QoS Aware Resource Discovery in Mobile Environments

Yun Huang, Shivajit Mohapatra, Qi Han & Nalini Venkatasubramanian

School of Information & Computer Sciences

University of California, Irvine, CA 92697-3425

{yunh,mopy,qhan,nalini}@ics.uci.edu

Abstract

In this chapter, we will address the problem of resource discovery that ensures sustained QoS (Quality of Service) for mobile applications. We present two aspects of the resource discovery problem: (i) static resource discovery that determines the best resources to service a request when it is initiated, (ii) dynamic resource reprovisioning that continues to find nearby resources as a mobile client moves arbitrarily. We explain the role of context information in effective resource discovery and discuss what constitutes relevant context information for mobile applications. By introducing a generalized mediation-based architecture, we show how context information is collected and applied to support static and dynamic resource discovery. We illustrate potential solutions to these issues through a case study that uses nearby grid resources to support mobile services.

Contents

1 Introduction

Recent advances in high quality digital wireless network technologies coupled with the unprecedented growth of mobile computing devices, such as personal digital assistants, laptop computers, mobile phones etc. are enabling new classes of mobile applications with diverse QoS requirements. Today, mobile applications[1, 2, 3] span a variety of domains - from business and entertainment to education, command and control, and crisis response. Mobile gaming, audio/video streaming and collaborative multimedia applications are becoming ubiquitous and are projected to be the dominant applications in next generation mobile systems. These applications have distinctive performance and processing requirements which tend to make them extremely resource hungry. They also have diverse Quality of Service (QoS) requirements that determine the utility of the (perceived) information to the end-user. QoS needs can be expressed as user-perceived quality needs (e.g. video quality) that translate into lower level application/system parameters. In addition, QoS statements may specify constraints on timing, availability, security and resource utilization at various levels of abstraction. For instance, timing based QoS requirements can be specified using abstract properties such as correct/timely data delivery and uninterrupted service. These properties can be translated to concrete application parameters such as jitter, end-to-end delay, synchronization skew and/or concrete resource requirements such as network and disk bandwidth and buffer requirements [4]. The notion of QoS can include bandwidth management, throughput control, timeliness, reliability (e.g. mean time to failure, mean time to repair), perceived quality and cost (e.g. communication cost, service cost) and even battery energy management [5]. Resources required to support these multidimensional notions of QoS in mobile applications can be in form of computation (CPU), storage, bandwidth, memory or services that must continue to be available as the user moves in the mobile infrastructure.

One approach is to overallocate and reserve resources to meet peak demands at all times. Overallocation is impractical in mobile environments due to the fact that it results in (a) low resource utilization and (b)it is difficult to predetermine where and when resources are needed. If resource availabilities are known in advance, static admission control techniques combined with resource reservation protocols can be used to admit requests if the QoS demands of the services can be met adequately. However, in mobile environments resource availabilities can change over time in a very erratic manner rendering static reservations invalid.

Resource discovery to ensure sustained QoS for mobile applications presents several interesting research challenges. These challenges arise at different levels (network, server, device etc)and are summarized as follows.

(a) Bandwidth-limited wireless networks: Wireless networks are often bandwidth constrained (e.g. 10 of Kbps for cellular, 10-100Mbps for WLAN); they are also characterized by irregular connectivity, transmission errors and frequent disconnections. Supporting high quality data intensive flows (e.g. multimedia) requires predictable mobile network behavior in terms of bandwidth availability, network losses and transmission delays.

(b) Uncertainty due to user mobility: user mobility introduces uncertainty in user locations and consequently in bandwidth usage at different points. This implies that resource discovery needs to cope with the uncertainties and ensure consistent resource availabilities that satisfy application QoS.

(c) Insufficient resources in mobile devices: portable devices have limited processing capabilities, memory and energy, while mobile applications have significant resource needs.

(d) Cost/quality tradeoffs: there exists an inherent tradeoff between application QoS and resource consumption. For instance, dedicated network resources could render higher quality multimedia applications; however, this may result in low network utilization and hence higher cost.

(e) Lack of accurate context: discovering optimal resources requires the knowledge of underlying context. Collecting and maintaining accurate context information in mobile environments is challenging due to user mobility and tradeoffs between context accuracy and maintenance overhead.

Extending existing approaches for resource discovery that have been developed in the context of wired networks directly for QoS-aware mobile applications is problematic. For example, CORBA [6] based approaches are too heavyweight for mobile applications; furthermore, they are designed for relatively stable environments where disconnections are not the norm. Java based middleware solutions such as JXTA [7] incorporate service discovery techniques that are based on object class matching. These approaches are primarily focused at the application level, and QoS guarantees are difficult to meet since the class files of mobile objects may be distributed over the network. In peer to peer systems [8], resources are discovered through the collaboration among peer nodes. However, applying the P2P approach to mobile applications suffers from high connection cost, high network traffic for overlay maintenance, low network efficiency and high latency [9]. In addition, the unpredictability of peer based networks makes it difficult to ensure application QoS. In general, none of these approaches is designed with the focus of ensuring QoS for mobile applications, hence it is desirable to investigate effective resource discovery strategies appropriate for mobile environments.

In this chapter, we will elaborate on why intelligent resource discovery and provisioning is needed to guarantee QoS requirements for mobile applications. We argue that context (application, network, resource, device) plays a crucial role in effective resource discovery for mobile applications. We discuss static and dynamic aspects of the context collection and resource allocation problem and discuss how dynamic adaptation can sustain QoS guarantees under changing network, device and system conditions in a cost effective manner. We first present approaches for performing static resource allocation (network and server resources). Subsequently, we describe how dynamic resource reprovisioning can be effectively used to handle dynamic changes in resource availability and system state. Using mobile multimedia as the driving application for a case study, we illustrate how to integrate the dynamic changes non-intrusively into a wide-area mobile infrastructure.

Mobile applications are typically run in two types of environments: mobile ad-hoc networks and infrastructure-based networks. Mobile Ad-hoc Networks (MANETs) are wireless networks consisting entirely of mobile nodes that communicate on-the-move without any infrastructure support such as base stations or access points. Nodes in these networks will both generate user and application traffic and carry out network control and routing protocols. Rapidly changing connectivity, network partitions, higher error rates, collision interference, and bandwidth and power constraints together make routing [10, 11, 12, 13] and topology management [14, 15] interesting and difficult problems in MANETs. To provide focus, in this chapter, we will mainly address the issues involved in resource discovery in infrastructure based wireless networks (e.g. cellular or WLAN). Typically, these networks are composed of mobile devices with wireless interfaces and a core infrastructure with fixed and wired hosts. Mobile users move and connect to the fixed network via wireless access points or base stations. They can access services and possible resources provided by the end systems from the fixed network.

2 A Mediation-based Architecture for Resource Discovery

We address the challenges that arise in cost-effective resource discovery for QoS-aware mobile applications from the following three perspectives:

- *Adaptive context collection*: The success of resource allocation relies on timely and accurate knowledge of underlying context. Efficient context collection and monitoring techniques are therefore needed to keep track of the current global and local states and possibly even predict future changes.
- *Static resource discovery*: QoS-based static resource allocation mainly solves the problem of scheduling and admission control at the initiation of an end-to-end interaction. Given an application request from a mobile device with corresponding QoS needs, the resource discovery and provisioning process allocates the resources to establish end-to-end interactions by evaluating resource availabilities and estimating the system performance. For instance, this process may include deciding a network path with minimum delay, choosing a video server with compressed data that match quality needs, or allocating a web server that is in close proximity to a mobile user.
- *Dynamic resource reprovisioning*: Dynamic service adaptation aims to maintain optimal QoS for mobile users during the service period. When users are mobile, their wireless network connectivity might change dynamically, causing some resources/data to be inaccessible. This implies that resource reprovisioning is crucial to ensure continuous services for mobile applications.

To provide coordinated resource discovery for multiple mobile applications, we introduce the notion of a mediation-based middleware architecture (Fig. 1).

In this architecture, a mediator maintains the necessary system context in a *context repository* using which admission control, resource allocation and reconfiguration decisions are made to ensure QoS for mobile applications. Traditionally, much of system context information is gathered and maintained independently; for instance, network topology can be maintained by routing information exchange, a replica map can be obtained from a distributed domain name service, network management software keeps track of topology and link parameters, load balancing services monitor server load patterns, and content management and replication services manage data placement and distribution. Integrating the above information into a common context repository (database or directory service) has several advantages. Firstly, effective resource discovery and provisioning algorithms can exploit information from various levels for better system utilization. Secondly, keeping track of dynamic changes in servers, networks and content availabilities can be decoupled from policies for resource discovery. This provides for a clean separation of concerns in system design. Thirdly, using well-defined and uniform representations for

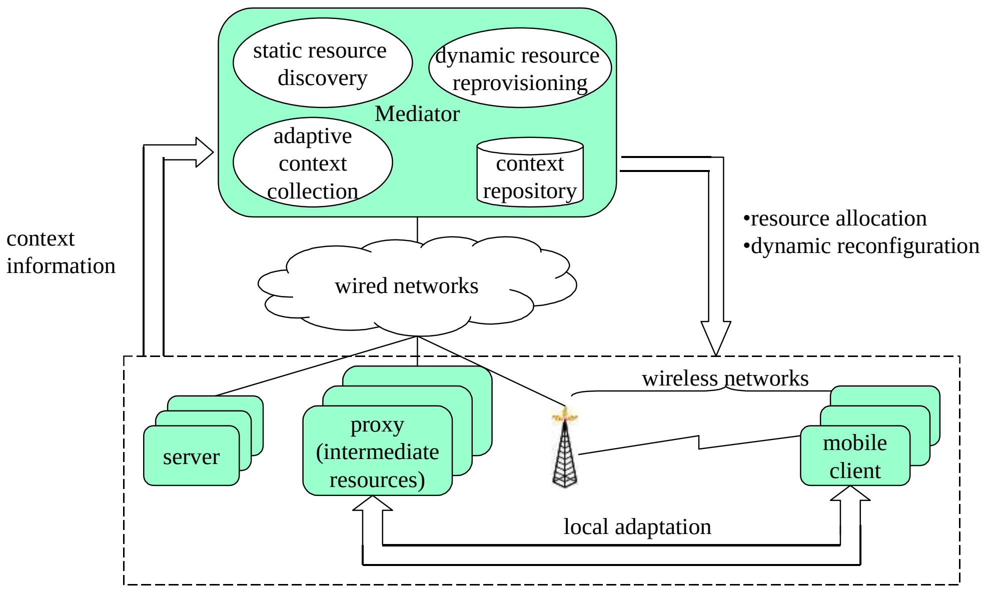

Fig. 1: A Mediator-based Architecture for Resource Discovery in Mobile Environments

the information allows easier manageability of data. Fourthly, knowledge of cross-layer information (from the network, application and devices) allows for flexible and efficient context collection. Such knowledge allows us to tailor the accuracy of the data in the context repository based on application needs, collection overhead and connectivity conditions. For instance, we may relax collection parameters when user QoS needs are not stringent, which will reduce collection overhead in an already congested network.

We also advocate the use of proxy nodes as intermediate resources in the mediation based architecture. Due to resource constraints in mobile devices and dynamic conditions in wireless networks, achieving sustained QoS between the service provider (server) and mobile device is difficult. The proxy approach attempts to use available resources in the wired networks within close proximity to the mobile device to support strategies such as proxy caching [16, 17], proxy-based transcoding [18, 19] or task-offloading [20] that can alleviate stringent resource needs of mobile applications. However, these intermediate resources must be discovered and deployed effectively.

QoS-aware mobile applications are supported by the generalized architecture as follows. A request containing QoS parameters is initiated at a mobile device. The mediator utilizes the resource availability information stored in the context repository (maintained by the adaptive context collection module) to decide an optimal allocation of network, server or proxy resources. Significant changes in the availability of the allocated resources will trigger the resource reprovisioning process to adapt the resource allocation accordingly. When the request terminates, the resources are reclaimed along the connection and the context collection module updates the resource availability status in the context repository. Using mobile multimedia as a driving application, we next describe techniques for addressing the key issues in resource discovery for QoS-based mobile applications - adaptive context collection, static resource discovery and dynamic resource reprovisioning.

3 Adaptive Context Collection for Effective Resource Discovery

Resource discovery and reprovisioning algorithms utilize information about current system context to ensure that applications meet their QoS requirements. For QoS-aware resource discovery, the relevant context can be classified into the following three categories:

- network core parameters: these include link bandwidth, link delay, loss rate, wireless channel conditions, mobility related parameters etc.
- network edge characteristics: these include parameters for servers (e.g., current server load, server CPU usage etc.), clients (e.g., device resource constraints, power levels, client locations) and content-specific attributes (e.g., number of replicas of video files, their locations etc.).
- intermediate resources: these include information on the capabilities of intermediate proxies, knowledge about when proxies are available and the relative stability of the proxy.

Accurate knowledge of context such as that described above enables optimal discovery and allocation of resources. Furthermore, knowledge of changing system context can be used by resource reprovisioning techniques for better QoS and performance. Therefore, the dynamic changes in system and network context must be captured rapidly with low overhead without interfering with the resource discovery and reprovisioning process.

The accuracy of context information can play a significant role in the efficacy of resource discovery techniques. For example, resource provisioning algorithms may select a network path for a flow/connection based on current resource availability, reserve the chosen path, and subsequently admit the request. In this process, imprecise system state information can lead to two types of failures. A *routing* failure may occur when a feasible path cannot be found for the new connection; a *setup* failure may occur when a seemingly feasible path is selected that ultimately does not have enough resources for the new connection. Neither failure is desirable; in particular, setup failures incur extra overhead to reserve resources that may never be used along the path. Maintaining accurate system/network status information can therefore help make right decisions, ensure the desired application QoS and consequently better user experience for mobile applications. However, maintaining accurate system context implies more frequent and tight monitoring; this in turn introduces significant network traffic resulting in poor utilization of underlying computation, communication and storage resources. The challenge then is to obtain *sufficiently* accurate state information to reduce the cost of collection while meeting user QoS needs.

There are various strategies for context representation and collection that address the cost-accuracy trade-off. Information representation and collection strategies are often intertwined. Any parameter in the context repository can be represented using either a single instantaneous value (i.e. the last measured value) [21] or by a range-based representation that approximates the value of a parameter by using an interval with an upper and a lower bound. The range size may remain static [22] or change dynamically [23, 24]. Corresponding collection policies [23] determine when and how often to sample the network components for current status information and whether to update the database with collected samples. Sampling periods may be fixed or may vary over time.

The need for flexible information collection is further aggravated in mobile environments due to the following reasons: (a) mobile devices roam across access points that connect them to wired networks, and the constant user mobility causes significant variations in the resource availability on various network links; (b) handheld devices typically have highly limited storage and computing resources; and (c) the resource availability can be substantially affected by computation and communication profiles of the applications executing on the device - this implies that capturing device limitations and the changes in device status as a part of the system image are necessary.

Collection and maintenance of location information for mobile hosts: Of particular interest in mobile applications is user location information. With accurate user location information, continuous service despite user mobility becomes possible, and nearby available resources can be discovered more effectively. There can be two ways using which location information can be collected. Fine-grained approaches maintain current location of each individual mobile client [25, 26], while coarse-grained collection captures information at an aggregated level for multiple clients. For example, *client aggregation*(i.e., mobile clients' population in each cell at a certain time instant) can be used as a coarse measurement for location information.

Fine-grained location information management has gained a lot of attention from researchers over the last few years. It typically involves three issues [25, 26]: location update strategies which decide when mobile users should inform the network about their current locations, paging strategies which decide when the base station should send out queries to search for the mobile user, and location information maintenance architectures which decide how to store and disseminate the location information. In addition, user mobility patterns (such as [27]) are studied concurrently in order to better capture current user location. However, gathering individual user location does not benefit the overall goal of efficient system status collection. Perturbation of residual resources caused by a single mobile user is almost negligible. Furthermore, keeping track of individual user mobility may entail significant overhead, since each mobile host needs to be probed separately and constantly.

A key observation that can be exploited for cost-effective collection of location information is that the movement of a large number of users may lead to non-uniform distribution of mobile users across cells. Previous work [28] has explored the use of coarse-grained mobility information that captures the distribution or aggregation of users in the mobile network to support cost-effective resource provisioning. In addition to lower overhead, collection using coarse-grained mobility information is independent of individual mobility models – this avoids inaccuracies introduced in modeling or predicting individual user's mobility. One measure of macro-level changes in mobile settings is the client aggregation status (i.e. number of users in a cell), which has the potential to significantly affect resource availability in the network/system. Client aggregation status can be obtained from cellular access points (i.e., base stations) that manage the communication of the mobile hosts residing within each cell. Base stations can apply a simplistic strategy (e.g., an update from the mobile host is triggered when handoff occurs) to maintain the total population of mobile hosts, or more complex prediction based approaches. In the absence of this information from base stations (which requires tight coordination with the service provider), it may be possible to derive/predict aggregate mobility status from individual mobility model. Prediction of aggregation status requires some knowledge about the distribution of mobile hosts and their mobility patterns in a region.

Once the coarse location information is obtained (either from base stations or via model-based predictions), it can be used to enhance system status collection. A family of collection strategies [28] have been proposed that use client aggregation status to drive the adjustment of sampling frequency and range size. The basic idea is as follows. The underlying topology is first partitioned into non-overlapping regions. Each region is equipped with a collection point that accumulates all the state information of the mobile hosts, servers, links for that region. A range with an upper bound and a lower bound is used to represent the mobile host aggregation status (e.g. number of hosts in a cell) in the directory service. The collection algorithm itself consists of two phases: Phase 1 derives the aggregate mobility patterns from individual user mobility patterns and utilizes the aggregation status and current resource utilization status to adjust the collection parameters such as sampling frequency and range size; Phase 2 utilizes feedback from mobile devices and the resource provisioning process for further customization of the collection process.

Fig. 2 demonstrates the performance of resource discovery and cost involved in maintaining resource availability information by using three different approaches to system status collection: mobility incognizant collection, collection using fine-grained mobility information and using coarse-grained mobility information. *Request completion ratio* (the percentage of requests that successfully complete) is used as a metric to measure the application performance. The request completion ratio is different from the request admission ratio. Admitted requests may not complete due to several reasons: there is no route with sufficient resources (a path failure); locating mobile hosts fails (a location failure); the alternate re-scheduling server may not have sufficient resources if path to original server is not available. We observe that the request completion ratio of resource discovery under the three approaches is close to each other; however, using fine-grained mobility information introduces significantly higher overhead, while using coarse-grained mobility information incurs the lowest overhead. This demonstrates the effectiveness of utilizing coarse-grained mobility information.

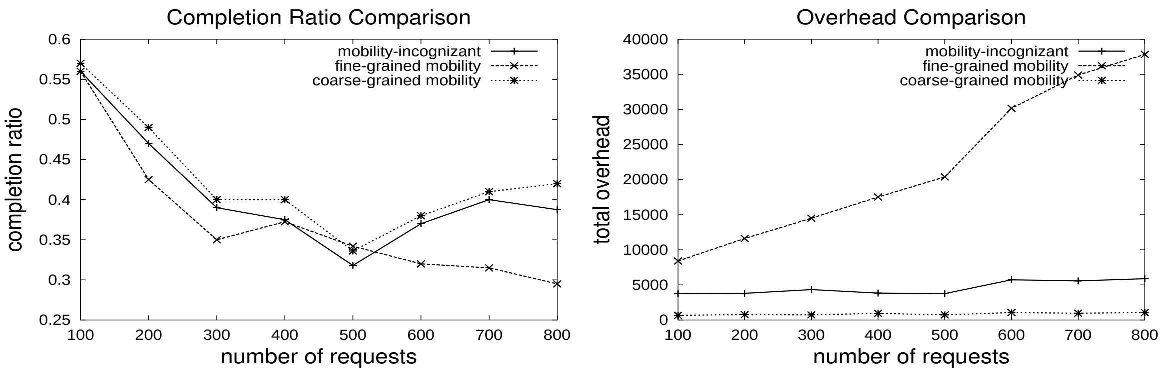

Fig. 2: Performance comparison of different system context collection approaches

With context collection strategies in place, we now describe how context information can be used in static resource discovery and dynamic resource reprovisioning to provide improved QoS for mobile applications.

4 Static Resource Discovery

In this section, we discuss the static resource discovery problem for mobile QoS-based services. Static resource discovery addresses the issue of discovering available resources (network, server, proxy) to provide acceptable services for a mobile user when the request is initiated. In this model, an incoming request from a mobile device expresses the service desired along with the QoS requirements (e.g. bandwidth needs, startup latencies, end-to-end delays) for the service. The incoming request goes through an admission control process that determines if end-to-end QoS can be satisfied under current (or predicted) conditions. If resources are unavailable to satisfy the QoS needs of the request, the incoming request is rejected.

Resource discovery mechanisms must ensure end-to-end application QoS while achieving an optimized resource allocation at the system level. To begin with, server and network resources must be discovered and provisioned. Selecting both network and server resources for multiple concurrent requests with varying QoS needs from a limited set of underlying resources is a challenging problem. In addition, resource constrained mobile devices can benefit significantly by using fixed resources within the wired networks (e.g proxies, idle machines, peer nodes) that are accessible through the wired/wireless infrastructure. Hence, the resource discovery mechanism must also address the discovery of proxy resources along the path of the service, preferably close to the mobile device. We advocate this approach because accessing data from nearby resources reduces network/server traffic; the proxy-based solution makes possible personalized services with high QoS satisfaction while improving overall system resource utilization.

There exists a fundamental performance-quality tradeoff that must be addressed to provide effective resource discovery for mobile applications. For instance, a solution that lowers the overall network traffic (for better performance) in the system might initiate the selection of the nearest resources; however, this can introduce frequent switches in the multimedia stream if the user moves rapidly, leading to increased jitter (i.e. lower QoS). Also, knowledge of the future (of application needs and resource availabilities) plays a useful role in supporting continued service with sustained QoS. Static resource discovery mechanisms that exploit knowledge of user and system needs (possibly through prediction techniques) in the long term can help better provisioning in two ways. Firstly, it enables the system to choose a suitable QoS service level that can be sustained for the entire duration of service. This is especially relevant when application QoS is constrained by available system resources and device energy limitations. This will minimize frequent changes in the QoS level leading to better user satisfaction. Furthermore the selection of network, proxy and server resources can be globally optimized for larger service durations; that will reduce frequent switching, thereby reducing service jitter and improving QoS.

There has been significant effort in discovering network and server resources for QoS-based applications. Server selection algorithms [29, 30, 31, 32] are often used to direct user requests to the optimal server based on chosen metrics (such as proximity or load) when data is replicated across multiple servers. These mechanisms often treat the network path leading from the client to the server as static. While this is useful for computation-intensive applications, interactive applications such as mobile multimedia must guarantee the availability of network resources as well. QoS-based routing techniques [33, 34, 35, 36, 37] typically aim to select the optimal path between a source-server pair and ignore the situation where there may be multiple servers that can serve the same request. Combined Path and Server Selection (CPSS) [38] is an integrated approach which allows load balancing not only between replicated servers, but also among network links. This has the potential to achieve higher system-wide utilization and allow more concurrent users.

In the remainder of this section, we present techniques for static discovery of proxy resources through a case study in infrastructure based wireless networks (e.g. cellular or WLAN). This case study illustrates how one might use idle grid resources as intermediate nodes/proxies for mobile applications. We will also discuss how knowledge/predictions of user mobility patterns, device capabilities and system resource availabilities can play important roles in proxy resource discovery for mobile applications.

4.1 Case Study: Using Grid Resources as Proxies for Mobile Multimedia Applications

Grid computing [39] is a distributed, high performance computing and data handling infrastructure that incorporates geographically and organizationally dispersed, heterogeneous resources. Traditional grid based research has focused on facilitating computation-intensive and data-intensive applications, e.g. AppLes [40], GrADS [41], and Nimrod [42]. Leveraging grid resources to facilitate mobile applications is motivated by the fact that the proliferation of freely available idle grid resources can be exploited efficiently to compensate mobile computing environment that is short of resources. Grid computing environment provides an ideal setting, where grid resources can act as proxies to improve power/performance of low-power mobile devices [43]. However, we need to address the challenge of identifying available grid resources that can be used as proxies.

Grid resource discovery for efficient mobile services poses distinct challenges due to the intermittent availability of heterogenous grid resources. The system needs to apply its knowledge of resource availability (e.g. using timemaps to quantify grid resource availability) when selecting a grid resource to build an end-to-end service channel for a mobile user. Note that availability of grid resources is unpredictable and the amount of available grid resources may fluctuate. Their stability features may also need to be investigated when resource discovery is performed. We illustrate how to apply knowledge of user mobility patterns to discover the nearby intermittently available grid resources, and how the system exploits knowledge of device information, e.g. energy sufficiency to perform better resource discovery. The proposed approaches have been implemented and evaluated in the context of the MAPGrid system [44]. The prototype system is illustrated in Fig.3.

Designing a Grid Resource Discovery Algorithm: In the MAPGrid system, we define a grid Volunteer Server (*VS*) as a machine that participates in the grid by supplying idle resources, i.e. *VS*s are intermittently available and geographically distributed. A *VS* (used interchangeably with proxy) in our case can be a wired workstation, server, cluster etc., which provides high capacity storage for storing multimedia data and CPU for multimedia transcoding, decompression and/or buffer memory. *VS*s are fixed machines and connect to the network using wired connections, whereas mobile hosts connect to the infrastructure using a locally available wireless network. A mobile user initiates a multimedia request, $R\!<\!VID,T,itinerary(opt)\!>$, where *VID* identifies

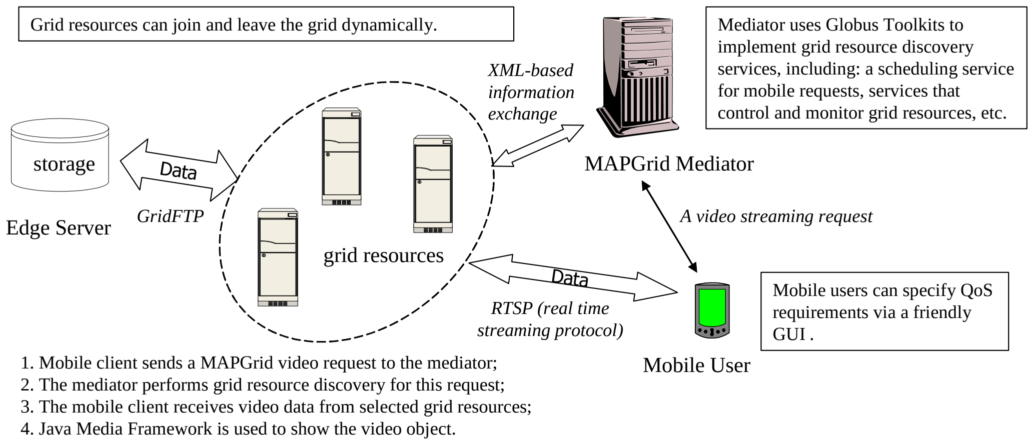
Grid resources can join and leave the grid dynamically.
Mediator uses Globus Toolkits to implement grid resource discovery services, including: a scheduling service for mobile requests, services that control and monitor grid resources, etc.
XML-based information exchange
storage
Data
Edge Server
GridFTP
grid resources
MAPGrid Mediator
A video streaming request
Data
RTSP (real time streaming protocol)
Mobile users can specify QoS requirements via a friendly GUI .
Mobile User
1. Mobile client sends a MAPGrid video request to the mediator;
2. The mediator performs grid resource discovery for this request;
3. The mobile client receives video data from selected grid resources;
4. Java Media Framework is used to show the video object.

Fig. 3: The MapGrid System Prototype

the requested video object, T represents the whole service period, and the *itinerary* contains user's mobility information (NULL, if no mobility information is available). Given mobile requests and information of grid resource availabilities, the static resource discovery tries to increase the overall acceptance of requests in the system by selecting optimal grid resources for each mobile request, meanwhile satisfying users' QoS requirements. The QoS requirements e.g. required network transmission bandwidth, will be determined by streaming a certain video streaming object.

The approach proposed in [45] is to divide the whole service period T into non-overlapping chunks (possibly of different sizes), each of which is mapped to an appropriate *VS,* e.g. the one that is geographically close and lightly loaded. Video objects are also divided into equal-sized segments. Corresponding video segments are downloaded onto selected *VS*s. The selected *VS* processes the request by transcoding the video segment and transmits the video stream via wireless links to bandwidth-limited and performance-limited mobile clients, such as PDA. Below we discuss a phased approach that exploits knowledge of user mobility patterns and grid resource availabilities.

In the first step, a time-based approach or a distance-based approach is applied to partition the service period T into chunks [45]. The time-based policy attempts to minimize the number of *VS* switches, and the distance-based policy uses knowledge of user mobility patterns and applies a well-known unsupervised neural learning technique called self-organizing map (SOM). If *itinerary* information is given, after service partitioning, the *Focus* of each chunk is calculated to identify the ideal resource location for each service period. Specifically, if $(a_i\ , b_i)$ represent the coordinates of the center for region i, and D_i represents the time duration spent in region i, we convert the problem of locating the chunk *Focus* into a Minisum Planar Euclidean Location Problem [46]. The objective is to minimize the overall distance cost *f(x,y)*, where *(x,y)* represents the coordinates of the *Focus* position of this chunk. The minimum value of the objective function *f(x,y)*, specified in equation (1), determines the *Focus* position of this chunk that is composed of N regions.

$$f(x,y) = \sum_{i=1}^{N} Di\sqrt{(x-a_i)^2 + (y-b_i)^2} \tag{1}$$

In the second step, a graph theoretic technique is applied for selecting an optimal set of grid resources to service each chunk. Decisions should be made by taking into consideration all the following factors: (a) intermittent availability of grid resources, (b) currently allocated workloads and predicted future workloads on grid resources, and (c) user's distance to grid resources. Basically, the goal is to select a lightly-loaded and nearby grid resource to service each chunk. In order to deal with heterogeneous grid resources in a unified way, and represent how much a request affects a server during each service period (chunk), a *VSFactor* is defined to measure a *VS*'s desirability as a grid resource for one service chunk of the mobile user. According to this definition, shown in equation (2), a *VS* with a larger *VSFactor* value is a better choice for servicing a particular service period [45].

$$VSFactor(VS,\ chunk) = \frac{Availability\ of\ VS}{VS\ workload \times Distance(\ VS\ ,\ the\ Focus\ of\ this\ service\ chunk)} \tag{2}$$

The problem of discovering intermittently available grid resources is further cast as a maximum flow problem, illustrated in Fig. 4. Nodes O and F are artificial nodes and represent the source vertex and sink node respectively. Node C represents the mobile client, and VS nodes represent volunteer servers. A set of time nodes TNs are introduced, each of which represents a period of service time. Weights for directed edges are also assigned as illustrated in Fig. 4. A feasible maximum flow solution that meets resource constraints corresponds to a possible scheduling solution; the basic solution has been adapted to develop a family of policies that cater to various application QoS needs [47].

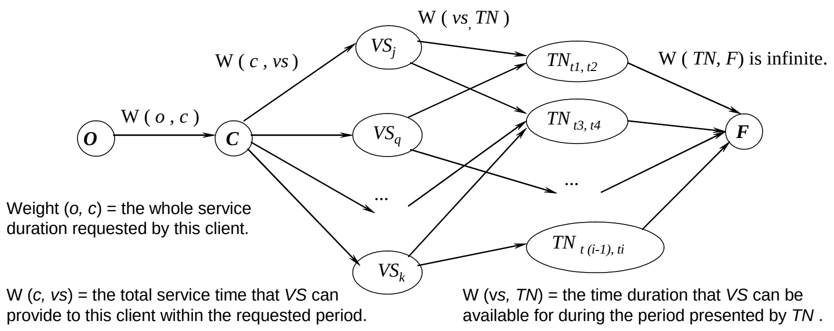
W (vs, TN)
W (c , vs)
W (TN, F) is infinite.
W (o , c)
O
C
VSj
VSq
VSk
...
TNt1, t2
TNt3, t4
TN t (i-1), ti
F
Weight (o, c) = the whole service duration requested by this client.
W (c, vs) = the total service time that VS can provide to this client within the requested period.
W (vs, TN) = the time duration that VS can be available for during the period presented by TN .

Fig. 4: Modelling the resource discovery problem

4.2 The Role of Device Constraints in Resource Discovery

One of the major constraints of executing multimedia applications on thin mobile devices is energy insufficiency. There exists tradeoff between QoS and energy that can be exploited to overcome energy limitations on mobile devices. This tradeoff is based on the fact that streaming lower quality video to power deficient mobile clients results in lighter traffic over the network and less computation for decoding video frames and therefore less energy consumption on the mobile device. One possibility is to use proxy resources to perform dynamic transcoding, which can help balance application QoS based on the device's residual energy. Since degrading video quality directly affects user perception (QoS), it is important to understand the notion of video quality for a handheld device and its implications on power consumption. Fig. 5 illustrates an E-Q (Energy/Quality) matrix for handheld computers (Compaq iPaq 3650) [19] to identify video quality parameters (a combination of bit rate, frame rate and video resolution) that produce user perceptible changes in video quality and noticeable shifts in power consumption for handheld computers.

QUALITY	Video transformation parameters	Avg. Power Windows CE (w)	Avg. Power Linux (w)
Q8 (original)	SIF, 30fps, 650Kbps	4.42	6.07
Q7 (Excellent)	SIF, 25fps, 450Kbps	4.37	5.99
Q6 (Very Good)	SIF, 25fps, 350Kbps	4.31	5.86
Q5 (Good)	HSIF, 24fps, 350Kbps	4.24	5.81
Q4 (Fair)	HSIF, 24fps, 200Kbps	4.15	5.73
Q3 (Poor)	HSIF, 24fps, 150Kbps	4.06	5.63
Q2 (Bad)	QSIF, 20fps, 150Kbps	3.95	5.5
Q1 (Terrible)	QSIF, 20fps,100kbps	3.88	5.38

Fig. 5: E-Q (Energy-Quality) Matrix for handheld computers (Compaq iPaq 3650)

Using the E-Q matrix, we can map each video quality level to a network transmission bandwidth and a power cost value (or vice versa). When a mobile request $R <VID,T,Q_{MIN},Q_{MAX},E_R,itinerary(opt)>$ specifies the lowest QoS level (Q_{MIN}) and the highest QoS level (Q_{MAX}), and gives information about current residual energy E_R, the E-Q matrix can be used to determine the best QoS level for this service. A straightforward extension of the static grid resource discovery algorithm to a energy-aware admission control algorithm using the E/Q matrix is shown in Fig. 6. Detailed explanations of the algorithm are presented in [48].

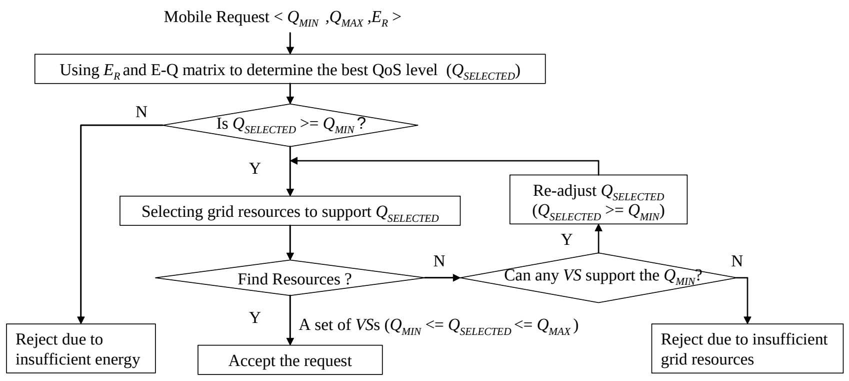
Mobile Request < Q_{MIN} ,Q_{MAX} ,E_R >
Using E_R and E-Q matrix to determine the best QoS level ($Q_{SELECTED}$)
N
Is $Q_{SELECTED}$ >= Q_{MIN} ?
Y
Re-adjust $Q_{SELECTED}$ ($Q_{SELECTED}$ >= Q_{MIN})
Selecting grid resources to support $Q_{SELECTED}$
Y
N
N
Find Resources ?
Can any VS support the Q_{MIN}?
Y
A set of VSs (Q_{MIN} <= $Q_{SELECTED}$ <= Q_{MAX})
Reject due to insufficient energy
Accept the request
Reject due to insufficient grid resources

Fig. 6: Energy-aware Admission Control Algorithm

Fig. 7 shows one experimental result that illustrates the performance of the energy-aware resource discovery algorithm [48]. Three different approaches are compared: (1) no QoS adaptation during resource discovery, (2) proxy-based transcoding only and (3) the energy-aware admission control algorithm described in Fig. 6. With the first approach, the system streams the video with the highest QoS level (original quality) to the devices (i.e. multiple quality levels are not supported at the servers). With the second approach, QoS degrades when the proxy (VS) resources in the system are not sufficient for the highest QoS level. The result shows that first approach leads to the lowest request acceptance rate, but the highest QoS provisioning for each service. However, as residual energy of the device is not considered, it also results in the largest number of incomplete services due to insufficient device energy. Approach 2 reduces the average QoS levels for requests, and therefore accepts more requests, while reducing the number of incomplete services. Approach 3 takes the residual energy of the device into account, while also performing quality transcoding; thus, it accepts the largest number of requests and completes all the accepted requests, assuming no other dynamic changes thereafter. Other experimental results [45, 48] also show that intelligent static resource discovery not only increases users' QoS satisfaction, but also significantly increases acceptance rate, completion ratio, as well as system throughput.

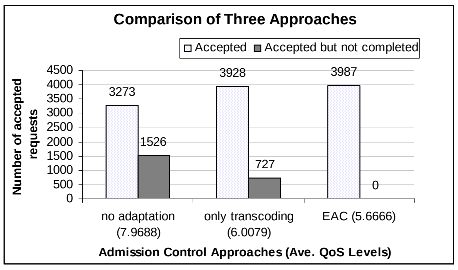

Fig. 7: Experimental results for static resource discovery: the Energy-aware Admission Control (EAC) techniques increases request acceptance and completion rates.

However, approaches for static resource discovery can often lead to over-provisioning of resources. That is, they may choose to deliberately over-estimate the number of resources a service is likely to require, and thereby sacrifice resource utilization. To a limited extent, static resource discovery methods can be further optimized by using profiled/historical information such as demand for a particular service, bandwidth and latency requirements for a service, and mobility patterns etc. However, in practice, it is still difficult to predict resource utilization and user mobility patterns in a mobile and wireless infrastructure. In the next section, we will describe approaches using which a system can track dynamic changes and adapt to these changes on-the-fly.

5 Dynamic Resource Reprovisioning

For mobile environments, it is difficult to accurately discover and provision resources using static methods for the entire duration of a service. This can be due to several reasons: (a) In wireless networks, disconnected operation and bandwidth fluctuations are common, making it impossible to discover/provision network bandwidth for the entire service duration. (b) device mobility makes it particularly difficult to provision resources, especially when there is no prior knowledge of how the user is expected to move. (c) Intermediate nodes might suddenly become unavailable (user unplugs the system) or system resources such as CPU, memory of servers and intermediate proxies might change unpredictably (starting or stopping applications can affect these resource availabilities). (d) Finally, mobile hosts may have unexpected changes in resource availability (e.g. new applications are started) which makes it hard to predict how resources are being consumed. These issues can be effectively addressed using dynamic resource provisioning where allocations of resources to services are automatically and continuously adjusted in response to either changing demands for a service and/or dynamic changes in resource availabilities in the system. This leads to a more accurate provisioning of resources and greater resource utilization.

However, dynamic reprovisioning in mobile environments complicates the problem of context collection and resource management. There are several difficult challenges that need to be addressed. What happens when an intermediate node (proxy) suddenly becomes unavailable or severely resource-constrained? In this case, services might be degraded, improved or terminated either to free resources or improve QoS. Strategies have to be designed to determine which services should be affected and how. Techniques to support on-the-fly assignments and revocations of resources must be developed. [49] outlines several issues that need to be addressed during resource revocation such as characterizing impact of revocation on services, handling deadlocks and designing revocation strategies. The above decisions are impossible to make without accurate knowledge of the system context. As the global state of the system changes dynamically, it is hard to maintain accurate context information. Who should maintain context information and how often should context be updated? What represents accurate context? Should context collection be distributed or centralized? Can we make global state estimation from local states? If so what is the accuracy (or error bound) on these estimates? Is there any general rule or systematic way for quantifying adaptations? All these open research issues and challenges are very pertinent to modern mobile environments and good solutions and insights to these problems will strongly impact mobile computing systems of the future.

To illustrate potential approaches to dynamic provisioning for mobile services we build upon the case study in section 4.1. Specifically, we address dynamic discovery and adaptation of resources for mobile multimedia applications that use grid resources as proxies. We focus on three aspects of the framework that are points of dynamic changes, namely: (i) the proxy (ii) the network, (iii) the mobile device.

5.1 Dynamic Changes in Proxy Resources

Proxies are participating machines on which applications can be randomly started or stopped, causing fluctuations in resource availability. Allocation mechanism must be capable of dealing with proxy failures and changes in proxy resources.

The worst case occurs when proxy is disconnected from the grid or switched off (e.g. unplugged) resulting in unavailability of the proxy itself. To deal with this problem, the broker needs to reallocate other available proxies to the interrupted requests in order to complete the interrupted services. When a specific proxy becomes unavailable, the broker retrieves information from the directory service about requests that are scheduled on the failed proxy, and triggers the re-scheduling process for each invalidated service. In order to reduce service failures and minimize service recovery time, the solution determines the order in which to migrate the disrupted services onto available proxies. To minimize service recovery time, invalid services are classified into two categories by the broker: (1) services that have been started and (2) services that are not yet started. Services in first category receive higher rescheduling priority than that of the second, whose service rescheduling can be postponed with an acceptable delay. Furthermore, within each category, requests with shorter remaining time of service and lower resource requirements receive higher rescheduling priority. If requests cannot be rescheduled, the broker downgrades a number of the disrupted services to accommodate them in the available proxies; if they still are not reschedule-able, even after downgrading the service, the broker notifies the clients that services has failed and releases pre-allocated resources for the services on the other proxies. The rescheduling process is then triggered for each invalidated service in order of decreasing priority. If requests cannot be rescheduled or postponed (for category 2 requests), the broker reports a request failure. Note that in the case of request failure, any resources reserved for this request on other selected proxies for the remaining service time should be released.

The proxy also needs to perform dynamic adaptations when its own resources reduce unpredictably (e.g. applications are started dynamically). If the resource changes are small, the proxy performs local adjustments to satisfy the QoS requirements of the current set of services. This might require downgrading the QoS (e.g. video quality) of an existing subset of services or allocating fewer resources temporarily to local applications. However, if there is a significant change in the resource availability at the proxy affecting the completion of certain services, then a subset of services have to be dynamically migrated away to another less loaded proxy. In this case, the proxy signals the broker to initiate a service migration algorithm, that migrates a set of services to another less loaded proxy. However, given system resource limitations, services that may not be schedulable on other proxies result in service failures. After successful migration of a service, the proxy re-adjusts the released resources and distributes them among the remaining services in order to minimize the number of migrations. The maximum number of migrations over the lifetime of a service can be achieved by placing an upper bound on the total number of migrations possible for a service. Fig. 8 shows that after proxies and broker perform the above adaptations there is a significant decrease in the number of requests that fail to complete due to dynamic changes in proxy availability [48].

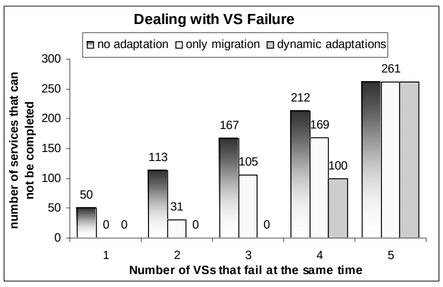

Fig. 8: Dealing with proxy failures.

5.2 Disconnections/Fluctuations in Wireless Network

The wireless network behavior depends closely on several factors such as wireless signal strength, congestion and noise. Each of these factors contributes to the connectivity and bandwidth availability of the network and can vary erratically over time, thereby making network provisioning a difficult problem. In order to effectively provision resources and adapt services for wireless networks, we need to (a) get accurate context information about changing network congestion and noise levels and (b) predict device mobility patterns. The congestion and noise information can be gathered from both the feedback from the device and by querying the wireless access points (section 3). With this information, the proxy can perform two different adaptations to improve QoS for multimedia applications: (i) proactive resource allocation and service adaptation based on device mobility and network congestion and (ii) adaptive network traffic management.

We explain the proactive adaptation approach by first differentiating it from the traditional reactive approachs that are representative of current best effort systems. In a more traditional *reactive* adaptation approach, a change in resource availability is first detected (possibly due to dropped packets, increased noise/congestion levels or low-power at the device), at a potential loss of QoS (video jitter). The proxy then reacts to this dynamic change by adapting the video stream (by either lowering or improving stream quality) to improve performance. However, the video/data packets already communicated might get dropped if mobile device suddenly enters a cell which is highly congested.

In dynamic environments, a proactive scheme can perform significantly better than a reactive scheme. In such a scheme, the proxy "proactively" predicts future system conditions and can determine how services can be adapted in advance. Specifically, the scheme exploits knowledge of system context and device mobility model to predict the number of users in a future target cell. With the knowledge of average traffic generated by each user, it can predict the dynamic congestion and noise levels within each target cell. This knowledge can be used in conjunction with the feedback from the device to "proactively" adapt either the video stream or the buffering (burst sizes) to maximize the application QoS. For example, the proxy predicts the noise/congestion level of a cell just before a user moves into the cell, and determines how to adapt the stream as the user enters the cell. In such a scenario, two factors significantly influence the performance of the schemes: predicted dynamic noise levels within each cell and the mobility (velocity) of the device. In [50] a comparison has been made between the proactive and reactive schemes for multimedia applications. The study also concludes that nature of distribution of noise induced by each mobile device has very little effect on the overall adaptations. Fig. 9 shows that the

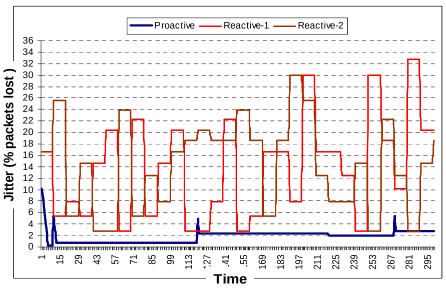

Fig. 9: Proactive vs. Reactive Adaptation. Proactive adaptation results in much smoother video with fewer quality fluctuations as opposed to the reactive scheme.

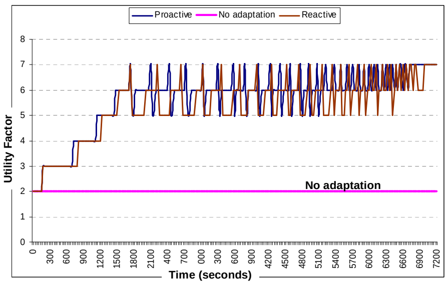

Fig. 10: Utility Factor vs. Adaptation scheme. A proactive scheme is able to provide higher utility to the system in terms overall improvement in video quality as well as battery energy savings.

proactive adaptation results in a much smoother video when compared to reactive adaptation. We see in Fig. 10, that a significant improvement in the overall system utilization is achieved using a proactive adaptation scheme .

An additional benefit is that proactive proxy-based adaptations can facilitate dynamic power management of the network interface card at the mobile device. Multimedia applications due to their periodic and predictable behavior present opportunities to use proxy based reprovisioning for improving the periods of inactivity for a wireless radio without affecting the application QoS. In [19], the authors describe an approach where a proxy can buffer video data (as opposed to sending data on a per frame basis) and send data to the mobile device in bursts along with control information containing the size of the bursts. This simple adaptation facilitates network card optimization as the device can now transition the radio to a low-duty cycle for longer periods thereby saving energy. The radio can then be switched back to active mode before the arrival of the next burst from the proxy, using the control information. The size of the bursts are dependent on the network congestion level, and buffering capabilities of the wireless access point as well as those of the mobile device and the quality of the video stream. The mobile host saves energy during periods of network inactivity as the radio consumes significantly less energy when operating in low duty cycle mode. Energy thus saved can result in improved QoS for the mobile service.

5.3 Fluctuations at the Mobile Device

On the mobile device, resource availabilities depend both on the number of applications and their demands on certain resources, either of which can change dynamically. We discuss two possible adaptations employed by the mobile client.

One approach is to dynamically migrate computationally expensive tasks from a mobile device to the proxy. While migrating tasks can reduce the computational load on a device saving both battery energy and reducing the load on the CPU, they often add extra communication overhead. Tasks such as media composing, encryption/decryption can be potential candidates for task migration. Such an approach can be profitable if the benefits of migration outweigh the overheads of migration and communication. [20] presents a graph theoretic analysis of how tasks can be partitioned and migrated from a local device to a proxy. When and how often such migrations need to happen are dictated by the number of applications and their computation and communication characteristics. Battery energy can also be incorporated into the adaptation by further modifying the algorithm to favor migrations when the residual battery power is low.

In addition to adapting to local variations in resources, a proxy can perform dynamic service adaptations based on feedback from the device as well as network state. For example, in the MAPGrid framework, we assume that a communication protocol is available that allows a proxy (grid volunteer server) to continually monitor the resource availability on a mobile device to which it is connected. This allows the proxy to dynamically adapt services to handle changes in the resource availability at a mobile device. We present a specific example of how a proxy can adapt the video streaming service in response to changes in device battery energy, but the concept can be extended to other resources.

If the residual battery energy of the device changes (e.g. either due to starting/stopping of applications on device or as a result of energy optimizations on device), the proxy reacts by changing the quality of the video stream to accommodate the changes. If the proxy determines that the device does not have sufficient battery energy to support the entire duration of the current service, it performs dynamic reprovisioning by streaming video at a lower quality to adjust to the lower residual energy at the device. If the proxy determines that the device cannot support even the lowest acceptable quality, it notifies the device about its depleted battery energy state. This implies that the device is consuming too much power (maybe due to other applications) and local proxy based adaptations cannot complete the service. Conversely, if there is an increase in the residual battery life of the mobile device either due to energy optimization strategies or reduction in the number of executing applications, then proxy based reprovisioning scheme can respond by increasing the video stream quality. The device can also employ dynamic adaptations at the operating system and hardware levels to optimize resource usage. For example, techniques like DVS [19] can be used to improve CPU utilization dynamically while saving energy. Note that these optimizations ultimately manifest themselves as higher QoS for applications executing on the mobile device.

Dynamic Changes in device/user mobility: As discussed in section 4, the static resource discovery solution optimizes assignment of proxies and their corresponding chunks for a particular service on the assumption that the overall mobility pattern of the mobile user is known. Specifically, a proxy that is geographically closest to the mobile device is preferred over more distant proxies. However, in real life, a user might dynamically choose to follow a different path in the middle of a service, thereby making the assignment of proxies sub-optimal (assuming the proximity criteria for optimality).

Initial studies have tried to study this problem by developing certain policies regarding how proxy assignments can be made when user mobility cannot be predicted in advance. One simple policy is to use a single proxy (single *VS*) for the entire duration of the service. A variation of this is the FastStartup policy, where a proxy is assigned some initial video segments to start the service immediately, while the algorithm searches for an ideal proxy or a set of proxies. Fig. 11 shows that policies such as FastStartup that systematically use multiple proxies performs better than the single proxy strategy under varying device mobility patterns [45]. A seamless way of incorporating dynamicity is to initiate the FastStartup policy when the mobile device has significantly departed from the assumed (predicted) path. Intelligent mobility prediction techniques can be also applied to further improve dynamic resource reprovisioning. For instance, knowledge of (predicted) mobility patterns can be used to design space/time based partitioning techniques for proxy allocation.

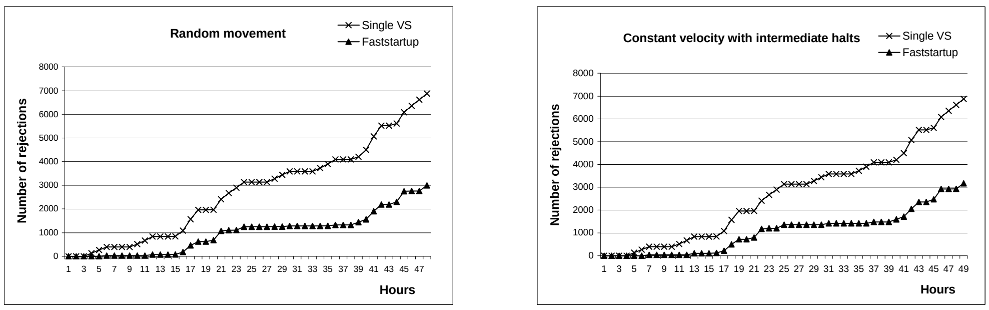

Fig. 11: Number of rejections over time with different device mobility patterns

6 Summary

In this chapter, we have described the problem of QoS-aware resource discovery for mobile applications and illustrated various elements of a solution to this issue using a generalized mediation based architecture. The chapter discussed in more detail on three key issues that must be addressed for resource discovery: (a) mechanisms to cost-effectively capture and maintain context information to enable resource discovery for mobile services, (b) algorithms for static resource discovery when a request is first initiated, using the available context information and (c) techniques for dynamic resource reprovisioning to deal with unanticipated changes in the applications, devices and the distributed infrastructure. Through a case study focusing on mobile multimedia services, we show how to take advantage of a priori knowledge of resource availabilities and mobility patterns to tailor the discovery and provisioning policies for better overall performance and enhanced user QoS. Efforts have also shown how context-aware intelligent policies for static and dynamic resource provisioning can support better application QoS, higher request acceptance rates and longer device lifetimes in mobile environments. One of the key observations in this chapter is the role of in-network proxy resources in enabling effective solutions to address the quality/energy/performance tradeoffs for QoS-based mobile applications. While service providers may choose to install dedicated proxy resources in-network, recent efforts have shown the possibility of leveraging additional heterogeneous machines within the proximity of mobile devices.

While the above steps are enabling technologies for the seamless execution of mobile applications under highly dynamic conditions, several challenges still remains to be addressed. Enabling Qos-based services in MANETs where a core wireless network infrastructure is unavailable poses new challenges. Issues of power-aware routing, QoS in the presence of topology changes and the ability to switch transparently between ad-hoc and infrastructure modes is still a challenge, which is further exacerbated in always-best-connected (ABC) networks where multiple access technologies (WiFi, cellular, BlueTooth, wired) may all be available to varying degrees simultaneously. Future work will also need to address the degree of location awareness required by the middleware for efficient allocation of mobile and fixed resources in a scalable fashion. Security (or lack thereof) of applications executing on wireless infrastructures is a big hurdle in the pervasive deployment of mobile services. Tradeoffs arise when timeliness requirements interfere with other application requirements such as security and reliability. Many of these challenges must be addressed to truly realize the eventual goal of widespread mobile services.

References

[1] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Dept. of Computer Science, Dartmouth College, Tech. Rep. TR2000-381, November 2000.

[2] D. Chalmers and M. Sloman, "Qos and context awareness for mobile computing," in *Proceedings of 1st Intl. Symposium on Handheld and Ubiquitous Computing (HUC'99).* Springer-Verlag, 1999, pp. 380–382.

[3] "Ercim news no. 54," July 2003.

[4] N. Venkatasubramanian, C. Talcott, and G. A. Agha, "A formal model for reasoning about adaptive qos-enabled middleware," *ACM Trans. Softw. Eng. Methodol.*, vol. 13, no. 1, 2004.

[5] D. Chalmers and M. Sloman, "A survey of quality of service in mobile computing environments," in *IEEE Communications Surveys*, 1999.

[6] S. Adwankar, "Mobile corba," in *IEEE DOA*, 2001.

[7] M. J. Yuan, *Enterprise J2ME: Developing Mobile Java Applications.* Prentice Hall, 2003.

[8] M. Hefeeda, D. Xu, A. Habib, B. Bhargava, and B. Botev, "Collectcast: A peer-to-peer service for media streaming," *ACM Multimedia Systems Journal*, 2005.

[9] B. Bakos, L. Farkas, N. Jukka, and G. Csucs, "Peer-to-peer protocol evaluation in topologies resembling wireless networks. an experiment with gnutella query engine," in *International Conference on Networks*, 2003.

[10] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994.

[11] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *INFOCOM (3)*, 1997.

[12] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353.

[13] C. Perkins, "Ad-hoc on-demand distance vector routing," 1997.

[14] L. Bao and J. J. Garcia-Luna-Aceves, "Topology management in ad hoc networks," in *ACM MobiHoc*, 2003.

[15] P. B. Godfrey and D. Ratajczak, "Naps: Scalable, robust topology management in wireless ad hoc networks," in *IEEE IPSN*, 2004.

[16] A. Singh, A. Trivedi, K. Ramamritham, and P. Shenoy, "Ptc: Proxies that transcode and cache in heterogeneous web client environments," *World Wide Web*, vol. 7, no. 1, pp. 7–28, 2004.

[17] M. S. Raunak, P. Shenoy, P. Goyal, and K. Ramamritham, "Implications of proxy caching for provisioning networks and servers," *SIGMETRICS Perform. Eval. Rev.*, vol. 28, no. 1, pp. 66–77, 2000.

[18] S. Chandra and A. Vahdat, "Application-specific Network Management for Energy-aware Streaming of Popular Multimedia Formats," in *Usenix Annual Technical Conference*, June 2002.

[19] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N.Venkatasubramanian, "Integrated power management for video streaming to mobile handheld devices," in *ACM Multimedia*, 2003.

[20] S. Mohapatra and N. Venkatasubramanian, "“PARM: Power-Aware Reconfigurable Middleware”," in *ICDCS-23*, 2003.

[21] "Ospf version 2," July 1991.

[22] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Quality of service based routing: A performance perspective," in *ACM SIGCOMM*, 1998.

[23] Q. Han and N. Venkatasubramanian, "Autosec: An integrated middleware framework for dynamic service brokering," *IEEE Distributed Systems Online*, vol. 2, no. 7, 2001.

[24] Z. Fu and N. Venkatasubramanian, "Adaptive parameter collection in dynamic distributed environments," in *IEEE ICDCS*, 2001.

[25] I. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, and W. Wang, "Mobility management in next-generation wireless systems," in *IEEE Proceedings Journal*, August 1999.

[26] V. W.-S. Wong and V. C. Leung, "Location management for next-generation personal communications networks," in *IEEE Network*, 2000.

[27] Z. Haas, "A new routing protocol for the reconfigurable wireless networks," in *IEEE Int. Conf. on Universal Personal Communications*, 1997.

[28] Q. Han and N. Venkatasubramanian, "Information collection services for qos-aware mobile applications," *IEEE Transactions on Mobile Computing*, 2005.

[29] J. Guyton and M.F.Shwartz, "Locating nearby copies of replicated internet services," in *ACM SIGCOMM*, 1995.

[30] A. Fei, G. Pei, R. Liu, and L. Zhang, "Measurements on delay and hop-count of the internet," in *Proceedings of Globecomm*, 1998.

[31] P. Francis, S. Jamin, V. Pasxon, L. Zhang, D. Gryniewica, and Y. Jin, "An architecture for a global internet host distance estimation service," in *IEEE InfoCom*, 1999.

[32] A. Myers, P. Dinda, and H. Zhang, "Performance characteristics of mirror servers on the internet," in *Proceedings of Globecom*, 1999.

[33] S. Chen and K.Nahrstedt, "Distributed qos routing in ad-hoc networks," *IEEE JSAC, Special Issue on Ad-hoc networks*, 1999.

[34] W. Zhao and S. K. Tripathi, "Routing guaranteed quality of service connections in integrated service packet network," in *Proceedings of ICNP*, 1997.

[35] I.Cidon, R.Rom, and Y.Shavitt, "Multi-path routing combined with resource reservation," in *Proceedings of Infocom*, 1997.

[36] L. Breslau and S. Shenker, “Best-effort versus reservations: A simple comparative analysis,” in *Proceedings of Sigcomm*, 1998.

[37] Q.Ma, P.Steenkiste, and H.Zhang, “Routing high-bandwidth traffic in max-min fair share networks,” in *Proceedings of SIGCOMM*, 1996.

[38] Z. Fu and N. Venkatasubramanian, “Directory based composite routing and scheduling policies for dynamic multimedia environments,” in *IEEE IPDPS*, 2001.

[39] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, 1998.

[40] F. Berman and R. Wolski, “The apples project: A status report,” in *The 8th NEC Research Symposium*, 1997.

[41] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski, “The grads project: Software support for high-level grid application development,” in *International Journal of High Performance Computing Applications*, 2001.

[42] D. Abramson, J. Giddy, and L. Kotler, “High performance parametric modeling with nimrod/g: Killer application for the global grid?” in *International Parallel and Distributed Processing Symposium*, 2000.

[43] L. McKnight, J. Howison, and S. Bradner, “Guest editors’ introduction: Wireless grids–distributed resource sharing by mobile, nomadic, and fixed devices,” in *IEEE Internet Computing*, July/August 2004, pp. Vol. 8, No. 4.

[44] “Mapgrid.” [Online]. Available: http://mapgrid.ics.uci.edu/

[45] Y. Huang and N. Venkatasubramanian, “Supporting mobile multimedia services with intermittently available grid resources,” in *Proc. of HiPC*, 2003.

[46] P. M. Kaminsky, “Ieor 251, logistics modeling,” 2002.

[47] Y. Huang and N. Venkatasubramanian, “Qos-based resource discovery in intermittently available environments,” Jul 2002.

[48] Y. Huang, S. Mohapatra, and N. Venkatasubramanian, “An energy-efficient middleware for supporting multimedia services in mobile grid environments,” in *IEEE International Conference on Information Technology*, 2005.

[49] M. Satyanarayanan, “Fundamental challenges in mobile computing,” in *ACM PODC*, 1996.

[50] S. Mohapatra and N. Venkatasubramanian, “Proactive energy-aware video streaming to mobile handheld devices,” in *IEEE MWCN*, 2003.

Biography

Yun Huang is a Ph.D. candidate studying in the Bren School of Information and Computer Science at the University of California, Irvine. Her research background is in resource management for distributed middleware systems. Specifically, she is devising efficient resource discovery algorithms and data placement strategies for providing mobile users with multimedia services by leveraging heterogeneous and intermittently available grid resources. She has an M.S. in Computer Science from the University of California, Irvine. She received her B.S. degree in Computer Science from Tsinghua University, China.

Shivajit Mohapatra is currently a PhD. candidate at the Donald Bren School of Computer Science at University of California, Irvine. His research interests include Distributed Operating/Middleware Systems, Multimedia and Real-Time Systems. His research focus has been on building theoretical solutions (graph theoretic and game theoretic) and designing adaptive middleware techniques for achieving power and performance benefits for low-power mobile devices. He is affiliated to the Distributed Systems Middleware Group as well as the Center for Embedded Systems at UC, Irvine. He expects to graduate in Spring 2005. Before joining the PhD. program at UCI, he worked as a Senior Software Engineer at Wipro Global R&D in Bangalore, India. He received his Bachelors degree in Computer Science from Birla Institute of Technology and Science (BITS), Pilani.

Qi Han is currently pursuing a Ph.D. from the Bren School of Information and Computer Science at the University of California, Irvine. Her research interests include distributed systems middleware, mobile and pervasive computing, and systems support for sensor applications. She is currently developing adaptive middleware techniques for collecting various dynamic context data in heterogeneous environments to support context aware applications. She has an M.S. in Computer Science from the Huazhong University of Science and Technology, China. She is a student member of the IEEE.

Nalini Venkatasubramanian is an Associate Professor at the Bren School of Information and Computer Science, University of California, Irvine. Her research interests include distributed and parallel systems, middleware, mobile environments, multimedia systems/applications and formal reasoning of distributed systems. She is specifically interested in developing safe and flexible middleware technology for highly dynamic environments. Nalini was a member of technical staff at Hewlett-Packard Laboratories in Palo Alto, California for several years where she worked on large scale distributed systems and interactive multimedia applications. Nalini has also worked on various database management systems and on programming languages/compilers for high performance machines. She has an M.S. and Ph.D. in Computer Science from the University of Illinois, Urbana-Champaign and is a member of the IEEE and ACM.