Destination Identifier for CONNECT operation

July 23, 1999
Akihiro Shimura
CANON INC.

Summary

Current draft (IDT_r0X.pdf as of July 5, 1999) specifies the SERVICE_ID parameter as a service name string registered with a global naming authority (like IANA).

The SERVICE_ID parameter is used in the following two places;

- Destination identifier for the CONNECT request operation, and
- Service descriptor for the SERVICE_DIRECTORY operation response.

By sharing a common name string in both places, the service descriptor describes destination identifier by itself.

I have a concern with employing the globally registered service name as a destination identifier for the CONNECT operation. (I.e., the use of the SERVICE_ID in the former place.)
And, I would like to propose the use of new parameter "DEST_ID" for a destination identifier that replaces "SERVICE_ID" parameter in the "connection establishment" part of the draft.

The "DEST_ID" is a number uniquely assigned to the service "entity" during its lifetime among all service entities within a fetch agent context (login) on a node. The "DEST_ID" is used as a destination identifier for the CONNECT operation in place of the SERVICE_ID. By introducing the "DEST_ID", the destination identifier for the CONNECT operation is isolated from the global name space.

Background of the problem

A service name registered with a naming authority is *globally* unique within a name space of the naming authority. A name should be registered globally and exclusively within the name space prior to use. The registered name statically stays valid at "any" time.

On the other hand, the requirement for the identifier used with the CONNECT operation is to uniquely identify the listening service "entity" within a *local* login context at connect time. It is enough for the identifier to distinguish between entities that exist at the "same" time within a login *locally*.

The registered service name by itself will not necessarily represent service "entity" uniquely because the service name usually represents only a class (or type) of the service. The "entities" should be differentiated by other means (e.g. attributes, locations, etc.).

Note: The word "entity" means the entity that is listening particular location identified by the destination identifier at CONNECT operation.

Problems and Solution

By employing a registered service name as an destination identifier for the CONNECT operation, the condition allowed to create a service "entity" becomes more restrictive than required due to the static and global nature of the service name like follows;

a) More than one service "entity" that has the same registered name cannot be co-exist within a login context at the same time. (The registered name itself will not necessarily identify the exact service "entity").

b) The name needs to be registered prior to listen even if the name of the listening service "entity" does not need to be *globally* registered with naming authority but only need to be assigned uniquely in the *local* login context. (It may be painful to identify "on-demand" service "entity" by the static globally registered name.)

Though preparing all possible names in *global* prior to use may solve the problems, the global name space in the naming authority (and IANA's name space for the name of naming authority) will be consumed in a twinkling, and finally, every name spaces will be exhausted if everyone try to register all possible names for the purpose of *local* identification of the connect destination.

Thus, I would like to propose to introduce "DEST_ID" parameter that enables *local* assignment of destination identifier for CONNECT operation and that replaces "SERVICE_ID" parameter.
(Though the registered service name will be still valuable with service descriptor, unifying its functionality with destination identifier will be harmful, and isolating destination identifier from registered service name will solve the problem.)

As described previously, The "DEST_ID" is a number uniquely assigned to the service "entity" during its lifetime among all service entities within a fetch agent context (login) on a node.

Note: the queue number(s) could not be used as a destination identifier because the queue number(s) does not become available until a connection establishment and a destination identifier is required **before** initiating a connection.

The service "entity" will be bound to the "DEST_ID" and listen to the CONNECT request. A client will make CONNECT request that specifies "DEST_ID" to connect to the service "entity".

Note: The "DEST_ID" works in the same way as a *destination port* of TCP/IP works. Because an established connection will be uniquely identified by the queue number(s) in the fetch agent context, another identifier similar to the *source port*, that is used as a part of the connection identifier in TCP/IP, will not be necessary as far as connection oriented service is concerned.

Example 1

For example, the "FTP" protocol (well known file transfer protocol in the Internet) uses two connections, one for protocol interpreter and another for actual file transfer.

A client connects to the ftp protocol interpreter service (*destination port* on the server) at first.
Upon a file transfer, the client listens a data port (*destination port* on the client), and tells the port number to the ftp service via protocol interpreter connection (if it is not default port).

Note: To make explanation simple, I eliminated the passive mode here.

The server then connects to given port (*destination port* on the client) and starts file transfer.

In this example, ftp protocol interpreter service can be named (statically) as "FTP".
It may be possible to name a data port, on which client is listening, as something like "FTP-DATA" as

far as only one data port exists at the same time (and default port is used every time). But in reality, there may be more than one on going ftp session, and names different from the "FTP-DATA" (e.g. "FTP-DATA2", "FTP-DATA3", ... and so on) need to be priorly registered to distinguish between listening data ports if the service names are used to address the destinations.
I think this kind of name registration, for identifying port in the *local* context, wastes the *global* name space, and isn't practical.

In fact, static *global* name is not necessary to uniquely address these (dynamic / on demand) data ports, and a "DEST_ID" within the login scope will be enough to identify the destination in the *local* context.

Example 2

Another example is that more than one service "entity" that has the same name but different "entity" co-exists in a *local* context at the same time.

Web services may be named as "WWW-HTTP" regardless of its contents. There already exists the web based printer monitoring service that uses different port (say 8000) from the normal web service (which uses default port 80), and co-exists with normal web service in an interface.

Note: Both entities are accessed by http protocol. When the client talk to these services, both service will return html document. The only difference is document contents obtained from the service. The contents will only be meaningfully distinguishable by human being. Thus, these services will not necessarily be named differently in the "global" name space that computer recognizes.
In the same reason, ftp service entities that provide access to the *different* file system may have the *same* global service name.

In this case, the *global* service name does not identify the exact "entity" and these "entities" are solely differentiated by its destination location (port number) and uniquely identified in the *local* context.

Note: The location will be found by any means other than the service name. It may be in the directory along with service name, attribute, etc (obtained via discovery/directory protocol), or it may be found by the computer recognizable URI (link) in the web page, e-mail. It may also be found on printed materials with human readable explanation, or may be told by someone via phone conversation, face to face conversation, or hand written paper.

The "DEST_ID" distinguishes these service "entities" in the local context in place of port numbers in the above example, and serves well, too.

Conclusion

As described above, employing the globally registered service name as a destination identifier will make the application of the transport very restrictive, or otherwise, the global name space will be exhausted.

By introducing "DEST_ID" that isolates destination identifier from *global* service name space, we will be able to eliminate unnecessary restriction to the application and/or impractical load to the naming authority.