

# ASM-BASED FORMAL MODEL FOR A NETWORK-LAYER GRID, THE ISEGRID

LakshmiPriya TKS, Ranjani Parthasarathi  
Dept. of Computer Science and Engg., College of Engg., Guindy,  
Anna University, Chennai, India  
tkslp@cs.annauniv.edu, rp@annauniv.edu

## Abstract

Recent advancements in network device technologies, along with the introduction of application-awareness in the network, are driving researchers to exploit the network for providing enterprise solutions. One such approach is to employ a grid, consisting of network entities to harness the idle compute power in the network and offer transparent network services to end-users. In this paper we present a formal framework, based on Abstract State Machines, for specifying and defining such a network layer grid. The ASM model serves to define the operations of the individual grid components and their interactions. The ASM model is presented with architecture-level and operational-level refinements of the individual grid components. Suggestions for algorithmic-level refinements are provided along with a list of uses for the ASM model.

## 1. Introduction

The proliferation of powerful networking devices such as the multi-core network processors [1,2] and the introduction of network-related concepts such as edge computing [3] and application-aware networking (AAN) [4,5] have paved way for trusting more computation/intelligence into the network. Networks today can handle multiple networking technologies and traffic flows (using Diffserv, Intserv [6], and SLA) with a satisfactory performance level. The reprogrammable nature of the networking devices has facilitated the accommodation of multiple, dynamic services at these nodes. Concepts such as edge computing have brought the computation to the edge of the network, thereby enabling personalization. AAN concepts have brought about the movement of application-level processing into the network thereby facilitating early-decision making. These advancements in the networking arena have resulted in a significant improvement in the network performance. To take this one step further, we foresee a need for the network devices to operate in a cooperative manner with a global knowledge of the network. Such an environment would enhance the application-awareness in the network.

Our approach to this is to employ a grid at the network layer of the ISO protocol stack, which harnesses the idle/under-utilized ‘networking-processing’ power that offers the networking-processing operations as a commodity to its users. Further this grid enables dynamic deployment of code and service components at its grid nodes by adopting the active network technology. We have presented the architecture, components and mode of operation of this network-layer grid and illustrated a few of its services in an earlier work [7]. We have evaluated the use of network processors (NPs) to act as candidates for playing the role of a network-layer grid node. Towards this we have tested the suitability of NPs to operate as active nodes [8] and evaluated NPs for application-awareness [9]. In this paper, we present an ASM based formal framework for our network-layer grid, which we call the iSEGrid. The purpose of the ASM model is to provide a formal specification and a clear demarcation of the minimal features required for a system.

ASMs [10] have been used in system design primarily as a formal specification methodology for a variety of systems such as the PVM architecture [11, 12], communication protocols [13,14], etc. They have also been used to compare distributed systems and conventional computational grids [15]. The ASM method has the advantage of being able to solve the language and communication problem (being algorithmic in nature), the verification problem (by inspection and by reasoning) and the validation problem (ASMs can be simulated) [16, 17].

The rest of the paper is organized as follows. Section 2 gives the background required to understand the iSEGrid. In Section 3 we present the ASM model for the iSEGrid. The refinements for the iSEGrid

ASM are discussed in Section 4 along with a list of uses for this ASM model. We conclude the paper in Section 5 giving pointers to future work.

## 2. Background – The iSEGrid

The iSE in the name iSEGrid, refers to the **in-network Service-aware Entities**, simply put, they are the network intermediaries. As the name suggests, the iSEGrid is a grid of iSEs. In this section we present an overview of the service architecture, the components and the mode of operation of the iSEGrid.

### 2.1. iSEGrid Service Architecture

The services offered by the grid form a four-layer service architecture, substituting the network layer in a typical layered network architecture. The four layers of the architecture are Basic Network-processing (BNp) layer, Local Decision-making (LDm) layer, Aggregate Decision-making (ADm) layer, and iSEGrid services layer. Of the four layers, the lower two layers, namely the BNp and the LDm layers perform the normal network processing functions. The BNp services are those that involve packets, headers, and tables; that are typical of any network node. The LDm services include those that consolidate the BNp services. Application-aware local policies and decision-making tasks are implemented at this layer using techniques such as Deep Packet Inspection (DPI). The other layers namely, the ADm and the iSEGrid services layers, are the grid extensions to the network layer. The ADm services include those that aggregate the LDm services, resource brokering services and other management and control services. The iSEGrid services are those that are offered to the users.

The iSEGrid consists of five major functional components (Fig. 1): iSEGrid service providers (iSEs), iSEGrid Resource Brokers (iSE\_RBs), iSEGrid Portal, iSEGrid Active-Code Repositories (iSE\_ACR) and the iSEGrid Directories (iSE\_Dir).

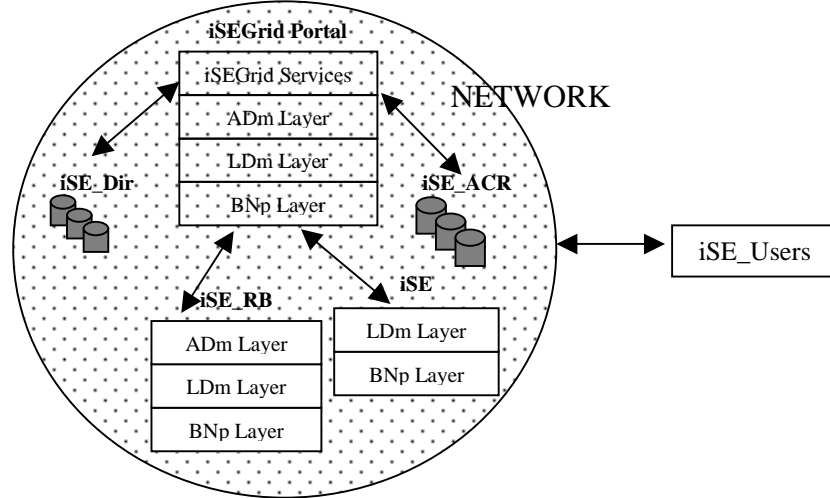


Fig. 1. iSEGrid Environment and the components with their service stack

The architecture components and the service stack at each of the grid components are shown in Fig. 1. The details of each iSEGrid components follow:

(i) iSEs: The nodes on the network edge and the core that can volunteer resources, may play the role of iSEs. The iSEs perform the services of the BNp and LDm layers. These services are controlled and monitored by the iSE\_RBs.

(ii) iSE\_RBs: The more powerful iSEs play the role of iSE\_RBs on the grid and perform the ADm services. The iSE\_RBs detect the need for iSEGrid services by intercepting the traffic and triggering the corresponding LDm services. Such implicit triggering of the services on the iSEGrid makes the grid and its operations transparent to the users. The iSE\_RBs are also responsible for aggregating the LDm services that run at several cooperating iSEs.

The iSEs and the iSE\_RBs are the nodes that offer services on the iSEGrid while the iSE\_ACRs, iSE\_Dirs and iSEGrid Portal are support nodes.

(iii) iSEGrid Portal: The entry point into the grid is a publicly accessible portal, which advertises the grid services. The portal negotiates and registers users for iSEGrid services. These services are partitioned into many Adm services, for each of which an appropriate iSE\_RB is chosen.

(iv) iSE\_ACRs: The nodes that can volunteer storage such as the storage servers play the role of iSE\_ACRs. The iSEGrid services are developed as active software components (Active Networking Technology [18, 19]) and are stored at iSE\_ACRs. The idea of having active code repositories on the grid is to enable movement of code and services in and out of the iSEs dynamically. The software components may be deployed at the grid nodes either during registration or on-demand, using techniques such as active networking or agent-based technology.

(v) iSE\_Dirs: The iSE\_Dirs are also storage volunteers located at various strategic points in the network that hold the data and metadata of the grid nodes and their services.

## 2.2. iSEGrid operation

The iSEGrid operates under two phases namely the iSEGrid setup phase in which registration and initial code deployment take place; and iSEGrid in-service phase, in which the iSEGrid services are offered. The events that occur during these phases are shown in Figure 2.

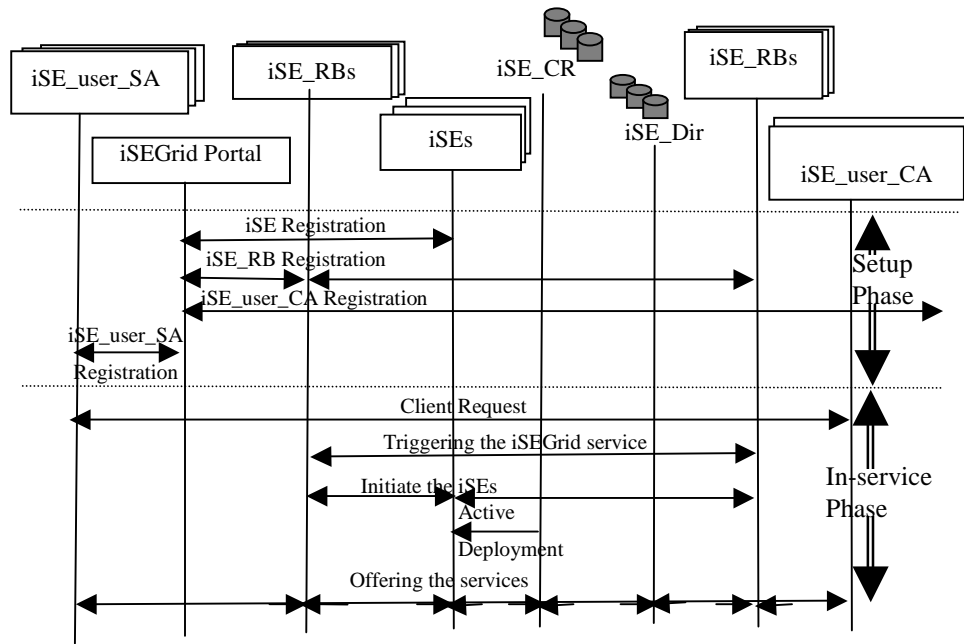


Fig. 2. Two-phase Operation of the iSEGrid

During the iSEGrid setup phase, network intermediaries register as grid nodes by approaching the iSEGrid Portal, while end-systems register as iSE\_users (Client Applications are termed iSE\_user\_CAs while the Server Applications are termed iSE\_user\_SAs). The registration process involves resource negotiation, initial active code deployment, user\_RB pairing and service partitioning tasks. Upon receiving the registration request, the iSEGrid Portal negotiates with the new node for resources and services. Once negotiation is successful, the iSEGrid Portal initiates the initial Active Code deployment operation during which appropriate code modules from iSE\_ACRs are deployed at the grid nodes. When iSE\_users register, pairing of RBs and users takes place. During the RB-user pairing operation, a suitable iSE\_RB, preferably at the user edge is chosen and the requested iSEGrid services are partitioned into appropriate Adm services to be run at the iSE\_RB. This iSE\_RB becomes the first point-of-access for those users. All further communications between the iSE\_user and the grid take place via this iSE\_RB. As a final stage of the setup phase, the iSE\_Dir is updated and initialization procedures are executed at the new grid node.

Soon after registration, the iSEGrid nodes enter the ‘in-service phase’ during which conventional client/server requests from the client systems, trigger or kick start the iSEGrid services, either explicitly or implicitly. ‘Explicit triggering’ occurs when the iSE module at the iSE\_user, explicitly requests the iSE\_RB for an iSEGrid service. On the other hand, implicit triggering occurs when the iSE\_RB intercepts the flow (using DPI mechanisms) at the user-edge and detects the need for an iSEGrid service. ‘Triggering’ an iSEGrid service, involves intelligent decision-making for identifying and initiating the iSE(s). Once identified, the iSEs are initiated and the parameters for the service including the location of the iSE\_ACRs are sent to them. After the on-demand code deployment, the iSEs begin offering the service. The iSE\_RBs periodically monitor the iSEs for the triggered services. When a service terminates, wind-up operations are done at the iSEs, and the iSE\_Dirs are updated to reflect this change.

### 2.3 The significance of the iSEGrid

To highlight how the iSEGrid helps in a typical networking application, let us consider a client group that makes regular requests to video servers for video files, and caches the viewed files for future use. During a flash crowd situation at the video server, the server may not be in a position to serve subsequent requests and hence many clients are deprived of service during the flash crowd period. However, at the client group under consideration, the cached copies may be shared among the clients, thereby minimizing such adverse conditions. Although such a setup offers improved system performance, it requires a lot of commitments from the client side, in terms of cache maintenance, detection of a flash crowd at the server, locating a local copy and redirecting the requests.

The iSEGrid solution to this is as follows: a node at the client edge, typically an edge router or a gateway (Client Edge Router - CER), is employed as the iSE\_RB for the clients. Similarly, an edge node on the server side (Server Edge Router - SER) is employed as the iSE\_RB for servers. Each client in the client group (CG) contributes to the collective cache that is maintained by the CER. The SER is responsible for monitoring the load on the video servers (S), detecting a flash crowd situation and communicating this information to the CER. The CER is also responsible for locating a cached copy and redirecting the request during a flash crowd situation. The datapath for this service is shown in Figure 3.

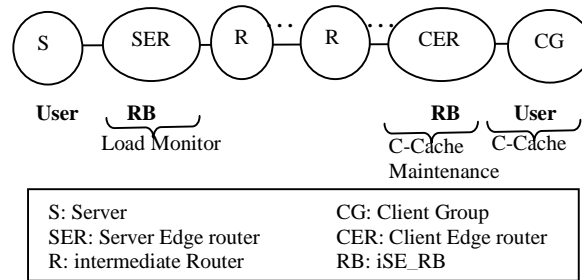


Fig. 3. Datapath for iSEGrid-based flash crowd control service

The two iSE\_RBs, located right on the datapath, are well positioned to intercept the traffic to/from the end nodes. They adopt the DPI mechanisms to inspect the packet flowing through them and detect the need for iSEGrid services. Further, based on the client requirements, they may employ in-the-net iSE nodes (R) to offer services in the network such as link-caching, media transcoding, prefetching files and virus scanning. The iSE\_RBs, with the global view of the network, control and coordinate the operation of these services.

The presence of the iSEGrid enables the CG to enjoy value-added networking services (in this case overcoming adverse conditions during flash crowds), in a transparent manner. Further, the data delivery to the CG can be improved by enabling in-the-net services at the appropriate iSEs as mentioned above.

It is evident from the above illustration that the iSEGrid relieves the burden on the end-systems and offers the service in a transparent manner with the help of a global view of the network and coordination between the in-the-net nodes.

### 3. ASM Model

ASMs represent a mathematical well-founded framework for system analysis and design. ASM, originally called Evolving Algebra [10], is a formal specification method for software and hardware systems. System modeling using ASMs involves identifying and defining the system's signature and state transitions. In ASMs, states are represented as modified logician's structures, i.e., basic sets (universes or domains) with functions and relations interpreted on them. The signature of an ASM consists of the number of the universes; the number of the operations with their arity, domains, and ranges; and integrity constraints. The objects of the physical reality are modeled as elements of universes and the relationship between the universes are modeled as functions.

### 3.1 Notations used

The notations used in this paper for universes and functions are as follows:

(i) Universes (Domains) are defined in one of the following ways.

Domain <domainName> or

Domain <domainName>= {<element1>, <element2>, ...< elementN> }, or

Domain <domainName>= {<domainName1>  $\cup$  <domainName2>  $\cup$  ...  $\cup$  <domainNameN>}

(ii) Functions of arity 'n' are denoted as follows:

<functName>: <domainName1> x <domainName2> x ...x <domainNameN>  $\rightarrow$  <domainName>

The iSEGrid is modeled as a multi-agent ASM with each iSE component forming an ASM agent. In this section we first give a list of the universes that belong to the iSEGrid ASM, followed by a list of the various messages used for communication between the various components and the list of the tables used by each component. We then provide the common functions used in the iSEGrid ASM and then describe the ASM rules for each of the iSEGrid components.

### 3.2 Universes belonging to the iSEGrid ASM

The five distinct types of components on the iSEGrid along with its users make up the Domain COMPONENT defined as Domain COMPONENT ={isePortal, iseRB, ise, iseACR, iseDIR, iseUser}.

Each component resides on network nodes represented as Domain NODE. These components interact with each other by exchange of messages (Domain MESSAGE). Each message has a name (in this work it is given the suffix MSG, eg., intimateRBUserMSG) and the message is defined by specifying its fieldnames and values (Domain FIELD, Domain FIELD\_VAL) as follows:

```
<msgName> =
  (
    <field1> = values or description,
    <field2> = values or description,
    ...
    <fieldn> = values or description
  )
```

Each component maintains data, metadata of its local resources (Domain RESOURCE), and the services it offers (Domain SERVICE) along with other information necessary for its operation, in data/information tables (Domain TABLE). Each information table has a name (with suffix Table, eg., iseComponentTable) and is defined by specifying its attributes and their values (Domain ATTRIB and Domain ATTRIB\_VAL).

```
<tableName> =
  (
    <attrib1> = value or description,
    <attrib2> = value or description,
    ...
    <attribN> = value or description
  )
```

Resources (Domain RESOURCE) are identified by their specifications and values for each specification (Domain SPEC, Domain SPEC\_VAL). The services (Domain SERVICE) offered by

each component belong to one of the four layers of the iSEGrid service architecture i.e., Domain SERVICE = ISEGRID\_SER  $\cup$  ADM\_SER  $\cup$  LDM\_SER  $\cup$  BNP\_SER. Services are distinguished by service type (Domain SER\_TYPE = {implicit, explicit}).

The software code for the iSEGrid components and for the iSEGrid services are represented by (Domain CODE) and codes are identified by code type (Domain CODETYPE).

### 3.3 Messages – The interaction between the components

The valid messages used for the intercommunication between the various iSE components are specified by the functions messageIn and messageOut. The first parameter in these functions, represent the component under consideration and the second parameter represents the component on the other end.

(i) Valid incoming messages at a component: messageIn: COMPONENT x COMPONENT  $\rightarrow$  MESSAGE

(ii) Valid outgoing messages at a component: messageOut: COMPONENT x COMPONENT  $\rightarrow$  MESSAGE

#### 3.3.1 Inter-communication between iSEGrid Portal and other iSE nodes

The messages to/from the iSEPortal are defined below.

The messages that an iSEPortal may receive from the nodes ise, iseRB, iseACR, iseDIR or iseUser are reqForRegMSG and negotiateAckMSG.

```
messageIn:(isePortal, {ise | iseRB | iseACR | iseDIR | iseUser} ) =
    {reqForRegMSG, negotiateAckMSG}
```

The outgoing messages common to all iSE nodes are:

```
messageOut:(isePortal, {ise | iseRB | iseACR | iseDIR | iseUser} ) =
    {negotiateReqMSG, regAckMSG}
```

The outgoing message specific to the iSE\_RB is:

```
messageOut:(isePortal, iseRB) = intimateRBUserMSG
```

The outgoing message specific to the iSE\_ACR and ise\_Dir is:

```
messageOut:(isePortal, {iseACR | iseDir}) = deployIseCompMSG
```

The description of each of these messages is provided below.

##### 1. Request for Registration Message

```
reqForRegMSG =
(
    msgType = reqForReqMSG,           // Message Type
    reqstType = iseRB|iseACR|iseDIR|iseUser, // type of registration
    resourceSpec                       //spec of the volunteered resources
)
)
```

##### 2. Negotiation ACK Message

```
negotiateAckMSG =
(
    msgType = negotiateAckMSG,
    reqstType = iseRB|iseACR|iseDIR|iseUser,
    resourceSpec //spec of the ACKed volunteered resources
)
)
```

##### 3. Negotiation Request Message - negotiateReqMSG // similar to negotiateAckMSG

##### 4. Registration Ack Message - regAckMSG // similar to reqForRegMSG

##### 5. Intimation message for RB-user pairing operation

```
intimateRBUserMSG =
(
    msgType = intimateRBUserMSG,
    updateType = updateAll, // fields to be updated in the tables
    serviceSpecList, //List of service spec.s
    impTrigType = timeBased | eventBased, //Type of implicit triggering
    impTrigParam = (startTime, endTime) |(startEvent, endEvent) |
        duration, //parameters for implicit triggering
    impTrigServPriority = 0 | 1 | ...//Priority for implicit triggering
)
)
```

##### 6. iSE Code deployment Message

```
deployIseCompMSG =
```

```

(
    msgType = deployIseCompMSG,
    codeType = iseACR|iseDIR|iseRB|ise|iseUser, //type of code
    serviceSpecList, //List of service spec.s for initial deployment
    nodeID //ID of the new node
)

```

### 3.3.2 Inter-communication between iSE\_RB and other iSE nodes

The messages to the iSE\_RB are defined below.

```

messageIn:(iseRB, isePortal) =
    {negotiateReqMSG, regAckMSG, intimerBUserMSG}
messageIn:(iseRB, iseUser)={ controlImplicitTrigMSG, controlServiceMSG }
messageIn:(iseRB, ise) = updateIseNodesDetailsMSG

```

The messages from the iSE\_RB to other nodes are defined below.

```

messageOut:(iseRB, ise) = { getIseNodesDetailsMSG, configureMSG,
    controlLDMSERVICEMSG }
messageOut:(iseRB, isePortal) = {reqForRegMSG, negotiateAckMSG }
messageOut:(iseRB, iseACR) = deployCodeMSG

```

The description of each of these messages is provided below.

1. Intimation message for RB-user pairing - IntimerBUserMSG // already defined
2. Control Message for implicit triggering services
 

```

controlImplicitTrigMSG =
(
    msgType = controlImplicitTrigMSG,
    compType = iseUser, // user for whom the services are offered
    controlType = start|stop|resume|pause|enable|disable,
    impTrigType = timeBased | eventBased | durationBased,
    impTrigParam = (startTime,endTime)|(startEvent,endEvent)|duration,
    impTrigServList = serviceList,
    serviceParamList //variable list of parameters to start this service
)

```
3. Service state control message - controlServiceMSG // similar to controlImplicitTrigMSG
4. Update table message with results of probing iSE nodes - updateIseNodesDetailsMSG  
Fields of this message are the probed parameters
5. Request message for the results of probing at the iSE nodes - getIseNodesDetailsMSG  
Fields of this message are the parameters for probing
6. Configure iSE Message - configureMSG  
Fields of this message are the parameters for configuring the iSE.
 

```

configureMSG =
(
    msgType = configureMSG,
    deploy = true | false,
    codeType, // Type of code to be deployed
    service //service for which the code is to be deployed
)

```
7. Control Message for LDm services- controlLDMSERVICEMSG // similar to controlImplicitTrigMSG
8. Deploy code on-demand Message – deployCodeMSG  
Fields of this message indicate the code for a service to be deployed on an iSEGrid node
 

```

deployCodeMSG =
(
    msgType = deployCodeMSG,
    codeType = component | service, // Type of code to be deployed
    componentType = ise | iseRB | iseACR | iseDIR | iseUser,
    serviceType = ADM_SER | LDM_SER | BNP_SER,
    service //service for which the code is to be deployed
)

```

### 3.3.3 Inter-communication between iSEs and other iSEGrid nodes

The messages to/from the iSEs are defined below.

```

messageIn:(ise, isePortal) = {negotiateReqMSG, regAckMSG}
messageIn:(ise, iseRB) = {getIseNodesDetailsMSG, configureMSG,
                           controllDMSserviceMSG}
messageOut:(ise, iseRB) = updateIseNodesDetailsMSG
messageOut:(ise, isePortal) = {reqForRegMSG, negotiateAckMSG}

```

The description of these messages has already been provided.

The inter-communication between iSE\_ACRs/iSE\_DIRS and other iSE nodes includes messages that request for code/information from the iSE\_ACRs/iSE\_DIRS and their corresponding responses. Due to space constraints they have not been included. However they will be introduced at the point of usage.

### 3.4 Tables - Information maintained at the iSEGrid components

The data tables maintained at the iSE\_Portal, iSE\_RB, and iSE are defined below.

```

iSE_PORTAL_TABLES = iseComponentTable //holds the details of each component on the iSEGrid
iseComponentTable =
(
    nodeType = ise | iseACR | iseDIR | iseRB,
    nodeID = IP, //or similar unique ID
    regTime, //Time of registration
    serviceSpecList, //list of service spec.s volunteered
    resourceSpecList, //list of resources spec. volunteered
    availDetails, //its availability details
    accounting, //its accounting spec.
    authentication, //its authentication spec.
    security //its security spec.
)

iSE_RB_TABLES = {
    userServicesTable, // service details of the services of the users paired with this RB
    iseNodesTable, //details of the user nodes paired to the RB & iSEs known to the RB
    impTrigServiceTable, // details of services to be triggered implicitly
    expTrigServiceTable } // details of services to be triggered explicitly

iSE_TABLES = {
    LDMServicesTable, // LDM service details
    LocalResTable //details of local resources
}

```

The iSE\_ACRs hold the `codeTABLE` that contains the details of the code modules and service modules available at the iSE\_ACR. The tables necessary for information maintenance (data and metadata) are available at the iSE\_Dirs.

### 3.5 Common functions defined in the iSEGrid ASM

- The dynamic nullary functions defined in the ASM for iSEGrid are: (i) `self`: allows a component to identify itself among other components, (ii) `any`: represents any arbitrary element of a domain. The static nullary functions (constants) defined in the ASM for iSEGrid are: `true`, `false`, `null`.
- `registered`: COMPONENT x NODE → {true, false}, denotes the registration of a network node as an iSEGrid component is complete.
- The functions defined on messages are:
  - Extracting the value of a field from a message: `get`: MESSAGE x FIELD → FIELD\_VAL
  - Framing a message with values for its fields: `put`: MESSAGE x FIELD x FIELD\_VAL → MESSAGE
- The functions defined on tables are:

- Insert into table the details present in the message. `insert: TABLE x MESSAGE`
- Select from table: `select: TABLE x ATTRIB_LIST → ATTRIB_VAL`, where `ATTRIB_LIST` is a list of table attributes.
- Update the table with details available in the fields of the message. `update: TABLE x MESSAGE x ATTRIB_LIST`

### 3.6 ASM rules for each iSEGrid component

The common ASM rules and the ASM rules for each of the five iSEGrid components and the iSEUser, namely `isePortal`, `iseRB`, `ise`, `iseACR`, `iseDIR`, and `iseUser` are given below.

#### 3.6.1 Common ASM rules

The ASM rules for the ‘send a message’ and ‘receive a message’ operations used in the ASMs for the various iSE components are `To` and `From` respectively. They are defined as follows:

- `To(m:MESSAGE, c:COMPONENT)` This rule sends the message ‘m’ to the iSE component ‘c’.
- `From(sc:COMPONENT)` This rule receives a message from the sender iSE component ‘sc’. The rule evaluates the receiver iSE component and the received message is available as ‘`msgRcvd`’.

The rules that represent the operations that control the iSEGrid Services are: `Start`, `Stop`, `Pause`, `Resume`, `Enable` and `Disable`. Eg. rule `Start(s:SERVICE)` starts the service ‘s’.

#### 3.6.2 ASM rules for iSE\_Portal

The iSEPortal component is represented by an ASM that consists of the following FIVE rules: `RECEIVE`, `NEGOTIATE_1`, `NEGOTIATE_2`, `INITIAL_DEPLOY`, and `RB_USER_PAIRING`. The sequence of rule execution during the registration of the iSEs, iSE\_RBs, iSE\_ACRs and iSE\_Dirs, is as follows: `RECEIVE`, `NEGOTIATE_1`, `RECEIVE`, `NEGOTIATE_2`, and `INITIAL_DEPLOY`. In the case of iSEUser registration, the rule `RB_USER_PAIRING` is fired in parallel with `INITIAL_DEPLOY`. `iseParam` denotes the type of iSE component that is currently under consideration and its value obtained from the `reqrType` field in the incoming message.

```
Rule ISEPORTAL =
SEQ
    RECEIVE
    NEGOTIATE_1
    NEGOTIATE_2
    If iseParam = iseUser Then // if the registration is from the iseUser
    SEQ
        INITIAL_DEPLOY
        RB_USER_PAIRING
    ENDSEQ
    Else // else if the registration is for any other ise component
        INITIAL_DEPLOY
ENDSEQ
```

1. The `RECEIVE` rule involves the receipt of a message (Rule `From`) from an arbitrary component (any). If the message is destined to the portal component itself (`self`), the component performs two checks on the received message (`msgRcvd`). First it checks if the message is one among the valid messages that it is expected to receive (`messageIn(self,any)`) and next it checks the genuineness of the message (`IsGenuineMsg(msgRcvd)`). By genuineness we refer to the authentication process. The `RECEIVE` rule sets the variable `secureMsg` (a dynamic function associated with that message) to `true` to indicate successful scanning of the message. The ID of the source node that sent `msgRcvd` is assigned to `otherEndNode` using `getSource` function.

```
Rule RECEIVE =
If From(any) == self Then //Receive a msg for self
    If msgRcvd ∈ messageIn(self,any) and IsGenuineMsg(msgRcvd)
        // is msg from expected sources and is msg authenticated
    Then
        secureMsg(msgRcvd) := true
```

```
otherEndNode := getSource(msgRcvd)
```

This RECEIVE rule is common to all the iSEGrid components and is also used within other ASM Rules that expect a message from another node.

2. The NEGOTIATE\_1 rule deals with initiating a 'negotiate' operation with the new node (otherEndNode - available as sourceID in the incoming message) by sending the negotiateReqMSG to it. This is done by comparing the resource specifications (offeredResSpec) mentioned in the reqForRegMSG and the expected resource specifications (expectedResSpec) for this type of iSE component, as per the iSEGrid policy. expectedResSpec is obtained by evaluating the rule ResourceSpecPolicy. The comparison is done by evaluating the Compare rule. If the Compare rule evaluates to false then a negotiateReqMSG is framed (put function) and sent to the otherEndNode (Rule To).

```
Rule NEGOTIATE_1 =
If msgRcvd.msgType == reqForRegMSG AND secureMsg(msgRcvd) == true Then
LET
  expectedResSpec := ResourceSpecPolicy(otherEndNode) AND
    offeredResSpec := get(msgRcvd, resourceSpec) IN

  agree := Compare(offeredResSpec, expectedResSpec)
  If agree == false Then
  SEQ
    negotiateReqMSG := put(negotiateReqMSG, resourceSpec, expectedResSpec)
    To(negotiateReqMSG, otherEndNode)
  ENDSEQ
  secureMsg(msgRcvd) := false // resetting the flag for reqForRegMSG
```

3. The NEGOTIATE\_2 rule is fired upon receiving the negotiateAckMSG message. The profile of the new node (available in negotiateAckMSG message) is inserted into the table holding the resource details of various iSEGrid components (IseComponentTable).

```
Rule NEGOTIATE_2 =
  If msgRcvd.msgType == negotiateAckMSG AND secureMsg(msgRcvd) == true then
    insert(iseComponentTable, negotiateAckMSG)
    secureMsg(msgRcvd) := false
```

4. The INITIAL\_DEPLOY rule uses the rule DeployCode to initiate the initial code deployment operation by sending the deployIseCompMSG to the iSE\_ACR. This rule takes as arguments a CODETYPE (iseCompType indicates iSE component type) and a COMPONENT (otherEndComp) to inform the iSE\_ACR that the code of iseCompType is to be deployed at the iSE component of type otherEndComp. This rule provides abstraction for the deployment mechanism i.e., either active networking approach or agent-based approach may be adopted. The rule also involves sending an acknowledgment for registration to the new node (regAckMSG). The registered predicate is set to true to indicate successful registration of otherEndNode as otherEndComp.

```
Rule INITIAL_DEPLOY =
  DeployCode (iseCompType, otherEndComp)
  To(regAckMSG, otherEndComp)
  registered(otherEndComp, otherEndNode) := true
```

5. The RB\_USER\_PAIRING rule performs the RB\_user pairing operation. The Rule SelectRB evaluates to an appropriate RB for the given component. The abstraction in this rule offers flexibility in the feature/characteristics matching algorithm used to select an RB. Upon successful selection of an RB, the iSEPortal partitions the iSEGrid services into many Aggregate Decision-making services (PartitionServices (COMPONENT: iseRB, COMPONENT, SERVICE: ADM\_SER)). The PartitionServices rule performs the partitioning task into many Adm services (newServices) for a given user (otherEndComp) and each of these Adm services will be carried out at the iseRB

(pairedRB). Notification of the formation of the RB\_user pair is sent to the RB and to the component communicating on the other end (the 2 Intimate rules).

```

Rule RB_USER_PAIRING =
  LET pairedRB := SelectRB(otherEndComp) IN
  If pairedRB NOT null Then
    SEQ
      newServices := select(iseComponentTable, otherEndComp, serviceSpecList)
      PartitionServices(pairedRB, otherEndComp, newServices)
      Intimate(pairedRB, otherEndComp)
      Intimate(otherEndComp, pairedRB)
    ENDSEQ

```

### 3.6.3 ASM rules for iSE\_RB

The ASM for the iSE\_RB consists of two main rules, corresponding to the background process (Rule ISERB\_BkGdProcess) and the regular process (Rule ISERB), running at each iSE\_RB.

#### a) The background process at the iSE\_RB

The Rule ISERB\_BkGdProcess is involved in the management and control operations that occur periodically and is modeled using the three rules IMPLICIT\_TRIGG, MONITOR\_ISE\_STATUS, and MONITOR\_RB\_RES\_STATUS.

```

Rule ISERB_BkGdProcess =
  IMPLICIT_TRIGG // monitor the packet flow and trigger implicitly
  MONITOR_ISE_STATUS // monitor iSE status
  MONITOR_RB_RES_STATUS // monitor local resource status

```

1. The IMPLICIT\_TRIGG rule performs the implicit triggering operation, which is a key task of the iSEGrid. The rule MonitorInFlow checks each packet flow to detect the need for an iSEGrid service. For this it uses the fields of the impTrigServiceTable table, which holds the services enabled for implicit triggering and the corresponding user. Upon encountering such an event, the iSE\_RB sets the predicate foundFlow to true and status of that service to detected, indicating the need for an implicit triggering event for the detected service. For each detected service 's', the iSE\_RB identifies one or more ADm services (RULE PartitionIntoADm) and starts them (Start). Further, the iSE\_RB selects one or more appropriate iSEs (RULE SelectISE), partitions each 'admService' into one or more LDm services (RULE PartitionIntoLDm), configures the chosen iSEs for these LDm services (RULE Configure), initiates the on-demand code deployment at the selected iSE (RULE DeployCode) and starts the services at the iSEs (RULE StartService).

The rule PartitionIntoADm identifies the ADm services (admService) corresponding to the detected iSEGrid service 's'. These ADm services are performed at the iSE\_RB. The rule PartitionIntoLDm, on the other hand determines the appropriate LDm services (ldmService) to be performed at the iSEs. The partitioning algorithms used in this rule must take into account the node-specific information about the iSEs and hence has to refer to the iseNodesTable table.

The rule SelectISE may contact the iSE\_Portal or the iSE\_DIR for details of various iSEs or refer to its local iseNodesTable table. The selection algorithm used by this rule must choose the iSEs along the natural datapath between the end-users.

The rule Configure initiates the initialization and configuration tasks at the iSEs using the message configureMSG and prepares the iSEs for the ldmService. The iSE now waits for code from the iSE\_ACR. The rule DeployCode uses the message deployCodeMSG to inform the iSE\_ACR to initiate the on-demand code deployment. The rule StartService uses the message controllDmServiceMSG to start the ldmService service at the activeIse.

```

Rule IMPLICIT_TRIGG =
  MonitorInFlow
  If foundFlow == true Then
    Forall s ∈ SERVICE and SER_TYPE(s) == implicit and status(s) == detected
      admService := PartitionIntoADm(s)
      Start(admServices)

```

```

LET activeIse := SelectISE(admService) IN
SEQ
  ldmService := PartitionIntoLDm (activeIse, admService)
  Configure (activeIse, ldmService)
  DeployCode(ldmService.codeType,activeIse)
  StartService(ldmService,activeIse)
ENDSEQ

```

2. The MONITOR\_ISE\_STATUS rule is responsible for monitoring the status of the various iSEs under its control. It uses timer1 to periodically probe (RULE Probe) each of the currently active iSEs (maintained in iseList at the iSE\_RB), and receive the message updateIseNodesDetailsMSG from the iSEs. This message contains the status of the iSEs and the status of the LDm services currently enabled at these iSEs. The iSE\_RB updates its tables (iseNodesTable and iseNodesDetailsTable) with this information. The rule Probe uses the message getIseNodesDetailsMSG for probing the iSE and receives the message updateIseNodesDetailsMSG from it. The parameters attribList1 and attribList2 in the update function represent the list of table attributes or message fields that are chosen for the update operation.

```

Rule MONITOR_ISE_STATUS =
  Forall e ∈ iseList
    msg := Probe(e, timer1)
    If msg NOT null AND msg.msgType == updateIseNodesDetailsMSG Then
      update( iseNodesTable, msg, attribList1)
      update( iseNodesDetailsTable, msg, attribList2)

```

3. The MONITOR\_RES\_STATUS rule is responsible for monitoring the operational status of the resources available with the iSE\_RB itself. It uses timer2, to periodically monitor (RULE MonitorResource) the status of each of its currently active resources 'r', maintained in resourceList. The rule MonitorResource evaluates to 'false' (resStatus = false), if the resource 'r' has malfunctioned or failed. The resource is rectified by evaluating the Rule Recovery.

```

Rule MONITOR_RB_RES_STATUS =
  Forall r ∈ resourceList
    LET resStatus := MonitorResource(r, timer2) IN
    If (resStatus == false) Then
      Recovery(r)

```

#### b) Foreground process at the iSE\_RB

The Rule ISERB is involved in performing the ADm services of the setup and in-service phases in response to messages from other nodes on the iSEGrid and is modeled using the rules RECEIVE, ADD\_NEW\_USER, CONTROL\_IMPLICIT\_TRIGGERING, and CONTROL\_REGULAR\_SERVICE.

```

Rule ISERB =
  RECEIVE
  ADD_NEW_USER
  CONTROL_IMPLICIT_TRIGGERING
  CONTROL_REGULAR_SERVICE

```

1. The RECEIVE rule is common to all the iSEGrid components and has been described under RULE ISE\_PORTAL.

2. The ADD\_NEW\_USER rule responds to the intimateRBUserMSG message, received from the iSE\_Portal, by updating the three tables (userServicesTable, expTrigServiceTable, impTrigServiceTable) that hold the details of the various services to be offered on the iSEGrid. This rule makes use of RULE GetServices to determine the services mentioned in the message.

```

Rule ADD_NEW_USER =
  If msgRcvd.msgType == intimateRBUserMSG AND secureMsg(msgRcvd) == true AND
    source == myiSEPortal Then
    update( userServicesTable, intimateRBUserMSG, attribList3)

```

```

LET serviceList := GetServices(intimateRBUUserMSG) IN
  Forall s in serviceList
    If (SER_TYPE(s) == implicit) Then
      update(impTrigServiceTable, intimateRBUUserMSG, attribList4)
    If (SER_TYPE(s) == explicit) Then
      update(expTrigServiceTable, intimateRBUUserMSG, attribList5)

```

3. The RULE CONTROL\_IMPLICIT\_TRIGGERING is responsible for controlling the services of type implicit, based on the information available in the controlType field of the message controlImplicitTrigMSG. The rule SetMonitoringFlags is responsible for updating the monitoring flags for the services, based on the controlType. The rule ControlAtISEs determines the corresponding Adm services running at various iSEs by referring to the table iseNodesTable and sends the control message controlLDMSERVICEMSG to the iSEs.

```

Rule CONTROL_IMPLICIT_TRIGGERING =
If msgRcvd.msgType == controlImplicitTrigMSG AND secureMsg == true AND
    source == myiSEList Then
    SetMonitoringFlags(controlImplicitTrigMSG.controlType)
    ControlAtISEs(controlImplicitTrigMSG.controlType)

```

4. The Rule CONTROL\_REGULAR\_SERVICE is responsible for controlling the explicit services on the iSEGrid, in response to controlServiceMSG. This rule behaves similar to the Rule IMPLICIT\_TRIGG when controlType is start.

### 3.6.4 ASM rules for iSE

The ASM for the iSE, like that of iSE\_RB, consists of two main rules corresponding to the background process (Rule ISE\_BkGdProcess) and the regular process (Rule ISE). The Rule ISE\_BkGdProcess is involved in the management and control of the local resources. The Rule ISE is involved in performing the LDM services of the in-service phases, apart from responding to messages from iSE\_RB. This rule is modeled using the rules RECEIVE, CONFIGURE\_ISE, DEPLOY, SETUP\_SERVICES, CONTROL\_SERVICES, and FEEDBACK.

#### a) The background process at the iSE

This process is similar to RULE MONITOR\_RB\_RES\_STATUS defined in the ASM for iSE\_RB.

```

Rule Ise_BkGdProcess =
    MONITOR_LOCAL_RES_STATUS

```

#### b) Foreground process at the iSE

This is the main rule in the iSE component and consists of RECEIVE, CONFIGURE\_ISE, DEPLOY, CONTROL\_SERVICES, and FEEDBACK rules. The RECEIVE rule is common to all the iSEGrid components and has been described under RULE ISE\_PORTAL.

```

Rule ISE =
    RECEIVE
    CONFIGURE_ISE
    DEPLOY
    CONTROL_SERVICES
    FEEDBACK

```

1. The Rule CONFIGURE\_ISE is evaluated in response to the configureMSG message from the iSE\_RB, which informs the iSE to configure its resources and get prepared to perform a new service. The iSE\_RB may indicate in the configureMSG message whether the iSE needs to initiate an on-demand deployment of the code for the new service. The rule RequestIseACR makes a request to the iSE\_ACR for the appropriate code. The rule AllocateResource is responsible for determining the resources required for the new service, allocating them, and initializing them. myIseRBLIST contains the ID of all the iSE\_RBs known to this iSE.

```

Rule CONFIGURE_ISE =
If msgRcvd.msgType == configureMSG AND secureMsg == true AND source == myIseRBLList Then
    If configureMSG.deploy == true Then
        RequestIseACR(configureMSG.service)
        AllocateResource(configureMSG.service)

```

2. The Rule DEPLOY is evaluated in response to the codeMSG message from the iSE\_ACR, which transfers the code to the iSE. The rule LoadCode performs security checks on the message and integrity checks on the code before loading it. myIseACR is the ID of the iSE\_ACR that this ISE has access to.

```

Rule DEPLOY =
If msgRcvd.msgType == codeMSG AND secureMsg == true AND Source == myIseACR Then
    LoadCode(msgRcvd)

```

3. The Rule CONTROL\_SERVICES is evaluated in response to the controlLDMSERVICEMSG message from the iSE\_RB, which is responsible for controlling the operational status of the LDm services at the iSEs. The rule SetLDmMonitoringFlags is responsible for updating the monitoring flags for the LDm services, based on the controlType. The rule ControlResource determines the corresponding BNP services running at the iSE and the physical resources used by these services by referring to the two tables namely LDMServicesTable, and localResTable. The rule controls the status of these BNP services and their resources.

```

Rule CONTROL_SERVICES =
If msgRcvd.msgType == controlLDMSERVICEMSG AND secureMsg == true AND
    source == myIseRBLList Then
    SetLDmMonitoringFlags(controlLDMSERVICEMSG.controlType,
                          controlLDMSERVICEMSG.service)
    ControlResource(controlLDMSERVICEMSG.controlType,
                   controlLDMSERVICEMSG.service)

```

4. The Rule FEEDBACK is evaluated in response to the getIseNodesDetailsMSG message from the iSE\_RB, which is a periodic probe message. The status of the various services and resources (statusList) is obtained by evaluating the rule ExtractStatus. This information is then sent to the iSE\_RB using the updateIseNodesDetailsMSG message.

```

Rule FEEDBACK =
If msgRcvd.msgType == getIseNodesDetailsMSG AND secureMsg == true AND
    source == myIseRBLList Then
    statusList := ExtractStatus
    msg:=put(updateIseNodesDetailsMSG, statusList)
    To(sourceRB,msg)

```

The ASM rules for iSE\_ACRs and iSE\_DirsT are the responses to the messages from other iSE components and correspond to table manipulation operations (Select, insert, update or delete) in the case of iSE\_Dirs or code retrieval operations in the case of iSE\_ACRs.

## 4. Refinement and Uses of the iSEGrid ASM model

In this section we list out the uses of this model and discuss the level refinements in our ASM model.

### 4.1 Uses of the iSEGrid ASM model

The ASM model presented here for the iSEGrid is useful in many ways. Its primary use is to formally specify and check the fundamental operations of the entire iSEGrid and each of its individual components. These specifications aid in building a new iSEGrid or upgrading a non-iSEGrid system to an iSEGrid. This involves identifying a suitable network node to play the role of an iSE component and in determining the functionalities required to upgrade the network node to an iSE component. For instance, in

order to play the role of an iSE\_RB, a typical network node must satisfy the negotiation parameters as per the iSEGrid policy for iSE\_RBs (`ResourceSpecPolicy(iSERB)` in Rule `NEGOTIATE_1` of the iSEPortal), as well as the requirements of the matching algorithm, which is abstracted by the Rule `SelectRB` in Rule `RB_USER_PAIRING` of the iSEPortal. Such a network node after registering as an iSE\_RB must possess necessary resources for initial code deployment (`DeployCode(isCompType,iSERB)` in Rule `INITIAL_DEPLOY` of the iSEPortal), for holding the tables mentioned in Section 3.4, for communicating with other iSE components as per the messages in Section 3.3.2 and for performing foreground and background operations mentioned in Section 3.6.3.

Another use of this model is to determine the amount of additional load that each network node has to take-up in order to play the role of an iSEGrid component. This can be evaluated from the requirements mentioned above for upgrading a network node to an iSEGrid component.

The specifications also help to distinguish the operations of a conventional grid/grid node from iSEGrid/iSEGrid component or to distinguish the operations of a regular network/network node from an iSEGrid/iSEGrid component.

Further, this ASM model can also be used to identify the overhead in the network due to the presence of the iSEGrid. This is done by calculating the overhead due to each additional commitment on the iSEGrid as compared to that of the conventional network. By additional commitments we mean the registration operations, iSEGrid specific message processing, resources volunteered, etc.

## 4.2 Refinement of the iSEGrid ASM model

The ASM model presented here facilitates the design of a network layer grid using a top-down approach. The model has been presented after applying architectural-level and operational-level refinements. The definitions of the domains of the iSEGrid components and the interactions between the individual components provide architectural-level refinement. The specification of the tasks performed at each iSEGrid component gives operational-level refinement, i.e., we have provided finer details of iSEGrid-specific tasks such as negotiation (`NEGOTIATE_1`, `NEGOTIATE_2`), RB\_user pairing (`RB_USER_PAIRING`), implicit triggering (`IMPLICIT_TRIGG`), monitoring (`MONITOR_RB_RES_STATUS`, `MONITOR_ISE_STATUS`, and `MONITOR_LOCAL_RES_STATUS`), controlling (`CONTROL_IMPLICIT_TRIGGERING`, `CONTROL_REGULAR_SERVICE`, and `CONTROL_SERVICES`), configuring (`CONFIGURE_ISE`), etc. This level of refinement provides an understanding of the design of the iSEGrid. It offers sufficient detail to design such a grid and gives the flexibility to choose implementation-specific options.

Further refinement for partitioning services (`PartitionIntoLDm`, and `PartitionIntoADm`), for security services (`IsGenuineMsg`), for resource management (`AllocateResource`, and `ControlResource`) would provide better insight to an algorithm-level design of each component. Refinement of the rule `DeployCode`, would be in terms of adopting Active network technology or agent-based technology for code transfer. Refinement of the component selection rules i.e., a 'matching' algorithm (`SelectRB`, and `SelectISE`) must choose iSE\_RBs at the edge of the network, close to the iSE\_users/iSEs along the natural datapath between the end-users. However, such refinements are implementation specific.

## 5. Conclusions and Future Work

In this paper, a formal framework using ASMs has been proposed for iSEGrid, a grid at the network-layer. We have provided the rules for each component of the iSEGrid, along with the messages for intercommunication and the data tables required at each component. Our ASM model has been provided with architectural-level and operational-level refinements. We have listed out the uses of this model and have provided pointers to algorithmic-level refinements. We are currently working on coreASM [20] for validating and verifying our ASM specification. Future work involves developing further refinement schemes for this model, determining the amount of additional load that each network node has to take-up in order to play the role of an iSEGrid component and identifying the overhead in the network due to the presence of the iSEGrid.

## Acknowledgement

We thank Prof. Dr. Egon Boerger, Dipartimento di Informatica, Universita di Pisa, Pisa, Italy, for his valuable suggestions and timely help to complete this work.

## References

- [1] Intel Network Processors: [www.intel.com/design/network/products/npfamily/index.htm](http://www.intel.com/design/network/products/npfamily/index.htm)
- [2] Mark Kohler, "Introduction to Network Processors", CMP Media, 2000.
- [3] Michael Rabinovich, Zhen Xiao and Amit Aggarwal, "Computing On The Edge: A Platform For Replicating Internet Applications", Sun Microsystems, 2003, <http://www.sun.com/software/grid/whitepaper.edge.pdf>.
- [4] "Application-aware network", AT & T White Paper, 2004: [www.business.att.com/content/whitepaper/att\\_application-aware\\_network.pdf](http://www.business.att.com/content/whitepaper/att_application-aware_network.pdf).
- [5] Taf Anthias and Krishna Sankar, "The Network's NEW Role", ACM Queue Vol. 4, No. 4 - May 2006.
- [6] DiffServ and IntServ: [www.ietf.org/html.charters/OLD](http://www.ietf.org/html.charters/OLD)
- [7] Lakshmi Priya TKS, Ranjani Parthasarathi, "Architecture for an Active Network Infrastructure Grid – the iSEGrid", International Workshop on Active Networks, France 2005.
- [8] Sharmila R, LakshmiPriya MV and Ranjani Parthasarathi, "An Active Framework For A WLAN Access Point Using Intel's IXP1200 Network Processor", International Conference on High Performance Computing, HiPC 2004, Bangalore, p.71-80.
- [9] TKS LakshmiPriya, V.Hari Prasad, D.Kannan, L.Karthik Singaram, G. Madhan, R.Meenakshi Sundaram, R.M. Prasad, Ranjani Parthasarathi "Evaluating the Network Processor Architecture for Application-Awareness", Second Int.l Conf. On Communication System Software and Middleware – COMSWARE 2007, Bangalore, Jan 7 – 12 2007.
- [10] E.Boerger, R.Staerk: "Abstract State Machines - A Method for High-Level System Design and Analysis", <http://www.di.unipi.it/AsmBook/>
- [11] Egon Börger and Uwe Glässer, "A formal specification of the PVM architecture." In *IFIP 13th World Computer Congress 1994, Volume I: Technology and Foundations*, eds. B. Pehrson and I. Simon, North-Holland, Amsterdam, 402-409.
- [12] E.Boerger and Uwe Glasser, "Modelling and Analysis of Distributed and Reactive Systems using Evolving Algebras", Technical Report, 1995.
- [13] L.Yuri Gurevich and Raghu Mani, "Group membership protocol : Specification and verification Methods", ed. E.Boerger, Oxford University Press, 1995, 295-328.
- [14] Giampaolo Bella and Elvina Riccobene, "Formal analysis of the Kerberos Authentication System", Journal of Universal Computer Science, vol. 3no. 12 (1997), 1337-1381.
- [15] Zsolth Nemeth and Vaidy Sundarem, "A Formal Framework For Defining Grid Systems", International Conference on Cluster and Grid Computing, CCGrid 2002.
- [16] Egon Börger, "The ASM Ground Model Method as a Foundation of Requirements Engineering". In N.Dershowitz, ed., *Verification: Theory and Practice*, Springer LNCS 2772, 2004, 146-161.
- [17] Egon Börger, "The ASM Refinement Method". *Formal Aspects of Computing*, 15:237-257, 2003.
- [18] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. V. Vincente, and D. Villela, "A survey of programmable networks," ACM SIGCOMM Computer Communication Review, Vol. 29 , Issue 2, April 1999.
- [19] David L. Tennenhouse et al, "A Survey of Active Network Research", IEEE Communications Magazine, 35(1):80-86, Jan 1997.
- [20] Coreasm source code: <http://coreasm.sourceforge.net>