

## Locate and Track

# Locate and Track

Bruce Cohen  
GemStone Systems, Inc.

### Abstract

*In a wireless world, objects cannot be expected to stay in one place, and finding them becomes significantly harder than if they were stationary. The Locate and Track pattern, similar to the Lookup pattern, addresses this problem of finding distributed objects and services, but may be used when location matters. Locate and Track uses the location of the client and the service, or the nature of the space or channel between them, to determine the suitability of a particular service for a client. Moreover, suitability is re-evaluated on a regular basis to determine if either client or server has moved so as to invalidate it.*

### Examples

Consider a mobile navigation system, in which a local computer in a vehicle uses a GPS receiver to determine its position, and then queries a centralized map database for a map of the immediate area around the current position of the vehicle and the current conditions of the roads, points of interest, etc. In this example, the client is mobile, but each of the potential services resides in a fixed location.

Now consider a robot construction crew, consisting of a large number of small, agile robots with similar capabilities, which can be dynamically organized into different groups to perform specific tasks. If a robot member of a crew digging a building foundation discovers a rock too large for the discoverer to remove, a subgroup of the robot crew can be organized dynamically to dig the rock out and haul it away. Only those robots in position to help with the rock should be discovered as acceptable services by the client coordinating the task.

And finally, consider a client that needs the result of a service request within some maximum latency time. For instance, an interactive client run by a human camera person working from a mobile vehicle that needs to operate a video camera on a separate mobile platform. The communication between the two must take less than about 100 to 200 milliseconds round-trip, or the delay could cause the operator to lose control because of positive feedback between command and platform movement.

## Locate and Track

In all these examples the choice of service depends on either location or communication medium attributes not necessarily knowable at the time the service registers itself for lookup, or that require additional protocol on top of lookup to use appropriately. In some of the examples, these attributes can change even while requests are being serviced, so that the appropriate choice of a service entity can change after the client locates one.

### Context

Distributed systems where the set of services that can be effectively used by a client, or the set of clients a given service can effectively support, must depend on their relative locations, or on the absolute location of one of them, or on the nature of the communication channel between them.

### Problem

Location transparent lookup assumes that the only attributes of services that are of importance to a client's selection of a particular service instance are attributes inherent in the service's behavior, or attributes known to the service at the time it registers itself with the lookup service (plus the information a client has about itself). But consider how a client selects a service when it communicates via a wireless network, and either the client or service, or both, are mobile. If the interaction between the client and server is dependent on where they are (absolute location of one or both) or how far apart they are (relative location of one with respect to the other), then selection requires information that may not be available prior to the instant at which the client starts the lookup.

Problems arise because of side effects of the location of client or server or in the communication between them. These problems are especially troublesome when the client and services have no previous knowledge of each other, when there are a large number of clients and services, or when the organization and communication flow among clients and services is dynamically organized as operation proceeds.

Mobile systems can also present a problem for the initial bootstrap of finding a service registry. How can a client determine which registry to use when the network the client is on is not a wired local area net, but a wireless, distributed internet. In such a network the set of network nodes that are immediately reachable can vary as the client or service nodes move around.

These problems produce the following forces which must be resolved by the solution the pattern provides:

- *Availability*: A client should be able to find out what services are available in its environment, to include its geographical location and communication latency distance, while as much as possible ignoring services outside those regions.
- *Restriction of service*: A service should be able to refuse to service a request if the location of the client requesting it would invalidate constraints on the service (e.g., real-time latency).
- *Termination of service*: A client must be able to recognize when a service is no longer appropriately located for its use, and locate a new service; conversely, a service must be able to recognize when a client can no longer guarantee the request constraints and terminate service to the client.

## Locate and Track

### Solution

Use the Lookup pattern to find services, but provide for additional criteria in finding and selecting services such that location and communication medium can become visible if required by the client or service for their function. Then have the client and the service keep track of changes in location or communication medium and terminate their association if the changes render it unusable.

### Registry Bootstrap

For the case where the client must find the appropriate registry for its current locality, one solution would be to have a single centralized registry containing the locations of local registries which register services in a particular locale. This solution is simple, but has several drawbacks:

- The centralized registry may need to be very large, and distant from the client, and may act as a bottleneck for lookup traffic.
- Going through 2 or more levels of registry to change the advertised location of a service that has moved may require sufficient time that the service will not be present at its new location long enough to be useful.

A more flexible solution is to have the client advertise its location when it looks for a registry, and have only those registries within an appropriate region respond. Services would register themselves with registries near their locations. The network of services would therefore be organized by locality.

### Service Registration

One solution for making the service location visible is to have the service register itself with a (perhaps regional) lookup service with a lease shorter than the time it expects it will require to move beyond a region in which its service will be continuously effective. Then when it moves, it can register itself again, perhaps with a different lookup service serving a different locality, or with a new location attribute.

If the service must be in particular region or location to maintain service for a particular client, then it must revoke the lease obtained by the client at lookup if it leaves that region. If the client/service interaction requires that the same service complete some set of operations then higher level patterns must be used by which the client negotiates for the service to remain in the required region for the required time.

Selection of services falls into two cases:

The client determines the algorithm for selecting a service based on data in the client (e.g., its location). This can be implemented by having the client compute the required attributes of a service and use them in lookup.

The service determines the algorithm for determining its appropriateness for a given client based on the location of the client. This can be implemented by registering a location independent broker service that fulfills requests for location dependent services based on the client location, which is passed with every request. If time code information or network packet routing information is passed with a request, the server can determine the characteristics of the intervening communication medium to

## Locate and Track

determine if it should accept the request.

### Termination of service

To allow for termination of service, the service must treat each request from the client in the same way it evaluates and accepts or rejects the initial request. So if the client data in a given request were to indicate an unacceptable location relative to the service's current position, the service would treat that request as if the client's lease had expired.

### Known Uses

Internet routing takes into account the "distance" in the communication medium (effectively, the cost of sending a message) between two nodes, and dynamically re-evaluates the distance as conditions change.

### Consequences

The benefits of Locate and Track are:

- *Geographic filtering*: A client can select from among those services which are in the correct region, or can communicate with it appropriately, without having to winnow through large numbers of unusable services.
- *Decentralization*: The need for centralized registries can be reduced, lessening long-distance traffic on the network.
- *Simplification of application code*: Removing the negotiation of location from the client and service application code makes the code simpler; one or a small number of negotiation routines can serve many clients and services.
- *Flexible negotiation*: The location negotiation algorithms or parameters can be modified independent of the client or service code; different instances of the same object can easily be configured with different location characteristics.

• The liabilities of Locate and Track are:

- *Additional overhead*: More message traffic between client and registry, service and registry, and service and client, may be required.
- *More complicated lease implementation*: Service termination can mean that lease handling algorithms are harder to implement, and may cause client application code to be more complicated.