Information Collection Services for QoS-aware Mobile Applications

Qi Han, *Student Member, IEEE,* Nalini Venkatasubramanian, *Member, IEEE*

Abstract

Efficient resource provisioning that allows for cost-effective enforcement of application QoS relies on accurate system state information. However, maintaining accurate information about available system resources is complex and expensive especially in mobile environments where system conditions are highly dynamic. Resource provisioning mechanisms for such dynamic environments must therefore be able to tolerate imprecision in system state while ensuring adequate QoS to the end-user. In this paper, we address the information collection problem for QoS-based services in mobile environments. Specifically, we propose a family of information collection policies that vary in the granularity at which system state information is represented and maintained. We empirically evaluate the impact of these policies on the performance of diverse resource provisioning strategies. We generally observe that resource provisioning benefits significantly from the customized information collection mechanisms that take advantage of user mobility information. Furthermore, our performance results indicate that effective utilization of coarse-grained user mobility information renders better system performance than using fine-grained user mobility information. Using results from our empirical studies, we derive a set of rules that supports seamless integration of information collection and resource provisioning mechanisms for mobile environments. These results have been incorporated into an integrated middleware framework *AutoSeC* (Automatic Service Composition) to provide support for dynamic service brokering that ensures effective utilization of system resources over wireless networks.

Index Terms

mobile environments, distributed applications, multimedia applications, middleware.

I. Introduction

The proliferation of mobile devices and wireless networks have increased the scope of applications that will execute in future pervasive environments. Increasingly, distributed applications (e.g. mobile multimedia) have diverse Quality of Service (QoS) requirements (such as transmission rate, delay and delay jitter etc.) that require guaranteed levels of resource availability (such as network bandwidth and server CPU etc.) from the underlying systems infrastructure. Resource provisioning algorithms such as

The authors are with the Department of Computer Science, University of California, Irvine, CA 92697. Email:{qhan,nalini}@ics.uci.edu; Telephone: (949)824-5898; Fax: (949)824-4056. Contact author: Qi Han.

QoS based network routing and server selection utilize information about current system status to ensure that applications meet their QoS requirements. The accuracy of system state information can play a significant role in the efficacy of the QoS management 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 distributed applications.

This paper addresses challenges in collecting and maintaining accurate information about system state for mobile applications. Maintaining accurate information about available system resources is complicated by dynamically changing network, server and client status and changing QoS profiles of applications. Gathering accurate information becomes even more challenging in a mobile environment due to two main reasons. Firstly, mobile devices roam between access points that connect them to wired networks, and the constant user mobility causes significant variation in the resource availability [1], [2] on various network links. Secondly, hand-held devices typically have highly limited storage, computing resources. The resource availability can be substantially affected by the computation and communication profiles of the applications executing on the device - this implies that capturing device limitations and changing device status as a part of the system image is necessary.

In this paper, we consider the scenario where mobile and stationary hosts co-exist in a mixed wired and infrastructure-based (cellular) wireless networks. In the cellular infrastructure, the coverage area is divided into a number of cells with one base station per cell. All the wireless devices residing in a cell communicate with other entities in the distributed system via the base station. The base stations, in turn are connected to the wire line network. Our work aims at ensuring the desired QoS levels for applications executing on mobile devices, despite frequent changes in system state. This is achieved by effectively monitoring system status in the wired and wireless networks and allocating resources accordingly. Fig. 1 shows a high level system overview. Satisfying user QoS implies accurate maintenance of system state via frequent 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*

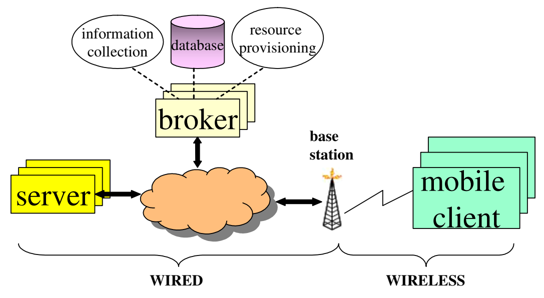

Fig. 1. *A high level system overview*

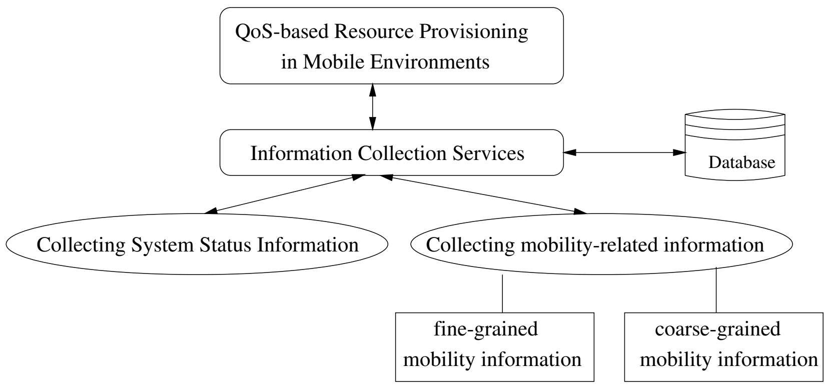
QoS-based Resource Provisioning in Mobile Environments
Information Collection Services
Database
Collecting System Status Information
Collecting mobility-related information
fine-grained mobility information
coarse-grained mobility information

Fig. 2. *Approaches presented in this paper*

accurate state information to reduce the cost of collection while meeting user QoS needs.

Our Approaches: To address the competing goals of accuracy and cost-effectiveness, we exploit the fact that users are able to tolerate system state imprecision at a certain level. Our strategy is to tailor the accuracy and fidelity of the captured data to ensure that applications can execute at desired levels of quality while minimizing resource consumption. The questions that need to be answered are: (1) what information needs to be gathered; (2) how frequently the information should be collected; and (3) at what granularity the information should be maintained. We aim to improve the performance of *QoS-based resource provisioning* by providing a reasonably accurate picture of the underlying systems, obtained and maintained by the proposed *information collection services*. The relevant information includes system status information such as network parameters, server and client characteristics etc., as well as mobility-related information such as current locations of mobile clients, power levels of mobile devices, and group mobility patterns etc. Collecting mobility information at a fine-grained level (i.e., for each individual mobile client) would facilitate adaptive resource provisioning; however, it may also incur significant overhead. We, therefore, explore the benefit of using coarse-grained mobility information (i.e., for each cell). We use the coarse-grained information to adjust information collection parameters, which indirectly improves the performance of resource provisioning. Fig. 2 illustrates the approaches studied in this paper.

Contributions of This Paper: Recent studies evaluated the impact of stale link state on the performance of QoS routing [3]; results indicated that the routing algorithm, network topology and link-state update policy have significant impact on the probability of successfully routing new requests, as well as the overhead of distributing network load metrics. Along similar lines, this paper aims to explore the impact of system status information collection policies on the efficacy of QoS-based resource provisioning for mobile applications. Specifically, this paper makes the following contributions:

• We discuss the issues involved in collecting system status information and present three general collection policies. Furthermore, we explore how system status collection policies can benefit from mobility related information. We show that collecting coarse-grained mobility information introduces significantly less overhead as compared to collecting fine-grained mobility information. We then present a family of strategies where coarse-grained mobility information is used to adjust the granularity of the system status information and the degree of dynamics with which current state information is obtained and updated.

• We empirically evaluate the impact of the information collection strategies on diverse resource provisioning techniques under varying mobility status, network/server conditions and different application workloads. Through the extensive performance studies, we identify the most effective information collection strategy for different scenarios and derive a set of service composition rules.

• We propose a prototype middleware framework, *AutoSeC* (**Auto**matic **Se**rvice **C**omposition), currently being developed at UC Irvine that can dynamically select an appropriate combination of information collection and resource provisioning policies according to current system conditions and user requirements. AutoSeC is designed to support heterogeneous application workloads (network or computation intensive) and manage resources in heterogeneous environments with both fixed and mobile devices while introducing minimal management overhead. We also present the preliminary results and discuss the integration of AutoSeC into existing wireless network infrastructures.

The rest of this paper is organized as follows. In Section II we discuss strategies for system status collection; in Section III we develop information collection policies that are driven by user mobility. We evaluate the impact of information collection algorithms on the performance of QoS-based resource provisioning and discuss the results in Section IV. Section V presents *AutoSeC*, a middleware framework for dynamic service brokering in wired/wireless environments, discusses implementation issues and the integration of AutoSeC into existing wireless network infrastructure. We present related work in Section VI and conclude with future directions in Section VII.

II. System Status Information Collection

Information used in QoS-based resource provisioning includes network parameters (e.g., available link bandwidth, link delay), server characteristics (e.g., current server load, server resource availabilities), client characteristics (e.g., resource constraints, power levels, locations) and content-specific attributes(e.g., number of replicas of video file, their locations) etc. Traditionally, much of this 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 maintain server load patterns, and content management and replication services keep track of data placement and distribution. Integrating the above information into a common repository (database) has several advantages. Firstly, effective end-to-end QoS provisioning algorithms can exploit information from various levels for better system utilization [4]. Secondly, keeping track of dynamic changes in server, network and content availability can be decoupled from policies for resource provisioning. This provides for a clean separation of concerns in system design. Thirdly, using well-defined and uniform representations for the information allows easier manageability of data. Fourthly, knowledge of cross-layer information allows for flexible and efficient data collection. Such knowledge allows us to tailor the accuracy of the data in the database based on application needs, collection overhead and connectivity conditions. e.g., we may relax collection parameters when user QoS needs are not stringent, which will reduce collection overhead in an already congested network.

Dynamic changes in system status must be captured rapidly with low overhead to allow applications to adapt to the changing conditions. The need for flexible information collection is further aggravated in mobile environments where the degree of dynamicity can vary significantly over time in different parts of the underlying system. The more accurate the database, the easier it is to satisfy more user requests in a timely fashion, but higher overhead involved in maintaining the database. An important issue here is to cost-effectively maintain imprecise system state information without degrading the performance of resource provisioning. In this section, we elaborate on various strategies for information representation and collection that address the cost-precision tradeoff.

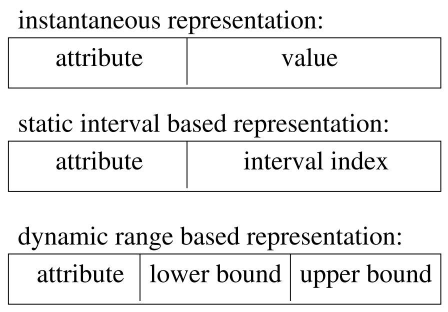
instantaneous representation:
attribute
value
static interval based representation:
attribute
interval index
dynamic range based representation:
attribute
lower bound
upper bound

Fig. 3. *Various Data Representations*

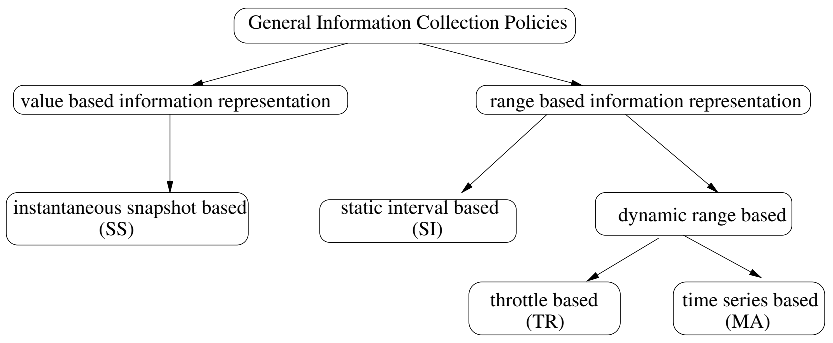
General Information Collection Policies
value based information representation
range based information representation
instantaneous snapshot based (SS)
static interval based (SI)
dynamic range based
throttle based (TR)
time series based (MA)

Fig. 4. *General Information Collection Policies Studied*

Information representation(Fig. 3) and collection strategies are often intertwined. Any parameter in the database can be represented using either a single instantaneous value (i.e. the last measured value) or by a range-based representation that approximates the value of a parameter by using an interval with an upper (U) and a lower(L) bound. The range size may remain static or change dynamically. Corresponding collection policies determine when and how often to sample the network components for current status information and whether to update the database with the collected sample. Sampling periods may be fixed or may vary over time. We classify policies for information collection as shown in Fig. 4.

System Snapshot Based Information Collection (SS): In this policy [5], system status information is stored in the database as an absolute value based on periodically sampling network and server nodes. The sampling frequency solely determines the accuracy of the information. To prevent information from being outdated, the sampling frequency should be very high; however, this would entail high overhead. Hence, a sampling period needs to balance the tradeoff between information freshness and information acquisition cost.

Static Interval Based Information Collection (SI): In this policy [6], we define a fixed interval B to partition the capacity of the collected information into a fixed number (say n) of equal size classes: $(0, B), (B, 2B), (2B, 3B), ..., ((n-2)B, (n-1)B)$. The residue capacity information is indicated by the corresponding class indices $0, 1, 2, ...(n-1)$. A probe is initiated at each sampling interval to obtain current information from the managed entities. If the obtained value is out of the range indicated by current index, the database is updated with another index, otherwise no update is needed.

Dynamic Range Based Information Collection: In this approach, the database holds the monitored information using a range with an upper bound U and a lower bound L. The central value of the range ($(U + L)/2$) is returned to represent current status. The range may be modified dynamically based on the sampled information.

The collection process follows the state transition shown in Fig. 5:Regular Probing(RP), Steadiness Confirmation(SC), Transient Noise Filtering(NF) and Range Adjusting(RA). Initially the system is in the RP state. At every sampling interval, the monitoring module sends out probes to collect current status information. If the sampled value falls within the current range, the system transitions to the SC state. In the SC state, the range size is tightened to provide higher accuracy if steadiness of state is confirmed; otherwise, the system goes back to the RP state and waits for more input. However, if the sampled value in the RP state falls outside the current range, the system goes to the NF state. In practice, the transition to the NF state can be caused by several events – measurement error, a transient burst or a significant load level change. In order to assist the underlying model to adapt to a confirmed change and filter out high frequency traffic components, the monitoring module sends N_p additional probes in period T_{probe}, where $T_{probe} = T/N_p$ with T as the original sampling period. If a significant change is confirmed, the system proceeds to the RA state to expand the range; otherwise, it goes back to the RP state.

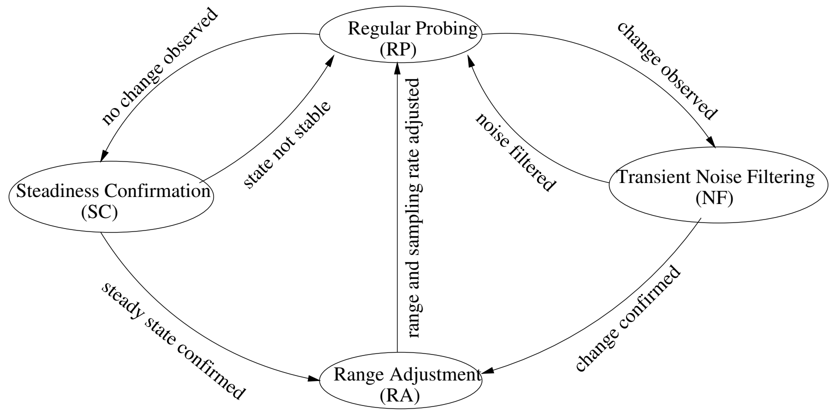

Fig. 5. *State Transition Diagram for Dynamic Range Based Information Collection*

We have explored two dynamic range based collection algorithms: (1) A Throttle based approach (TR) which uses periodic sampling and adjusts ranges by looking at the average values of recent historical samples; and (2) A time series (moving average) based policy (MA) [7], which applies a prediction model using moving average to adjust both range size and sampling period dynamically. The two policies differ in how and when range adjustment is performed. Our performance studies show that TR outperforms MA for QoS-based resource provisioning [8], so we restrict our discussions on dynamic range based approaches to TR. TR uses the average value of samples in the previous monitoring window to decide whether a range adjustment is needed. If deemed necessary, the range may be increased or decreased exponentially using a pre-specified throttle factor.

Range tightening in TR: In TR, a history table is updated with the sampled value $v(s)$ and its time stamp $t(s)$. The history table is searched in inverse time order to find out the time period tp during which the sampled values fall into current range. If $tp \geq T_{steady}$, where T_{steady} is a given threshold parameter, then we anticipate more steadiness for the link in the future and the current range R_a is tightened. To tighten the range, we first predict a future value v_n, averaged from samples taken within the specified sliding window T_{window}: $v_n = AVG(v(s_i))$, where $t(s_i) > t(current) - T_{window}$. Next, the range is tightened according to a predefined throttle factor TH_r to get a new range R_n: $R_n = R_a * TH_r$. Finally, the new range R_n is adjusted such that the average value v_n lies in the middle of the new range: $U(R_n) = min(v_n + 0.5 * R_a, capacity(link))$, $L(R_n) = max(v_n - 0.5 * R_a, 0)$. If $tp < T_{steady}$, the process goes back to the RP state and waits for more input.

Range relaxation in TR: To confirm significant changes, additional probes are sent out in the NF state. We average these additional sampled values to determine if it falls within the current range R_a. If yes, the noise is transient and can be ignored; otherwise, it is a significant change and the range R_a is relaxed as follows: $U(R_n) = max(v(s_i), U(R_a))$, $L(R_n) = min(v(s_i), L(R_a))$, where $i = 1, 2, .., N_r$ and $t(s_i) > t(current) - T_{window}$.

III. Customizing Information Collection for Mobile Applications

In mobile environments, an accurate picture of the underlying system must incorporate not only the information about network components and services, but also information about the mobile clients. To accommodate potential client mobility patterns, information collected may include intermittent connectivity patterns, location, residual energy level etc. QoS-based resource provisioning can be enhanced by adapting to these constantly changing factors. For instance, accurate information about residual energy level of mobile clients can be used to decide how the requested content should be delivered (e.g. burst sizes and schedules), information about client connectivity can be used to provide effective synchronization and caching. Similarly, accurate information about current and future locations of mobile clients can be exploited to provide reservations for smooth handoff. We are specifically interested in location information since this has implications on which links in the wired network will be used to serve the mobile user. In this section, we discuss different approaches to collecting location information for mobile applications, and then present techniques for improved system status collection using location information.

A. Approaches for Collecting Location Information

We explore two ways using which location information can be collected. Fine-grained approaches maintain current location of each individual mobile client [9], [10], while coarse-grained collection captures information at an aggregated level for multiple clients. In particular, we use the notion of *client aggregation*(i.e., mobile client population in each cell at a certain time instant) 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 [9], [10]: 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 [11]) are studied concurrently in order to better capture current user location.

Our strategy is to ensure resource availability for mobile applications by providing *reasonably* accurate system status information. We refrain from using fine-grained location information to determine overall resource availability. 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 we need to probe each mobile host separately and constantly. Hence, gathering individual user location does not benefit our overall goal of efficient system status collection.

However, the movement of a large number of users may lead to non-uniform distribution of mobile users across cells. We conjecture that using **coarse-grained mobility information** to adjust collection parameters will be sufficient to support cost-effective resource provisioning. In addition, collection using coarse-grained mobility information is independent of individual mobility models – this relieves us from the inaccuracies introduced in modeling or predicting individual user mobility. One measure of macro-level changes in mobile settings is the client aggregation status (i.e. number of users in a cell), this has the potential to significantly affect resource availability in the network/system [1], [2]. Client aggregation status can be obtained from cellular access points (i.e., base stations) that manage the communication of the mobile hosts residing within that cell. The 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.

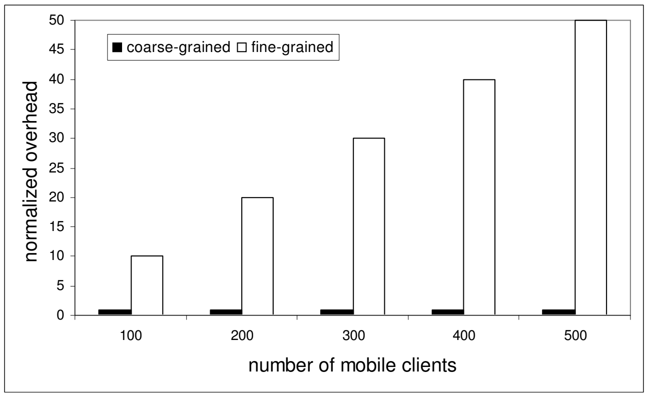

Fig. 6. *Overhead comparison of collecting coarse-grained vs. fine-grained information: The overhead is the number of probing and update messages normalized over the number of messages used for coarse-grained mobility information collection.*

Fig. 6 shows the results of a simple experiment that compares the overhead of collecting coarse-grained(aggregate) and fine-grained (individual) location information. Without loss of generality, we applied the system snapshot based collection policy to gather mobility information. We set the sampling period to be 1 second and count the number of total sampling messages as the metric for collection overhead. In our experiment, the coverage area consists of 10 cells. In order to obtain coarse-grained mobility information (i.e number of users in a cell), we probe each base station every second. Since the number of base stations remain unchanged irrespective of the changes in the number of mobile hosts, the sampling cost involved remains 10 per second; however, in the case of collecting fine-grained mobility information, we need to sample all the mobile hosts and the number of sampling messages increases linearly as the number of mobile hosts increases.

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 models. Prediction of aggregate status requires some knowledge about the distribution of mobile hosts and their mobility patterns in a region. For example, the incremental mobility model [11] is often used to capture the mobility pattern of users in a metropolitan area where the mobility of most users is restricted by the layout of streets and freeways. In this model, mobile hosts are distributed and can move randomly in a closed coverage area with the size of (X_{max}, Y_{max}). The location of the mobile host (x, y) and its velocity vector (both speed and direction) incrementally change as time progresses. Change of the location is determined based on previous velocity and direction; change of velocity is uniformly distributed between the minimum and maximum acceleration/deceleration of the host.

we now describe the derivation of aggregate mobility from the incremental individual mobility model, similar derivation can be obtained for other mobility models. The closed coverage area is divided into a number of non-overlapping equal sized regions with the size (X_{region}, Y_{region}). Each region has a collection point (e.g. base station) that serves as the wired network access point for all the mobile hosts in its controlled region. If we let $X_{dim} = \lceil \frac{X_{max}}{X_{region}} \rceil$ and $Y_{dim} = \lceil \frac{Y_{max}}{Y_{region}} \rceil$, then there are $N_{region} = X_{dim} \cdot Y_{dim}$ regions in the area. Mobile hosts either move or stop in this area, the population/aggregation in each region changes all the time and can be captured. At a certain time t, a mobile host located at $(x(t), y(t))$ is in region $\lfloor \frac{x(t)}{X_{region}} \rfloor + X_{dim} \cdot \lfloor \frac{y(t)}{Y_{region}} \rfloor$. At time t, the aggregation of region i $(A_i(t))$ is the number of mobile hosts located in region i, i.e., the cardinality of the set of the mobile hosts who are in this region.

$$A_i(t) = \|\{j | \lfloor \frac{x_j(t)}{X_{region}} \rfloor = i \quad mod \quad X_{dim}, \lfloor \frac{y_j(t)}{Y_{region}} \rfloor = \frac{i}{X_{dim}} \}\|. \tag{1}$$

Once the coarse location information is obtained (either from base stations or via model-based predictions), we would like to utilize it to enhance system status collection.

B. Mobility-aware System Status Information Collection

Given that sampling frequency and range size are the two most important factors that impact the efficiency of the collection process, in this subsection, we present a family of collection strategies(Fig. 7) that uses client aggregation status to drive the adjustment of sampling frequency and range size. These policies are not only independent of the individual mobility model adopted, but are also expected to provide reasonable levels of data accuracy with significantly less overhead, enabling efficient QoS support. since our empirical results (presented later) show that dynamic range based collection policies outperform the instantaneous value based or static interval based approaches, we restrict our discussion to customizations for the dynamic range based policy.

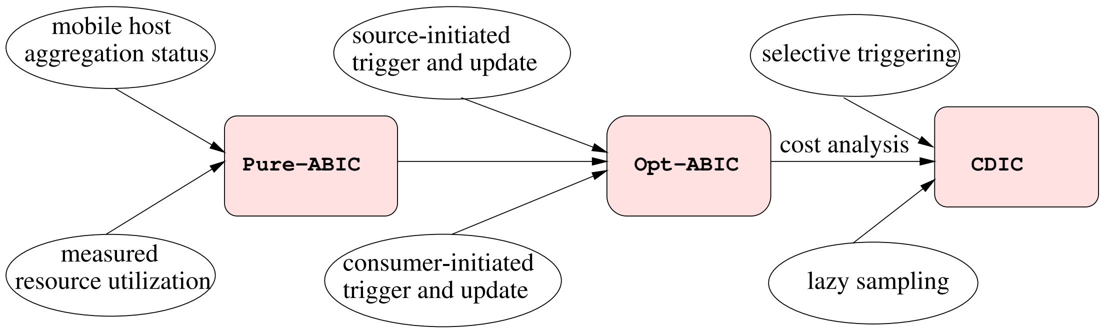
mobile host aggregation status
measured resource utilization
Pure-ABIC
source-initiated trigger and update
consumer-initiated trigger and update
Opt-ABIC
cost analysis
selective triggering
lazy sampling
CDIC

Fig. 7. *Evolution of information collection policies for mobile applications*

We consider all the components in the system that provide information of interest to users as *information sources*; while all the components that use the available information to make corresponding decisions are referred to as *information consumers*. The consumers can be middleware services such as QoS-based resource provisioning algorithms, or user applications such as location-aware information delivery. Consumers may have varying data accuracy needs that are different from database values, which will cause *consumer-initiated triggers and updates*; Similarly, information sources may be intelligent enough to notify the collection process of sudden changes in their states, which will cause *source-initiated triggers and updates*.

Fig. 7 illustrates the various aggregation based information collection techniques proposed in this paper. In the Pure Aggregation based (Pure-ABIC) approach, the collection process adjusts sampling frequency and range size based on mobile host aggregation status and current resource utilization. The Opt-ABIC [12] approach extends the Pure-ABIC approach to accommodate heterogeneity of both information source and information consumer. In addition, we propose a Cost Driven Aggregation based (CDIC) approach [13] where collection cost is analyzed to derive an optimal range size, and to explore the conditions under which maintenance overheads may be further reduced.

Pure Aggregation Based Information Collection (Pure-ABIC): Pure-ABIC utilizes the combination of mobile host aggregation status and resource (server/link) utilization to dynamically adjust the collection parameters such as sampling frequency SF and range size R. It is a relatively simple policy that assumes uniform consumer needs and homogeneous source behavior. We classify aggregation and resource utilization levels into three categories - high, medium and low. The threshold utilization values that demarcate the boundaries between high, medium and low are implementation-specific and dependent on the traffic characteristics and system capacities. It is difficult to determine an optimal threshold since system conditions (such as resource availability, traffic patterns) are varying dynamically. In our simulation, we consider the utilization level higher than 0.9 as 'High', and that lower than 0.4 as 'Low'. These parameters are chosen based on our previous experimental studies [13], where we observe that when utilization level goes above 0.9 or below 0.4, the system is more sensitive to sampling frequency and information granularity. In the case of AL (Aggregation Level), assuming that N_{mh} is the total number of mobile hosts and N_{region} is the total number of regions, we use $\frac{N_{mh}}{N_{region}}$ and $\frac{N_{mh}}{8}$ as two boundary values. The choice of these parameters is obtained by observing the traffic variation and mobility patterns under the chosen mobility model in the simulation. Using different parameter settings and/or other mobility models could yield different threshold values.

When resource utilization is *high* or *low*, the aggregation levels have little impact, since most of the requests are either rejected or accepted. The choice of a small SF and a small R is desirable. When resource utilization is *medium*, aggregation status comes into play: we increase SF and decrease R when aggregation level increases. In other words, $SF(t) = min\{(1 + \alpha) * SF(t - 1), SF_{max}\}$ and $R(t) = min\{(1 + \beta) * R(t - 1), R_{max}\}$, where α is set to be $\alpha_H = 0.2$, $\alpha_M = 0.4$ and $\alpha_L = 0.8$ respectively for high, medium and low aggregation level. Similarly, β is set to be $\beta_H = 0.8$, $\beta_M = 0.4$ and $\beta_L = 0.2$ respectively for high, medium and low aggregation level.

Optimized Aggregation Based Information Collection (Opt-ABIC): In Pure-ABIC, the adjustment of information collection parameters is performed by a local collection module. This localized decision making reduces communication overhead among different components in the system; however, when information sources and consumers participate in determining collection parameters, the system is expected accommodate to unexpected changes in system load and consumer specific accuracy needs. The design of a more generalized Aggregation Based Information Collection (Opt-ABIC) grew out of this observation.

In addition to coarse-grained adjustment of collection parameters driven by aggregate mobility status as in Pure-ABIC, Opt-ABIC (Fig. 8) utilizes feedback from the sources and the consumers for further customization of the collection process. Every source maintains the current exact value and the approximation is held in the database. There are three reasons that may cause the exact source value to deviate from the stored interval: (1) resource reservation; (2) resource release; and (3) changes in application load. Application load is dynamic and is often caused by unmediated requests (those without QoS needs) that are accepted or completed. These best-effort requests are not accounted for by the resource provisioning process since admission control is not required. When the current value changes unexpectedly and falls outside of the current range, a *source-initiated trigger* is issued to the collection module for potential range size relaxation. Every so often, admission decisions cannot be made based on the approximate information in the database. In this case, the resource provisioning process (which is the information consumer in our case) may request data at a certain level of accuracy, generating a *consumer-initiated trigger*. The collection process responds to these two types of triggers by determining if the range size should be adjusted. The adjustment process and subsequent collection are similar to the dynamic range based collection approach discussed in the previous section.

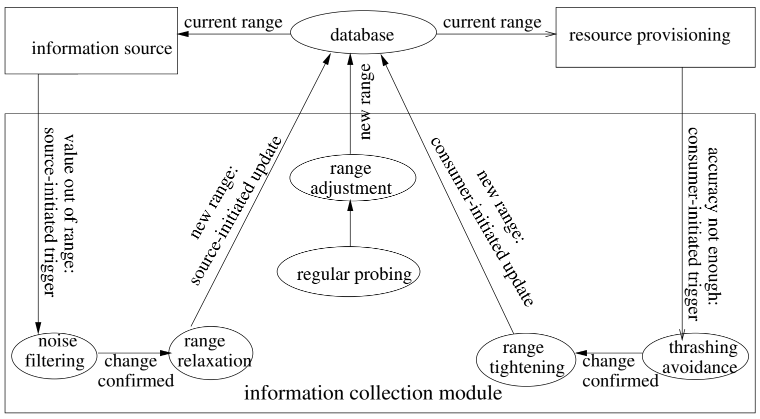

Fig. 8. *State diagram of Opt-ABIC*

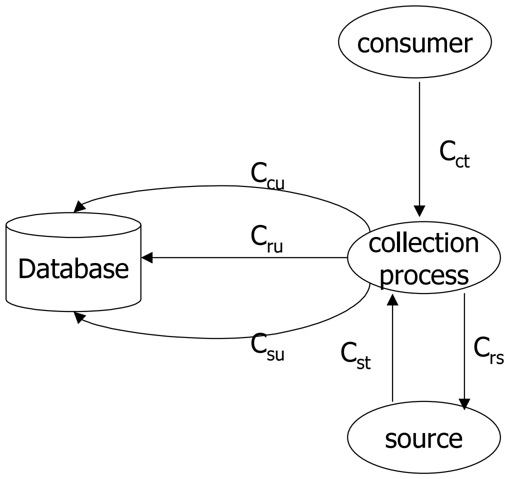

Fig. 9. *Cost factors in Opt-ABIC*

Cost Driven Aggregation Based Information Collection (CDIC): In this approach, we analyze the cost factors involved in a generic information collection process and incorporate the cost measures into Opt-ABIC. Careful study of a generic information collection process reveals that the communication cost involved (Fig. 9)consists of:

• regular sampling overhead C_{rs}: This is introduced by probing messages issued by the resource provisioning module to sources;

• regular update overhead C_{ru}: This is introduced by update messages which report current source value to the resource provisioning module and update the database with the new value.

• source/consumer-initiated trigger overhead C_{st} and C_{ct}: This is introduced by messages from the sources/consumers to the resource provisioning module that can potentially cause range expansion and tightening.

• source/consumer-initiated database update overhead C_{su} and C_{cu}: This is introduced by the messages sent by resource provisioning module to update database with the new range.

Overhead incurred by regular sampling and update (C_{rs} and C_{ru}) is dependent upon the sampling frequency, which is driven by mobile host aggregation status and actual resource utilization as shown in previous section. Therefore, to minimize the overall communication cost, we need to minimize the overhead introduced by source/consumer-initiated trigger/update. We observe that both types of triggers/updates (source- and consumer-initiated) provide an opportunity for the collection module to adjust the range size in the database. Our objective in selecting a good range size is to avoid the need for future updates and consequently minimize the communication costs. To avoid source-initiated triggers and updates, the range size should be large enough to incorporate changes in source values. On the other hand,to avoid consumer-initiated triggers and updates, the range size should be small enough to satisfy consumer accuracy constraints. Hence, it is not obvious how to choose a range size that minimizes the possibility of both source and consumer-initiated triggers and updates.

In Opt-ABIC, one source/consumer-initiated trigger does not necessarily cause one source/consumer-initiated update since a significant change needs to be confirmed. Unlike Opt-ABIC, CDIC triggers always result in updates. Our objective here is to minimize the weighted sum of consumer-initiated cost ($C_{cc} = C_{ct} + C_{cu}$) and source-initiated cost ($C_{sc} = C_{st} + C_{su}$). Assuming that P_{cc}, P_{sc} represent the frequency of consumer-initiated triggers/updates and source-initiated triggers/updates, we can rewrite the cost formula as $\Omega = C_{cc} \cdot P_{cc} + C_{sc} \cdot P_{sc}$. Intuitively, a smaller range size would lead to more frequent source-initiated triggers/updates, and less consumer-initiated triggers/updates. Therefore, the range size needs to be adjusted carefully in order to minimize the overall overhead. Previous studies [14] have shown that in approximate caching, for a given R, we have $P_{sc} = K_{sc}/R^2$, $P_{cc} = K_{cc} \cdot R$, where K_{sc}, K_{cc} are parameters that depend on the nature of the source and the frequency of consumer requests and the distribution of consumer requirements. This result can be applied in our context here. Hence, we can re-write the cost expression as $\Omega = C_{sc} \cdot K_{sc}/R^2 + C_{cc} \cdot K_{cc} \cdot R$.

Finding the value for R that minimizes this expression can be reduced to finding the root of the derivative. Using this approach, the optimal value for R is $(2 \cdot \frac{C_{sc}}{C_{cc}} \cdot \frac{K_{sc}}{K_{cc}})^{1/3}$. Even though consumer-initiated cost C_{cc} and source-initiated cost C_{sc} vary depending on system conditions, we can still reasonably assume the ratio $\frac{C_{sc}}{C_{cc}}$ to be constant. This implies that there is no need to choose values for each type of cost. For example, consumer-initiated cost is 2 messages (i.e., 1 message from information consumer to the collection process; and 1 message from the collection process to the database). Similarly, source-initiated cost is also 2 messages. Hence, $\frac{C_{sc}}{C_{cc}} = 1$.

Setting the optimal range size based on the cost analysis assume that source behavior and application workloads remain stable. In practice, parameter values and application needs may change dynamically at run-time. The overhead of monitoring these factors at run-time in order to set K_{st}, K_{su}, K_{ct} and K_{cu} appropriately affects the system performance significantly. Therefore, we apply the following strategies to further decrease the communication cost without sacrificing overall QoS:

Selective Triggering: The triggering process can be significantly optimized to suit the needs of the information consumer (the resource provisioning process in our case). The approach described below is especially suitable in mixed wired/wireless networks. Consumer initiated triggers are used when the consumer determines that the supplied data is not accurate enough. Such consumer initiated triggers are impractical in the case of link related information due to the following reasons: (a) QoS-based routing techniques select a path based on aggregate link characteristics, hence it is difficult to determine whether more accurate information is needed for links on an individual basis; and (b) since the number of links can be large, customizing the collection process on a per-link basis is not a scalable proposition.

Furthermore, we observe that consumer initiated triggers may be necessary only when the required resources for a request closely match the available resources (if there are ample available resources, the request is likely to be accepted anyway). In such a case, we can assume that the server or the network is overloaded [1]; at this time, there is a high likelihood that incoming requests are rejected; hence maintaining accurate database information will have little use. For the above reasons, we expect that turning off consumer-initiated triggers and updates would not degrade QoS-provisioning performance greatly, but the collection overhead can be significantly decreased.

Lazy Sampling: Once consumer-initiated triggers and updates are eliminated, the database modifications can only be driven by the regular probing process or by source-initiated triggers. To further reduce communication overhead, we apply lazy sampling strategies that can decrease the probing cost even more. The sampling frequency can be reduced in two cases: (a) If the number of source-initiated triggers in a given period is less than a predetermined value, we can infer that the current approximation of the source value is reasonable (i.e., the source value does not change very dramatically), therefore we can reduce the sampling frequency; (b)if the range is relaxed to exceed a certain value ($1/4$ of the maxima as indicated in our experiments), it is likely that the range is large enough to accommodate reasonable

[1]Typical request sizes and system configurations are such that an incoming request will occupy only a small fraction of the total system capacity (most servers and networks are designed to deal with hundreds of requests.)

changes in the source value, hence we can reduce the sampling frequency. In both cases, we use the following equation to reduce the sampling frequency: $SF_{new} = \frac{SF_{old}}{1+\alpha}$.

A more focused study of this approach as applied to network management appears elsewhere [13].

IV. Performance Evaluation

Information collection is an underlying support service, so its effectiveness is reflected by improvements in overall system performance. In this paper, we use QoS-based resource provisioning as a sample information consumer and evaluate the impact of the proposed information collection policies on QoS-aware mobile applications. The interaction between the information collection service and resource provisioning process is as follows. Resource provisioning policies consider current residual network and server resources as well as mobile client aggregation status to ensure better overall performance. Given the server and/or path assignment, the client proceeds to set up a connection along the assigned network path from the base station where the mobile host resides to the server. During the connection setup, the routers and the servers along the path check their residual capacity and either admit the connection by reserving resources or reject the request. When a client moves out of current region/cell, another path/server that better serves the client is found and allocated accordingly. When the connection terminates, the client sends termination requests and resources are reclaimed along the connection path.

Resource Provisioning Policies Studied: We evaluate the following three representative resource provisioning policies that cover a range of mobile applications in composition with various information collection techniques: (a) server selection (load based); (b) server selection (proximity based); and (c) combined path and server selection (CPSS).

Server Selection(SVRS): As distributed information services like the World Wide Web become increasingly popular on the Internet, scalability becomes a very serious issue. Performance degradation can be caused by excessive server load due to the request of popular documents, wasted network bandwidth due to redundant document transfer, and excessive latency in delivering documents to the client due to the potential for transfers over slow paths. A promising technique that addresses all of these problems is service/document replication where the same data/service is partially replicated across distributed servers. However, when a service is replicated, clients must be able to discover an optimal replica. Server selection schemes attempt to choose the best replica/server for a given request. Although requests from all the mobile clients within the same cell go through the same base station, different servers may be chosen for different mobile clients for improved load balancing. We apply the following two server selection policies for evaluation.

• **Least Utilization Factor Policy(SVRS-UF)**: In this policy, the capacity of a server can be specified as four parameters: CPU cycles, memory buffers, disk bandwidth as shown in [15]. In determining the server utilization factor, the bottleneck resource (i.e. one that is impacted the most) is conservatively used to estimate the impact of the request on a server. This policy chooses the server with the minimal utilization factor. If there is a tie between the servers, the policy randomly picks one.
• **Shortest Hop Policy(SVRS-HOP)**: This policy chooses the nearest server to the mobile host in terms of the number of hops. If there is a tie between the servers, the policy chooses the least loaded one as in SVRS-UF described above.

Combined Path and Server Selection(CPSS): Server selection algorithms discussed above 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 typically aims to select the optimal path between a source-server pair and ignores the situation where there may be multiple servers that can serve the same request. CPSS [16] is an integrated approach which allow 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. We have developed extensions for CPSS that account for mobile user aggregation status in a cell and adapt well to request locality (i.e., users in a region may request similar information). Previous work has studied a family of CPSS policies, in this paper we choose the policy which selects the server and links that maximize the overall utilization of resources.

We now describe the simulation setting used for performance studies in this paper, and then present the extensive simulation results and analysis, whose objective is to determine the most appropriate information collection policy for a given resource provisioning mechanism based on current system state and application workload.

A. Simulation Environment

Our simulator is a message driven multi-threaded system intended to study the dynamics of QoS sensitive network environments. We use the cellular architecture in which each cell has one base station as its access point to wired networks (Fig. 10). 800 mobile hosts are distributed in 20 cells. The movement based approach [17] is used for mobile hosts to perform location update. In other words, we count the number of cell boundary crossings for each mobile user and location is updated when this counter exceeds a predefined threshold (5 in our simulation). Note that our aggregation mobility based approach is in fact independent of underlying location management scheme used, since it does not need the knowledge of each mobile host location at any time instant. Therefore, we resort to a simple yet relatively reasonable scheme, movement-based updates, for our simulation purpose. The choice of 5 as the threshold for cell boundary crossing is based on our observation of traffic variation and mobility patterns under the chosen incremental mobility model and corresponding parameters in our experiments. In addition, ideally, QoS-based resource provisioning in mobile environments integrates appropriate paging schemes to determine the location of mobile hosts in advance, and incorporates predictive resource reservation. However, the focus of this paper is on information collection service itself instead of novel resource provisioning approaches, hence, we abstracted out paging schemes. The location management is purely triggered by location update requests from mobile hosts. Furthermore, we model two levels of mobility: high mobility and low mobility. Specifically, we set maximal velocity to be 65 mph, maximal acceleration/deceleration to be $0.9meter/sec^2$, maximal direction change per unit time to be $0.1745radian/sec$. With high mobility, the velocity change is twice that of low mobility.

To better emulate the real network, the capacities of network links are selected from various widely used link types from 1.5Mbps to 155Mbps, with the mean value being 64Mbps. When defining the capacity of the server nodes, we calibrate CPU units using a basic 64Kbit voice processing application, memory units to be 64Kbytes, and disk bandwidth units to be 8Kbytes/s. The server capacities are also selected from popular models of multimedia servers and web servers, with the mean CPU, memory, disk bandwidth to be 1845, 6871 and 5770 calibrated units respectively.

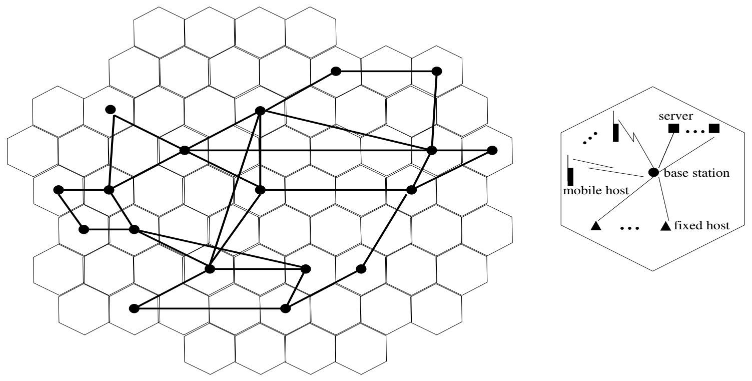

Fig. 10. *Simulation Topology: The base stations are connected as in a typical MCI ISP network. There exist a number of mobile devices, fixed hosts and servers in each cell.*

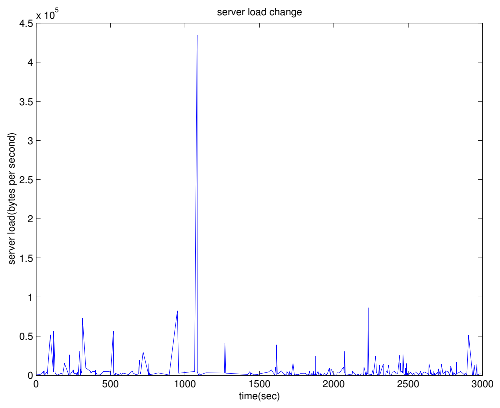

Fig. 11. *Server load Change Pattern: It is a day's worth of all HTTP requests to the EPA WWW server located at Research Triangle Park, NC.*

We model request arrival at the source nodes as a Poisson distribution, and the request holding time is exponentially distributed with a pre-specified average value. The request holding time is the time for which the requested network and server resources such as link bandwidth, CPU, buffer, disk bandwidth etc. are reserved. We pre-define a set of client request templates to capture typical multimedia connection request patterns in terms of network bandwidth, CPU, memory, disk bandwidth and end-to-end delay. For each request generated, the requested parameters are randomly selected from the set of request templates, with the mean requested bandwidth being 2.5Mbps, mean end-to-end-delay being 400ms and CPU, memory and disk bandwidth being 150, 374 and 271 calibrated units respectively.

Network traffic is classified into two categories: non-uniform traffic (NUT) and uniform traffic (UT). To represent non-uniform traffic, we designate some sets of candidate destinations as being “hot”, (i.e. serving popular videos, web sites etc), and they are selected by the clients more frequently than others. To reduce the effect of local and nearby requests, we choose three pairs of source-destination sets from the topology. The requests arrive at these hot pairs, as foreground traffic, at a higher rate than other background traffic. In our non-uniform traffic pattern, we set the foreground arrival rate to be 5 times higher than the background rate, and in uniform traffic pattern, we set them to be equal to each other. Specifically we set the foreground arrival rate to be 10 seconds, and the background rate to 50 seconds.

app. workload	*HM-NUT*	*LM-NUT*	*HM-UT*	*LM-UT*
pure QoS requests	M1	M3	M5	M7
mixed requests	M2	M4	M6	M8

TABLE I

Scenarios Studied

Application workload can be classified into two categories based on the types of requests sent to network: (a) pure QoS requests: Servers such as VoD servers and computation bounded servers are designed for a specific purpose and thus only accept requests with QoS requirements, therefore, all the changes on the server load are exposed to resource provisioning; (b) mixed requests: Servers like distance learning servers are general-purpose, i.e., they accept both QoS-based requests and best-effort requests. Generally there is no admission control for best-effort requests, which makes the server load change without the notice of the resource provisioning process. For this simulation, we use the trace [18] to represent the server load change (Fig. 11). Table I depicts all the different scenarios we studied in this paper.

B. Performance Results

The following three metrics are used to compare the performance of resource provisioning mechanism:

– *request success ratio*: it is the ratio of the number of successful requests over the number of all requests. In traditional networks where no mobile clients exist, a request is considered successful if resources along the chosen path and server are successfully reserved for it. In mobile environments, a *successful request* is a *completed* request. Since mobile hosts constantly move, dealing with request handoff is unavoidable. Incoming requests go through an admission control process where they may be accepted or rejected. An admitted request executing on a mobile host will require re-provisioning of resources (network links, servers etc.) as it migrates through different regions. The intermediate re-provisioning may or may not be successful. Admitted requests may not complete due to several reasons: there is no route with sufficient resources (path failure); locating mobile hosts fails (location failure); the alternate re-scheduling server may not have sufficient resources if path to original server is not available.

– *information collection overhead*: Basically it is the total communication overhead involved in the whole collection process, including the cost incurred by sampling information sources and updating databases. Note that the collection overhead consists of both system status collection overhead and mobility related information collection overhead. when fine-grained mobility information is maintained, the cost involves all the overhead of communicating with each individual mobile host; on the other hand, with coarse-grained mobility, the cost refers to the overhead introduced by maintaining region or cell based mobility. For simplicity, we assume the collection overhead is proportional to the number of messages exchanged among different system components. i.e., it is the sum of the number of samplings, the number of database updates, and the number of triggers from both sources and consumers.

– *overall performance efficiency*: It is a metric defined as the ratio of the number of successful requests to the information collection overhead.

1) Basic Performance Results: In the previous sections, we presented three different approaches to system status information collection: mobility incognizant collection, collection using fine-grained mobility information and using coarse-grained mobility information. We now compare the performance of resource provisioning under these approaches. We use the Throttle based (TR) algorithm as the representative for mobility incognizant collection policies, since it can be easily extended and customized to incorporate mobility information. We use Pure-ABIC to represent the approaches that use coarse-grained mobility information.

From Fig. 12, we observe that the completion ratio of CPSS 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, which in turn exhibits the lowest overall efficiency. This

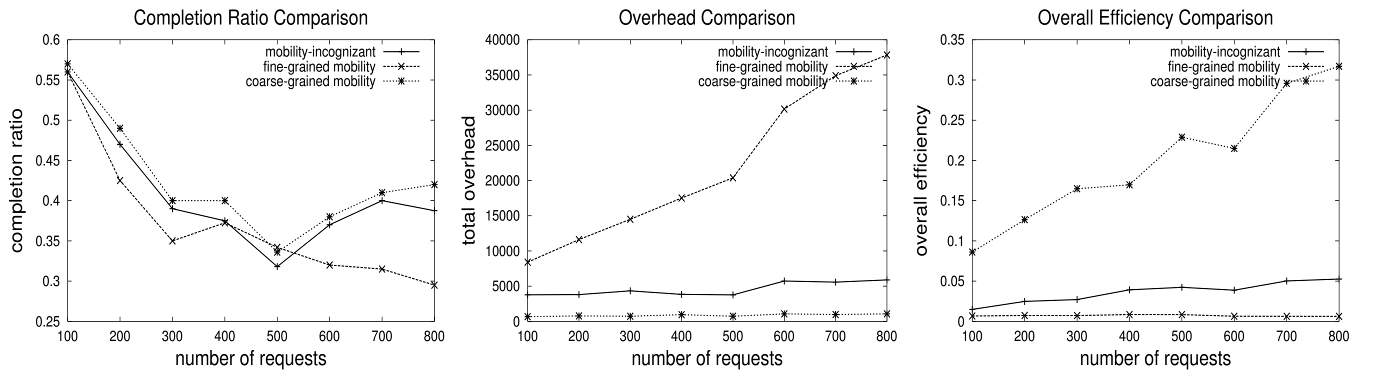

Fig. 12. *Basic performance comparison*

result remains the same with SVRS-UF or SVRS-HOPS under other scenarios.

2) Performance Comparison of Underlying Collection Policies: The system status collection policies (SS,SI,TR) presented in Section II are the underlying policies based on which the coarse-grained location information driven system status collection approach is customized. We conducted a series of experiments to determine the best combination of a collection policy (SS, SI, TR) and a resource provisioning policy(SVRS-UF, SVRS-HOP, CPSS) under varying application workload. For fair comparison, we compare these combinations using the parameter settings that exhibit the best performance for each of them. Performance studies indicate that the best overall efficiency is achieved by using 10 seconds as the sampling period for snapshot based(SS), static interval based (SI) and throttle based (TR) collection policies. In our experiments, we observed that the performance comparison of different policy combinations under varying scenarios is very similar, so we only show the results of case M1 (high mobility, non-uniform traffic, pure QoS requests) here. Fig. 13 shows the performance of SVRS-UF with the three information

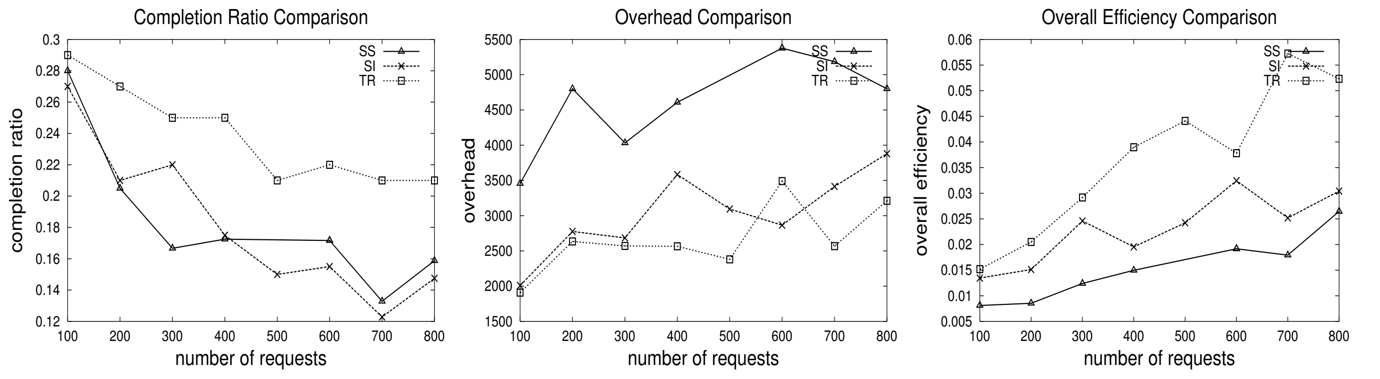

Fig. 13. *Performance of SVRS-UF with different underlying collection policies under Case M1 (High Mobility, Nonuniform Traffic, Pure QoS Requests)*

collection policies under case M1. The completion ratios do not show much difference. However, it can be seen that SS introduces the highest overhead, and TR incurs the lowest overhead. In terms of overall efficiency, TR outperforms the other policies.

We study the performance of CPSS under different information collection policies and scenarios. Fig. 14 shows the performance of CPSS with the three information collection policies under case M1. The SS,

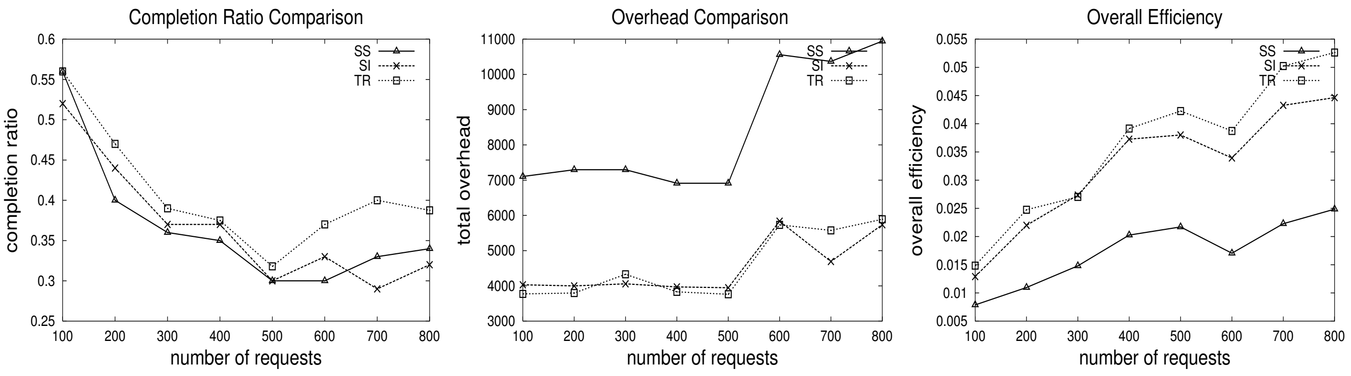

Fig. 14. *Performance of CPSS with different underlying collection policies under Case M1 (High Mobility, Nonuniform Traffic, Pure QoS Requests)*

SI and TR algorithms exhibit similar completion ratio to each other. In general, the overhead of all the strategies increases with increase of the number of requests because more dynamic change is expected with more requests and the collection process performs more monitoring. While SI and TR incur similar amount of overhead, as expected, the SS algorithm introduces far more overhead than the other three due to frequent sampling and database updates containing exact values. Hence, SS exhibits the lowest overall efficiency among all the three algorithms. The efficiencies of the dynamic range based algorithm and static range based algorithm are very close to each other. The reason is that in mobile environments, we must account for host mobility in addition to network and link changes. Not all requests admitted can be completed and the range change may not keep up with the movement of mobile hosts, hence changing the range size does not always exhibit better results.

Among the three mobility-incognizant policies we studied here, TR outperforms the others in terms of overall efficiency, therefore, we use TR as underlying policy to support integration with coarse-grained mobility information.

3) Impact of Integrating Coarse-grained Mobility Information: In Section III, we presented three different ways to customize TR to take into account coarse-grained mobility information. We now use experiments to determine the best combination of a coarse-grained mobility information driven collection policy (Pure-ABIC, Opt-ABIC, CDIC) and a resource provisioning policy(SVRS-UF, SVRS-HOP, CPSS) under varying application workload.

Fig. 15 shows the performance of SVRS-UF with the three information collection policies under case

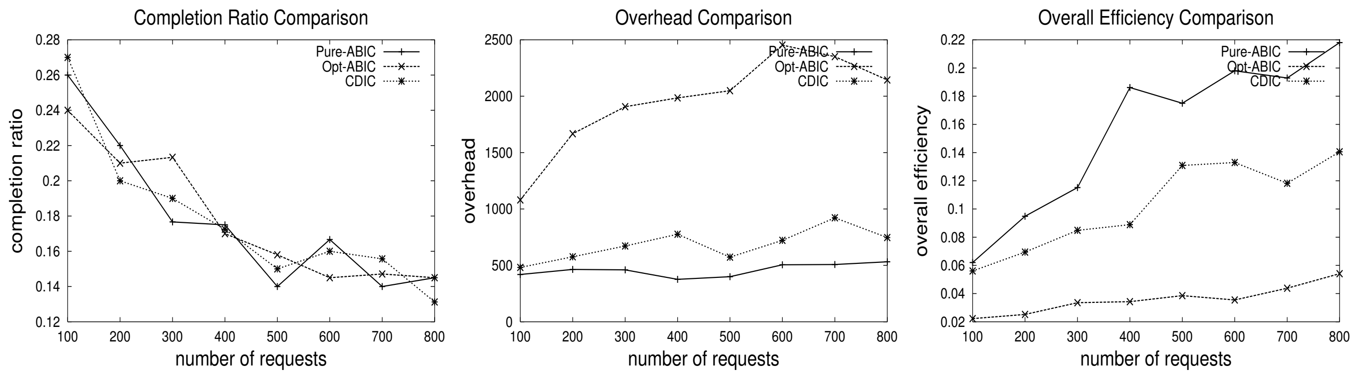

Fig. 15. *Performance of SVRS-UF with different coarse-grained mobility information driven collection policies under Case M1 (High Mobility, Nonuniform Traffic, Pure QoS Requests): Pure-ABIC shows similar completion ratios to the other policies, but performs the best overall due to its lowest overhead.*

M1. The completion ratios of the three policies are similar to each other. CDIC and Pure-ABIC have much lower overhead than the others which results in highest overall efficiency. Overall, Pure-ABIC performs the best for SVRS-UF, CDIC also performs comparably. CDIC has the advantage that it gives more flexibility to information sources and consumers to control the collection process.

We study the performance of CPSS under different information collection policies and scenarios. Fig. 16 shows the performance of CPSS with the three information collection policies under case M1. In general, the completion ratio of the three policies is very close and they follow a similar trend: with an increase of the number of requests in the system, the completion ratio decreases. As the system saturates (the number of requests reaches 500 in this case), the completion ratio reaches a minimum and then rises gradually due to completing requests.

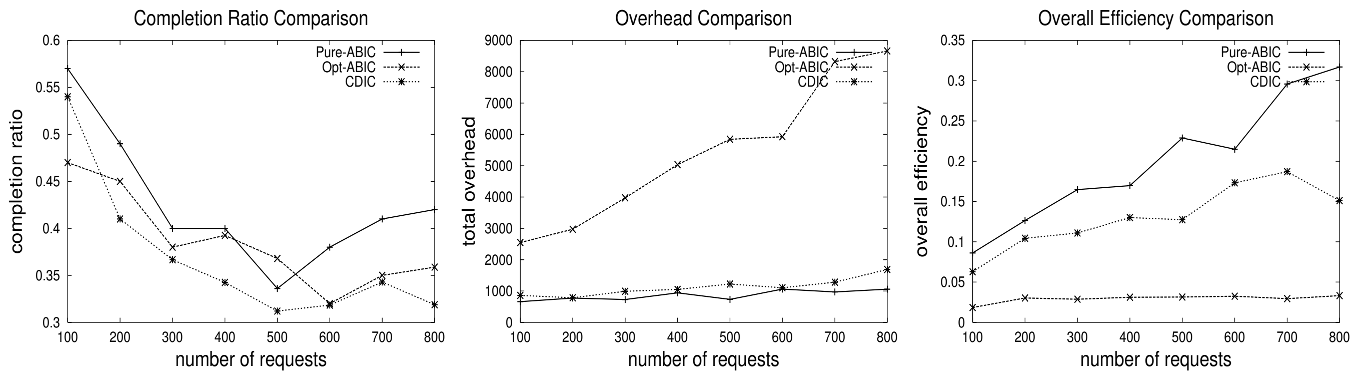

Fig. 16. *Performance of CPSS with different coarse-grained mobility information driven collection policies under Case M1 (High Mobility, Nonuniform Traffic, Pure QoS Requests): Pure-ABIC exhibits the best overall efficiency which is followed by CDIC.*

The overheads of Pure-ABIC and CDIC are low and close to each other. The reason for the low overhead of Pure-ABIC is that coarsely selecting collection parameter such as sampling frequency and range size based on the aggregate mobility status and resource utilization status is good enough to render satisfactory completion ratio; adding source- and consumer-initiated triggers and updates just make the collection process more complex and adds more overhead without increasing the completion ratio. Overall, Pure-ABIC shows the highest efficiency, which is followed by CDIC; Opt-ABIC and CDIC exhibits much lower overall efficiency.

In summary, if balancing the tradeoff between system performance and the overhead is the goal, then Pure-ABIC is the most desirable strategy collecting network and server information in mobile environments; however, in scenarios where resources are very limited, CDIC is a good candidate because of its very low overhead; Opt-ABIC can always be used when more flexibility needs to be given to the information sources and consumers.

C. Deriving Automatic Service Composition Rules

Our evaluation indicates that both the accuracy and overhead of information collection policies have a significant impact on the performance of resource provisioning process. Although snapshot based collection can obtain very accurate information, the huge overhead introduced by frequent sampling and database updates makes it a bad choice. In contrast, the throttle based algorithm turns out to be the best choice among the mobility-incognizant collection algorithms. Although seemingly naive and ad-hoc, it adapts very well to the constantly changing environment. We further observe that the family of aggregation based information collection policies works uniformly better than the mobility-incognizant policies. This finding shows the effectiveness of utilizing mobility status in designing collection policies.

In summary, Pure-ABIC works the best for both CPSS and Server Selection under all the scenarios studied if overall system efficiency is the major concern. Client aggregation information is beneficial for collecting not only link resource availability status but also server resource usage status. This is because when user density increases in a certain region, potentially they more frequently access the servers in that same region, which leads to more variation in server resource status. In addition, CDIC outperforms the other policies if maintenance overhead is the major metric. For example, users issuing computation-bounded requests tend to remain in a certain region for a while , i.e., they do not constantly change their locations. In this case, request completion ratio is relatively stable and overhead introduced by collecting the information becomes the major concern, hence CDIC should be used.

Using the results from empirical study, we derive general heuristics to select appropriate composition of information collection and resource provisioning policies for an incoming request. We classify the client requests into: (a) web requests that include normal HTTP, FTP or TELNET sessions that do not present stringent and explicit QoS requirements; (b) multimedia requests that include requests for audio/video streaming or for rendering 3D graphics/images; and (c) computation requests that include complex scientific computations and usually CPU bounded in nature. We also classify servers into four types: (a) web servers that support web requests and also multimedia requests; (b) multimedia servers that deal with multimedia requests; (c) computation servers that support high speed computations; and (d) general purpose servers that support all the different types of servers.

	server types			
request types	web	multimedia	computation	general-purpose
web req.	n/a			
multimedia req.	*CPSS+Pure-ABIC*	*CPSS+Pure-ABIC*	n/a	*CPSS+Pure-ABIC*
computation req.	n/a		*SVRS+CDIC*	*SVRS+CDIC*

TABLE II

Sample Service Component Selection Rules

With web requests, there is no need to reserve either server or path resources, so we do not provide any resource provisioning or information collection policies for them. For multimedia requests, CPSS should be used to find the best path to the best server since requested items are often replicated on different servers, and there are a number of paths available between the user and the servers. However computation-bounded requests only need to reserve resources at the server since relatively smaller amount of information is transmitted along the path. Performance results indicate that given the specific resource provisioning policy (SVRS-UF, SVRS-HOP or CPSS), there exists a particular information collection policy that works the best for it under varying network/system conditions. This shows that the adaptations of collection parameters in information collection policies are sufficient, and policy switching among collection policies is not necessary. We propose the service component selection rules illustrated in Table II.

V. AutoSeC: Dynamic Service Broker Framework

The AutoSeC (Automatic Service Composition) framework provides a user-transparent middleware infrastructure that supports automatic and adaptive selection of policies for managing dynamic distributed environments. In particular, it supports selection of optimal combinations of information collection and resource provisioning services, thus sheltering the applications from the complexity of decision making by hiding the diversity of the underlying provisioning services. *AutoSeC* has five components (Fig. 17): Information Source, Information Repository, Resource Provisioning, Information Mediator and Dynamic Service Broker. We describe these components in detail below.

The *Information Source* corresponds to different components in the system, such as the server, link,

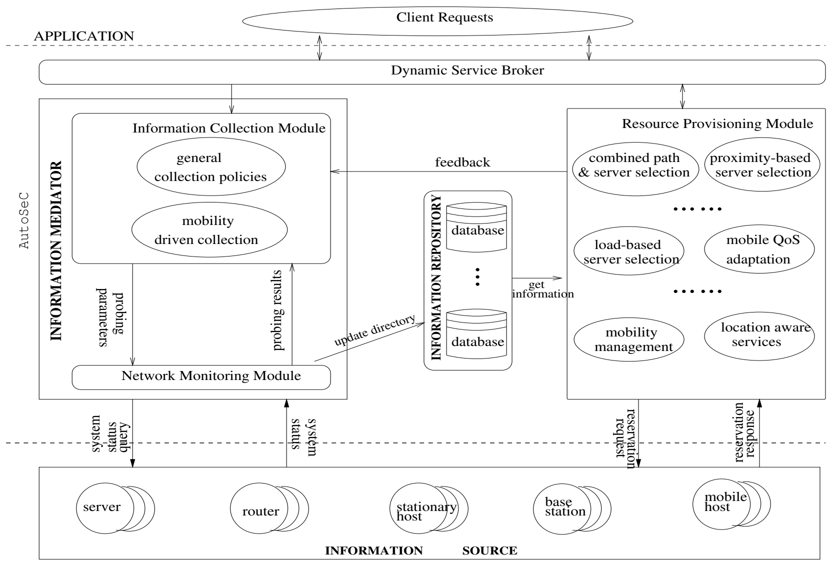

Fig. 17. *The AutoSeC Framework*

mobile host or stationary host. The state information about sources includes network parameters(such as residual link bandwidth, end-to-end delay on links etc.), server parameters (such as CPU utilization, buffer capacity, disk bandwidth, etc.), stationary host parameters (such as client capacity, connectivity, etc.), and mobile host parameters (such as mobile host location, connectivity, power level etc.).

The *Information Repository* consists of distributed databases which hold system state information about information sources. In this paper, for simplicity we assume a centralized database within which adaptive data representation is supported. Each data item can be represented by an instantaneous value or a range bounded by an upper and a lower value. The information in the database is updated based on current network conditions and user requirements using different update policies.

The *Resource Provisioning Module* uses information held in the database and feedback from resource provisioning module to perform admission control and resource allocation.

The *Information Mediator* serves as the decision point for the information collection. It listens to notifications from sources or the resource provisioning module and invokes suitable actions so that the database maintains information at a suitable level of accuracy. It consists of an information collection module and a network monitoring module. The *information collection module* determines whether and how to update the database with the current system image. The *network monitoring modules* are distributed in the network and each module monitors portions of the entire domain based on sampling parameters supplied by the information collection module. The monitoring module issues probes to network components to collect system state information. The probes consolidate the collected sample values and then hand them to the monitoring module. The monitoring module returns the collected results to the information collection module that updates the database.

The *Dynamic Service Broker* implements the service composition rules in Table II. It selects the best combinations of resource provisioning and information collection policies to match current user requests and system conditions. It also determines when to switch among different policies if necessary.

Prototype Implementation: Building *AutoSeC* is an important step in understanding and controlling the complex dynamics of QoS-based resource provisioning under varying levels of information precision. In our prototype system, the information mediator and the database reside on Sun Ultra5 workstations running Solaris 2.7, and information sources execute on Pentium III PCs running Windows 2000 connected via a 10/100Mbps Ethernet link. AutoSeC is implemented in Java to facilitate portability. Fig. 18 shows the prototype system architecture.

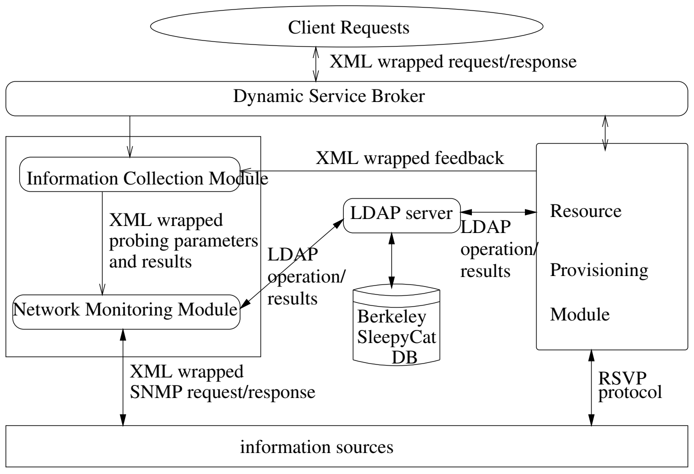
Client Requests
XML wrapped request/response
Dynamic Service Broker
XML wrapped feedback
Information Collection Module
XML wrapped probing parameters and results
Network Monitoring Module
LDAP server
LDAP operation/ results
LDAP operation/ results
Berkeley SleepyCat DB
Resource Provisioning Module
XML wrapped SNMP request/response
RSVP protocol
information sources

Fig. 18. *AutoSeC Prototype System Architecture*

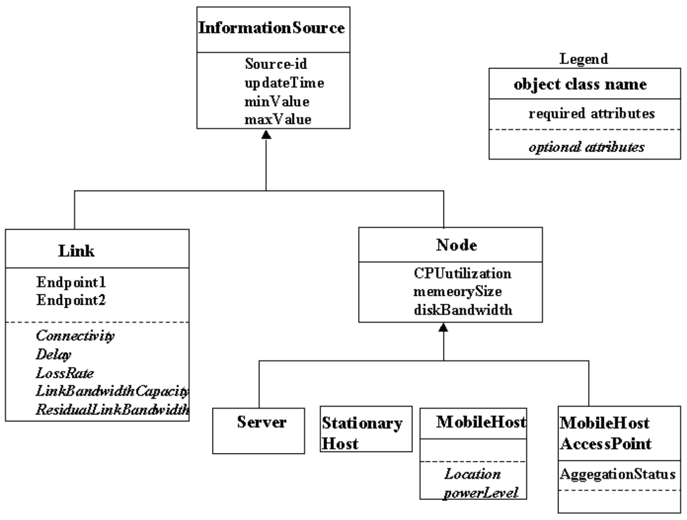

Fig. 19. *Schema Diagram of AutoSeC*

XML is used as a messaging format to communicate between components (sources, consumers, mediator, repository) in the distributed environment since it allows flexible development of user-defined document types. It provides a robust, non-proprietary, persistent, and verifiable file format for the storage and transmission of text and data. In order for each component to verify that the data it receives is valid, we define DTD's (Document Type Definition) where the legal building blocks of an XML document are specified. We collect server and network related information in our prototype system. This information is obtained using Simple Network Management Protocol (SNMP), which is readily available on most servers and routers. Through SNMP, the mediator can obtain current status from the sources; In addition, traps can be defined for sources so that sources can report serious conditions to the mediator. AdventNet SNMP API is a comprehensive toolkit for rapid development of SNMP based management applications, so we leverage it to monitor and track the performance of the sources. As illustrated, the SNMP messages are wrapped in

XML format to enable system interoperability and facilitate integration of non-SNMP protocols. The status of information sources is stored in the information repository, which is implemented using OpenLDAP's slapd LDAP (Lightweight Directory Access Protocol) server together with the Berkeley Sleepy Cat Database as the backend. slapd is a stand-alone LDAP daemon that listens for LDAP connections on a port and responds to the LDAP operations it receives over these connections. Fig. 19 shows the LDAP schema used for our prototype.

Operation	*Latency (ms)*
probing source	3000
XML wrapping	1
SNMP call	0.6

TABLE III

Source probing latency

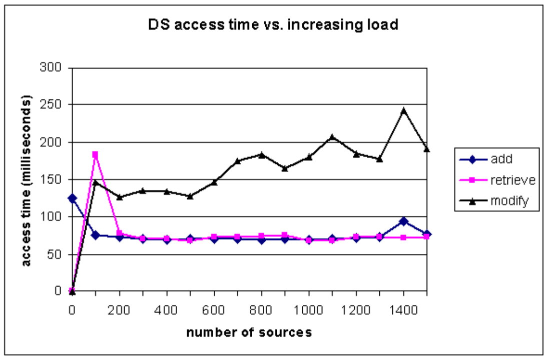

Fig. 20. *Database Access Time*

A preliminary study was carried out to determine the communication latency between different system components. The source probing latency consists of the following parts: (a) XML wrapping of probing request; (b) transmission of the wrapped request from the mediator to the source; (c) unwrapping of probing request; (d) SNMP request; (e) XML wrapping of the result, (f) transmission of the wrapped result from the source to the mediator; and (g) unwrapping the result. Table III indicates that in our implementation the time spent on XML parsing and SNMP calls is negligible comparing to network delay. As the database is a vital part of the information collection framework, we conducted a preliminary study on the scalability of the database by increasing the number of the sources in the system. Fig. 20 indicates the variation in the time of database operations (addition, retrieval and modification) as the number of sources in the database increases. As can be seen, attribute modification of a source in the database incurs significantly more overhead than addition of a source or retrieval values of a source. This trend is fairly consistent.

Integrating AutoSeC into existing wireless infrastructures: The proposed AutoSeC framework can be easily integrated into existing or next-generation wireless networks. In particular, cellular handsets/mobile hosts and base stations can be incorporated into the AutoSeC architecture as information sources to supply provide device specific parameters (by mobile hosts) and information on the population of mobile hosts (by base stations). Typically, cellular networks have pre-defined access points to the wired network (see

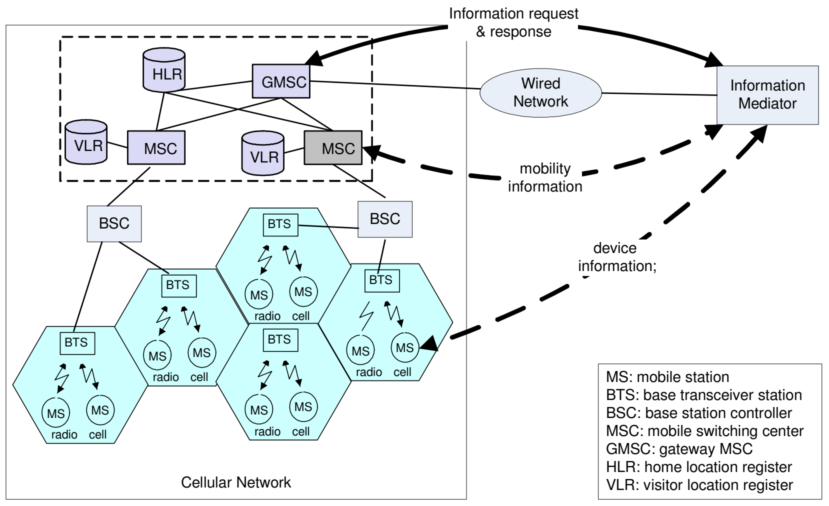

Fig. 21. Integration of AutoSeC and next-generation wireless networks UMTS

GMSC in Fig. 21). Hence, all the information requests from AutoSeC are forwarded by the access point to specific entity in the wireless network. We now illustrate a typical cellular network infrastructure (UMTS) and its integration into the AutoSeC data collection process.

UMTS [22] is one of the premium standards for cellular networks evolving from GSM. In this hierarchical system, coverage area is divided into a number of radio cells and each cell has one *base transceiver station (BTS)*. A mobile station (MS) or a handset communicates with the BTS in its cell. A *base station controller (BSC)* manages several BTSs. It reserves radio frequencies, handles the handover from one BTS to another, and pages the MS when needed. Several BSCs in a geographical region are managed by a single entity, the *mobile services switching center (MSC)*. In addition, each MSC is associated with one *home location register (HLR)* and one *visitor location register(VLR)*. HLR is a database that stores all user relevant information, and VLR is a very dynamic database which stores all important information needed for the mobile stations currently in the region that is associated to the MSC. MSCs are managed by a *Gateway MSC (GMSC)*, which is the access point through which external networks (e.g. wired networks, Wi-Fi networks) can integrate with the cellular infrastructure.

In the UMTS infrastructure, the GMSC serves as the conduit through which mobility related information is delivered to the mediator. GMSC collects mobility related information from MSC and device-specific parameters from MS. In order to manage cellular networks, SNMP agents are typically deployed in MSC or MS. These SNMP agents form the communication mechanism of the UMTS components (GMSC, MS) with AutoSeC.

The collected information can be utilized by both traditional and emerging algorithms. Consider, for example, the open-shortest-path-first (OSPF) routing protocol [5] currently used in the Internet. The routing is done in a dynamic and adaptive fashion based on Dijkstra's shortest path algorithm. The term "shortest path" implies the optimum path by considering multiple factors such as geographical distance, delay, throughput or connectivity etc. A router that detects a change in the network immediately multicasts the information to all other routers in the network so that all will have the same routing table information. In addition, in the proposed QoS extensions to OSPF protocol [19], each switch maintains its own view of the network status, selects a path capable of meeting the QoS requirements of a given request, establish and maintain the path by reserving the resources using resource reservation protocol (RSVP). These examples indicate that network connection parameters and bandwidth availability information are maintained at each router and exchanged among routers either periodically or when a change in the network is detected. By maintaining the system status information in a common repository, we can improve the scalability of the routing protocols in large scale distributed systems. Furthermore, our proposed architecture provides source-based routing algorithms a global view of system status. Standards such as MPLS and RSVP can be used together with the information in the database to find a feasible path to the optimal server, reserve the resources along the path before actually routing the request. This enables improved resource allocation and better QoS guarantees.

More recently, researchers have proposed numerous QoS-based resource allocation algorithms (in the areas of replicated service selection, QoS-based routing, load balancing services, and content distribution), which can benefit from the information collected. For example, dynamic server selection policies have been studied in the context of replicated web data. One such effort [20] uses the notion of a Predicted Transfer Time (PTT) to reduce the response time. PTT is calculated using parameters such as available bandwidth and round-trip delay. As another example [21], link parameters such as delay, available bandwidth, etc. is described using probability distribution and used to find the most probable path satisfying the requested bandwidth and end-to-end delay. The Combined Path and Server Selection (CPSS) algorithm used in this paper utilizes the information from both server and network routers to assist the selection of an optimal server and a path for better resource utilization.

The development of AutoSeC framework and its integration into existing wireless infrastructures has the potential to improve system and network resource provisioning while ensuring application QoS.

VI. Related Work

In this section, we compare our work to related research in network status collection, data caching and QoS-based resource provisioning.

System status collection for *wired networks* has been widely studied in network management community. Existing collection techniques may be classified as being passive or active. Passive collection observes the existing traffic and infers network performance from the collected measurements, e.g.. Tcpdump, NetFlow. In contrast, active collection injects probes into the network and then derive network performance from the end-to-end measurements, e.g. *traceroute*, *netperf*,*netstat*. Tools such as *Pathchar* [23], *Packet pair* technique [24] and *Packet tailgating* [25] measure link and/or path capacity by inspecting the change in the latency of the test packets sent out. Systems that use active probing for the measurement of communication resources across Internet wide networks include Network Weather Services(NWS) [26], topology-d [27], and Remos [28]. Similarly, the snapshot based collection policy (SS) has been used in routing protocols such as OSPF [5] and QoS-based path selection [29], where link state information (e.g. dynamic residual bandwidth) is periodically exchanged among routers. To decrease the overhead introduced by active probing, a straightforward strategy is to control the probing frequency. Adaptively adjusting probing frequency has been studied using threshold or history measurements [30], application information [31] or bandwidth availability [32]. An alternative strategy to reduce the overhead is to degrade information accuracy to a certain extent. For example, the Static Interval based policy(SI) [6] stores an interval rather than an instantaneous value for each data item. In addition to all the existing work in wired network monitoring, network monitoring for *wireless networks* has been focusing on mobile devices' location information management [10]. In this paper, instead of focusing on mobile host location management, we are interested in collection of system status information where mobility status is accounted for improved performance. In addition, we studied the impact of SS and SI on the performance of QoS based resource provisioning in mobile environments and compared the performance against our proposed approaches.

Our information collection policies address the tradeoff between information accuracy and maintenance overhead, which bears similarity to adaptive data caching. In adaptive data caching, one of the strategies is to use approximations to exact values in the cache in order to reduce cache maintenance cost. Divergence caching considers setting the precision of approximate values in a caching environment [33], where precision is inversely proportional to the number of updates to the source value not reflected in the cached approximation, independent of the actual updates. The TRAPP system automatically selects a combination of locally cached bounds and exact master data stored remotely to deliver a bounded answer to a query consisting of a range that is guaranteed to contain the precise answer [14], [34]. In addition, research on data streams([35]) deals with continuous monitoring/querying of fast changing data in both traditional Internet environments and emerging ad-hoc sensor networks, etc. These projects include

OpenCQ, Niagara, Telegraph, STREAM and Aurora. Our work differs from the aforementioned research in the evaluation methods of cache maintenance strategies. We use QoS-based resource provisioning as a motivating example and measure the effectiveness of proposed collection policies by comparing their impact on the performance of resource provisioning.

Network resource provisioning has been extensively studied in the context of QoS-based routing both in wired networks and mobile ad hoc networks. QoS routing is to find a network path to satisfy user quality requirements while achieving system efficiency of resource utilization. Based on how the network information is maintained and how the feasible path is found, QoS routing can be classified into source routing, distributed routing and hierarchical routing [36]. Among a number of challenges faced by QoS routing, imprecise state information is one that is relevant to this paper. Source routing algorithms are proposed [21], [37] to deal with imprecise state information. In their approaches, link parameters such as delay, available bandwidth, etc. can be described using probability distributions and can be used to find a *most probable* path satisfying the requested bandwidth and end-to-end delay. In order to avoid the expensive centralized computation at the source, a distributed routing scheme [38] is proposed, which searches multiple paths in parallel for a satisfactory one. The state information of intermediate nodes is collectively used to guide the routing messages along the most appropriate paths in order to maximize the success probability. Further studies [3] show that the routing algorithm, network topology and link-state update policy have significant impacts on the probability of successfully routing new requests, as well as the overhead of distributing network load metrics. While the research mentioned above focuses on routing itself, CPSS (Combined Path and Server Selection) [16] is proposed to consider a joint optimization in searching both a path and a server for requests with QoS requirements. In mobile environments, the problem of imprecise state information becomes even more severe due to node mobility. Predictive location-based algorithm [39] is proposed to compute the future route in advance before existing routes break. In this paper, we use QoS based resource provisioning as a consumer of the information provided by the collection services. We evaluate the impact of information collection policies on QoS-based resource provisioning under different system conditions and application workload.

VII. Conclusions

A key concern of our work is to preserve the desired QoS of applications in mobile environments without introducing significant overhead. In this paper, we have indicated that instantaneous representation of information obtained via frequent periodic sampling incurs significant overhead and is not necessary; on the contrary, adaptive dynamic range-based information collection mechanisms exhibit superior performance under most system conditions and user workloads. We also observed that resource provisioning process benefits significantly from the information collection mechanisms that utilize the mobile user aggregation status. We further showed that coarse-grained mobility information (i.e., aggregate user mobility status) is sufficient for effective resource provisioning, while fine-grained mobility information (i.e., individual user's location information) introduces very high overhead and hence not desirable. The integrated middleware framework AutoSeC presented in this paper is a result of incorporating the insights gained from studying the impact of various information collection policies on the performance of a family of resource provisioning mechanisms. In general, the dynamic nature of applications such as those of multimedia under varying network conditions, request traffic etc. implies that information collection policies must be dynamic and customizable. Therefore, intelligent policy composition mechanisms such as those explored in AutoSeC, must be developed to achieve effective utilization of resources while ensuring user QoS.

Adaptive information collection policies must not only accommodate statistical fluctuations in network and server loads, handle changes in user mobility status, but also deal with a variety of application workloads. The distributed object paradigm is often used to facilitate the development of large scale distributed applications. However, the traditional object messaging layer operates with limited awareness of underlying system/ network conditions; whereas current system/network monitoring tools operate at network layer with little awareness of application level object communication requirements. We have explored the possibility, mechanisms and benefits of filling the gap between object messaging and system monitoring. We introduced the connection abstraction as the mechanism for these two layers to communicate and exchange information. Through this integration, object messaging can proactively adapt to changing system conditions; system monitoring policies and parameters can be optimized based on inter object communication properties [40].

In this paper, a specific information consumer – QoS based resource provisioning is used to evaluate the information collection policies presented here. In fact, information consumers can be extended to a large scope of context-aware applications, and our eventual goal is to develop a fundamental middleware service for context information collection. To achieve true context awareness, systems must produce reliable and real-time information in the presence of uncertain, rapidly changing and partially true data from multiple heterogeneous sources. With the emergence of sensor networks, a large amount of context information is captured by sensors. Distributed sensor environments are very different from traditional mobile environments: Sensor devices are expected to interact with the physical world by allowing continuous monitoring and reaction to natural processes, hence real-time communication is crucial. In addition, sensor nodes are energy constrained and sensor networks are subject to high fault rates. The concept of QoS for distributed sensor applications differs from QoS concept for multimedia applications and can be defined by information delivery latency, information accuracy, and energy efficiency. Therefore, context information collection for distributed sensor environments should address timeliness/accuracy/energy consumption tradeoffs while ensuring this particular QoS. Middleware techniques for adaptive context information collection such as those described in this paper are key to guaranteeing application QoS in highly dynamic heterogeneous environments. This is a necessity to achieve the goal of true ubiquitous computing.

ACKNOWLEDGMENT

The authors would like to thank Zhenghua Fu for useful discussions that led to the framework described here and his contributions to designing the simulation environments. This research was supported by the Office of Naval Research under MURI Research Contract N00014-02-1-0715 and the National Science Foundation under Award Numbers 0331707 and 0331690. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the research sponsors.

REFERENCES

[1] A. Acampora and M. Naghshineh, "Control and quality-of-service provisioning in high-speed microcellular networks," *IEEE Personal Communications Magazine*, vol. 1, no. 2, 1994.

[2] S. K. Das, R. Jayaram, and S. K. Sen, "An optimistic quality-of-service provisioning scheme for cellular networks," in *Proceedings of ICDCS*, 1997.

[3] A. Shaikh, J. Rexford, and K. G. Shin, "Evaluating the impact of stale link state on quality-of-service routing," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, April 2001.

[4] Z. Fu and N. Venkatasubramanian, "Combined path and server selection in dynamic multimedia environments," in *ACM Multimedia*, Orlando, FL, 1999.

[5] "Ospf version 2," RFC 1247, July 1991.

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

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

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

[9] I. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, and W. Wang, "Mobility management in next-generation wireless systems," *IEEE Proceedings Journal*, vol. 87, no. 8, August 1999.

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

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

[12] Q. Han and N. Venkatasubramanian, “Aggregation based information collection for mobile environments,” *Journal of High Speed Networks*, vol. 11, no. 3,4, 2002.

[13] Q. Han and N. Venkatasubramanian, “A cost driven approach to information collection for mobile environments,” in *Proceedings of IEEE MWCN*, 2002.

[14] C. Olston, B. T. Loo, and J. Widom, “Adaptive precision setting for cached approximate values,” in *ACM SIGMOD*, 2001.

[15] N. Venkatasubramanian and S. Ramanathan, “Load management for distributed video servers,” in *IEEE ICDCS*, May 1997.

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

[17] A. Bar-Noy, I. Kessler, and M. Sidi, “Mobile users: To update or not to update?” *ACM/Baltzer Journal of Wireless Networks*, vol. 1, no. 2, July 1995.

[18] L. Bottomley, “Epa-http single www server trace,” http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html.

[19] “Qos routing mechanisms and ospf extensions,” RFC 2676, January 1998.

[20] R. Carter and M.E.Crovella, “Dynamic server selection using bandwidth probing in wide-area networks,” in *IEEE InfoCom*, 1997.

[21] D. Lorenz and A. Orda, “Qos routing in networks with uncertain parameters,” in *IEEE InfoCom*, 1998.

[22] J. Schiller, *Mobile Communications*. Addison-Wesley, 2000, ch. 4.

[23] V. Jacobson, “pathchar- a tool to infer characteristics of internet paths,” ftp://ftp.ee.lbl.gov/pathchar/, 1997.

[24] S. Keshav, “Measuring link bandwidths using a deterministic model of packet delay,” in *ACM SIGCOMM*, 1991.

[25] K. Lai and M. Baker, “Measuring link bandwidths using a deterministic model of packet delay,” in *ACM SIGCOMM*, 2000.

[26] R. Wolski, N. Spring, and C. Peterson, “Implementing a performance forecasting system for metacomputing: The network weather service,” in *Proceedings of IEEE/ACM SC Conference*, 1997.

[27] K. Obraczka and G. Gheorghiu, “The performance of a service for network-aware applications,” in *Proceedings of the SIGMETRICS symposium on Parallel and distributed tools*, 1998.

[28] T. Dewitt, T. Gross, B. Lowekamp, N. Miller, P. Steenkiste, and J. Subholk, “Remos: A resource monitoring system for network aware applications,” CMU, Tech. Rep. CMU-CS-97-194, 1997.

[29] Q. Ma and P. Steenkiste, “On path selection for traffic with bandwidth guarantees,” in *Proc. IEEE ICNP*, 1997.

[30] P. Dini, G. V. Bochmann, T. Koch, and B. Kramer, “Agent based management of distributed systems with variable polling frequency policies,” in *IEEE/IFIP IM*, 1997.

[31] P. Moghe and M. Evangelista, “An adaptive polling algorithm,” in *IEEE/IFIP NOMS*, 1998.

[32] K. Yoshihara, K. Sugiyama, H. Horiuchi, and S. Obana, “Dynamic polling scheme based on time varation of network management information values,” in *IEEE/IFIP IM*, 1999.

[33] Y. Huang, R. Sloan, and O. Wolfson, “Divergence caching in client-server architectures,” in *IEEE PDIS*, 1994.

[34] C. Olston and J. Widom, “Offering a precision-performance tradeoff for aggregation queries over replicated data,” in *VLDB*, 2000.

[35] M. Garofalakis, J. Gehrke, and R. Rastogi, “Querying and mining data streams: You only get one look,” Tutorial at VLDB, 2002.

[36] S. Chen and K.Nahrstedt, “An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions,” *IEEE Network Magazine*, vol. 12, no. 6, November-December 1998.

[37] R. Guerin and A. Orda, “Qos-based routing in networks with inaccurate information: Theory and algorithms,” in *Proceedings of IEEE InfoCom*, 1997.

[38] S. Chen and K.Nahrstedt, “Distributed qos routing with imprecise state information,” in *IEEE ICCCN*, 1998.

[39] S. H. Shah and K. Nahrstedt, “Predictive location-based qos routing in mobile ad hoc networks,” in *Proceedings of IEEE ICC*, 2002.

[40] Q. Han, S. Gutierrez-Nolasco, and N. Venkatasubramanian, “Reflective middleware for integrating network monitoring with adaptive object messaging,” *IEEE Network*, January 2004.