Planning with Middleware Services

Wednesday 1st of November, 2017

Individual Middleware Services

These are the services we've seen so far:

- Remote execution, server activation
- Discovery (directory or p2p)
- Event notification
- Mobility management
- Content adaptation
- Context-awareness

We only use services when they're needed, e.g. only use a discovery service if we're in a dynamic environment where services are coming and going.

When a middleware service is required, the next question is "What is the supporting architecture?". E.g. with content adaptation, adaptation can take place on the device or on the server, but if it's on the server, the server needs to be provided with information about the clients.

When designing the architecture that supports a middleware service, it's important to consider the efficiency.

The set of protocols implementing the service is also important.

Performance Goals

- achieve the main functional goal without errors
- minimum execution time / minimum overhead
  - middleware services are all simple to aim for this
  - they aim for this because all middleware services will be running in conjunction with an application
- minimum resource consumption
  - especially energy

        * only keep the most important parts of the service active all the time
    - often compromise between this and execution time
- robustness
    - can't really measure this
    - want the system to withstand changes and still provide the correct result
- reliability
    - does the service provide the expected result?
- scalability
    - similar performance results irrespective of the number of clients of the service

Use Case 1

[...]

A directory-based solution allows for creation of a federation of directories (one at each site), allowing for service discovery across the networks. It would also allow for logging. Finally, security and service provisioning can be better controlled with a centralised solution. However, the directory-based approach has worse scalability than the p2p approach.

Protocols

1. Service discovery
2. Service offer registration (in the directory)
3. Service offer de-registration
4. Directory service lookup
5. Remote service use
6. Lease allocation and renewal
7. Service event notification

Evaluation

Is the system achieving the functionality? What's the cost/overhead? Is it robust, reliable, and scalable? (some of this determined by testing)

Use Case 2

Small organisation with one site that may include mobile devices. As the set of services on offer changes dynamically, middleware for service discovery and remote execution is required.

P2P looks more appropriate, as it's more suitable for mobile, short-lived services.

Server would broadcast its services periodically, and clients can listen to discover the services.

A useful benefit here is that the broadcasting can be controlled – the offer window can be varied. During this window, servers advertise and clients listen.

This would also consume less resources – for example, no dedicated directory service which is always running.