9: Mobility Management Service

Access Point Based Mobile Networks

- the alternative would be ad-hoc networks

The problem:

- mobile devices subscribe to events or need fetch data from internet servers
- moving from access point to access point should be transparent
- the middleware needs to minimise duplication and loss of information

You can solve this problem with proxies and caching. Caching, however, raises the problem of inconsistency between different caches – one solution to this is to have the access point periodically broadcast invalidation reports that contain data that has been changed during the last time interval. (Access points are already broadcasting beacons periodically.)

Here the supporting architecture is:

- proxies are stationary components running at or close to each access point
- while the client is disconnected, and during the hand-over phase, subscriptions and publications are managed by the proxy/caching system

Note we are requiring significant resources with each access point.

Mobile Publish/Subscribe Service

Mobile clients use a mobility service client library, linked to the client application.

When the client is connected, the client library mediates subscriptions – it stores a copy of client subscriptions. Before disconnecting, the client calls the move-out function, which causes the client library to move the stored subscriptions to the mobility proxy (which is on the access point).

The proxy then subscribes to and buffers all incoming messages. When the client reaches the destination, it calls the move-in function, to contact the local proxy. The two proxies execute a protocol that results in the transfer of all subscriptions and buffered messages to the proxy that is now near the mobile device.

The call to the move-out function can be triggered in response to a reduction in signal strength from the access point – this way it will happen if the client is leaving (though it may also happen for other reasons).

The proxy will have to subscribe to publishers on behalf of each client that has called move-out and hasn't yet reached another proxy.

When the client reaches the new access point, it needs to tell the local proxy where to find its old proxy, so that the client can receive the stored subscriptions, etc. It will also need to give an ID to the local proxy, so that the old proxy knows which device's subscriptions are no longer needed.

Issues

There are several issues here.

First of all, security – the proxies need to be verified when they talk to each other. This can be solved by a symmetric key – the mobile device tells the first proxy it will use a specific key, and it then reports that to the second proxy.

Second, duplication – a client may receive copies of the same notification. Some notifications may also be lost. Duplication happens between a client calling move-out and receiving confirmation of that, and also between calling move-in and the old proxy removing its subscription, when it is more likely.

Data loss is possible if a client leaves an access point too quickly (and so doesn't receive an acknowledgement from the access point), or if it is disconnected for too long (a proxy can only store a limited amount of data).

How can you make this a robust system?

We can use shadowing. The access point is aware that the client is shadowing the proxy, and so doesn't deliver to both – just to the proxy.

Another possibility is to use ping messages between the two proxies(using the Publisher/Subscriber system itself), and another is to introduce a delay before removing subscriptions issued by the move-out proxy.

Cache Consistency

Mobile devices on a certain access point may be running several applications that interact with remote servers (maybe for synchronisation). Each application will have a cache memory, so that it can find data there instead of having to access the server. This cache must be kept consistent with the data on the server.

Systems

- stateless server (server has no information about the mobile clients)
- stateful and asynchronous server (server stores information about the clients)

Stateless Server

We use invalidation reports – the server sends messages that report all data that was changed. Clients can then refresh any data in their cache that has changed.

Invalidation reports must include:

- the server ID
- list of data that was changed
  - note this is variable

The size of the broadcast window (interval between two invalidation reports) determines how much data can be invalidated in one IR.

If an application is checking the cache, typically it will delay the cache until the next invalidation report, to see if the data is still valid. If it isn't valid, it will request it from the server.

Mobile applications can listen for the beacons to synchronise with the server.

Benefits:

- mobile devices don't have to always be connected

Drawbacks:

-

Updated Invalidation Reports

As before, there is a periodic beacon. In between beacons, we can also allow time-critical data invalidation reports (updated invalidation reports).

Now, applications can wait for just the updated invalidation reports, and not the main ones.

Stateful Server

Each server stores info about its clients (e.g. HomeCache).

[need to check this – really don't get it]

Server sends updates to clients, and keeps track of whether the data has been acknowledged by the client.

?

When a mobile device disconnects and then reconnects, it needs to update its cache data.

It sends a list of its caches to the access point proxy, along with the timestamps for the most recent modification of each of the caches.

Everything before that timestamp can be removed – only updates after the timestamps need to be invalidated.