# Services-Oriented Computing in a Ubiquitous Computing Platform

Ji Hyun Kim[1], Won Il Lee[1], Jonathan Munson[2], and Young Ju Tak[1]

[1] IBM Ubiquitous Computing Laboratory, Seoul
[2] IBM T. J. Watson Research Center, Hawthorne, New York
jihkim@kr.ibm.com, wilee@kr.ibm.com, jpmunson@us.ibm.com, yjtak@kr.ibm.com

**Abstract.** Current telematics services platforms are tightly integrated, relatively fixed-function systems that manage the entire end-to-end infrastructure of devices, wireless communications, device management, subscriber management, and other functions. This closed nature prevents the infrastructure from being shared by other applications, which inhibits the development of new ubiquitous computing services that may not in themselves justify the cost of an entire end-to-end infrastructure. Services-oriented computing offers means to better expose the value of such infrastructures. We have developed a services-oriented, specification-based, ubiquitous computing platform called TOPAZ that abstracts common ubiquitous computing functions and makes them accessible to any application provider through Web-service interfaces. The nature of the TOPAZ services, as enabling long-running sessions between applications and remote clients, presents peculiar challenges to the generic issues of service metering and resource management. In this paper we describe these challenges and discuss the approach we have taken to them in TOPAZ. We first motivate and describe the TOPAZ application model and its service set. We then describe TOPAZ's resource management and service metering functions, and its three-party session model that forms the basis for them.

**Keywords:** Ubiquitous computing, telematics, resource management, service metering.

## 1 Introduction

The term "ubiquitous computing" is often applied to the applications that serve mobile users such as drivers, healthcare workers, and emergency personnel. They typically link personal or embedded devices with centrally operated services, using information (context) gathered from the devices. These applications are particularly popular in the automotive world, having millions users worldwide, and are now appearing in other domains as well. In the Republic of Korea, for example, several municipalities are launching ubiquitous computing initiatives under the collective name "u-City." The term encompasses a general vision where information systems for healthcare, education, and even private residences are responsive to collective contextual data, and these information systems have a pervasive presence in homes, streets, buildings, and public places such as convention centers.

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 601 – 612, 2006.

Meeting the diverse and changing needs of these user communities can be prohibitively expensive given the cost of developing, building, and operating ubiquitous computing systems serving millions of users, over wireless networks, using a diverse set of devices. Organizations that could provide valuable services to telematics or other ubiquitous-computing consumers face a high barrier of entry, because they must either integrate their service with an existing service provider, or provide their own end-to-end solution. As a result, business models that can be supported in this kind of ecosystem are limited.

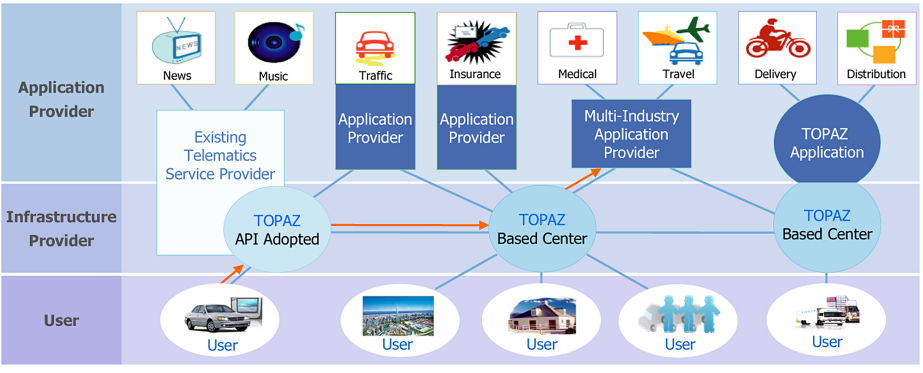


**Fig. 1.** The role of TOPAZ in the ubiquitous computing applications marketplace

We envision a marketplace in which application providers are able to easily compete for the business of users and user communities. They would be able to inexpensively develop, operate, and maintain the applications, and they would be able to quickly make them available to end users. In this paper we describe a ubiquitous computing framework named TOPAZ that we designed to enable this kind of marketplace. TOPAZ is a platform of core services for ubiquitous-computing applications, the purpose of which is to factor out the telematics-intensive parts of telematics applications and make them available in a uniform way to all application providers, through public applications-programming interfaces. The TOPAZ service set is provided by TOPAZ Platform Operators (TPOs), who make the services available to any application provider. Fig. 1 illustrates.

We designed TOPAZ based on this vision and the technology of services-oriented computing. At the outset of the project we identified services-oriented computing as the model that would enable us to realize our vision of a new kind of ubiquitous computing marketplace. Service interfaces would allow us to abstract out ubiquitous-computing infrastructure functions and provide them to applications through standards-based interfaces. Web services, in particular, would enable us to clearly separate the execution environments of the TOPAZ Platform Operator and the application providers, and enable TPOs to provide their services to any application provider on the Internet.

The position of TOPAZ with respect to applications and ubiquitous-computing clients (e.g., cell phones, telematics devices) is illustrated in Fig. 2.

The services that TOPAZ provides include managing content flows from the applications to the clients; managing user-interaction flows from the clients to the applications; managing the flow of sensor data from clients to applications; detecting application-specific situations involving clients; and supporting peer-to-peer flows of data and content among clients. All of these services are accessed through industry-standard Web-service interfaces. Like other Web-service-based platforms of application services, TOPAZ has the same concerns of service metering, quality of service guarantees, and resource management. However, while we have been able to use some existing models for these functions, we have had to adapt them to the peculiar nature of our services, and we have had to develop our own implementations of them. In this paper we describe the nature of our specific application model and discuss the particular services our platform offers. We then describe our approaches to the utility-computing issues of resource management, quality-of-service guarantees, and service metering.

## 2 TOPAZ Application Services

The set of application services offered by TOPAZ was defined by a process of requirements gathering and factoring. We sought to define a platform that met the needs of a wide range of applications with a relatively small number of services. Automotive and fleet telematics was the applications domain of initial focus. We began by compiling a large set of application scenarios through surveying a number of existing telematics systems and industrial telematics consortia. Then, through analyzing their requirements, we factored out a set of core services that would support the applications. The services are implemented as WSDL-based Web services.

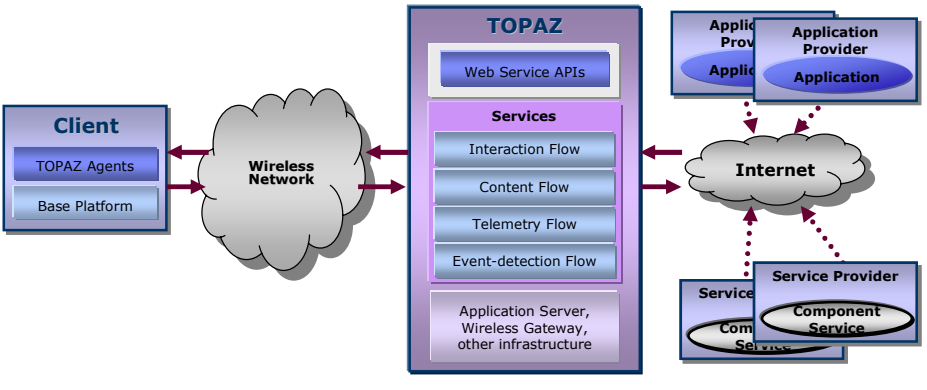


**Fig. 2.** TOPAZ in End-to-end System

TOPAZ services are in general concerned with facilitating and managing data flows between applications and clients or between peer clients, where the data may be content flowing from applications to clients, or data and events from clients to

applications. The services include Telemetry Subscription, Content Push, Client-to-Client Communication, Spatiotemporal Event Detection, and Event-Based Content Push. We describe these services in more detail below.

**Telemetry Subscription.** This service enables service providers to receive dynamic vehicle data (a.k.a. telemetry data) from individual vehicles or groups of vehicles. Service providers specify the data that is to be sampled, simple conditions under which the data is to be sent, the period at which the conditions are to be evaluated, and an address of a Web service that will receive the data. A TOPAZ Telemetry Agent fulfils the subscription at the vehicle end and sends to a TOPAZ telemetry receiver server. The server collects the telemetry from multiple vehicles for a short time and then forwards it to the requesting application.

**Content Push.** The Content Push service enables application providers to push multimedia messages, map features, Web forms, and other content to customers' in-vehicle displays. This may be used, for example, by real-time navigation applications to push routes, route information, and just-in-time turn instructions to drivers. The Content Push service offers different priority levels and different reliability levels to the application programmer. Content received from multiple applications concurrently is aggregated for delivery to the client, for greater efficiency in wireless networks.

**Client-to-Client.** TOPAZ also provides a service enabling application providers to connect groups of users directly, for message- or connection-oriented communications. This service enables applications to link communication applications with the interactive viewers provided by a TOPAZ client, and it eliminates the need for them to act as their own communications hub.

**Spatiotemporal Event Detection.** The Spatiotemporal Event Detection Service serves those ubiquitous-computing applications that involve sensing conditions in a user's physical context and responding to those conditions in real time. Examples include fleet-management services that alert drivers when they are off-route or off-schedule, and employee safety applications that notify plant staff that a new employee has entered a restricted area without an accompanying supervisor. Programmers represent events of interest in the form of rules that "trigger" when the situations are detected.

**Event-Based Content Push** is an extension of Spatiotemporal Event Detection that enables application providers to associate content with a rule. The ECPS will push the content to a user that caused the rule to trigger.

**Resource Services.** TOPAZ provides several services for managing specific kinds of resources. The User Group service enables applications (and application providers, through a portal) to create groups and assign users to groups. Each of the services above allows groups, as well as individual users, to be subjects of service calls. The Rule Resource service enables applications to create, update, and delete data objects (such as geographical polygons) that are referenced in rules of the Spatiotemporal Event Detection service. The User/Device Resources service enables applications to associate arbitrary data with a user or device, for general-purpose use.

**Application Example.** A typical TOPAZ application is one offering drivers turn-by-turn driving instructions to a given destination. Use of the application by a client involves three flows: one for application requests from the user to the application, another for telemetry data (the vehicle's position) from the vehicle to the application, and another for content from the application to the user: routes and turn instructions. Fig. 3 illustrates. Execution begins with a request from the user to start navigation, the request including the destination. The request takes the form of an HTTP Post, using an HTML form provided by the application.

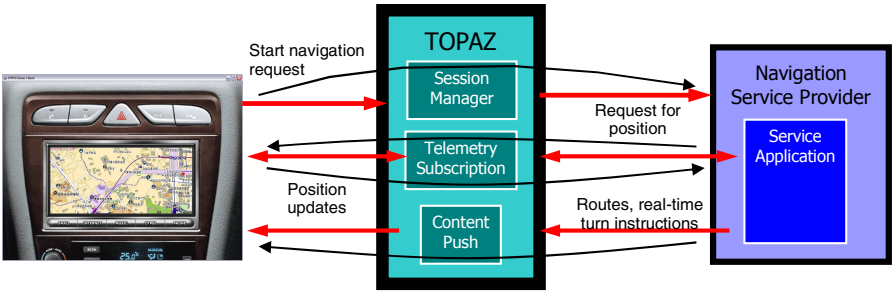


**Fig. 3.** Application execution using Telemetry Subscription and Content Push services

When the application receives the request, it invokes the Telemetry Subscription service, which initiates the second flow, of position data from the vehicle to the application.

When the application decides it needs to notify the driver of an upcoming turn, or if it has determined that the driver is off-route, and it needs to send the driver new instructions, it will compose the content and invoke the Content Push service to do so. These pushes constitute the third flow.

## 3 TOPAZ Session Model

The data flows (of content, application requests, telemetry data, and rule-triggering events) managed by TOPAZ services take place in the context of sessions. TOPAZ provides a multi-layered session model, shown in Fig. 4. A session may be a device session, user session, or application session. Each session has a parent session. For application sessions, the parent session is a user session; for user sessions, the parent session is a device session. For device sessions, the parent session is the System session, not shown. Parent relationships of the sessions in Fig. 4 are shown by the tree in the figure.

A device session is created when a device first connects to a TOPAZ Session Manager. The Session Manager automatically creates a user session for the device owner. Then the Session Manager will start any device-dependent auto-startup applications, creating application sessions for them. Normal user sessions are started when users log in. Non-device-dependent applications set for auto-startup are then started at this time. Users can manually start and stop applications at any time following this. When a device is shutdown, all children sessions of the device session are closed.

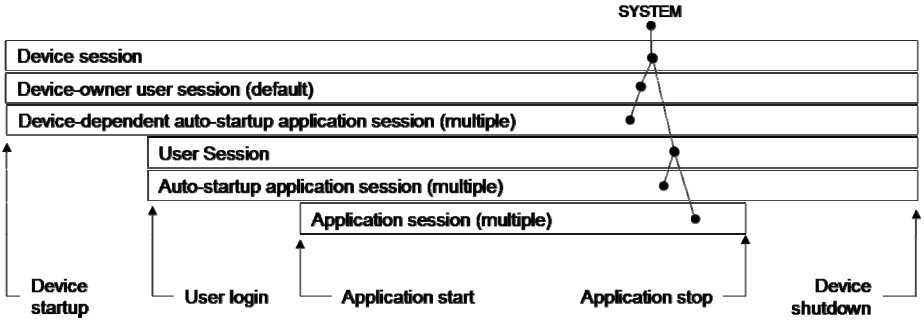


**Fig. 4.** TOPAZ Session Model

## 4 Resource Reclamation in TOPAZ

A variety of resources are involved in the facilitation of any given flow in TOPAZ. What we term a resource is a data object residing on the server relating to the operation of an application, that lasts at least as long as an application session. A user group is an example of a resource, as is the internal record of a telemetry subscription. Resources may be created explicitly through service invocations, such as user groups created through calls to the User Group Service, or may be created implicitly by services, such as when telemetry subscriptions are created as the result a rule subscription to the Spatiotemporal Event Detection Service. Resources may be passive data objects, such as telemetry subscription records, or they may be active objects, such as the objects in the Content Push Service that manage content pushes to individual clients. Applications can elect to manage the lifetimes of resources themselves, or they can allow TOPAZ to manage the lifetime for them.

**Table 1.** Resource lifetimes

| Lifetime Name | Duration |
|---|---|
| DEVSESSION | The life of the associated device session. |
| USERSESSION | The life of the associated user session. |
| APPSESSION | The life of the associated application session. |
| APPLICATION | The life of the application that created the resource. |
| PROVIDER | The life of the provider of the application that created the resource. |

Automatic resource reclamation is an important function in TOPAZ because forcing applications to do it explicitly is too great a burden on programmers. They may not properly clean up; the client or the application may quit unexpectedly, or clients or applications may suffer long periods of disconnection. Without automatic resource reclamation, dead resources would grow continuously with no way to reclaim them.

**Table 2.** Resource types

| Service | Resources | Allowable Lifetimes |
|---|---|---|
| Telemetry Subscription | Telemetry subscriptions | APPSESSION or APPLICATION, chosen by application |
| | Per-client subscription optimizers | DEVSESSION |
| Content Push | Per-client content-push priority managers | DEVSESSION |
| User Group | User groups | APPSESSION, APPLICATION, or PROVIDER, chosen by application |
| Client-To-Client | Telemetry subscriptions | APPSESSION |
| Spatiotemporal Event Detection | Rules | APPSESSION or APPLICATION, chosen by application |
| | Rule subscriptions | APPSESSION or APPLICATION, chosen by application |
| | Telemetry subscriptions | APPSESSION or APPLICATION, chosen by application |
| Rule Resources | Rule resources | APPLICATION |
| | User rule resources | APPSESSION or APPLICATION, chosen by application |
| Event-based Content Push | Rules | APPLICATION |
| | Rule subscriptions | APPLICATION |
| | Telemetry subscriptions | APPLICATION |
| | Event/content table | APPLICATION |
| User/Device Resources | User/device resources | APPLICATION |

To facilitate the reclaiming of resource objects, each resource is associated with the lifetime of a session, an application, or a provider. Table 1 lists the possible resource lifetimes. Programmers declare the lifetime of resources they create explicitly, while TOPAZ services declare the lifetime of resources that are created implicitly, as the result of creation of other resources.

**Modular Resource Management.** The model for resource management in TOPAZ is that each application service manages its own set of resources. Each service module is notified of lifetime events (e.g., session ended), and each module performs the appropriate resource-management functions. The table below lists the resources managed by each service.

The lifetime events that each module uses to dispose of resources (and in some cases create them) originate with the Session Manager, the Applications Manager, and the Provider Registration Manager. The Resource Manager is a component on the

server that acts as a central clearinghouse for these events, distributing them to the other server-side modules that require them.

Internally, the Resource Manager uses an asynchronous messaging mechanism (J2EE Message-Driven Beans) to decouple the callers of the Resource Manager from the execution of the management logic at the receivers of the Resource Manager's lifetime events.

**Table 3.** Quality of Service Parameters in TOPAZ Services

| Service | Parameter | Description |
|---|---|---|
| Telemetry Subscription Service | Sampling interval | Applications specify the sampling interval of the requested telemetry data. |
| | Report aggregation | Applications can request that the telemetry reports be delivered in as large batches as practically possible. |
| Content Push Service | Push priority | Applications set the priority of content delivery, on a per-push basis. |
| User Group Service | None | |
| Client-To-Client Service | Same as Telemetry Subscription Service | |
| Spatiotemporal Event Detection Service | Rule evaluation interval | Applications specify how frequently rules are evaluated. |
| Rule Resources Service | None | |
| Event-based Content Push Service | Same as STED Service and Content Push Service | |
| User/Device Resources Service | None | |

## 5 Quality-of-Service in TOPAZ

The conventional notion of quality-of-service in Web services is the response time of a service invocation. However, since TOPAZ services are client-session-oriented, not request/response oriented, its notions of quality-of-service are correspondingly different. Rather than requesting a certain response time, TOPAZ service callers specify certain qualities of the client session, on a per-session, or finer, basis. Table 3 lists the quality-of-service parameters offered by each service.

In the sections following we discuss the quality-of-service parameters offered by the Telemetry Subscription Service and the Content Push Service.

**QoS for the Telemetry Subscription Service.** The Telemetry Subscription Service offers two parameters related to quality of service, sampling interval and report aggregation.

*Sampling Interval*. Applications express the sampling interval of a telemetry subscription as minimum and maximum intervals between any two samples of the telemetry requested. In order to make most efficient use of its bandwidth to the TOPAZ server, a TOPAZ client will attempt to send the data for multiple subscriptions at the same time, and so allowing some variance in the sampling period of subscriptions is helpful.

Applications choose a maximum interval according to the response-latency requirements of their application. The shorter the interval, the more quickly the application can respond to changes in the client's context. Applications set the minimum interval to a level high enough to avoid unnecessary expenses due to too-frequent telemetry reports (because more frequent telemetry result in increased service charges from the TOPAZ platform operator). However, the usage fee structure for the Telemetry Subscription service encourages applications to set a reasonably wide range between the minimum and maximum intervals.

*Report Aggregation*. Group telemetry subscriptions offer a "maximize aggregation" parameter, which, when true, instructs TOPAZ to aggregate client telemetry destined to a single application as much as possible. This will result in fewer telemetry transmissions to the application, each one aggregating more client reports. The resulting load on the application server handling subscriber telemetry should therefore be lower. However, the cost of transmitting data from clients may increase because TOPAZ has less flexibility in scheduling transmissions from clients to the TOPAZ servers. Thus the TOPAZ platform operator's service charges to the application provider may increase.

The mechanism for doing this is to approximately synchronize the sampling and transmission of the application's requested telemetry data at all clients in the group. TOPAZ synchronizes (roughly) the telemetry streams by telling each client to start sampling at a common UTM-specified time and to sample at a common interval thereafter. Clients who miss the start time can synchronize with the group by beginning sampling at any multiple of the specified maximum sampling interval.

Whether or not clients synchronize their sampling, the Telemetry Subscription Service will buffer their telemetry reports for a short time before forwarding them together to the application.

**QoS for the Content Push Service.** The Content Push service offers two parameters related to quality of service: the priority of a push, and the reliability required for the push.

*Priority***.** The Content Push Service is used for content that is time-sensitive, such as a message from a real-time navigation service instructing a user turn right in 50 meters, and for content that is not time-sensitive, such as telemetry requests. Because it must handle content push requests from multiple applications simultaneously, it must therefore make decisions about which content goes first. A simple first-come, first-served approach would mean that time-sensitive content may unnecessarily wait for non-time-sensitive content. Therefore, in order to serve applications more effectively, the Content Push Service offers a "push priority" parameter that can take on the values URGENT and NORMAL. Callers use URGENT for content that represents a time-sensitive communication. NORMAL delivery should be used for all other content. Content sent with NORMAL delivery may experience slight delays for

efficiency reasons. Content sent as URGENT will not suffer these delays, but will be charged a higher rate.

*Reliability***.** Not all content has the same importance, and the Content Push Service offers two levels of reliable delivery: BESTEFFORT and ASSURED. With BESTEFFORT delivery, the CPS will attempt delivery a limited number of times before it gives up and discards the push request. With ASSURED delivery the CPS will retry delivery unless it is clear that delivery is not possible (e.g., user has unsubscribed from the application).

## 6 Service Metering

TOPAZ does not mandate a particular business model used by a TOPAZ Platform Operator, but it does provide models for service metering that a TPO can use as a basis for its business model. We expect a typical business model for a TPO to be one in which it charges users for subscriptions to applications, and it charges application providers for the use of TOPAZ application services. TOPAZ's service metering models provide a basis for a TPO's charges to an application provider.

Rather than metering per service invocation, as do some Web-services platforms (for example, ESRI's ArcWeb services [6]), or using monthly or annual fees (for example Microsoft's MapPoint services [11]) TOPAZ's metering models are based on the aggregated cost of providing flows. In this section we describe how this orientation toward flows has determined out metering models. Our work in this area is not complete; we are currently refining our models and determining the various constants in them empirically.

Each application service has its own metering model, according to how flows in the service consume system resources. A service's metering model is a function whose inputs are the parameters used in invoking the service that generated the flow and any statistics recorded for the flow, and whose outputs are abstract "cost units". These cost units translate directly to monetary charges. The terms in the formulas for the models reflect how invocation parameters impact the consumption of particular system resources.

In this section we focus on two services, Telemetry Subscription and Content Push.

**Metering Telemetry Subscription.** The various parameters used in invoking the Telemetry Subscription service impact its consumption of system resources in various ways, but the most important are the minimum and maximum sampling intervals. For each telemetry report received by the server, it must allocate a thread and a database connection. Relatively few CPU cycles are consumed in processing the data. In order to not block the client while sending the data to the application, the TSS decouples the process of receiving the telemetry from the process of sending it, but this means the service must allocate another thread for sending. In order to reduce the number of threads required for this, the service will batch reports together for a short time before sending them to the application.

The initial TSS metering model we are working with now charges application providers according to how their subscriptions consume bandwidth, modified by how the subscriptions impact other system resources. The charge to any one application for

a telemetry session is a simple summation of the charges for handling each report, where the charge for each report is a product of the size of the report and the sampling parameters in the application's subscription. Applications reduce their charges by specifying a generous tolerance between minimum and maximum sampling intervals, thus allowing the client more flexibility in combining telemetry reports destined for different applications in the same message sent to the server. Therefore the server needs to allocate only one thread and one database connection to process multiple reports. This is reflected in the metering model by applying a discount to the per-report charges, where the discount is a function of the range between minimum and maximum sampling intervals.

**Metering Content Push.** Threads are the primary resources consumed by the Content Push service. For each URGENT push a thread must be allocated to invoke the transport mechanism that will carry the data. NORMAL pushes, however, are queued for a short time before the entire queue is emptied and sent to the client. Therefore, the metering model for Content Push is a function of the priority used, as well as the size of the content pushed.

## 7 Related Work

Our technique for resource reclamation is similar to the "soft state" approach used in management of network-entity state in Internet protocols[5, 13, 14], RMI and Jini [12], and more recently the Open Grid Services Architecture [8], in that each reclaimable resource is associated with a lifetime. In our technique, however, the lifetime is not an actual time, but instead a link to the lifetime of an entity in the system—a session, application, or application provider. Resources are known to be reclaimable when the entity to which they are associated has ceased to exist. In this respect our technique is similar to reference counting in a distributed system [2], except that references point in the other direction. Thus, knowing the ID of a deceased entity, we can query a resource set directly for resources whose lifetime is associated with it.

Work in quality of service for Web services—how to specify it, measure it, and monitor it—focuses, for performance metrics, on generic qualities such as response time, throughput, availability, and reliability [3, 10]. However, our flow-oriented services require different performance metrics, such as sampling regularity in the Telemetry Subscription service. In this respect our concerns are more nearly aligned with multimedia systems, but with looser real-time constraints. See [4].

The abstract cost units our metering models are based on are similar to the credits used by ESRI's ArcWeb Services [6]. Our cost units, however, are based on how a particular service's flows consume critical resources, and the cost of providing those resources. We have based our models partly on the modeling of thread and database connection resources in [7]. While we have implemented our own metering subsystem, we could also use metering services such as that in the UMI utility infrastructure [1].

## 8 Conclusions

TOPAZ is a Web-services-based platform of services designed to facilitate a marketplace of ubiquitous computing applications, by making these applications radically less expensive to develop, deploy, and operate. As with any utility-computing infrastructure, it faces challenges on how to manage the objects and resources consumed in the delivery of its services, how to meter the use of its services, and what quality-of-service parameters to offer to applications. We have presented our own approaches to these challenges, which take into account the nature of TOPAZ services as providing "flows" of content, data, and event-detection between clients and applications. We continue to refine our metering models based on observations of our system's runtime characteristics.

We have developed a number of applications for TOPAZ, and external developers have developed others. We are currently in the process of measuring the performance of TOPAZ when serving large numbers of clients.

## References

1. Albaugh, V., Madduri, H., The Utility Metering Service of the Universal Management Infrastructure. IBM Systems Journal, Vol. 43, No. 1, 2004, 179–189.
2. Bevan, D.I., Distributed Garbage Collection Using Reference Counting. In *Parallel Architectures and Languages Europe*, 1987, Springer-Verlag, LNCS 259, 176–187.
3. Bhoj, P., Singhal, S., Chutani, S., SLA Management in Federated Environments. In *Proceedings of the Sixth IFIP/IEEE Symposium on Integrated Network Management (IM '99)*, IEEE, 1999, 293–308.
4. Campbell, A., Coulson, G., Garcia, F., Hutchison, D., Leopold, H., Integrated Quality of Service For Multimedia Communications. In *Proceedings of the 12th Annual Joint Conference of the IEEE Computer and Communications Societies - IEEE INFOCOM '93*; 1993, 732–739.
5. Clark, D.D. The Design Philosophy of the DARPA Internet Protocols. In *SIGCOMM Symposium on Communications Architectures and Protocols*, 1988, ACM Press, 106–114.
6. ESRI ArcWeb Services. http://www.esri.com/software/arcwebservices/index.html
7. Ferrari, G., Ezhilchelvan, E., Mitrani, I. Performance Modeling and Evaluation of E-Business Systems. CS-TR 954, School of Computing Science, University of Newcastle, March 2006.
8. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S. Grid Services for Distributed Systems Integration. *Computer*, Vol. 35, No. 6, 2002.
9. Frølund, S., Koistinen, J. 1998. Quality-of-Service Specification in Distributed Object Systems, Distributed System Engineering 5: 179–202.
10. Keller, A., Ludwig, H., The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management, Vol. 11, No. 1, March 2003, 57–81.
11. Microsoft MapPoint. http://www.microsoft.com/mappoint/default.mspx
12. Oaks, S., and Wong, H. *Jini in a Nutshell*. O'Reilly, 2000.
13. Sharma, P., Estrin, D., Floyd, S., Jacobson, V., Scalable Timers for Soft State Protocols. In *IEEE Infocom '97*, 1997, IEEE Press.
14. Zhang, L., Braden, B., Estrin, D., Herzog, S., Jamin, S., RSVP: A New Resource Reservation Protocol. In *IEEE Network*, 1993, 8–18.