Context

John-Austen Francisco

1. ABSTRACT

We present a system to allow users to locate network services across an internetwork.

2. INTRODUCTION

While many services may exist across a network, the network itself only exists to transfer data from point to point, putting the burdern of discovering services on the user. Not only does the user need to know what services are available, but also what their network address is. Knowing all the types and network locations of all services available across a ubiquitous, dynamic, worldwide network, like the Internet, is all but impossible. The Context Service Index (CSI) remedies this by acting as a semantic lookup service that records announcements from context services about the kinds of data inferences they can provide and matches client service requests against them. The CSI sends a list of the GUIDs of all services with matching inference announcements to the requesting client(s).

3. ORGANIZATION

Interactions with the CSI are divided into two categories; client interactions and service interactions.

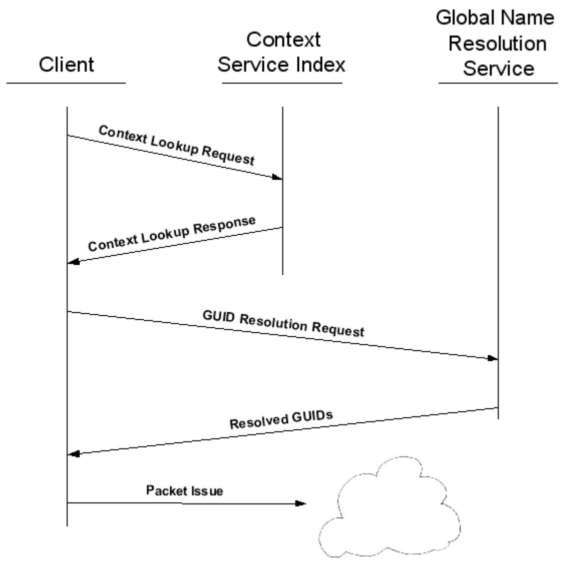

Figure 1: CSI Client Interactions

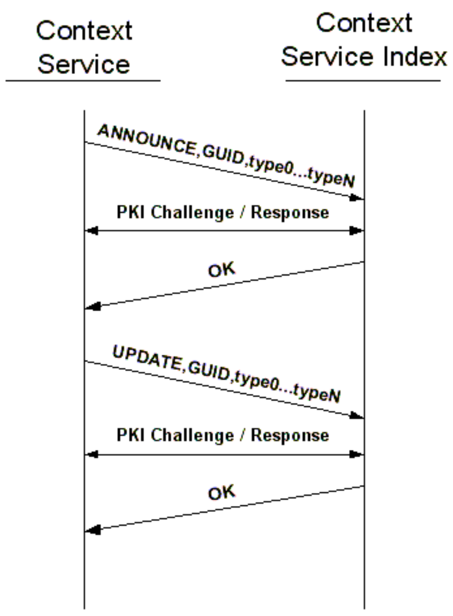

Figure 2: CSI Context Service Interactioons

3.1 Client Interactions

Clients can only query the CSI. This is done using a query message type and specifying the kinds of context fields or inferences the client wants to find. The CSI will respond with either GUIDs of services that can provide those kinds of context, or with a failure message. Clients can also query the CSI to obtain a list of context types that a given GUID can deliver.

3.2 Service Interactions

Services announce to the CSI what kinds of context they can provide. While there is no provision for a service to query the CSI, there is nothing preventing a service from using the client protocol to do so. Services are expected to be ready to deliver judgements and handle large amounts of traffic as soon as they announce their services. Services may overwrite their service announcments in order to add or delete from their service list, although all operations that alter state require a PKI authentication.

3.3 Protocol

- All characters outside of <> and [] are literal symbols.

- All fields, literal or variable, are separated by a comma (',')
- All commands end in a period ('.')
- Angle brackets ('<>') denote a requied field that is variable. The text between the brackets names the field, but is not literal.
- Hard brackets ('[]') denote optional fields that are variable. The text between the brackets names the field, but is not literal. 0 or more optional fields can be included.

3.3.1 Announce

Message type used to announce a context service's capabilities to the CSI. The Announce message is only expected for a GUID that does not exist in the CSI.

Service->CSI: initiate

1. ANNOUNCE,<GUID>,<capability name>[,<capability name>].

CSI->Service: response

1. OK,<GUID>.
   no problem
2. WARN,<GUID>,CAPEXIST,<capability name>[,<capability name>].
   This GUID has already announced the attached capabilities within this announcement. Logical error, but not an error state.
3. WARN,<GUID>,CAPTYPEDEF,<capability name>[,<capability name>].
   The CSI does not know about this capability and does not support any functions on it. The CSI will not accept any data for this type. The CSI will however store the capability/GUID tuple and will answer lookups for this type.
4. ERR,<GUID>,CAPTYPEILL,<capability name>[,<capability name>].
   The following capability types are, for some reason, illegal types. This is a functional error. The CSI will not store these capabilities or accept data based on them.
5. ERR,<GUID>,INTCAPSTORE,<capability name>[,<capability name>].
   The CSI has failed to store the capability/GUID tuple. This is an internal error. It's expected that the client will either repeat their request or attempt to contact a different CSI.
6. ERR,<GUID>,GUIDEXIST.
   The CSI already has an entry for this GUID, precluding an Announcement-type message. The CSI will not store any of the capabilities sent. The service is expected switch to an Update message.

3.3.2 Update

Message type used to alter a service's current capability list in the CSI any time after an initial Announce.

Service->CSI: intiate

1. REMOVE,<GUID>,<capability name>[,<capability name>].

CSI->Service: response

1. OK,<GUID>,<capability name>.
   No problem
2. WARN,<GUID>,NOCAPDEF,<capability name>[,<capability name>].
   The following capabilities that were removed are not defined for that GUID. This is only a warning since the effect, no tuple binding for that capability, is already the case.
3. ERR,<GUID>,INTCAPREM,<capability name>[,<capability name>].
   The CSI has failed to update the capability/GUID tuple. This is an internal error. It's expected that the client will either repeat their request or attempt to contact a different CSI.

3.3.3 Index

- Any client (C) can query the CSI at any time to generate a GUID/capability index

Client->CSI:

1. INDEX,<capability name>[,<capability name>]. -expect response 0 or 1

CSI->Client:

1. OK,<GUID>[,<GUID>].
   All matches
2. NO,<GUID>.
   No GUID has all the capabilities listed
3. ERR,<GUID>,NOCAPDEF,<capability name>[,<capability name>].
   The CSI has no definition for some of the capabilities listed
4. ERR,<GUID>,INTCAPIND,<capability name>[,<capability name>].
   The CSI has failed to find a GUID list for the capability list. This is an internal error. It's expected that the client will either repeat their request or attempt to contact a different CSI.

4. USE CASES

4.1 Send to Teatime

The CSI can be used to send data to a dynamic group, for instance to send a message to all persons in a kitchen at teatime. The client would first request the CSI to find services that can perform the location function. The client would then choose a service GUID to communicate with, ask for the locations of all GUIDs that service can locate, determine which ones are in the teatime area, and then send a message to them.

4.2 Decentralized Taxicab Dispatch

The CSI can aid in the decentralization of some storngly-centralized resource allocation and management problems. If taxicabs can themselves receive calls, they need not either accept them or route them to the centralized dispatcher, but can forward the calls themselves if the taxi can discover which of the other taxis is nearer to the call than itself. One way to do this is to intellligently name context services that are exported in the CSI. If all taxicabs act as context services and have a way of discovering what stree they are on, they can update the CSI with that name. The driver who takes a call could then query the CSI to see if any taxis are on the street in question, or can get the current 'services' of all the taxi GUIDs it knows about, revealing their current positions, allowing the driver who received the call to decide which taxi should have the call forwarded to it.

5. LIMITATIONS

While designed to operate within the Mobility First Internet, and link clients to service providers, the CSI has a range of fundamental limitations.

5.1 No understanding of context

While it succeeds in linking clients to services, the CSI does not understand context at all. It requires the clients and the services to, a priori, understand all the types of context services available. Unless the client looks for the correct type of context inference, they will not find a service that produces what they desire. This is akin to searching for websites based on keywords, if websites could only be described by keywords. Unless the correct set of keywords are searched for, the desired site would never appear as an option.

5.2 No unified view of context

While it succeeds in facilitating connection between client and service, the CSI does not guarantee or enforce any unified view of the way context is interpreted or represented. For instance, a 'location' could be a name of a state, street or building. It could be GPS coordinates, lattitude/longitude, r/Pi from a center point, or a convex hull, to name a few. It could also represent a point, adjacency, the state of being in a certain room, the state of existing within a certain area. There are also issues in encoding any of these representations of the semantic meaning of 'location'. The CSI requires both the client and service to understand all of this or synchronize on it out of band.

5.3 No understanding of context services

The CSI can deliver network-level information, namely GUIDs, but it does not understand anything about the network itself. It can not for instance provide a client with service GUIDs that are 'closer', 'lower latency', or 'less loaded'. The CSI can not reason about any qualitative or quantitative aspect of context services.

6. REFERENCES