Architecture for an Active Network Infrastructure Grid – the *iSEGrid*

LakshmiPriya TKS, Ranjani Parthasarathi

Department of Computer Science and Engineering, College of Engineering, Guindy, Anna University, Chennai, India
tkslp@annauniv.edu, rp@annauniv.edu

Abstract. Although the net processing power in the network is increasing steadily, it is heterogeneous. Hence the immense compute-power may be underutilized at certain points while it remains inadequate at others. This paper proposes an active network-based framework that views the entire network as a single-entity to effectively utilize the network resources. The single-entity model is enabled by establishing an infrastructure grid *at the network layer*. Such a grid has the advantage of supporting a wide range of application-layer services in the network. Network processors and Active Network technology work in tandem to facilitate this. The network processors with their deep-packet-processing capabilities allow offloading of application-level processing into the network. Active Network technology allows this to take place on-demand. We present the design and architecture of the infrastructure grid, called *iSEGrid*, and illustrate its use for streaming services. We provide experimental results to indicate the potential and scope of the concept.

Keywords: Network Layer Grid, Network Infrastructure Grid, Active Networking, Network Processors, and Grid Architecture.

1. Introduction and Motivation

Active Networks (AN) technology has been proposed to support dynamic deployment of services in the network. This involves execution of code carried along with the data packets, at the intermediate nodes of the network. Researchers have extensively studied the potential benefits of this approach to various performance issues in the network [1,2,3]. Application specific tasks such as providing QoS, security, policy management, network resource management, translation, etc., have been shown to benefit from this approach. However, a significant challenge to this technology is the requirement for programmable network elements; especially in a scenario where the routers and switches in the network are built using ASICs and custom-hardware. Custom hardware is used to provide higher performance, however, it lacks the flexibility required for active networks.

In this context, the advent of Network Processors (NPs) which provide programmability without compromising on performance, serves as a boost to the AN technology. The benefits of this marriage of NPs and AN have just begun to be explored [4]. The entire spectrum of services from basic packet processing operations (such as classification and routing) to QoS specific operations (such as scheduling and queue management) [5], to application specific processing in the network (such as deep-packet inspection, filtering, and caching) can be supported actively using NPs.

NPs with their multi-core, multi-threaded architecture targeted at network processing functions have the potential to efficiently perform these operations and much more, at wire-speed. Recently, even application layer functions have been ported onto the NPs [19]. NPs can be positioned at the network-core, at the network-edge or as an attached processor at the end systems - both client and server. The functionality at these points may vary in complexity due to the heterogeneity of the end systems, and the traffic in the network. Thus with NPs pervading the network, the processing power in the network is bound to increase manifold. However, it may not be uniformly distributed. The processing power may be underutilized at certain points, but inadequate at certain others. We propose that this imbalance be exploited by viewing the entire network of intermediate elements as a single, coordinated entity. To this end, we propose a grid-framework that pools the in-network resources, and makes the network services available as a commodity. In this framework we propose the use of AN technology for dynamic deployment of network services on the NPs to suit the varying demands of applications. It is to be noted that this proposed grid framework operates at the network layer as opposed to conventional grids (computational grids, data grids, etc), which focus on the application layer [20]. Even the Active Grid framework of the RESO project [13,14], which is aimed at providing network services for the conventional grids using AN technology, focuses on the higher layers. Thus, our proposed grid framework is different from existing grids in that it is an infrastructure-level grid of active NPs.

The grid features that we exploit are: use of idle resources, large-scale sharing of heterogeneous resources spanning across different administrative domains, and a single-system view of the network. The different network devices play the role of service providers, resource brokers and coordinators depending on their processing capability and resource availability. The single-system view emphasizes end-to-end performance as opposed to localized solutions in conventional networks, and benefits both high-end and low-end clients of the network. Thus, the significant benefits that we foresee are handling of high-bandwidth applications with reduced burden on the end-systems, and offering of customized value-added services in the network, to low-end clients like handheld devices, in a transparent manner.

The proposed grid operations are facilitated by the use of NPs and AN technology. The NPs with their deep-packet-processing capabilities enable application-aware processing and allow offloading of application-level services into the network. AN technology allows this to take place on-demand. The primary goal of the AN technology is to decouple the network services from the networking element thereby enabling on-demand code deployment. The re-programmable nature of the NPs qualifies them to be Active Nodes. Thus NPs and AN technology together, enable on-demand deployment of application-aware services in the network.

This paper presents the conception of this infrastructure grid, describes the proposed architecture in detail, and illustrates its use for a specific application. It examines the suitability of NPs and AN technology and provides a proof-of-concept implementation of select key components. The organization is as follows. Section II presents the design of the proposed grid, its architectural components and the mode of operation. In Section III various scenarios of the *iSEGrid* are illustrated for multimedia traffic. This is followed by the evaluation of the *iSEGrid* in Section IV. Section V presents comparison with related work and Section VI concludes the paper with a briefing on work-in-progress.

2. Design of the *iSEGrid* – a network infrastructure Grid

The proposed infrastructure grid consists of network entities, which are in-network-service-aware entities (*iSE*s). Hence this grid is named '*iSEGrid*'. The purpose of this network infrastructure grid is to harness the tremendous network-processing power, and offer it as a commodity to the grid users - the end-systems. Here, the term 'end-systems' includes the server applications (*iSE_user_SA*s) and the client applications (*iSE_user_CA*s). The servers associated with the Internet service providers, media service providers, mail-service providers and content providers, along with their clients are the *iSE_user_SA*s. The *iSE_user_CA*s that benefit from the services of the *iSEGrid* may run on PCs, laptops, mobile phones or any other computational gadget. The grid environment is depicted in Figure 1.

The *iSEGrid* spans across the entire Internet, edge-to-edge, consisting of all sorts of edge nodes as well as core nodes, as its grid-resources – the *iSE*s. These resources posses diverse characteristics in terms of processing power, memory, data rate, type of protocol handled, QoS characteristics, data medium, type of interface, etc. In addition, being part of different administrative domains, these resources, follow different policies and practices. The requirement for an in-network node to be an *iSE* is the availability of 'excess' resources that it can volunteer to the grid and the ability to be an Active Node. Resources that may be volunteered are computational threads, CPU time, memory or buffers, and ability to handle an additional flow of packets. The providers of the network infrastructure, who volunteer their resources to the *iSEGrid*, constitute the *iSEGrid* service providers.

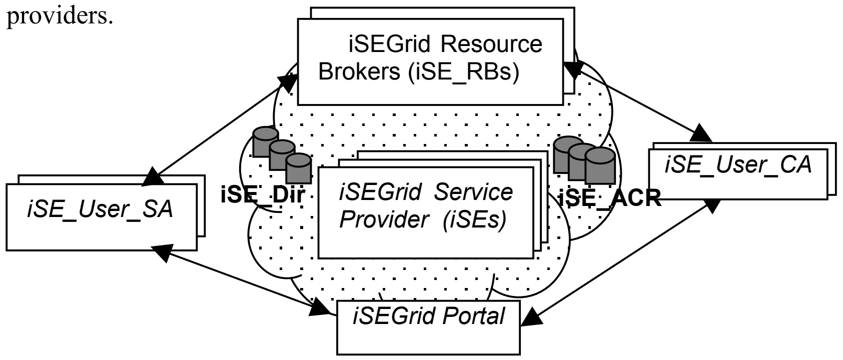
iSEGrid Resource Brokers (iSE_RBs)
iSE_User_SA
iSE_Dir
iSEGrid Service Provider (iSEs)
iSE_ACR
iSE_User_CA
iSEGrid Portal

Fig. 1. *iSEGrid* Environment and components

The *iSEGrid* coordinates its heterogeneous, distributed resources, to solve a common problem – the end-to-end performance. It employs resource brokers (*iSE_RB*s) which are powerful intermediate nodes, for typical resource brokering operations such as managing idle resources, delegating tasks to the *iSE*s, aggregating services from individual *iSE*s, enforcing policies, resource accounting and charging, and triggering the grid activities.

With the help of a rule-base, the *iSE_RB*s make intelligent decisions on the aggregate information collected from other *iSE*s. The *iSE_RB*s may also handle issues relating to fault tolerance, reliability and availability (esp., transient nodes) of the *iSEGrid*. They may cooperate with each other or form a hierarchical resource broker structure if necessary, to provide a service. In this paper, however, these extensions are not dealt with.

The *iSE_RB*s and the *iSEs* must possess the necessary code modules while offering the *iSEGrid* services. These modules are developed as active software components and are stored in the *iSE* Active-Code Repositories (*iSE_ACR*). In addition to the storage available at the *iSE Portal*, nodes like storage servers may volunteer storage to this repository. The active components are deployed at the *iSEGrid* nodes either during registration or on-demand.

From the kind of the operations performed at the various *iSEGrid* nodes, it is obvious that these nodes maintain a variety of information for normal operation. The data structures that maintain these data and metadata are collectively maintained as 'directories' (*iSE-Dirs*), located at various strategic points for use by the grid nodes.

The entry point into the *iSEGrid* is a publicly accessible portal, which advertises the *iSEGrid* services. The in-network node owners can register their nodes as *iSE*s via this portal while the server apps and the client apps can register themselves as *iSE_user_SA* and *iSE_user_CA* respectively. This portal maintains the static part of the *iSE_Dir*s while the rest is maintained at the *iSE*s that volunteer storage resources to the *iSEGrid*.

Thus the *iSEGrid* is seen to consist of four major functional components: *iSEGrid Portal, iSEGrid Resource Brokers (iSE_RBs), iSEGrid service providers (iSEs), iSEGrid Active Code Repositories (iSE_ACR) and the iSEGrid Directories (iSE_Dir)* as shown in Figure 1.

2.1 *iSEGrid* Service Architecture

The *iSEGrid* service architecture is a four-layered one, which replaces the network layer in a typical layered network architecture. For instance, in the TCP/IP model, this grid can be viewed as an extended IP layer, sitting below the TCP layer and above the MAC layer. The four layers of the proposed architecture are *Basic Network-processing (BNp)* layer, *Local Decision-making (LDm)* layer, *Aggregate Decision-making (ADm)* layer and *iSEGrid services* layer as shown in Figure 2. Of the four layers, the lower two layers, namely the BNp and the LDm layers perform the normal network processing or IP functions. The other layers namely, the ADm and the *iSEGrid* services layers, are the grid extensions to the IP layer.

The BNp layer includes services like packet processing, classification, header processing, flow identification, etc., that are performed at the individual in-network entities.

Above the BNp layer is the LDm layer. The services of this layer include the local policy and decision-making services. By 'local' services, we mean the consolidation of the BNp services that are performed at an individual *iSE* without the global view or interaction with other *iSE*s. The LDm services may be general purpose or may be specific to the applications. This layer exposes the network resources to the *iSEGrid*. For this purpose, in addition to the LDm services, it includes the communication and authentication protocols associated with the resources. The services of this layer include the policies for analysis of packets, the decision-making rules for operating on particular type of flow, the access-control policies and authentication mechanism for each *iSE*.

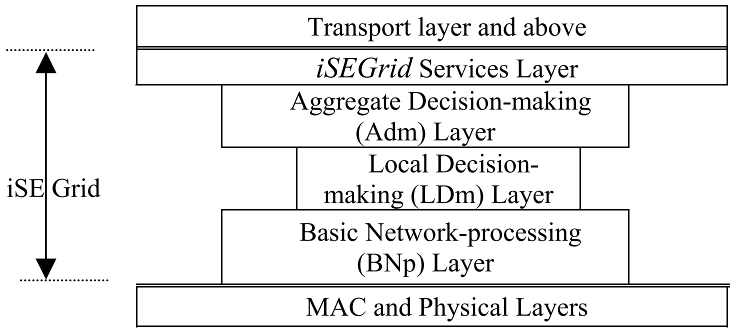
Transport layer and above
iSEGrid Services Layer
Aggregate Decision-making (Adm) Layer
Local Decision-making (LDm) Layer
Basic Network-processing (BNp) Layer
MAC and Physical Layers
iSE Grid

Fig. 2. *iSEGrid* Service Architecture

The LDm services of the *iSE*s are aggregated by the coordination of the *iSE*s. These collective operations constitute the ADm layer and are performed by *iSE*s that coordinate the network processing services of individual *iSE*s, namely the *iSE_RB*s. This layer exhibits intelligence in the network.

The aggregated services of the ADm layer are customized and offered as '*iSEGrid Services*' to the *iSEGrid* consumers as per their requirements. These services form the *iSEGrid Services Layer*. It is via this layer that the *iSEGrid* communicates its services to its consumers.

The *iSEGrid* architecture is hourglass shaped with the LDm layer at the neck of the hourglass. It is preferred to keep the LDm layer thin due to the development of a diverse range of in-the-net nodes, so that the set of core LDm services is small and a broad range of services at the ADm layer can be implemented on top of these.

Services at each layer or across layers are developed as active code components and are made available at the *iSE_ACR* s. The granularity of the code varies with the requirement.

2.2 *iSEGrid* operations

The *iSEGrid* operations can be explained under two phases namely the *iSEGrid setup phase*, which involves registration and module deployment, and the *iSEGrid-in-service phase*, in which the *iSEGrid* services are offered. The interactions between the *iSEGrid* nodes and consumers during these phases are depicted in Figure 3.

Phase 1- iSEGrid Setup phase

The *iSEGrid* is set up as the grid nodes (i.e., *iSE*s, *iSE_RB*s, *iSE_CR*s and *iSE_Dir*s), and the grid consumers, register (Figure 3). Network providers, the owners of a large variety of in-network entities including base stations, access points and CDNs, approach the *iSE Portal* to register their nodes as *iSE* grid nodes. Information regarding the configuration, capability, and constraints of these nodes are conveyed to the *iSEGrid*. Negotiations regarding security, accounting, type and service parameters, are carried out. As each *iSEGrid* node is registered, the active code modules required for the services sought, are deployed at the respective nodes. The *iSE_Dir* is updated and initialization procedures are executed at the new *iSEGrid* node.

At the time of registration of an *iSE_RB*, each new *iSE_RB* is associated with a set of *iSE_user_SA*. Similarly, when a new *iSE_user_SA* registers, the corresponding startup *iSE* modules are deployed and a specific *iSE_RB* is associated with it. The *iSE_RB* becomes its *first point of contact to the iSEGrid*. All further communications from the *iSE_user_SA* to the *iSEGrid* will take place via this *iSE_RB*. Initially, the known client groups of the SA are made the *iSE_user_CA*s. However, *iSEGrid* also permits adding client groups dynamically. This can be done in two ways. The *iSE_RB*s may automatically detect these clients by monitoring traffic at the server-edge, or receive intimation from the server on the arrival of requests from the clients. Figure 3 indicates registrations occurring in a particular order, but in practice, this varies.

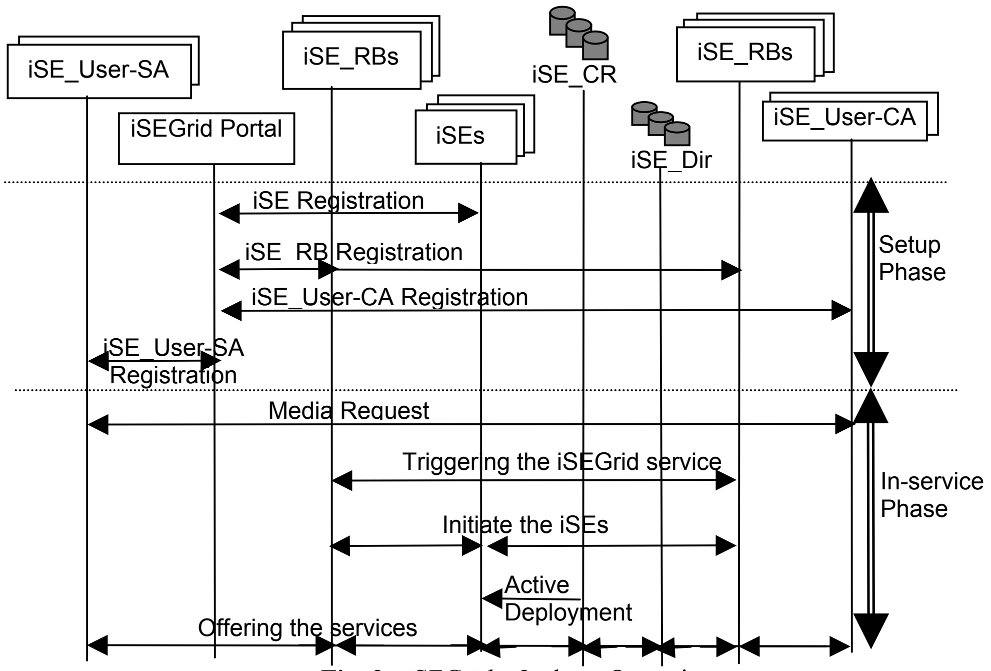

Fig. 3. *iSEGrid* – 2-phase Operation

Participation in the *iSEGrid* is transient. By this we mean that grid nodes and consumers are permitted to register and deregister alternately. However, a node may deregister only after, either completing or migrating, the committed services.

Phase 2 - iSEGrid-in-service phase

Soon after registration, the *iSEGrid* nodes enter the *iSEGrid in-service phase*. Step 2 is a typical client/server request. The arrival of the client request at the server triggers the *iSEGrid* service. This can be done in two ways: explicitly or implicitly. Explicit triggering occurs when the *iSE_user_SA* requests the *iSE_RB* for an *iSEGrid* service. On the other hand, implicit triggering occurs when the *iSE_RB* intercepts (i.e., deep packet processing) the flow at the server-edge and detects the need for an *iSEGrid* service. 'Triggering' an *iSEGrid* service, involves identifying and initiating the *iSE*(s). Identifying the right *iSE*(s) necessary to service the request involves intelligent decisions. Typical rules for this purpose may be based on, proximity to the client group; *iSEGrid* services offered by the *iSE*; ability to provide the service at that point of time; or expected response time. The identification of *iSE*s may be performed by *iSE_RB*s in isolation or in coordination with others. Once identified, the *iSE*s are initiated and the parameters for the service including the location of the *iSE_ACR* s are sent to them (Step 4).

The *iSE*s begin offering the services (Step 5) after the 'on-demand' deployment of the *iSE* code modules from the *iSE_CR*. Once triggered, the *iSE_RB*s periodically probe the *iSE*s and maintain the services. When a service terminates, wind-up operations are done at the *iSE*s and the *iSE_Dir*s are updated to reflect this change.

2.3 *iSEGrid* – Modes of usage

It can be seen from the above description that global view and coordinated functions are two key characteristics of the *iSEGrid*. We now present different modes of usage of the *iSEGrid* that exploits these two characteristics.

1. Integration of services

A straightforward application of the *iSEGrid*, would be to integrate the currently available localized mechanisms, from a global perspective.

2. Code and service movement

Since the grid allows dynamic deployment of services and code movement, any of the existing network services can be moved to the appropriate location (sometimes even to multiple locations) to provide efficient service. That is, services provided at the network edge may be moved into the network or those at the end systems may be moved to the edge and vice-versa. Thus the network resources can be effectively utilized and the burden on the end systems can be reduced. This has an added advantage that services that were hitherto available only to powerful end-systems can now be provided to less powerful end systems like the hand-held devices too.

3. Novel in-network service

In order to tap the full potential of the *iSEGrid*, novel solutions that exploit the in-network capabilities can be identified. One such solution is in terms of setting up a chain of services at the intermediate *iSE*s, along the path. These services may be aggregated or used in isolation. Even though this requires a paradigm change in the networking domain, it can be seamlessly integrated using the various features of the *iSEGrid*.
The next section illustrates each of these modes of usage as applied to multimedia services.

3. *iSEGrid* for Multimedia Services – An illustration

Multimedia traffic requires special attention especially because of the differences in its characteristics from those of other traffic in a network. Researchers have been working on various issues to improve the performance of multimedia applications by making the underlying network services, streaming-aware [6-10]. One of the major issues is timely arrival of media packets at the client node. Mechanisms like prefetching and caching; media-specific packet scheduling, network congestion avoidance using multipath-transmission, rate adaptation, and minimizing end-to-end retransmission have been proposed to reduce the latency of the packets. Yet another issue, namely high bandwidth requirement, is being handled by mechanisms like transcoding and multiple source streaming.

Each of these mechanisms operates at various points along the transport path i.e., some at the end systems, some at the network edge, and some at the core. Typically they provide solutions based on a localized view of the problem. Hence they do not guarantee an end-to-end solution, which adapts to varying network conditions. It is in this situation that the *iSEGrid* provides an alternative. Any of the three modes of usage mentioned in the previous section could be applied. We take a few instances to illustrate each of these modes.

1. Integration of Services:

We consider a full-fledged application, namely, flash crowd control in a p2p network, serviced by media servers. Flash crowds occur when an unexpected number of requests hit the server within a very short duration of time. One solution to this is the maintenance of a coordinated cache at the client end, which serves the clients locally during flash crowds [11]. Here, one or more of the clients performs the coordination. The physically distributed cache, which is the key component of this service, is made up of portions of memory volunteered by each client in the peer group.

An *iSEGrid-based* approach to this solution employs a **s**erver-**s**ide *iSE_RB* (*SS_RB*) and a **c**lient-**s**ide *iSE_RB* (*CS_RB*). The *SS_RB* monitors the load on the server and communicates peak-load conditions to the *CS_RB*. The *CS_RB* along with the peers in the network performs the pre-flash-crowd operations, i.e., caching the recently viewed clips at the clients and transparently maintaining their indices at the *CS_RB*. On the occurrence of flash crowd conditions at the media server, the *SS_RB* sends an intimation to the *CS_RB*. This intimation provides timely detection of flash crowds. The *CS_RB* immediately redirects further media requests to the locally existing objects, ensuring continuous delivery. Since the *iSE_RB* takes up most of the maintenance tasks, the load on the client is reduced. The *iSEGrid*-specific messages either have no payload or are light in weight and can be piggybacked. Thus, this application illustrates the unification of the load-monitoring service, which is typically performed at the server, and the coordinated cache service implemented at the client end for undisturbed service.

2. Code and Service Movement

We consider three different mechanisms that can be enhanced by code/service movement – feedback-based rate adaptation, QoS mechanisms for wireless network and transcoding.

Feedback-based rate adaptation at the server: Feedback is normally obtained from the end system or the network edge. In the *iSEGrid*, this feedback generation can be moved to the *iSE*s in the network and an aggregated feedback can be obtained at the server. The advantage gained is that information about the entire path is available at the server for rate adaptation and adverse conditions along the path are detected earlier.

Similarly, QoS mechanisms are normally deployed at the core of the network, for both wired and wireless networks. However, for a wireless network, it will be useful to move this service to the edge where the wired meet the wireless. The nodes at the junction of wired and wireless networks, which experience the varying characteristics of two different networks, are ideal *iSE*s, to impose QOS. These *iSE*s can perform QoS-specific scheduling, classification and queue management on the flows and adaptively cater to the changes in the wireless applications. The dynamic deployment feature of the *iSEGrid* can be used to enable the on-demand loading of the desired algorithm [5]. An *iSE_RB* can be used to detect the change in flow and initiate code transfer from the code repository.

Transcoding is normally employed at the edges – either server or client edge – to adapt to the client requirements in terms of bandwidth and resolution. In the *iSEGrid* environment, this service can be moved to any position in the path – server edge or client edge or any volunteering intermediate *iSE* – to dynamically accommodate variation in service and demand. Prefetching and caching services can also be offered in a similar manner. Here again, an *iSE_RB* will be used to coordinate this adaptive service movement.

3. Novel in-network solution

The in-network feedback generation described above is an example of an in-network chain of services. Similarly, a chain of link-cache can be maintained at the *iSE*s to cache high priority media packets at each link until the subsequent *iSE* (link) acknowledges. This provides early detection of link-level packet loss thereby avoiding end-to-end packet retransmission [12].

4. Evaluation of *iSEGrid*

To evaluate the proposed *iSEGrid*, we consider two aspects – the underlying technology, and the performance benefits to an application that uses the *iSEGrid*. Since the idea of the grid is motivated by the use of AN and NP technologies, it is important to study the feasibility of using the NP as an *iSE* and as an Active node. The evaluation is based on the scenarios described in the previous section.

1. Using NP as an *iSE_RB*

The *iSE_RB* functionality, illustrated in the previous section that coordinates transcoding, prefetching and caching has been developed on an Intel IXP1200 NP [17]. This *iSE_RB* is assumed to be located on a Base Station Controller (BSE) at the wired-wireless junction. This *iSE_RB* is responsible for detecting, scheduling and allocating volunteer *iSE*s for offering streaming service to mobile clients. This involves a sequence of events E1 to E6 as follows. The volunteers first register with the *iSEGrid* (E1). The media requests from the mibile are intercepted by the *iSE_RB* (E2). The *iSE_RB* then probes the volunteers for their availability (E3). On receiving a response (E4), it allocates the volunteer for the service (E5) and intimates the client (E6). The volunteer *iSE* then begins prefetching, caching and transcoding before streaming the object to the mobile client.

The ability of the IXP1200 NP-bases *iSE_RB* to handle these requests has been analyzed under two different scenarios: (1) a volunteering *iSE* is available till the end of the service (Figure 4a) and (2) the *iSE leaves* the system before completing the service (which requires reallocation) (Figure 4b). It is assumed that all packets arrive on 100Mbps lines with an inter packet gap of 960 nanoseconds. The μ-engines of the NP, operate at 232 MHz.

Figure 4 shows the time line diagram of the events that occur for scenario 1 and 2. The clock cycles at which the events occur are given.

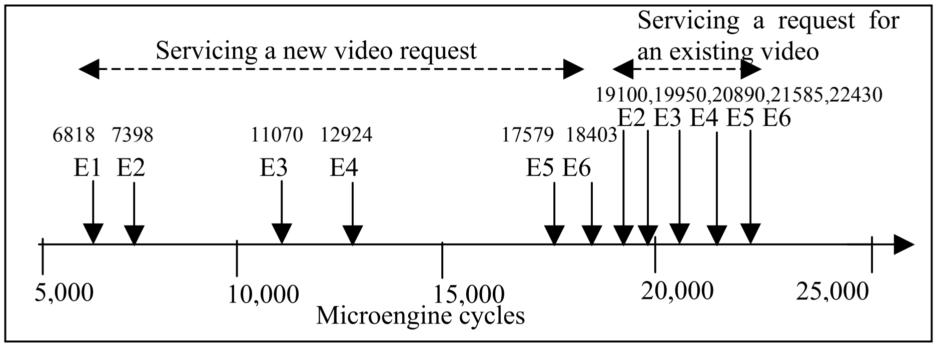

Fig 4a. Scenario 1

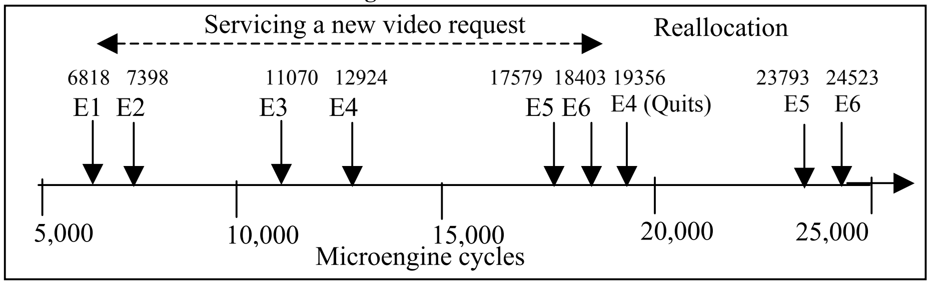

Fig 4b. Scenario 2

EVENTS

a) iSEGrid setup phase **:** E1 - volunteering iSE registers

b) iSEGrid-in-service phase : E2 to E6 (Explicit triggering)

E2 - streaming request packet arrives from a mobile
E3 - iSE_RB sends probe packet to the iSE
E4 – iSE sends response to the iSE_RB
E5 - iSE_RB sends start-service packet to the iSE
E6 - iSE_RB sends service-intimation packet to mobile

Fig. 4. Time line showing the events during the services of two requests

Scenario 1: Initially, the volunteer *iSE* registers (E1 at 6818). The sequence of events (E2 at 7398 to E6 at 18403), occur during the service of a video object for the first time (i.e., a new video). This is followed by another request for the same video (E2 at 19100 to E6 at 22430). The *iSE_RB* requires 11,005 μ-engine cycles (47.4 μ sec) for the first request, while for a subsequent request to the same object; it is only 3,330 μ-engine cycles (14.3 μ sec).

Scenario 2: Registration and service to a new video request (i.e., E1 to E6) are the same as in previous scenario. When the *iSE* leaves the system, it intimates the *iSE_RB* (E4 at

19356) of its unavailability. The interval between the points (6,18403) and (4,19356) indicates the partial streaming interval. Since the *iSE*'s service has not been completed, the *iSE_RB* does a reallocation (i.e., 19356 onwards) and sends the service-intimation (E6 at 24523) to the new *iSE*. The reallocation latency is 5167 μ-engine cycles (22.7 μ sec), for this scenario. The overall latency involved when a volunteer leaves, is found to be 16172 μ-engine cycles (70.1 μ sec).

The overheads of using an NP as an *iSE_RB* are viewed in terms of the resources (i.e., microengines of the NP) required and in terms of the message exchanges specific to *iSEGrid* operation.

The *iSE_RB* design utilizes all the six microengines of the IXP1200. Hence it is recommended to use the IXP1200 as an attached processor on the BSC or choose a higher version NP that can house both the functionality of the BSC as well as that of the *iSE_RB*. The *iSEGrid* specific message exchanges have been listed above as 'events'. Most of these messages either do not have a payload and hence may be piggybacked or are light in weight. The probe and start-service messages have no payload. The probe-response message contains the parameters, like channel-quality between the *iSE* and the mobile, and hence require a few tens of bytes as payload.

This experiment evaluates the effectiveness of NPs as RBs on the *iSEGrid* and shows the reduction in service.

2. Using NP as an Active iSE

The adaptive QoS service applied at the edge of wireless network as illustrated in the previous section has been developed on an NP-based WLAN Access Point (AP) [5]. This receives active packets (in the ANEP format) and dynamically loads the embedded QoS function. Active modules for various classification, queue management and scheduling algorithms have been developed on the active framework for IXP1200 based NPs. An active code handler module has been developed specifically to handle the active code and load it onto the microengines. The system has been tested with various active code modules. The NP-based active test-bed was found to receive the active packets, stop the currently running algorithm and load the new one appropriately.

Normally, the switching time is a major overhead of dynamic loading operations. However, the *iSE_RB* at the WLAN AP has been designed with two sets of queues, one for the currently running algorithm and another for the incoming algorithm to-be-loaded.

Here, the feasibility of an active NP-based *iSEGrid* component has been established.

3.An application on the iSEGrid

The flash crowd control application, as illustrated in the previous section, has been tested in an *iSEGrid* environment consisting of a *CS_RB*, an *SS_RB* and a peer-to-peer group of 12 clients sharing 20 media files. The effectiveness of the service is shown in the graph in Figure 5.

The percentage of requests that were *not serviced* by the server during a flash crowd, are plotted for five different bandwidth reservations at the server (data series 1). These server-rejected requests are handled by the *CS_RB*. The percentage of the total requests serviced by the *CS_RB* is indicated by the second data series in the graph. The difference between the first and the second data series indicate the usefulness of this service. The usefulness is calculated as *Effective Service Percentage* (ESP) of the *CS_RB*.

$$\mathrm{ESP} = \frac{\text{Percentage of requests serviced by } \mathit{CS\_RB}}{\text{Percentage of requests rejected by the server}} \times 100$$

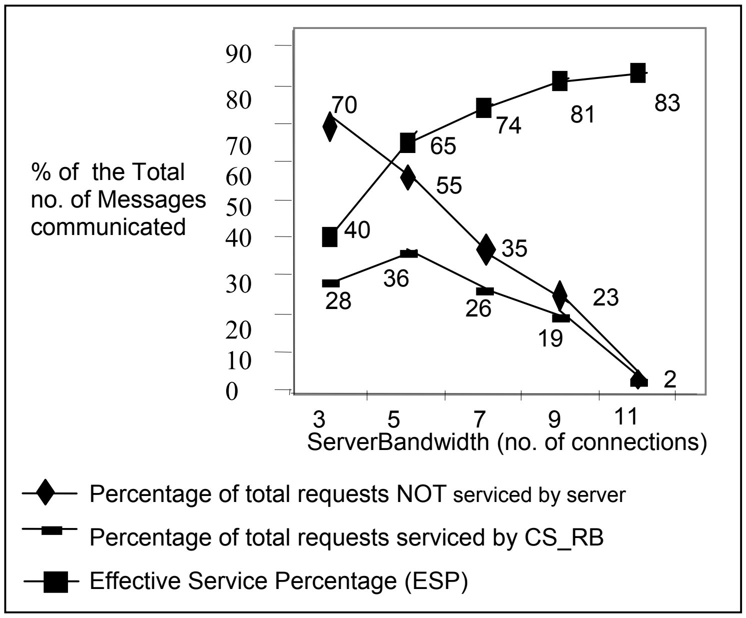

Fig. 5. Flash Crowd Control service of the *CS_RB* component

The ESP is the percentage of the server-rejected requests that are serviced by the *CS_RB*. The third data series gives the ESP. It can be seen that the ESP value is high when a large bandwidth is reserved at the server, and is lower otherwise. The ESP is the parameter that determines the usefulness of the *CS_RB*, for a given bandwidth reservation at the server. We find that the *CS_RB* is more useful when the bandwidth reservation at the server, is low. About 67 requests out of the rejected 167 (i.e., 40%) have been serviced by the *CS_RB* with a reservation of three connections. During such situations the *CS_RB* experiences maximum load.

The *CS_RB* for this service has been developed on an Intel IXP1200 NP to test its performance under various scenarios and varying flash crowd durations. The design of this service requires only two microengines, indicating that any NP-based edge device that can volunteer two microengines and 12 x ‘n’ Bytes of SRAM for table lookup can play the role of such a *CS_RB* with a client group size of ‘n’ peers.

The above evaluation also showcases two out of three modes of use of the *iSEGrid* – integration of services and code and service movement. The third aspect in-network novel service – has also been demonstrated, but the results have been presented elsewhere [12].

5. Comparison with related work

The goal of the infrastructure grid, proposed in this paper is to pool the network resources and to exploit the imbalance in their processing power to bring about better end-to-end performance and for enabling value addition for the applications. To do this effectively, the *iSEGrid* employs the positive aspects of NP and AN technologies.

The idea of combining active networking with powerful network resources is not entirely new and is very close to the idea of active grids proposed by the RESO project [13, 14]. Active grids focus on offering intelligent adaptive transport services to support computational grids (grid environments and apps).

While the RESO project is concentrating on developing application-aware components (AACs) for deployment at the edge nodes of the core network, the functionality of which is specifically for computational grids and their applications; *iSEGrid* concentrates on implementing the application-level services at the network layer thereby offering a common base for a wide range of technologies operating at the higher layer – computational grids, peer-to-peer networks, overlays, internet, and web services. It can be viewed as complementing the Active Grid by providing support at the network layer.

Another architecture that is similar to the *iSEGrid* is the Open Pluggable Edge Services (OPES) architecture [15,16], which brings application awareness into the network. However, OPES is an overlay of application-level network intermediaries while *iSEGrid* is an overlay of all network intermediaries. Hence the *iSEGrid* has a wider scope for in network functionalities. It is a complementary technology in the sense that it can also be used to support OPES.

In terms of the underlying concepts of resource sharing and coordination, the P2P computing paradigm and conventional grids are similar to the *iSEGrid*. However, while P2P networks operate at the end systems, *iSEGrid* spans across the network. In that sense the P2P concept is embedded in the *iSEGrid*. Similarly, conventional grids treat the entire network, as an individual resource, whereas *iSEGrid* goes deeper and focuses on the network components themselves. Thus the *iSEGrid* presents another dimension in the grid space.

6. Conclusion and future work

This paper has introduced the *iSEGrid*-framework that pools the in-network resources of NPs, and makes the network services available as a commodity. This infrastructure grid allows dynamic deployment of application-aware network services using ANs to suit the varying demands of applications. In essence, it has been shown that the synergy of three different technologies namely grid technology, ANs and NPs, can be exploited cleverly to expend the capabilities of today's networks for the future. The layered architecture and the design of the proposed *iSEGrid* have been presented. A few multimedia-specific scenarios have been illustrated to bring out the usage of the *iSEGrid*. Even though the use of the *iSEGrid* has been showcased for multimedia applications, it is to be noted that all the above services benefit non-streaming applications as well. Further challenge lies in exploring more issues to be solved by this paradigm. The evaluation of the *iSEGrid* in terms of the underlying technology, and in terms of different usages of the *iSEGrid*, has been presented.

To conclude, the *iSEGrid* has exposed a whole new paradigm for enabling networking services and solutions. Our architecture implies that an intelligent, efficient underlying grid is available to the application developers. The challenge now is in finding issues that can be solved better using this paradigm, and in exploring services that can be provided 'in the network'. We have initiated activity in this direction for both wired and wireless media streaming applications. We also plan to explore support for text processing applications. Work in progress includes the development of a network level simulator for the proposed grid. Further analysis is required in terms of security issues at the *iSEGrid* components, storage issues at the resource brokers, and the development of effective protocols for communication between the *iSEGrid* components with reduced overloads.

Acknowledgement
We thank M.Nithish, C.Ramakrishnan, J.Ramkumar, R.Sharmila, and V.Krishnan, for their contribution in the implementation of the *iSEGrid* modules.

References

1. Dan Decasper, Bernhard Plattner, "*Distributed Code Caching for Active Networks*", Infocom 1998.
2. William La Cholter, Priya Narasimhan, Dan Sterne, Ravindra Balupari, Kelly Djahandari, Arvind Mani, Sandra Murphy. "*IBAN: Intrusion Blocker Based on Active Networks*," DANCE, Vol. 00, p. 182, 2002.
3. S. Subramaniam, E.Komp, M.Kannan and G.J. Minden, "*Building a Reliable Multicast Service based on Composite Protocols for Active Networks*", IWAN 2004.
4. Andreas Kind, Roman Pletka, and Marcel Waldvogel, "*The Role of Network Processors in Active Networks*", ANTA 2002.
5. Sharmila R, LakshmiPriya MV and Ranjani Parthasarathi, "*An Active Framework For A WLAN Access Point Using Intel's IXP1200 Network Processor*", Bangalore, HiPC 2004.
6. Mohamed M. Hefeeda, Bharat K. Bhargava, and David K. Y. Yau, "*A Hybrid Architecture for Cost-Effective On-Demand Media Streaming*", Journal of Computer Networks, 2004.
7. Ralph Keller, Sumi Choi, Marcel Dasen, Dan Decasper, George Fankhauser and Bernard Plattner. "*An Active Router Architecture for Multicast Video Distribution*", Proceedings of Infocom, 2000.
8. Thinh PQ Nguyen and Avideh Zakhor "*Distributed Video Streaming Over Internet*", in Multimedia Computing and Networking 2002, Proceedings of SPIE, San Jose, California, January 2002, Vol. 4673, p. 186-195.
9. Xuan Chen and John Heidemann, "*Flash Crowd Mitigation via Adaptive Admission Control based on Application-level Observations*", ISI-TR-2002-557, May 2002 (updated on March 25th, 2003)
10. Turgay Korkmaz,, and Marwan M. Krunz, "*Routing Multimedia Traffic With QoS Guarantees*", IEEE Transactions On Multimedia, VOL. 5, NO. 3, September 2003
11. A.Stavrou, D.Rubenstein, S.Sahu, "*A Lightweight, Robust Peer-To-Peer System to Handle Flash Crowds*", 10th IEEE International Conference on Network Protocols (ICNP'02) 2002.

12.M.Nithish, C.Ramakrishna, J.Ramkumar, TKS LakshmiPriya, "*Design And Evaluation Of Intermediate Retransmission And Packet Loss Detection Schemes For MPEG 4 Transmission*", ITCC, 2004.
13.RESO Project : http://www.inria.fr/recherche/equipes_ur/reso.en.html
14.F. Bouhafs, B. Gaidioz, J.P. Gelas, L. Lefevre, M. Maimour, C. Pham, P. Primet, B. Tourancheau "*Designing and Evaluating An Active Grid Architecture*", The International Journal of Future Generation Computer Systems (FGCS)-Grid Computing: Theory, Methods and Applications, February 2005.
15.IETF OPES: http://ietf-opes.org/
16.Tuomas Nurmela, "*Analysis of Open Pluggable Edge Services*", Seminar On Hot Topics In Internet Protocols, 2004.
17.Intel IXP1200 Network Processor: www.intel.com/design/network/ products/npfamily/
18.V.Krishnan, "*InGRA-Intelligent Grid resource allocation for Mobile Clients*", Project Report, Dept. of CSE, CEG, Anna University, Chennai, India, May 2005.
19. Oyvind Hvamstad, Carsten Griwodz and Pal Halvorsen *"Offloading Multimedia Proxies using Network Processors",* International Network Conference 2005.
20. Foster, I. et al. *"The Anatomy of the Grid: Enabling Scalable Virtual Organizations"*, International Journal of Supercomputer Applications and High Performance Computing, 2001.

* * * * *