CS464/564 Project4

Given: Wednesday November 15, 2000

Due: Wednesday December 13, 2000, at the beginning of class

Overview:

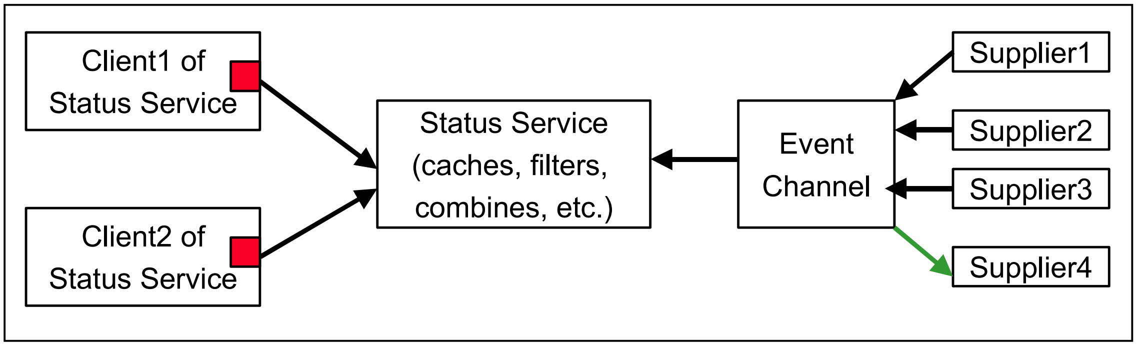
In this project you write a simple status service, using the CORBA Event Services provided by VisiBroker. The architecture of the example is provided, but you will have to devise the specific examples of the service to form a realistic (or at least plausible) status service.

Instructions on how to turn it in, further details on the project, and where you can copy theTA's Project2 from, will be forthcoming via email from the TA. You will demonstrate the running project to the TA between the time you turn it in and the final examinations' date.

Problem Setting:

CORBA event services are very useful for allowing data-driven or event-driven communications to occur. However, every consumer gets every event from every subscriber, which is far more than many kinds of clients need. A status service is a kind of service that gets around this problem, by being the consumer for events from low-level probes or sensors, and then providing a higher-level status service to clients. Typically such clients can subscribe to certain status conditions (e.g., temperature in a given city) and get callbacks when it changes too much or over a threshold. Or they can call the status service for the latest values of the conditions of interest.

You will develop a simple status service. It will use 4 different sensors (Event services suppliers), each which you will design. It will also utilize one or two different client applications (one gets you more credit). You only have to use one event channel between the suppliers and the status service. However. The architecture is as follows:


Client1 of Status Service
Client2 of Status Service
Status Service (caches, filters, combines, etc.)
Event Channel
Supplier1
Supplier2
Supplier3
Supplier4

Example Baseline:

Here is an example baseline, which you will get no credits for originality for doing, though you may use it in Supplier #1 is a temp. sensor at the Moscow Mall, and Supplier #2 is at the U. of Idaho's student union bulding. Supplier #3 is a temp. sensor at the WSU student union building, and Supplier #4 is in downtown metropolitan Pullman. The Status Service stores the current temperatures in Moscow and Pullman (the latest update from either site) as well as the average temprature in the Palouse region (average between Moscow and Pullman). Client 1 is a casual user, who just queries the service occasionally. There is no Client2.

Grading Criteria:

The grading is based on the following:

40% baseline with "no frills"

10% demo (runs OK; 5% max if no GUI, but only a simple one is needed for 10%)

20% extra features (564 only; 10% each). This can include

- A different kind of client (not just a different copy of the same client), o
- Using the event model in more than just the canonical push model
- Using an object wrapper to cache a value at the client
- Having the status service offer a callback, for example when a value gets over a certain threshold
- Some other extension you come up with (this helps your originality points too!)

20% originality/realism of the exact status service, suppliers, clients, etc. And discussion of the tradeoffs etc if you had to deploy the application for real. If you use the baseline example, and have very little writeup, you get 20% off here.

10% for the rest of the writeup. I will provide the required sections later; it will probably be 3-5 pages.

Collaboration with Other Students:

This is an individual project, and you need to do the design and coding by yourself. You certainly may discuss with classmates what kinds of status service or event suppliers are interesting, and why. However, each student should do his or her own design of the IDL for the different components (certainly do not share code here!), do his or her own coding and debugging, analysis, writeup, etc.

For More Information:

We will be discussing this project in the beginning of the next few lectures. But to make sure that you understand the issues discussed here, and you get off to a good start, please start working on this!