made by a DEVS simulator, the activity over a time interval T, and the quantum size D.

Fact: The number of threshold crossings in an interval of length T for threshold levels that are equally spaced by quantum size, D, is:

$$NumberofThresholdCross(T, q) = \frac{A(T)}{D} \tag{2}$$

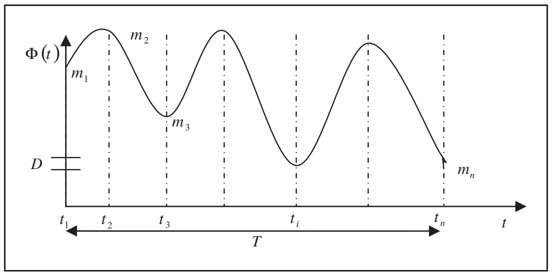

Figure 2. Definition of activity for continuous time segments.

Proof: Break up a curve with a finite number of extrema (minima and maxima) into segments between successive extrema (i.e. between a minimum and the next maximum, or a maximum and the next minimum). These segments are either non-decreasing or non-increasing, as illustrated in Figure 3, showing a non-decreasing segment with minimum, a and maximum, b. Divide up the interval of length $b - a$ into intervals of size D (the quantum size) by a grid as shown. There are $(b - a)/D$ such intervals and no matter what the continuous curve $f(t)$ looks like, it must cross each of the grid lines exactly once (where, if as illustrated by the last crossing, it stays on the grid line we count this as one crossing). Thus for any non-increasing segment, the number of threshold crossings is the distance from the minimum to the maximum divided by the quantum size. It is easy to see that a similar situation holds for a non-decreasing segment (where the distance is the absolute value of the difference). So in any inter-extrema segment we have

$$\#crossing = \frac{|m_{i+1} - m_i|}{D}$$

Now since the activity is the sum of successive distances between extrema (Equation (1)), it easily follows that the number of threshold crossings is the activity divided by the quantum size (Equation (2)). Equation (2) holds for continuous curves with a finite number of extrema in an interval – there will be slight error that is bounded by the quantum size, which will disappear as the quantum goes to zero. By definition, Equation (2) will be true for any quantizer that takes exactly one quantum step at each transition and tracks the curve exactly. ■

2.3. Information and energy – activity and weighted activity of quantizer

Consider a quantizer that processes information (the external input) using the quantization-based principle described above. The activity of the input stream is

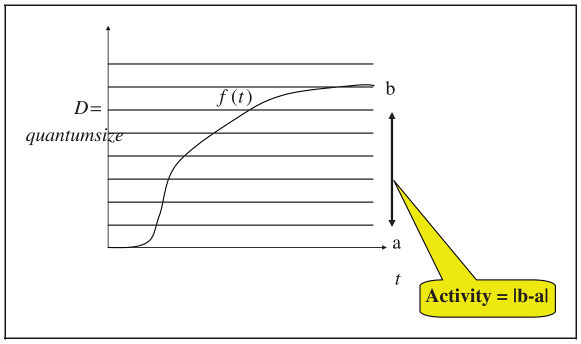

Figure 3. Activity of a non-decreasing segment.

measured by its activity A, as computed in Equation (1). This represents the amount of information that needs to be processed by the quantizer. We now show that activity A is directly relatable to the weighted activity of the quantizer. As described before, the weighted activity is measured by the number of transitions counted with its transition weighting functions. The weighted activity indicates the energy consumption of the quantizer. Below we examine a close relationship between the activity and the weighted activity.

Fact: The weighted activity experienced by a quantizer with a quantum size D is approximately equal to A/D where A is the activity of the input segment for suitable choice of weighting functions. As shown above for continuous curves, A is easily computed from their successive extrema (maxima and minima).

Proof: As D gets smaller, the number of threshold crossings of any continuous curve with activity A approaches the ratio A/D. We choose weighting functions for the quantizer so that a transition is counted as unity just in case the input differs from the last value by more than the quantum. Under these conditions the number of transitions counted in an interval is equal to the number of threshold crossings for the same quantum. Thus the weighted activity of the quantizer approaches the input activity divided by the quantum size, an approximation that becomes better as the quantum size becomes smaller.

From the above proposition, we can see a close relationship between the two activity perspectives – in the ideal case the weighted activity (measurement of energy) is proportional to the activity (measurement of information). In general we can expect that for a well-designed real-world quantizer, the major contribution of its weighted activity would come from the (external) activity of its sensed input stream (with the other part considered as overhead). When we relate weighted activity to energy, say in terms of battery power consumption, then a useful prediction of such consumption will come from the anticipated pattern of activity of the input stream.

Section 3 examines such a relationship using an example of a quantization-based wireless sensor node for information processing.

2.4. Activity-based energy-aware information processing

The foregoing activity results provide a formal foundation to study the relationship and interaction between energy and information. Among the myriad of uses of this framework is to control the operation of the information processing model, such as a quantizer, executed in real time. As illustrated in Figure 4, the controller can adjust parameters of the quantizer, such as the size of the quantum, based on the weighted activity that represents the energy consumption. Given a certain 'energy budget' (the total energy that can be consumed), a control policy might be to increase the quantum size when the remaining energy is insufficient to support information processing under the current quantum size and to decrease the quantum size otherwise (the remaining energy is the total energy minus the consumed energy, which is measured by the weighted activity). In this way, the controller can adjust the sensitivity of the quantizer in line with the energy left, and thus support energy-aware information processing.

Consider a wireless sensor node for detecting and monitoring wildfires as an example. When the sensor node runs low on power, increasing the quantum size (thereby decreasing the transition rate) will save draining the battery and result in increased life time for the sensor to monitor the fire temperature. Conversely, when the sensor node has sufficient energy to cover the fire event, decreasing the quantum size will provide more accurate sensor data for monitoring the fire temperature. As a result, the 'smart' sensor node is able to trade energy consumption in for information precision and vice versa by adaptively adjusting its quantum size in information processing. Section 3 gives a concrete example of such a 'smart' sensor node for detecting and monitoring wildfires.

2.5. Implementation

To implement the weighted activity in DEVS needs extensions at both the modeling level and simulation level. At the modeling level, a modeler defines the weighted transition mapping and considers this definition to be part of the model itself. This means each atomic model is extended to include an internal transition weighting function and an external transition weighting

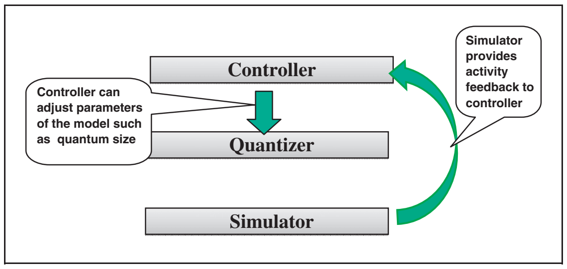
Controller
Quantizer
Simulator
Controller can adjust parameters of the model such as quantum size
Simulator provides activity feedback to controller

Figure 4. Activity feedback to a controller.

function (a confluent transition weighting function can be defined too if needed). At the simulation level, the weighted counting of transitions is readily implemented in the DEVS Abstract Simulator. The simulator for atomic models can tell when a particular transition is about to occur and can therefore invoke the appropriate transition weighting function just before it makes this transition happen. The following modifies the atomic model simulator to implement this concept.

```
Atomic DEVS-Simulator
    variables:
    parent      -- parent coordinator
        tl          -- time of last event
        tn          -- time of next event
    n           -- number of transitions
        DEVS        -- associated model    with total state (s, e)
        y           -- output message bag
    //initialization with i-message
    when receive i-message (i, t) at time t
        $tl = t - e$
        $tn = tl + ta(s)$
    $n = 0$
    //internal event triggered by *-message
    when receive *-message (*, t) at time t
    if $t = tn$ then
        $y = \lambda(s)$
    send y-message (y, t) to parent coordinator
    //external event triggered by x-message
    when receive x-message (x, t) {
    if ($x = \Phi$ and $t = tn$) then
        $n = n + wt_{\text{int}}(s)$
        $s = \delta_{int}(s)$
    else if ($x \mathrel{!=} \Phi$ and $t = tn$) then
        $s = \delta_{con}(s)$
    else if ($x \mathrel{!=} \Phi$ and ($tl \le t \le tn$))
        $e = t - tl$
        $n = n + wt_{ext}(s,e,x)$
        $s = \delta_{ext}(s, e, x)$
    }
    $tl = t$
    $tn = tl + ta(s)$
```

Algorithm 1. Simulator for Atomic DEVS modified to accumulate weighted transitions

The coordinator of a coupled model is extended to include collaboration with all the simulators of its coupled model in which the coordinator can query the simulators to obtain the weighted activity each has accumulated. Through a method, for example, *getSimActivity()*, an atomic model simulator can make the collected weighted activity of its model available upon request. Such requests might come from the coordinator of a coupled model in which the atomic model is a component. The coordinator can in turn provide an array of weighted activities collected from the component simulators for use by the simulation experimenter, whether human or programmatic.

This implementation can easily support the functionality shown in Figure 4. We can place the controller and quantizer into a coupled model to be executed by a real-time coordinator (denoted as *Simulator* in Figure 4). The real-time coordinator obtains the weighted activity from the atomic simulator as the real-time simulation proceeds (see Algorithm 1). The activity outputs of the coordinator can be sent to the controller as external inputs through a dedicated activity input port (e.g. as in DEVSJAVA). Alternatively, the controller can query the coordinator to get the current weighted activity. This activity is used by the controller to adjust parameters of the model, such as the quantum size.

3. A wireless sensor node example

We consider a wireless sensor node for detecting and monitoring wildfires. The sensor node has a global positioning system (GPS) sensor and a temperature sensor for sensing its location and the temperature of the environment. It has a microcontroller for processing data and sending output (here we treat the memory and radio as part of the microcontroller). The sensor node uses a quantization-based temperature sensor (a quantizer) that outputs a temperature to the microcontroller only when the temperature change reaches the quantum size. The quantum size can be set by the microcontroller. The sensor node starts in the inactive state and after activation via an activating input will remain active for a finite time before deactivation. The activating input may come from another sensor node or be triggered internally when detecting some events. Here we assume the activating input is internally generated when there is significant rise in temperature (e.g. the change of temperature bypasses a pre-defined threshold). While active, the sensor node can process temperature as well as GPS location updates, both of which impact its time to remain active. The sensed temperature is paired with the current location to provide location-based output. Such a device has a myriad of applications when replicated and deployed to multiple monitoring locations. Once deployed it can sense its location and remain fixed in place as in a network to detect wildfire behavior. Alternatively, it can be mobile and report geo-referenced data on the move. This would be the case if worn by firefighters in their hats, providing dynamic information on the fire-front perimeter.

3.1. The DEVS model with weighted state transitions

A DEVS model that implements the behavior of the sensor node is given below. Tables 1 and 2 show the external transition function $\delta_{\text{ext}}(s, e, x)$, the internal transition functions $\delta_{\text{int}}(s)$, the time advance function $ta(s)$, and the output function $\lambda(s)$ of the model. In the tables, the *inNextValue* port receives inputs from the quantization-based temperature sensor, and the

Table 1. Internal transition function, time advance function, and output function

Phase	$\delta_{\mathrm{int}}(s)$	$ta(s)$	$\lambda(s)$
WaitForActivation		∞	
WaitForNextValue	WaitForActivation	timeToDeactivate	
sendValue	WaitForNextValue	0	(lastValue, location)

Table 2. External transition function

Phase	Input port	$\delta_{\mathrm{ext}}(s, e, x)$
WaitForActivation	inActivation	WaitForNextValue
WaitForNextValue	inNextValue(val)	lastValue = val
		WaitForNextValue
	inLocation(loc)	location = loc
		WaitForNextValue

Table 3. Parameters of energy consumption weights

Parameter	Definition	Sample value
whileInactiveRate	Time rate at which transition counts increase while not active	1 (per second)
whileActiveRate	Time rate at which transition counts increase while active	10 (per second)
processingActivation	Transition weighting for processing activation input	20
processingLocation	Transition weighting for processing location input	50
processingInput	Transition weighting for processing temperature input	500
sendingOutput	Transition weighting for sending location-based output	20

inLocation port receives inputs from the GPS sensor. The *timeToDeactivate* is a parameter determining how long the sensor node stays in the active state before automatically returning to the inactive state. Another important parameter not shown in the tables is the quantum size of the temperature sensor.

It would be crucial for such a sensor node to be designed with battery power consumption in mind. Both the quantum size of the temperature sensor and the *timeToDeactivate* duration are design choices in configuring the sensor node to optimize power consumption in a particular environment. In this paper, we focus only on the quantum size of the temperature sensor.

The weighted activity approach provides a well-defined basis for studying and designing the energy-aware sensor node. Table 3 lists the parameters of energy-consumption weights that need to be considered in the design decisions. These parameters and their values are derived from the wireless sensor literature (see, e.g. Sinha and Chandrakasan,[5] He et al.,[6] Raghunathan et al.[7] Antoine-Santoni et al.[8,9]). Some adjustments are made in order to better illustrate the activity-based framework. For example, we use a one-time value $processingInput = 500$ to represent the energy consumption for processing a temperature input. This is abstracted from the real energy consumption, which actually depends on the duration for processing the input. In addition, we do not consider the latency overhead associated with transitioning to, and from, the inactive state. The latency overhead exists due to the transition interval – for example, when a processor wakes up, it spends the transition time waiting for the phase-locked loops to lock, the clock to stabilize, and the processor context to be restored. During the transition interval, no productive work can be done and the sensor could miss detecting important events.[5]

Tables 4 and 5 show the internal and external weighting functions defined in terms of these parameters. Note that in the active state, transition counts accrue at the same rate with respect to elapsed time whether accumulated at internal or external transitions.

This kind of consistency is facilitated by defining parameters such as *whileActiveRate*, which can be used in different places to convey equivalent meanings.

The well-defined weighting functions make it possible to analyze some of the weighted transition accumulations. The accumulation for a single activating input after an elapsed time e with no subsequent inputs and ending at the return to inactive state is (for simplicity, in the following, we omit the *intTime* function from the equations)

$$processingActivation + whileInactiveRate \times e + whileActiveRate \times timeToDeactivate$$

which represents the overhead incurred by activating the device with no subsequent sensing.

The accumulation for m temperature inputs and n location updates spread anywhere over an active period of time t is

$$Accum = m \times (processingInput + sendingOutput) + n \times processingLocation + whileActiveRate \times t$$

which represents the part attributable to the external activity of the temperature input stream – $m \times (processingInput + sendingOutput)$ – plus the rest, which is the overhead incurred by processing GPS inputs and the continuous consumption of power while active during the period t. Clearly, the design should attempt to make *whileActiveRate* as small as possible while also reducing *whileInactiveRate*, recognizing that the former will always be larger than the latter. Of course, the processing and transmitting energy consumption should be minimized as well.

3.2. *Activity-based energy-aware sensing using changing quantum size*

Let us consider only the processing of temperature data and analyze how the activity-based approach can support energy-aware sensing by dynamically changing the quantum size of the temperature sensor. Omitting the part related to the GPS data, the accumulated activity for m temperature inputs over an active period of time t is

$$Accum(t) = m \times (processingInput + sendingOutput) + whileActiveRate \times t$$

Recall Equation (2), for a quantization-based temperature sensor m is the number of threshold crossings: $m = A(t)/D$, where $A(t)$ is the external activity (the temperature change) in time interval [0, t]. Assuming that the quantum size is constant, this leads to an expression for the weighted activity accumulated over the interval:

$$Accum\ (t) = (A(t)/D) \times (processingInput + sendingOutput) + whileActiveRate \times t \quad (3)$$

In a typical wildfire scenario, when the fire spreads and gets closer to a sensor node, the sensed temperature gradually increases and reaches its peak point when the fire spreads to the location of the sensor node. After that the fire burns the biomass (represented by a fuel model) in the area and gradually dies out when the biomass is consumed. The expected temperatures at any height above a surface fire can be estimated or calculated from ambient temperature and fireline intensity.[10,11] We denote the peak temperature as R_{max}, and the ambient temperature as $R_{ambient}$. The ambient temperature usually does not vary much and can be treated as a constant. Figure 5 illustrates the typical pattern of temperature change of a sensor node in a wildfire field. At time T_{start} the fire approaches the sensor node and thus the temperature begins to rise above the ambient temperature. At time T_{end} the fire dies out and the temperature returns back to the ambient temperature. The time duration between when the temperature begins to rise and when the temperature drops to normal is the duration of the fire event, denoted as T ($T = T_{end} - T_{start}$). Ideally, the sensor node should monitor the entire duration of the fire event before running out of energy. Note in this paper we do not consider the case that the sensor may be destroyed by the fire.

The overall external activity (the temperature change) of a fire event, as shown in Figure 5, is $A(T)$

Table 4. Internal transition weighting function

Phase	wt_{int} (s)
WaitForNextValue	*whileActiveRate* $\times$ *intTime*(σ)
sendValue	*sendingOutput*

Table 5. External transition weighting function

Phase	Input port	wt_{ext} (s, e, x)
WaitForActivation	inActivation	*processingActivation* + *whileInactiveRate* $\times$ *intTime*(e)
WaitForNextValue	inNextValue	*processingInput* + *whileActiveRate* $\times$ *intTime*(e)
	inLocation	*processingLocation* + *whileActiveRate* $\times$ *intTime*(e)

$=2\times(R_{\max}-R_{\text{ambient}})$. Replacing $A(T)$ in Equation (3), we have

$$Accum_{\text{total}} = (2\times(R_{\max}-R_{\text{ambient}})/D) \times (processingInput + sendingOutput) + whileActiveRate \times T \quad (4)$$

Equation (4) shows that the total weighted activity (the energy consumption) of the sensor node for monitoring the whole fire event is a function of the quantum size D, the peak temperature $R_{\max}$, and the duration of the fire event T. The quantum size D is a parameter that can be controlled by the sensor node. When D is small, the energy consumption would be higher and when D is large the energy consumption would be lower. Different from D, both $R_{\max}$ and T are determined by the fire behavior and cannot be controlled by the sensor node. The peak temperature $R_{\max}$ depends on the fuel type, terrain, and weather condition of the place. The fire duration T depends on how fast the fire spreads, as well as the fuel model at the location. In general, T is smaller if the location is at the head of the fire where fire spreads fast; it is larger if the location is at the tail of the fire. This is because the spreading speed at the tail is much slower. As a result, the biomasses at the tail area are ignited at a slower rate, leading to a longer duration before the fire dies out. Some experimental results of energy consumption with different D and different fire behaviors are given in Section 4.1.

Since quantum size D is a parameter that can be controlled to affect energy consumption, next we study the control policies that dynamically change D for supporting energy-aware sensing. Let us define an energy budget E_{budget}, representing the total energy that is available for monitoring the fire event. We assume the sensor node knows its energy consumption at real time at any given time t. One way it can monitor its energy consumption is to use the accumulated weighted activity computed from weighted state transitions. In this approach, the energy consumption at time t: $Accum(t) = m\times(processingInput + sendingOutput) + whileActiveRate \times (t\text{-}T_{\text{start}})$, where m is the number of temperature inputs and T_{start} is the start time of the fire event. Here, the device maintains the count m of inputs at any time. This can be called a model-based approach and depends on accurate calibration of parameters (*processingInput*, *sendingOutput*, and *whileActiveRate*), as well as the validity of the weighted transition model.

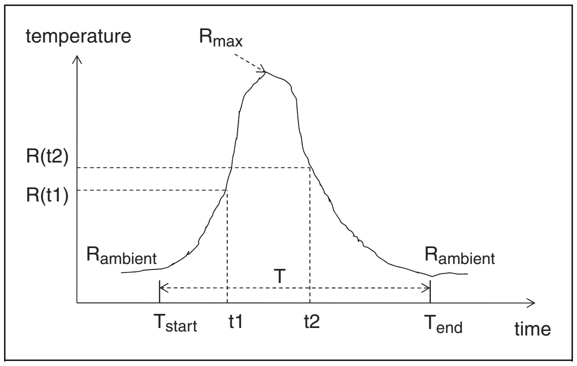

Figure 5. The temperature change pattern of a sensor node.

Let $E(t)$ denote the energy consumption at time t, and $E_{\text{remaining}}(t)$ denote the remaining energy budget. We have: $E_{\text{remaining}}(t) = E_{\text{budget}} - E(t)$. The goal of dynamically changing D at time t is to compute a new quantum size so that the sensor node can finish monitoring the fire event (or monitor the fire event as long as possible) before running out of the remaining energy.

We define $R(t)$ as the sensed temperature at time t, and $A_{\text{remaining}}(t)$ as the expected remaining external activity (temperature change) for the fire event. Based on the observation (see Figure 5) that the temperature always increases to the peak point $R_{\max}$ and then decreases to the ambient temperature R_{ambient} at the end of the fire event, we can calculate $A_{\text{remaining}}(t)$ at time t, regardless of the temperature change curve. To do this, we need to know if at time t the peak temperature has been reached or not (this can be decided, e.g. based on if the current temperature is increasing or decreasing). Figure 5 illustrates two sample time points $t1$ (where the peak temperature is not reached yet), and $t2$ (where the peak temperature has been reached). The values of $A_{\text{remaining}}(t)$ for these two time points are calculated as below:

$$A_{\text{remaining}}(t1) = R_{\max} - R(t1) + R_{\max} - R_{\text{ambient}}$$
$$A_{\text{remaining}}(t2) = R(t2) - R_{\text{ambient}}$$

Knowing the expected remaining external activity $A_{\text{remaining}}(t)$, the expected energy consumption for the sensor node to cover the rest of the fire event can be computed from Equation (3). This energy consumption should not be larger than the remaining energy $E_{\text{remaining}}(t)$. Thus we have

$$E_{\text{remaining}}(t) \geq \left(A_{\text{remaining}}(t)/D\right)\times(processingInput + sendingOutput) + whileActiveRate \times (T_{\text{end}} - t)$$

This equation defines the constraint for selecting the new quantum size. Let $D_{\text{new}}(t)$ be the new quantum size at time t. $D_{\text{new}}(t)$ is computed as

$$D_{\text{new}}(t) \geq A_{\text{remaining}}(t)\times(processingInput + sendingOutput)/\left(E_{\text{remaining}}(t) - whileActiveRate \times (Tend - t)\right) \quad (5)$$

This constraint first imposes the precondition that $E_{\text{remaining}}(t) \geq whileActiveRate \times (T_{\text{end}}-t))$ so that the remaining energy covers the remaining power consumption just for being active. It then stipulates that the quantum be big enough so that the weighted transitions for processing the remaining activity are possible.

There are two unknown parameters in Equation (5) in order to compute $D_{\text{new}}(t)$. The first one is $R_{\max}$, which may be needed to calculate $A_{\text{remaining}}(t)$ ($R_{\max}$ is needed only when the peak temperature is not reached yet at time t). For this unknown parameter, we propose to use some empirically defined value based on the fuel model and typical weather condition[10,11] as an estimation of $R_{\max}$. We note even if the real $R_{\max}$ is different from the empirically defined value, the system still works because eventually the real $R_{\max}$ will be recorded (when the peak is detected). The second unknown parameter in Equation (5) is T_{end}, that is, the ending time of the fire, which is determined by the fire behavior. Below we present two different ways to treat this unknown parameter, which result in two different control policies for dynamically changing the quantum size. We name them the *Adjusted Quantization Policy* and the *Adaptive Quantization Policy*, respectively.

3.2.1. Adjusted quantization policy. In the adjusted quantization policy, we do not estimate the remaining duration of the fire event, denoted as ($T_{\text{end}} - t$) in Equation (5). The simplest way of doing this is to totally ignore the $whileActiveRate \times (T_{\text{end}}-t)$ element, that is, setting $(T_{\text{end}}-t) = 0$. In this way, the result is always optimistic in the sense that the computed D_{new} is always smaller than it should be. This is because the energy consumption of staying in the active state is not taken into account for computing D_{new}. To alleviate this problem, an alternative approach is to always treat ($T_{\text{end}} - t$) as a constant number, which can be empirically defined. Section 4.2 shows some results of the adjusted quantization policy by setting $(T_{\text{end}} - t) = 0$ all the time.

3.2.2. Adaptive quantization policy. In the adaptive quantization policy, we dynamically estimate the remaining duration ($T_{\text{end}}-t$) of the fire event and use that estimation to calculate the quantum size D. We carry out the estimation based on a simple technique similar to extrapolating the slope of a curve in numerical solutions. Specifically, we compute the rate of temperature change from the last quantized input, and use this rate to estimate the duration for the remaining temperature change $A_{\text{remaining}}(t)$. Let $c(t)$ denote the rate of change computed from the last quantized input, then the remaining during ($T_{\text{end}} - t$) is estimated as $(T_{\text{end}} - t) = A_{\text{remaining}}(t)/c(t)$. The adaptive quantization policy is more 'intelligent' in the sense that it adaptively adjusts the quantum size based on what is the current rate of temperature change. Section 4.3 shows some experimental results of the adaptive quantization policy.

Since several estimations are used in computing the quantum size D, the effectiveness of adjusting D will depend on the accuracy of these estimations. However, even when imprecise estimations are used, the control policies still work to some extent. This is because D is dynamically computed at every update and the system has the capability of adjusting itself to overcome the estimation errors.

4. Experimental results

We present several experiment results to demonstrate the activity-based approach described in this paper. The experiments are carried out using wildfire spread simulations based on the DEVS-FIRE model.[12,13] The sensor node model and the state transition weighting functions are described in Section 3.1. Since we focus on the temperature sensor data only, the logic corresponding to the GPS sensor is not implemented in the experiments. In addition, we set the $timeToDeactivate = 600$, which is long enough so the sensor node will not transition to the inactive state before the fire event ends. This allows us to study the quantum size as the only factor for affecting the energy consumption of the sensor node.

Figure 6 shows a snapshot of the wildfire spread simulation and the corresponding temperature map at the moment. In Figure 6(a), the different colors in the background represent different fuel models. The wind speed used in the simulation is 12 miles/hour, while the wind direction is from south to north. The fire front is depicted in red, while the burned area is depicted in black. Figure 6(b) shows the temperature map in the fire field. The temperatures range from 27 to about 300 Celsius, which are displayed in different colors. We consider two identical sensor nodes placed at two different locations in the fire field, as marked in Figure 6(b). Sensor node 1 (denoted as sensor1) is placed at the head of the fire (because the fire spreads from south to north due to wind direction). Sensor node 2 (denoted as sensor2) is placed at the left flank of the fire.

4.1. Quantum size, sensor data and energy consumption

Our first experiment aims to show the effect of quantum size to the sensor data and to the energy consumption

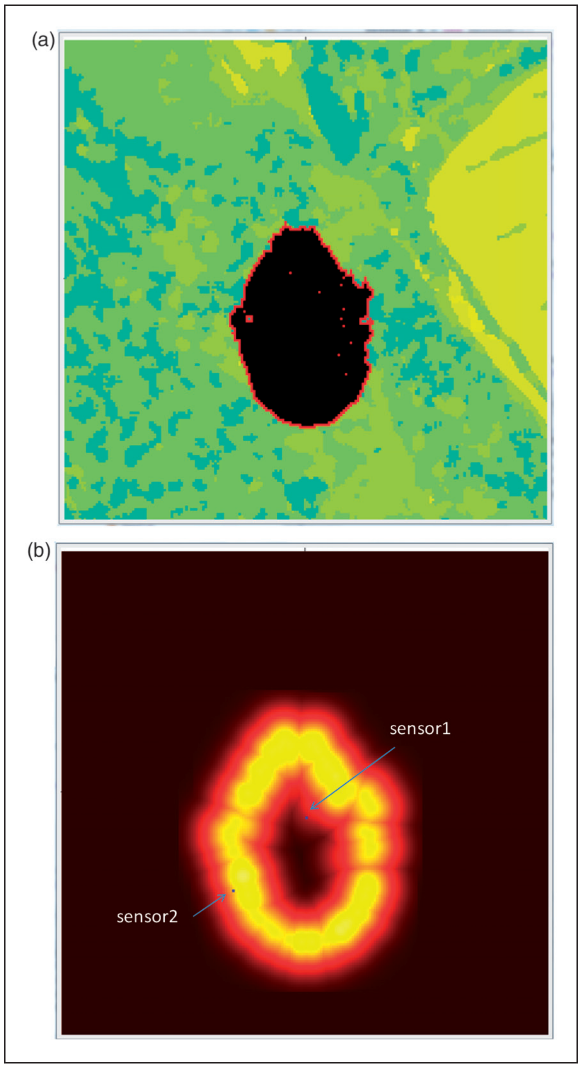

Figure 6. Wildfire spread simulation, temperature map, and sensor nodes. (a) A wildfire spread simualtion. (b) The temperature map and sensor nodes (color online only).

of the sensor nodes. Figure 7(a) show the actual temperature of the locations and the sensed temperatures from the two sensors when both sensors use a fixed quantum size $D = 10$. The actual temperature is calculated in the simulation based on a temperature model that takes into account the distance to the closest fire front and the diminish effect of temperature after the area is burned out. Figure 7(b) shows the same information when quantum size $D = 30$. The figures show that the two sensor nodes have different temperature curves. Sensor1's temperature starts to increase (at a rapid pace) at time 260 s. It reaches its peak temperature at time 580 s, and returns to the ambient temperature at time 1340 s. At time 960 s sensor1's temperature fluctuates due to the non-linear fire-spreading behavior caused by the non-uniform terrain and fuel model around the location of sensor1. Note that sensor1's temperature changes faster than sensor2's temperature, because it is located at the head of the fire where the fire spreads fast. Different from sensor1, sensor2's temperature rises at time 360 s at a much slower pace. It reaches its peak at time 1400 s, and returns to the ambient temperature at time 2860 s. The duration of the fire event (between temperature rises and returns) as sensed by sensor2 is 2500 s, which is much longer than that of sensor1 (1080 s). Figures 7(a) and (b) show that the quantization-based sensors are able to keep track of the curve of the actual temperature. The max error between the sensor data and actual temperature is constrained by the quantum size. The sensed temperatures are stepwise constants due to the discrete event-based updates of the sensor data. Comparing Figures 7(a) and (b), one can see that the smaller the quantum size, the more accurate the sensor data.

Figure 7(c) shows the energy consumption of the two sensors when using the two different quantum sizes. The two sensors have different rates of consuming the energy. In the beginning, sensor1 consumes more energy because it has more frequent temperate updates. Then sensor1 becomes inactive after the fire burns out. Sensor2 has a smaller rate of energy consumption in the beginning. However, in the end sensor2 consumes more energy than sensor1 consumes because it stays active for a longer duration. In this example, sensor1 deactivates at time 1340 s, while sensor2 deactivates at time 2860 s. For the same sensor node, as predicted, when the quantum size increases, the number of temperature updates decreases. As a result, the energy consumption becomes smaller. Below we show how the quantum size can be dynamically changed based on the energy budget.

4.2. Results of adjusted quantization policy

Our second experiment shows results of adjusted quantization policy. The experiment uses the same configurations as in Section 4.1. The only difference is that instead of maintaining a fixed quantum size, the two sensor nodes dynamically change their quantum sizes according to the adjusted quantization policy described in Section 3.2. In the experiment, the initial quantum size for both sensors is 10. We define an upper bound, 50, and a lower bound, 10, for the quantum size adjustment. We carried out the experiment using two different energy budgets: one is 35,000 and the other is 25,000. Figure 8(a) show the results of sensor data when the energy budget is 35,000; Figure 8(b) shows the results of sensor data when the energy budget is 25,000. Figure 8(c) shows the adjusted quantum sizes over

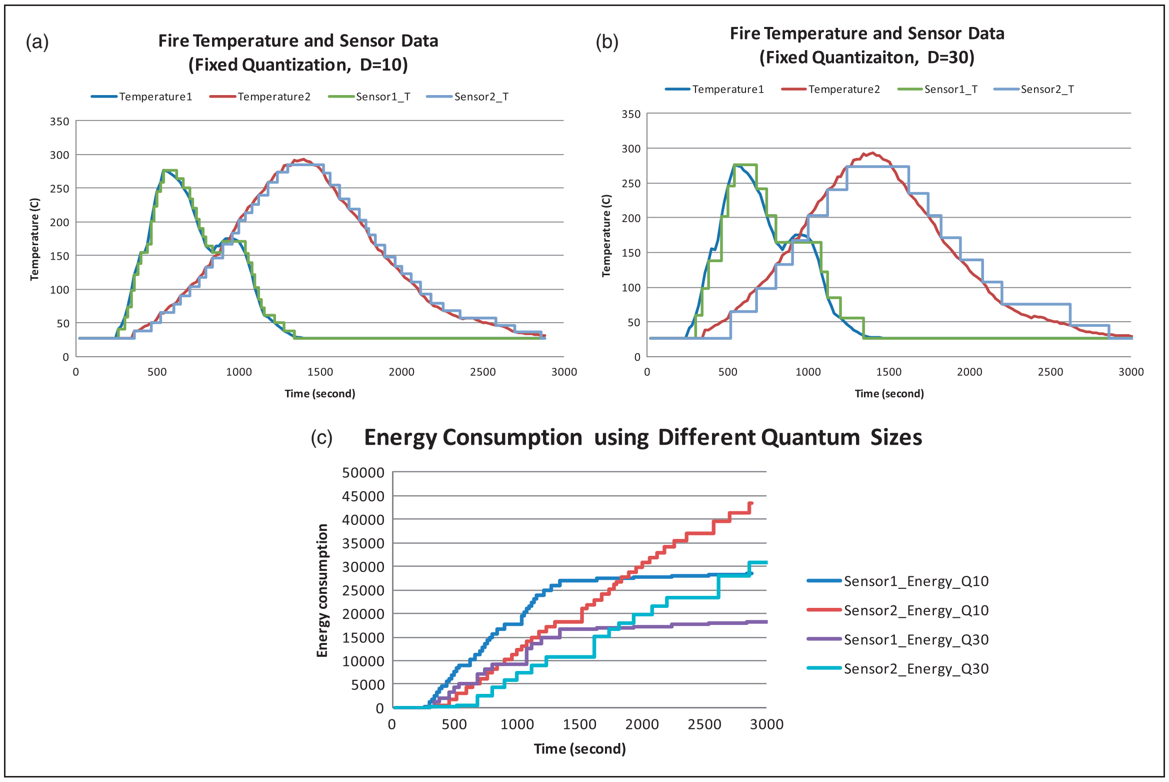

Figure 7. Temperature data and energy consumptions when using different quantum sizes. (a) Quantum size $D = 10$. (b) Quantum size $D = 30$. (c) Energy consumptions.

time for both sensor nodes under the two different energy budgets.

Figure 8 shows that the quantum sizes of the two sensors are not fixed any more – they increase dynamically based on the remaining energy budget. When the energy budget is 35,000, sensor1's quantum size is maintained at 10 (because there is enough energy for sensor1); sensor2's quantum size increases to 11, 12, 14, and finally to 48. When the energy budget is 25,000, sensor1's quantum size slightly increases to 12; sensor2's quantum size quickly increases and finally reaches the upper bound of 50. Figure 8(c) shows that the smaller the energy budget is, the more likely it is that the quantum size increases to a large value. This is expected because a large quantum size leads to less energy consumption and thus can better meet the energy budget. Figures 8(a) and (b) show that because of the adjusted quantum size the sensor nodes are able to monitor the fire event for longer (compared to using a fixed quantum size 10) under a given energy budget. In this example, when the energy budget is 35,000, sensor2 runs out of energy at time 2340 s. This is longer than 2260 s if a fixed quantum size 10 is used (see the energy-consumption figure in Section 4.1). When the energy budget is 25,000, sensor2 runs out of energy at time 1940 s. This compares to 1740 s if a fixed quantum size 10 is used. For both energy budgets, sensor1 was able to finish monitoring the whole fire event, which ends at time 1340 s. However, if a fixed quantum size 10 is used, sensor1 will run out of energy at time 1280 s.

This experiment demonstrates how the adjusted quantization policy works. It shows that the adjusted quantization policy is able to dynamically increase the quantum size of a sensor node based on the remaining energy budget and thus increases the life time of the sensor node. However, the adjusted quantization policy does not work in a very effective manner. This is because the policy does not take the *whileActiveRate* into account for adjusting the quantum size (see Section 3.2).

4.3. Results of adaptive quantization policy

Our third experiment shows results of adaptive quantization policy. The experiment setup is the same as in Section 4.2, with the only difference being changing the quantum size according to the adaptive quantization

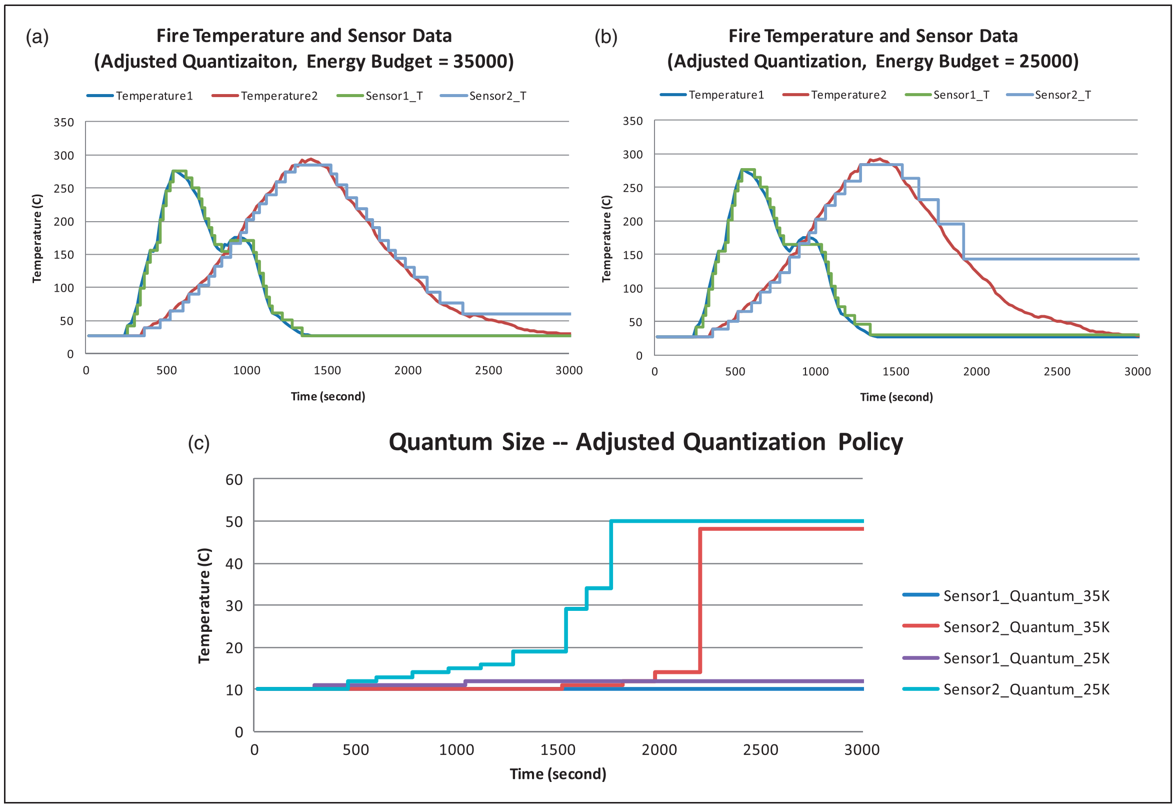

Figure 8. Temperature data and quantum sizes when using adjusted quantization policy. (a) Energy budget = 35,000. (b) Energy budget = 25,000. (c) Quantum size adjustment over time.

policy instead of the adjusted quantization policy. Figure 9(a) show the results of sensor data when the energy budget is 35,000; Figure 9(b) shows the results of sensor data when the energy budget is 25,000. Figure 9(c) shows the changing quantum sizes over time for both sensor nodes under the two different energy budgets.

From Figure 9(c) we can see that the quantum sizes of the two sensor nodes change dynamically too. However, different from the adjusted quantization policy that increases (not decreases) the quantum size only, the adaptive quantization policy increases and decreases the quantum size adaptively according to the remaining energy budget. In addition, the adaptive quantization policy gives better results than the adjusted quantization policy in terms of the sensor nodes' life time. For example, when the energy budget is 25,000, sensor2 runs out of energy at time 2340 s. This compares to 1940 s if using the adjusted quantization policy and 1740 s if using a fixed quantum size of 10. When the energy budget is 35,000, sensor2 is able to finish the whole fire event, which ends at time 2860 s. However, sensor2 runs out of energy at time 2340 s if using the adjusted quantization policy.

Figure 10 compares the energy consumptions of the adjusted quantization policy and the adaptive quantization policy. Figure 10(a) shows the energy consumptions of sensor1 and sensor2 for energy budget = 25,000. It can be seen that the adjusted quantization policy and the adaptive quantization policy result in different trajectories of energy consumptions. Compared to the adjusted quantization policy, the adaptive quantization policy gives a slower rate of energy consumption. This is especially obvious for sensor2 – sensor2 was able to monitor the fire event for a significantly longer time period when using adaptive quantization policy. Figure 10(b) shows the energy consumptions of sensor1 and sensor2 for energy budget = 35,000. Similar to in Figure 10(a), sensor2 has a slower energy-consumption rate when using adaptive quantization policy. The energy consumption of sensor1 follows about the same rate for the two quantization policies. This is because there is enough energy to monitor the whole fire event for sensor1, and thus both policies suggest using the smallest quantum size (quantum size = 10). Figure 10 clearly shows the advantage of the adaptive quantization policy over the adjusted quantization

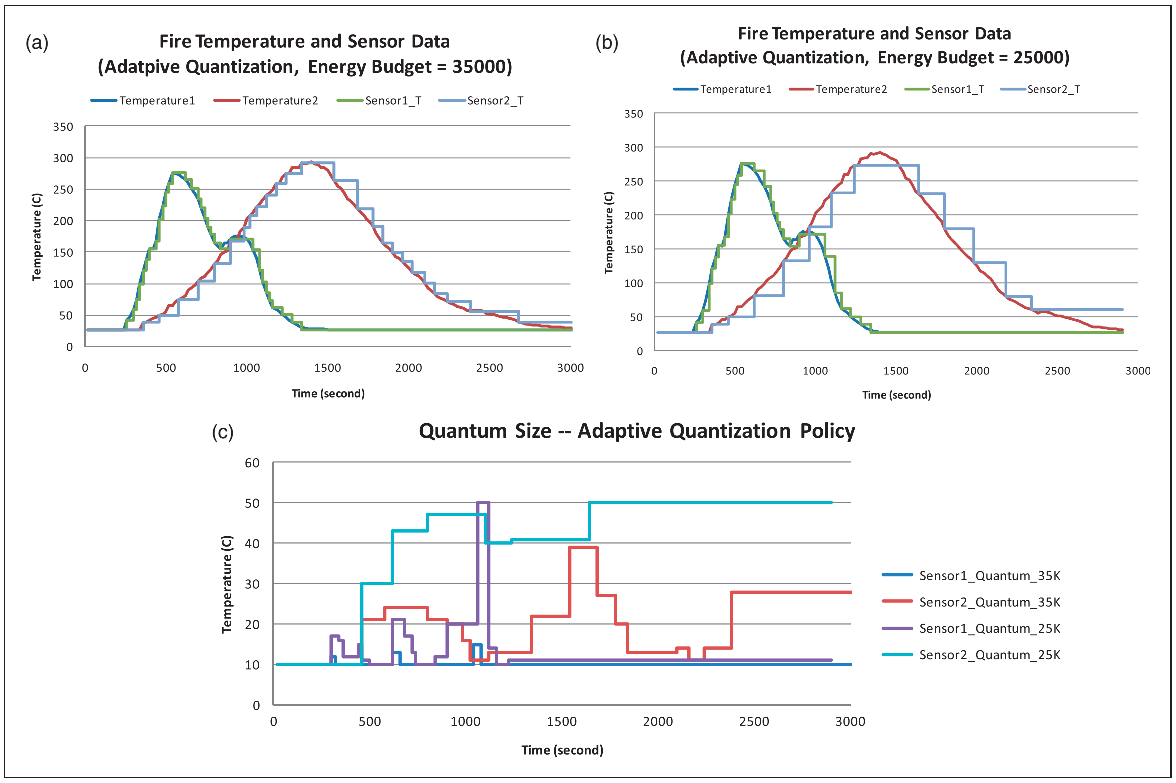

Figure 9. Temperature data and quantum sizes when using adaptive quantization policy. (a) Energy budget = 35,000. (b) Energy budget = 25,000. (c) Quantum size adjustment over time.

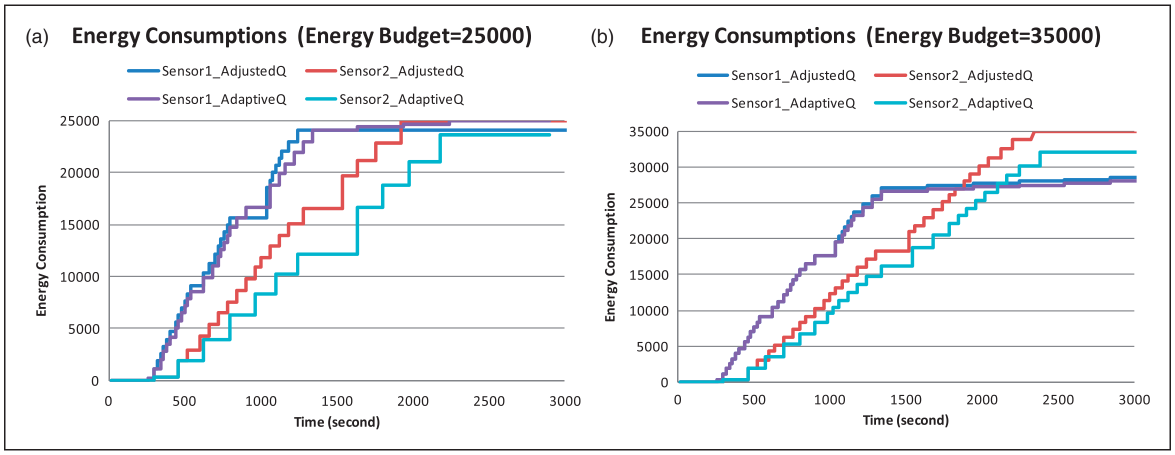

Figure 10. Energy consumptions of adjusted quantization policy and adaptive quantization policy.

policy in situations when there is a limited energy budget.

The adaptive quantization policy gives better results because it takes into account the *whileActiveRate*, as explained in Section 3.2, and dynamically estimates the remaining duration of the fire event. In this sense, the adaptive quantization policy is ‘smarter’ than the adjusted quantization policy. We note that an accurate estimation of the remaining duration of the fire event is important for the adaptive quantization policy to work well. In our current implementation, the controller of the adaptive quantization policy computes the temperature change rate from the most recent quantized input for calculating the remaining duration of the fire event. Better results could be reached if the temperature change rate is computed from the last several inputs instead of only the most recent input.

5. Conclusions

We presented an activity-based framework that links information and energy, and showed how this framework can be used to support energy-aware information processing. For sensing applications, the activity concept links the time course of a signal to the energy required by a quantizer to monitor it over a finite interval. An application to a wireless sensor node for detecting and monitoring wildfires was presented. Two policies of dynamically changing quantum size based on energy budget were studied. Results obtained from simulation experiments show that effective policies for setting the quantum size can be developed to provide energy-aware sensing using the activity-based framework developed in this paper. From the application point of view, the activity awareness sensing discussed in this paper can be expanded in two directions in future work. The first involves direct hardware implementation as in the DEVS System-on-a-Chip.[14] The other direction is the application of activity in learning and adapting to provide lower cost and power consumption, and increased life time for sensor nodes in deployments in stressful environments. More generally, the activity concept might prove to be a useful concept in the design of cyber-physical, and other information technology systems, that need to account for their use of energy, as well as their other requirements.

Funding

This research was supported in part by the National Science Foundation [CNS-0841170 and CNS-0941432].

Conflict of interest statement

None declared.

References

1. Zeigler BP, Jammalamadaka R and Akerkar SR. Continuity and change (activity) are fundamentally related in DEVS simulation of continuous systems. *Lect Notes Comput Sci* 2005; 3397/2005: 1–13.
2. Zeigler BP, Praehofer H and Kim TG. *Theory of modeling and simulation*, 2nd ed. New York, NY, USA: Academic Press, 2000.
3. Muzy R, Jammalamadaka BP and Zeigler J. Nutaro, the activity tracking paradigm in discrete-event modeling and simulation: the case of spatially continuous distributed systems. *Proceedings of SIMULATION 2010*, 2010.
4. Jammalamadaka R. Multilevel methodology for simulation of spatio-temporal systems with heterogeneous activity: application to spread of valley fever fungus. *PhD Dissertation*, Electrical and Computer Engineering Department, University of Arizona, 2008.
5. Sinha A and Chandrakasan A. Dynamic power management in wireless sensor networks. *IEEE Design Test* 2001; 18: 62–74.
6. He T, Krishnamurthy S, Stankovic JA, Abdelzaher T, Luo L, Stoleru R, et al. Energy-efficient surveillance system using wireless sensor networks. In: *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, Boston, MA, 6–9 June 2004.
7. Raghunathan V, Schurgers C, Park S and Srivastava MB. Energy-aware wireless sensor networks. *IEEE Signal Process Mag* 2002; 19: 40–50.
8. Antoine-Santoni T, Santucci JF, De Gentili E and Costa B. Discrete event modeling and simulation of wireless sensor network performance. *Simulation* 2008; 84: 103–122.
9. Antoine-Santoni T, Santucci JF, De Gentili E, Silvani X and Morandini F. Performance of a protected wireless sensor network in a fire. Analysis of fire spread and data transmission. *Sensors* 2009; 9: 5878–5893.
10. Van Wagner CE. Convection temperatures above low intensity forest fires. Ca*n For Serv Bi-mon Res Notes* 1975; 31: 21–26.
11. Van Wagner CE. Height of crown scorch in forest fires. Ca*n J For Res* 1973; 3: 373–378.
12. Hu X, Sun Y and Ntaimo L. DEVS-FIRE: design and application of formal discrete event wildfire spread and suppression models. *Simulation* 2010; [in press].
13. Ntaimo L, Hu X and Sun Y. DEVS-FIRE: towards an integrated simulation environment for surface wildfire spread and containment. *Simulation* 2008; 84: 137–155.
14. Hu X, Zeigler BP and Couretas J. DEVS-on–a-Chip: implementing DEVS in real-time Java on a tiny internet interface for scalable factory automation. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, October 2001.

Author biographies

Xiaolin Hu is an Associate Professor in the Computer Science Department at Georgia State University, Atlanta, Georgia. He received his PhD degree from the University of Arizona, MS degree from the Chinese Academy of Sciences, and BS degree from the Beijing Institute of Technology in 2004, 1999, and 1996, respectively. His research interests include modeling and simulation theory and application, agent and multi-agent systems, and complex systems science. He has served as program chairs for several international conferences/symposiums in the field of modeling and simulation, and is an associate editor for Simulation: Transaction of The Society for Modeling and Simulation International. Dr Hu is a National Science Foundation (NSF) CAREER Award recipient.

Bernard P Zeigler is Chief Scientist for RTSync, Emeritus Professor of Electrical and Computer Engineering at the University of Arizona and Research Professor in the C4I Center at George Mason University. He is internationally known for his seminal contributions in modeling and simulation theory. He has published several books including 'Theory of Modeling and Simulation' and 'Modeling & Simulation-based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange'.