Service-oriented middleware for mobile sensing

A recent survey from Forrester suggests that in 2016 one billion people will own smart mobile devices with powerful sensing, computing, and communication capabilities. The widespread adoption of such devices creates an unprecedented opportunity towards harvesting and harnessing tremendously large volumes of sensed information so as to select and inform the most appropriate context aware servicesto support human users.

This practical aims to design and develop a service-oriented middleware for mobile phone sensing. This practical is an opportunity to get better acquainted with middleware systems and service-oriented architectures; to think about scalability and robustness when building with components.

The practical will be conducted in small groups of 2-3.

The challenge

Build a service-oriented middleware for mobile sensing that's as scale and robust as you can make it, and demonstrate this is the case. This will consist of three kinds of components: service producers (i.e., mobile phones), service registry and broker (i.e., hold sensing services), and service consumers (i.e., any application or users that request or use mobile data). The first two are the primary components in this practical and they interact with each other in two phases: (1) registration — each mobile phone is expected to periodically advertise its services to the Registry; (2) lookup and access — When data about the physical environment of a remote region is required, the Registry is queried for devices with the relevant sensing services to then access them and acquire their measurements, which can be processed to obtain the desired results.

Put them together into an appropriate network, and
then extend the system to address different issues that cause
problems, including but not limited to, temporary interruptions of
connection; crashing queues; crashing consumers; increasing
number of queries; mobility of mobile phones; long delays in network
traffic; dropped messages; out-of-order messages; duplicated messages;
and so forth.

The key to the practical is to *demonstrate* that your solution works. Just imagine you have millions of mobile phones, and how your middleware is designed to support location, failure, load, and upgrade transparencies.

You can use any base middleware you like: Java RMI, CORBA, SOAP, XML RPC, something with a binding into your favourite language. As long as you don't choose one that is basically a robust MOM system itself, it's fine :-) In many ways SOAP is the obvious modern choice, but the techniques are pretty similar whichever you choose.

Assessment

Submit via MMS your code and a short (no more than 8 page) report for the group that describes your solution and demonstrates that it solves -- and to what extent -- the scalability and robustness challenges you've chosen to address. The assignment is due to Wednesday of Week 8; that is, 23:59 2nd April.

Marks will be given for the scalability and robustness the system demonstrates and for the quality of evidence mustered in support of these claims.

This assignment is worth 60% of your total coursework mark. Marking will follow the mark descriptors as set out in the student handbook (see http://www.cs.st-andrews.ac.uk/student-handbook). Late submission of coursework is penalised by 1% per hour.

Suggested reading list:

In case you are not familiar with mobile sensing concepts, here are the list of relevant academic papers:

- Service-Oriented Computing: Key Concepts and Principles, by Michael N. Huhns and Munindar P.Singh. IEEE Internet Computing, 2005 Feb, pp.75-81.
- Service-oriented middleware for large-scale mobile participatory sensing, by Hachem et al. Pervasive and Mobile Computing 10(2014), pp.66-82. [To note, the idea of this practical is built on top of this paper, but in a much smaller and simpler scope.]

Hints

Given that this practical is about implementation and evaluation, there's no point making a user interface or even allowing user interaction: it's probably easier to have everything driven from a script, or use simulators, to make it repeatable.

The practical is clearly open-ended, since the quest for scalability never ends. I would choose three challenges to focus on, extend the system gradually to cope, and add measurement and analysis as you go. If you do more, all well and good.

Think from the start about what you can measure: if you can't measure it, you can't really claim it. It's better to claim less, but demonstrate it convincingly, than to claim a lot of not have proper evidence in support.

Think about what would convince you, as a developer or buyer, that the system could do what it claimed: what sorts of things would you want to see?