Applying a Nested Petri Net Modeling Paradigm to Coordination of Sensor Networks with Mobile Agents

Lily Chang and Xudong He

School of Computing and Information Sciences, Florida International University
Miami, FL 33199, USA
Email: {lchan003, hex}@cis.fiu.edu

Juzheng Li and Sol M. Shatz

Department of Computer Science, University of Illinois at Chicago
Chicago, IL 60607, USA
Email: {jli48, shatz}@uic.edu

Abstract In this paper, we provide a formal definition of a framework of two layered nested predicate transition nets. We address the various technical issues in such a definition. We show the applicability of our approach in addressing different cooperation behaviors of mobile agents in modeling a two-tiered network consisting of a top-level mobile-device network and a lower level network of sensor nodes, and demonstrate the usefulness of formal modeling in revealing hidden and missing requirements.

1 Introduction

Nets-within-nets paradigm [10] provides a natural abstraction of many complex software systems such as multiple agent systems where agents are mobile. In recent years, several approaches based on this paradigm were proposed for modeling the mobility of mobile agents ([8], [2]). In [8], the communication between an agent net and the system net was synchronized through the concept of a channel defined by a fusion of two enabled transitions [1]. The firing of these transitions allows bi-directional information exchange between the agent and the agent system, and the movement of the agent. In [2], the synchronized communication between an agent net and the system net was through some input and output transitions, however no formal definitions were provided. In this paper, we propose a nested Petri net framework for modeling two-tiered networks. We provide a new definition of channels and extend predicate transition nets [4] with this channel concept. The new channel concept adapts the input and output commands in Hoare's process algebra CSP [7]. The communication between an agent and an agent system consists of synchronous control and unidirectional information flow, with which we can nicely model the essential characteristics of reactivity (input) and pro-activeness (out) of agents.

Unlike many existing nets-within-nets approaches, our new definition has the following features: (1) it supports the use of both data tokens and agent tokens in a higher level net (agent system net); (2) it provides a flexible and concise dynamic channel representation; and (3) it addresses the interplay between synchronized communication and timing constraints.

We demonstrate the use of our paradigm by modeling the coordination mechanism proposed in [9] to efficiently deal with the communications incurred in a multi-tiered network consisting of a lower level wireless sensor network and an overlay higher level network of mobile devices. Additionally, we show how a formal model can help reveal subtle behaviors of such a two-tiered network and help us to better understand interactions among various participants under normal and abnormal situations, which is necessary for a correct realization of such a system.

2 A Nested Petri Net Framework

To model various aspects of a mobile agent and an agent system, we need to not only capture the communications and movements, but also address data, functionality and other issues such as real-time requirements, which were not adequately dealt with in existing nets-within-nets approaches. We found that predicate transition nets (PrT net) [4] form a sound basis for our framework.

2.1 Predicate Transition Nets

We use the PrT net definition in [6]. A PrT net is a tuple (*N*, *Spec*, *ins*), where:

(1) *N= (P, T, F)* is a net structure. *P* and *T* are finite sets of places and transitions of *N*, where $P \cap T = \emptyset, P \cup T \neq \emptyset$ and $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs, which define the flow relation,

(2) *Spec* is an algebraic specification, which includes sorts, operators, and equations. Terms defined in *Spec* include tokens in *P*, labels on *F* and constraints associated with *T* ,

(3) *ins*= (φ, L, R, M_0) is an inscription that maps net elements to their denotations in the algebraic specification *Spec*. φ is a mapping from *P* to the set of sorts; *L* is a sort-respecting mapping from *F* to the set of labels; *R* is a mapping from *T* to the set of constraints; and M_0 is the initial marking – a mapping from *P* to the set of tokens.

The dynamic semantics of a PrT net can be defined as follows:

(1) A marking of a PrT net is a mapping from *P* to sorts defined in *Spec*;

(2) An occurrence mode of *N* is a substitution $\alpha = \{x_1 \leftarrow c_1, \ldots, x_n \leftarrow c_n\}$, which instantiates typed label variables. We use $e{:}\alpha$ to denote the result of instantiating an expression *e* with α, in which *e* can be either a label expression or a constraint;

(3) Given a marking M, a transition $t \in T$, and an occurrence mode α, t is α_enabled at M iff the following predicate is true: $\forall p$: $p \in P.(\bar{L}(p,t):\alpha) \subseteq M(p)) \wedge R(t):\alpha$; where

$$\bar{L}(x,y) = \begin{cases} L(x,y) & \text{if } (x,y) \in F \\ \varnothing & \text{otherwise} \end{cases}$$

(4) If t is α_enabled at M, t may fire in occurrence mode α. The firing of t with α returns the marking M' defined by $M'(p) = M(p) - \bar{L}(p,t):\alpha \cup \bar{L}(t,p):\alpha$ for $p \in P$. We use $M[t/\alpha>M'$ to denote the firing of t with occurrence α under marking M. As in traditional Petri nets, two enabled transitions may fire at the same time as long as they are not in conflict;

(5) For a marking M, the set $[M>$ of markings reachable from M is the smallest set of markings such that $M \in [M>$ and if $M' \in [M>$ and $M'[t/\alpha>M''$ then $M'' \in [M>$, for some $t \in T$ and occurrence mode α (note: concurrent transition firings do not produce additional new reachable markings);

(6) An execution sequence $M_0T_0M_1T_1\ldots$ of N is either finite when the last marking is terminal (no more enabled transition in the last marking) or infinite, in which each T_i is an execution step consisting of a set of non-conflict firing transitions;

(7) The behavior of N is the set of all execution sequences starting from the initial marking.

2.2 Modeling Time Concepts with Predicate Transition Nets

Many time/timed Petri net models have been proposed in the past three decades, and a discussion of those different models and their trade-offs can be found in [11]. It is well known that a high-level Petri net model can handle time concepts adequately by representing time information as an additional element of tokens and adding time constraints as additional conjuncts to transitions [5].

Here we introduce a special variable τ, and use it exclusively as part of a transition constraint. A time expression has the following general form $a \le \tau \le b$, where a and b indicate the lower and upper time bounds respectively. This time expression has the same meaning as a time interval $[a, b]$ associated with a transition in classical time Petri nets; such that an enabled transition t with time constraint $a \le \tau \le b$ is *fireable* within the relative time interval $[a, b]$ or the absolute time interval $[\theta + a, \theta + b]$. θ denotes the moment (absolute time) that transition t was enabled. Furthermore, we adopt the *strong fireability* rule, i.e., a fireable transition must fire by the time limit $\theta + b$. Transitions without timing constraints can be viewed to have fireable intervals of $[0, \infty]$.

To model timing concept, we need to modify the dynamic semantics of PrT nets. Since tokens now carry timing information, we do not explicitly add a time element into the definition in markings. Thus we only need to make the following changes to the definitions of dynamic semantics of PrT nets:

(3) Given a marking M, a transition $t \in T$, and an occurrence mode α, t is α_enabled at M iff the following predicate is true: $\forall p$: $p \in P.(\bar{L}(p,t):\alpha) \subseteq M(p)) \wedge R_u(t):\alpha$;

where $R(t) = R_u(t) \wedge R_\tau(t)$. $R_u(t)$ is a non-timing constraint, and $R_\tau(t) = a \le \tau \le b$ is a timing constraint. We do not explicitly write the time interval $[0, \infty]$ for non-timed transitions;

(4a) An enabled transition t with timing constraint $R_\tau(t) = a \le \tau \le b$ under marking M with occurrence α is *fireable* within time interval $[\theta + a, \theta + b]$, and must fire at $\theta + b$ if it is continually enabled;

(4b) The firing of fireable transition t under M with α returns the marking M' defined by $M'(p) = M(p) - \bar{L}(p,t):\alpha \cup \bar{L}(t,p):\alpha$ for $p \in P$. We use $M[t/\alpha\rangle M'$ to denote the firing of t with occurrence α under marking M. As in traditional Petri nets, two fireable transitions may fire at the same time as long as they are not in conflict.

2.3 Nets within Nets

There are several ways to use Petri nets to model a coordination mechanism between a two layered communication network; however a nested Petri net framework seems to be a natural and logical choice, which provides a nice abstraction of the system under study. Although we can define a general structure of deeply nested nets as in other works [8], here we elect to just define a two-level net structure that is adequate for our study and is much simpler.

The lower level nets are used to model the behaviors of individual mobile agents and thus PrT nets are appropriate. Since lower level nets serve as tokens of the higher level net, some care must be taken to ensure the higher level nets are well defined.

First, the higher level net has its own data abstraction and processing capabilities; thus we still need to have the full description power of PrT nets. Second, agents have their own behaviors that cannot be described by static data. As a result, we have to treat agent tokens as black boxes. Only their identities are visible and accessible in the higher level net. This treatment allows these agent tokens to be grounded and thus well-defined. This treatment also respects the autonomy characteristic of an agent. Third, the creation and removing of an agent token can be done through some boundary transitions without input and output places, respectively. We can leave the functionality (constraints) of these transitions open and view these as the responsibilities of an external environment. Fourth, we only model the logical mobility of an agent token and thus do not consider the implementation details with regard to whether to use value or reference in passing an agent's information from one location to another as discussed in [8].

To model synchronized communications between nets at different levels, we extend the constraint definition to include channel expressions. We borrow the input and output commands in CSP [7] for representing channel expressions. Thus a channel expression is $n!e$ (output) or $n?x$ (input), where n is a channel name, e is an expression, and x is a variable. Channel names are the identifications of agent nets and the identification of the system net. A synchronized communication occurs when two fireable transitions at two different net levels have a *matching pair* of input and output

channel expressions, i.e, a transition in the system net with identification *sys-id* contains *agent-id* ! *exp* (or *agent-id* ? *x*), and a fireable transition in an agent net with identifier *agent-id* contains *sys-id* ? *x* (or *sys-id* ! *exp*). To enforce well-definedness of communications, the channel names in agent nets must be constants; however, a channel name *n* in the system net can be a variable ranging over the agent identifications, which is instantiated with an enabling agent token identification. This allows great flexibility and concise representation of synchronized communications. During the synchronization, a unidirectional information flow occurs such that the value in *e* of the output command is assigned to the variable *x* of the input command.

We further revise the definitions of dynamic semantics of PrT nets to capture the synchronized communications as follows:

(3) Given a marking M, a transition $t \in T$, and an occurrence mode α, t is α-enabled at M iff the following predicate is true: $\forall p: p \in P.(\bar{L}(p,t):\alpha) \subseteq M(p)) \wedge R_u(t):\alpha$; where $R(t) = R_u(t) \wedge R_\tau(t) \wedge R_c(t)$. $R_u(t)$ is a non-timing constraint, $R_\tau(t) = a \leq \tau \leq b$ is a timing constraint, and $R_c(t) = n!e \mid n?x$ is a channel expression. We do not explicitly write the time interval $[0, \infty]$ for non-timed transitions;

(4a-1) An enabled transition t with a channel expression $R_c(t)$ is *ready* if a transition with a matching channel expression is also ready;

(4a-2) A ready transition t with timing constraint $R_\tau(t) = a \leq \tau \leq b$ under marking M with occurrence α is *fireable* within time interval $[\theta + a, \theta + b]$, and must fire at $\theta + b$ if it is continuously enabled.

Note: The effect of information flow during a synchronized communication has been reflected in the arc label expressions and thus no change to the definition of (4b) from Section 2.2 is needed. Of course, the value of input variable *x* can affect the new marking defined by the firing rule in (4b).

It is obvious that this modified semantics is only meaningful in the context of a net model that consists of two levels. The synchronized communications only happen vertically not horizontally (i.e., no direct communication between two agent nets.) With the input and output commands, we can naturally define the reactivity and pro-activeness of an agent.

3 A Modeling Example

3.1 Wireless Sensor Network with Mobile Devices

Mobile devices are introduced into a wireless sensor network system (WSN) to take advantage of the mobile devices' storage, communication ability and computation power [3]. In order to gather and share sensor data, mobile devices have to effectively cooperate with each other. To help develop such a system, we use the nested net framework defined in the previous section to describe the behavior of a coordination mechanism for a WSN with mobile devices [9]. A key idea of the coordination is that

mobile devices share common interest (sensor) data within the same group or among different groups.

The coordination mechanism is a form of publish and subscribe and provides a paradigm for organizing multiple mobile devices through an interest-management server, in which groups of mobile devices are formed based on the location of their interested information. Mobile devices that issue queries for sensor data are subscribers. The server manages subscribers' information but does not directly publish sensor data; that task is actually done by mobile devices using a multicast mechanism that sends the shared data to a group address given by the server. The sensor nodes in WSN are divided into clusters according to their geographic locations and the information of clusters is kept in the server. Each cluster, identified by a unique cluster id, has a cluster head and the sensor data within a cluster is periodically sent to the cluster head, which serves as a communication point with entities outside the cluster. We have identified the following entities involved in communication behaviors shown in Fig. 1. The arrows represent the message flows.

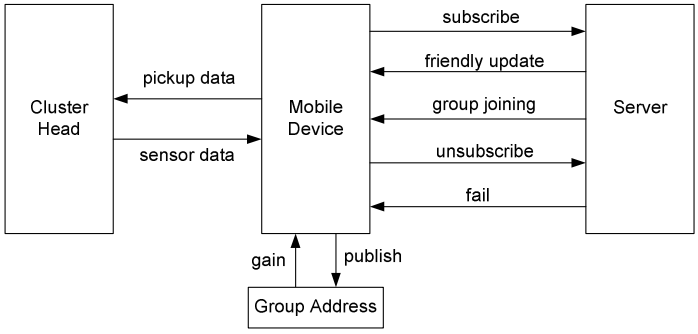

Fig. 1. Message flows between entities.

Based on the communication relations of these entities, the mobile devices are modeled as tokens of a server net since there are multiple mobile devices that communicate with the server. Mobile devices as tokens are active elements in the server net, i.e., a mobile device has its own behavior of issuing a query (subscribe), pickup sensor data, publish shared data, gain shared data and unsubscribe when a result for the query has returned. Thus a mobile device is modeled by a mobile device net, which communicates with a server net through channels. In [9], there were no explicit descriptions of behaviors of sensor nodes, and the interaction between mobile devices and a cluster head was not precisely specified. Thus, we mainly focus on the interaction behaviors between the mobile devices and the server, and define the server net and the mobile device nets. We discuss the behavior models separately in the following sections.

3.2 The Interest-Management Server Model

In response to a query request from a mobile device, the server performs two actions. First, it checks whether there exists some common interest group with interest in the current location of the mobile device. If there is an existing common interest group in the region of current location of the requesting mobile device, a message containing the group address is sent back to the mobile device to instruct the mobile device to perform 'friendly update' by picking up the sensor data from the relevant cluster head and multicasting the data according to the group address received from the server to share sensor data. Second, followed by the friendly update, the requesting mobile device is added to the identified common interest group(s) that have the same query region as the subscribing mobile device. In the case that no existing cluster region covers the query region, a failed message is sent to mobile device. Mobile devices can unsubscribe after getting the result; therefore, when an unsubscribe message is received, the server removes the information of unsubscribing mobile device. Table 1 shows the relevant actions of the server with regards to the algorithm in [9].

Table 1. Conditions of the server's actions.

Actions	Pre-conditions	Post-conditions
Subscribe	Mobile device sends a query	Query received
Instruct friendly update	Mobile device's current location exist common interest sharing group.	Sends a message to mobile device to instruct a friendly update at its current location.
Group Joining	Mobile device's query region is covered at least by one cluster	Adds the mobile device to the common interest sharing group (s) with the same query region and sends a joining message to mobile device.
Fail	None of the cluster regions covered the query region.	Sends a fail message to mobile device.
Unsubscribe	Received the unsubscribe message from a mobile device	Removes the information of the unsubscribing mobile device

We have used the following conventions: (1) our model does not restrict the multiple subscriptions of the same group for the same mobile device; (2) we assume that the information required for multicasting is kept in the server, and the information is accessible during multicasting; (3) there is no action for a mobile device to temporarily join a group to perform a friendly update; (4) we use the original query content as the unsubscribing message and assume that the group address described in the requirement is a unique and fixed value like cluster id.

Fig. 2 presents the server's behavior model according to information and control flows identified in Table 1.

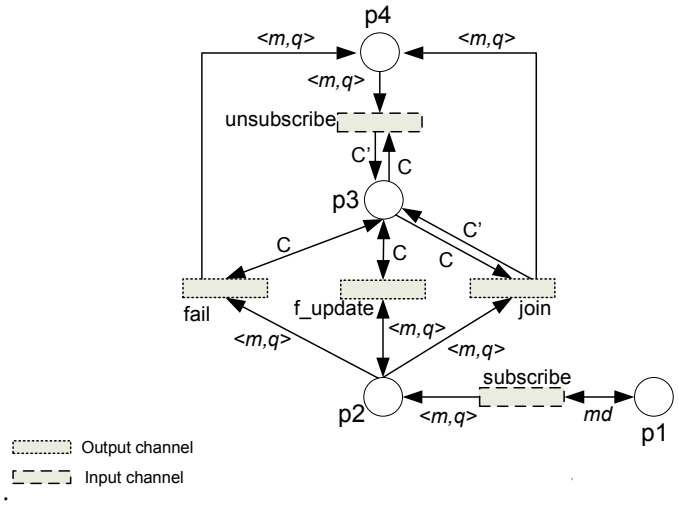

Fig. 2. Server's Behavior Model.

Semantic Definitions of Server's PrT Model

Token Types:

$QUERY = QueryID \times MobileID \times CurrentLocation \times Query\_region \times TimeToLive$

$CLUSTER = ClusterID \times Cluster\_region \times GroupAddress \times \wp(MobileID)$

$\varphi(p1) = \wp(MobileNet)$

$\varphi(p2) = \varphi(p4) = \wp(MobileNet \times QUERY)$

$\varphi(p3) = \wp(CLUSTER)$

// *QUERY* and *CLUSTER* are Cartesian product of predefined data types that define the contents of a query and subscribed information about common interest group respectively. *Query_region* and *Cluster_region* are of the same type that can be used to represent a set of coordinates that define the cover region of a query and cluster region respectively. The *MobileNet* is the net token defined by a mobile device net.

Transition Constraints:

$R(subscribe) = n?q \wedge m = md$

$R(f\_update) = \exists c \in C.(q[3] \in c[2]] \wedge c[4] \neq \varnothing) \wedge q[2]!(c[2], c[3], q)$

$R(join) = \exists c \in C.(c[2] \cap q[4] \neq \varnothing) \wedge \forall c \in C.(c[2] \cap q[4] \neq \varnothing \wedge c'[4] = c[4] \cup q[2]) \wedge q[2]!(c[2], c[3], q)$

$R(fail) = \forall c \in C.(c[2] \cap q[4] = \varnothing) \wedge q[2]!'subscription\_failed'$

$R(unsubscribe) = n?q \wedge \forall c \in C.(c[2] \cap q[4] \neq \varnothing \wedge c'[4] = c[4] - q[2])$

// Transition *subscribe* is fireable when a matched output channel *n* in other net is ready. Transition *f_update* is enabled if mobile device *q*[2]'s(id) current location *q*[3] is in cluster region *c*[2] and a non empty set *c*[4] of mobile devices subscribed to cluster region *c*[2]. After firing, the friendly update message is sent through channel. Transition *join* add mobile device id *q*[2] to the set of subscribed mobile devices *c*[4]. If query region *q*[4] is not covered by any cluster region *c*[2] in *C*, transition *fa*il will fire and fail message is sent through channel *q*[2]. Transition *unsubscribe* removes the active query token and its subscribed information.

3.3 The Mobile Device Model

A mobile device communicates with the user, server, other mobile devices and cluster heads. First, a user may generate a query through the interface on the mobile device when he/she needs information. Then, the mobile device sends a query request message to the server for subscription. Various actions may be taken according to the responding message from the server. If a friendly-update instruction message is received, the mobile device has to pick up the sensor data from the cluster head at the region of its current location, and publishes the data to a group address received in the friendly-update message for sharing. Upon receiving the group-joining instruction message, by the expiration of Time-To-Live time frame, the mobile device may receive the data through other mobile devices. In other case, if the mobile device traveled to the same region with its query region before receiving the interested data through group sharing performed by other mobile devices, the mobile device will pick up the sensor data itself and share the data with the members in the same group. If no such group sharing event happens when the time of a valid query is going to expire, i.e., no one in the same group ever reaches the query region or no friendly update is performed, the mobile device will perform the 'direct injection', which is a way of getting the sensor data from the query region through sensor network routing. After getting the data and publishing the data, the mobile device will unsubscribe from the server. The data received is properly extracted and displayed on the interface of the mobile device for the user. Table 2 shows the possible actions of a mobile device.

Table 2. Conditions of a mobile device's actions.

Actions	Pre-conditions	Post-conditions
input query	The user of the mobile device needs information	A query is generated containing query id, mobile id, current location, query region and Time-To-Live(valid time)
Subscribe	A query message is ready to send to the server.	Waits for the reply from the server.
friendly update	The server instructs a friendly update	Get the friendly update message and ready to pickup data from cluster head in the current region
join	Server has sent a group-joining message	(1) received sharing data from group members, or (2) picks up the sensor data directly by itself (3) performs direct injection to acquire the data and publishes
Fail	Server has sent a failed message	Outputs the error message to the user
Gain	Cluster data is published by group members	Got the data and ready to unsubscribe
Publish	Sensor data has been picked up.	Published the data and ready to unsubscribe

Direct pickup	Current location is inside the query region	Interact directly with cluster head to acquire sensor data
Direct injection	Current location is not in the query region and the valid time for the query is about to expire and no sharing data is available	Request sensor data from query region cluster head remotely through sensor network routing
Unsubscribe	Received the data and sends an unsubscribe message to the server	Outputs the data to the user

We use two pairs of channels to represent the communication behaviors between these entities. Based on the actions listed in Table 2, we developed the behavior model of a mobile device shown in Fig. 3. Fig. 4(a) and 4(b) provide the group address multicast sharing data behavior model and cluster head's behavior model. It is assumed that a mobile device knows its location by GPS or other localization techniques. Place *p8* in Fig. 3 keeps the mobile device's id and its current location. Place *p2* in Fig. 4(a) refers to the same database as place *p3* in Fig. 2, which is assumed to be accessible during multicasting. In our model, the content of the responding message from the server contains the query information.

Semantic Definitions of Mobile Device's PrT Model

Token Types:

$QUERY = QueryID \times MobileID \times CurrentLocation \times Query\_region \times TimeToLive$

$\varphi(p1) = QueryID \times Query\_region \times TimeToLive$

$\varphi(p2) = \wp(QUERY)$

$\varphi(p3) = \wp(Cluster\_region \times GroupAddress \times QUERY)$

$\varphi(p4) = \wp(GroupAddress \times QUERY)$

$\varphi(p5) = \wp(DATA \times GroupAddress)$

$\varphi(p6) = \wp(QUERY \times DATA)$

$\varphi(p7) = \wp(DATA)$

$\varphi(p8) = MobileID \times CurrentLocation$

// *QUERY*, *Cluster_region* and *Query_region* are of the same type as defined in Server's behavior model. *QueryID*, *MobileID*, *CurrentLocation*, *TimeToLive*, *GroupAddress* and *DATA* are predefined data types.

Transition Constraints:

$R(subscribe) = q[1] = r[1] \wedge q[2] = cl[1] \wedge q[3] = cl[2] \wedge q[4] = r[2] \wedge q[5] = r[3] \wedge S!q$

$R(f\_update) = S?(cr, ga, q)$

$R(fail) = S?d$

$R(join) = S?(cr, ga, q)$

$R(gain) = ga?d$

$R(publish) = s[2] = ga \wedge ga!s$

$R(direct\_pickup) = cl[2] \in cr \wedge cr!(q[2], ga)$

$R(direct\_injection) = cl[2] \notin cr \wedge \tau = q[5] - \varepsilon \wedge cr!(q[2], ga)$

$R(unsubscribe) = S!q$

$R(sensor\_data\_in) = cr?d$

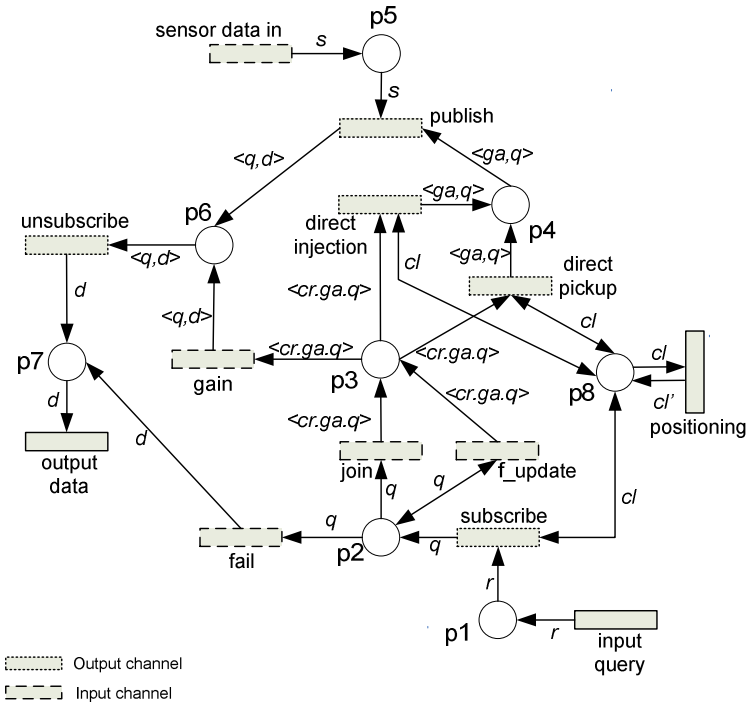

Fig. 3. Mobile Device's Behavior Model

// ε is a predefined and small constant value, which is determined by the designer to enforce the firing of the transition. Transition *input query* and *output data* are boundary transitions for user interface. Transition *positioning* is the boundary transition that has the function of obtaining the current location (coordinate) of mobile device itself.Transition *subscribe* is enabled whenever there is token *r* available from the input place *p1* and a matched input channel *S* in server net is ready. Transition *f_update*, *join* and *fail* are unconditionally ready for fire whenever output channel *S* in the server is ready to output messages. Transition *gain* is unconditionally ready for fire whenever output channel *ga* from group address multicast is ready to broadcast messages. Transition *publish* is ready for fire when sensor data *s* is received through transition *sensor data in* through input channel *cr*. Transition *direct pickup* is enabled if current location *cl*[2] is in query region *cr*. After firing, a message is sent through output channel *cr* to cluster head to request for data.Transition *direct injection* is enabled if current location *cl*[2] is not in the query region and the *time to live q*[5] is going to expired within ε. Transition *unsubscribe* is enabled if query *q* is finished and

data *d* is available in *p6*. After firing, the data *d* is output to *p7* and query *q* is sent through output channel *S* to server to be removed from subscribed information.

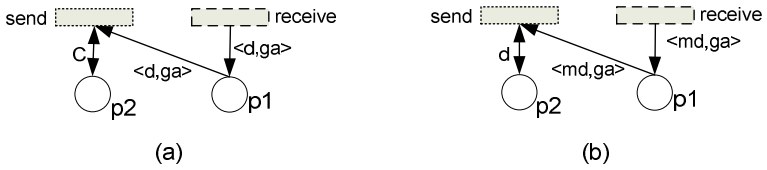

Fig. 4. (a) Group address multicast (b) Cluster head data communication

Semantic Definitions of Group Address Multicast's PrT Model

Token Types:

$CLUSTER = ClusterID \times Cluster\_region \times GroupAddress \times \wp(MobileID)$

$\varphi(p1) = DATA \times GroupAddress$

$\varphi(p2) = \wp(CLUSTER)$

Transition Constraints:

$R(receive) = md?(d, ga)$

$R(send) = \forall c \in C.(c[3] = ga \wedge \forall md \in c[4].(md!d))$

Semantic Definitions of Cluster head Data Communication's PrT Model

Token Types:

$MD = \wp(MobileID)$

$\varphi(p1) = MD \times GroupAddress$

$\varphi(p2) = DATA$

Transition Constraints:

$R(receive) = md?(md, ga)$

$R(send) = md!(d, ga)$

3.4 A Query Request Scenario

Let us look at a simple scenario based on the behavior models defined in the previous sections. A campus WSN system with information-management server keeps the information of three cluster regions: library, cafeteria and gymnasium. The students on campus are usually interested in the conditions of these locations (for example, available space). There are four students, John, Mary, Peter and Eric with their own mobile devices, and each of them is at different locations on campus. John and Mary are currently at the library; Peter is at the cafeteria and Eric is at the gym. Eric finished his workout and is heading for the cafeteria but wondering if the cafeteria is too crowded or not. So Eric sends a query request through his mobile device to subscribe for information about the cafeteria. In the meantime, John and Mary finished their study at the library and are also heading to the cafeteria for lunch; they

have sent query requests and subscribed for the information about the cafeteria. Peter has finished his lunch at the cafeteria and is heading for the gym having subscribed for information for the gym. Therefore, two different common interest groups have been formed based on the interested regions: John and Mary for the cafeteria, and Peter for the gym. We use Eric's query request scenario as an example to demonstrate the behaviors. The current marking M_0 of the server for the above scenario is:
M_0 (p1)={John, Mary, Peter, Eric}
M_0 (p2)= ∅
M_0 (p3)={(1,library,1, ∅), (2,cafeteria,2,{md1,md2}), (3,gym,3,{md3})}
M_0(p4)={(John,q1,md1,library,cafeteria,10), (Mary,q1,md2,library,cafeteria,10), (Peter,q1,md3,cafeteria,gym,10)}

For simplicity, we only demonstrate Eric's mobile device net, which has interactions with the server net, group address multicast and the cluster head. Initially, only place *p8* has token, which stores the mobile id and current location of the mobile device. After Eric sent a query request through the mobile device's interface, a query request message containing the information of query id, mobile device id, current location, query region and a maximum waiting time for the query is generated. A token *q* = {*q1, md4, gym, cafeteria, 10*} is generated. Since the server is ready to input message through transition '*subscribe*', together with the transitions '*subscribe*' in the mobile device net, a synchronized communication occurs as a result of firing a pair of matching transitions. After firing the matching transitions, the server received the query request. A token {*Eric, q1, md4, gym, cafeteria*, 10} is output to *p2* in both nets, resulting a state that is ready to join a common interest group. This query simply says that the mobile device *md4* is interested in the information of region *cafeteria*; and its location is at *gym*; and this query is valid only within 10 seconds after the query is sent.
Next, the server checks if there exists a common interest group in mobile device *md4*'s current location, *gym*. Refers to place *p3*, that holds the current information of subscriptions and regions in the server net (Fig. 2), there is a common interest group indicating cluster id '3', which is region *gym*, linking a group address '3' and mobile device id '*md3*'. The server then responds with the friendly update message containing {*gym, 3*} through output channel '*md4*', which is the communication channel to Eric's mobile device. The mobile device's input channel '*f_update*' simultaneously receives that message. Eric's mobile device then requests the sensor data through the communication channel '*direct pickup*' with the cluster head of the current situated region. As soon as it gets the response through '*sensor data in*' from the cluster head, the data is published to group address '3'. This is done through transition '*publish*' in Fig. 3. The shared data is sent through multicast, and mobile device '*md3*', which is Peter, will receive that data. After instructing a friendly update, the server adds Eric's mobile device to the group of region '*cafeteria*' through the simultaneous firing of transition '*join*' in Fig. 2 and 3, Eric's mobile device receives the group joining message containing {*cafeteria, 2*}, indicating that he has joined in the group address 2 in region cafeteria. After Eric joined the group, Mary happens to arrive at the cafeteria and gets the information of the cafeteria. She publishes the information for sharing to group address '2' and all the mobile devices in the same group, John and Eric, received the information of the cafeteria through input channel '*gain*'. After Eric

receives the shared data, an unsubscribe message is sent through transition '*unsubscribe*' with output channel *S* to server. Server receives the message and removes Eric's subscription information. The firing sequence of mobile device net regards to Eric's query request is:
M_0 [*input query*> M_1 [*subscribe*> M_2 [*f_update*> M_3 [*direct pickup*> M_4 [*sensor data in*> M_5 [*publish*> M_6 [*join*> M_7 [*gain*> M_8 [*unsubscribe*> M_9 [output data>

The firing sequence of the server net regards to Eric's query request is:
M_0 [*subscribe*> M_1 [*f_update*}> M_2 [*join*> M_3 [*unsubscribe*>

The firing sequence of the group address multicasting net and the cluster head net regards to Eric's query request is:
M_0 [*receive*> M_1 [*send*>

The corresponding pairs of communication channels are listed in Table 3.

Table 3. Communication channels

Server	Mobile Device	Group Multicast	Cluster head
Subscribe(in)	Subscribe(out)		
F_update(out)	F_update(in)		
Join(out)	Join(in)		
Fail(out)	Fail(in)		
Unsubscribe(in)	Unsubscribe(out)		
	Publish(out)	Receive(in)	
	Gain(in)	Send(out)	
	Direct pickup(out)		Receive(in)
	Sensor data in(in)		Send(out)
	Direct injection(out)		Receive(in)

Let Eric's request query firing sequence be the sequence called *ReqSeq* and the transitions fired in these nets are represented in the order [server net, mobile device net, group address multicast net, cluster head net]. We use λ to represent no transition firing in a net. The concurrently firing sequence *ReqSeq* of these nets regards to Eric's request query is:
ReqSeq = [λ , *input query*, λ , λ], [*subscribe*, *subscribe*, λ , λ],
[*f_update* ,*f_update*, λ , λ], [λ , *direct pickup*, λ , *receive*], [λ , *sensor data in*, λ , *send*], [λ , *publish*, *receive*, λ], [*join*, *join*, λ , λ], [λ , *gain*, *send*, λ],
[*unsubscribe*, *unsubscribe*, λ , λ], [λ ,*output data*, λ , λ]

4 Concluding Remarks

We applied the nested net paradigm to model the coordination processes in a two-tiered network. During this study, we examined the applicability of this paradigm in modeling various aspects of an agent – autonomy, reactivity, pro-activeness, sociability, and mobility; as well as an agent system. We further demonstrated the

usefulness of formal modeling that forces us to explicitly deal with all hidden assumptions and detect missing requirements. To implement such a nested net paradigm, many subtle research issues exist such as the interplay between synchronized communications and timing constraints, and the manipulations of data tokens and behavior tokens (nets). These issues need to be solved in order to develop a suitable simulation tool to facilitate the automated system analysis. Our future research work includes how to extend one-to-one communications to one-to-many (broadcasting) communications, and how to generalize the two layered nested net framework to deeply nested net framework.

Acknowledgements:

We thank the anonymous reviewers for their comments that help improve the presentation of the paper. Lily Chang and Xudong He's research was partially supported by NSF grants HRD-0317692 and IIP-0534428. Juzheng Li and Sol M. Shatz's research was partially supported by U.S. Army Research Office under grant number W911NF-05-1-0573.

References

1. Christensen, S., Hansen, N.: "Colored Petri nets extended with channels for synchronous communication", *Proc. of International Conf. on Application and Theory of Petri Nets*, (1994), 159-178.
2. Ding, J., Clarke, P., Xu, D., He, X., Deng, Y.: A formal model-based approach for developing an interoperable mobile agent system, *Multiagent and Gird Systems – An International Journal*, vol. 2, no.4 (2006), 401-412.
3. Ekici, E., Gu, Y., Bozdag, D.: Mobility-based communication in wireless sensor networks. *IEEE Communications* Magazine, vol.44, no.7 (2006), 56-62.
4. Genrich, H.J.: Predicate/transition nets. In: *Advances in Petri Nets*. (1986) 207–247.
5. Ghezzi, C., Madrioli, D., Morasca, S., Pezze, M., A unified high level Petri net formalism for time critical systems, *IEEE Transactions on Software Engineering*, Vol. 17, No. 2,(1991) 160-172.
6. He, X., Deng, Y.: A framework for developing and analyzing software architecture specifications in SAM. *Computer J*. 45 (2002) 111–128.
7. Hoare, C.: Communicating Sequential Processes. Prentice Hall (1985).
8. Kohler, M., Moldt, D., Rolke. H.: Modeling mobility and mobile agents using nets within nets", *Proc. of International Conf. on Application and Theory of Petri Nets*, *LNCS* vol. 2679 (2003), 121-139.
9. Li, J., Shatz, Sol M.,: A Coordination Mechanism for Mobile Devices to Gather and Share Common-Interest Sensor Data. Submitted for publication (2008).
10. Valk, R.: Petri nets as token objects: An introduction to elementary object nets. *Proc. of International Conference on Application and Theory of Petri Nets* (1998), *LNCS*, vol. 1420, 1–25.
11. Xu, D., He, X. & Deng, Y.: Compositional schedulability analysis of real-time systems using time Petri nets. *IEEE Trans. On Software Engineering*, *28*(10), (2002), 984-996.