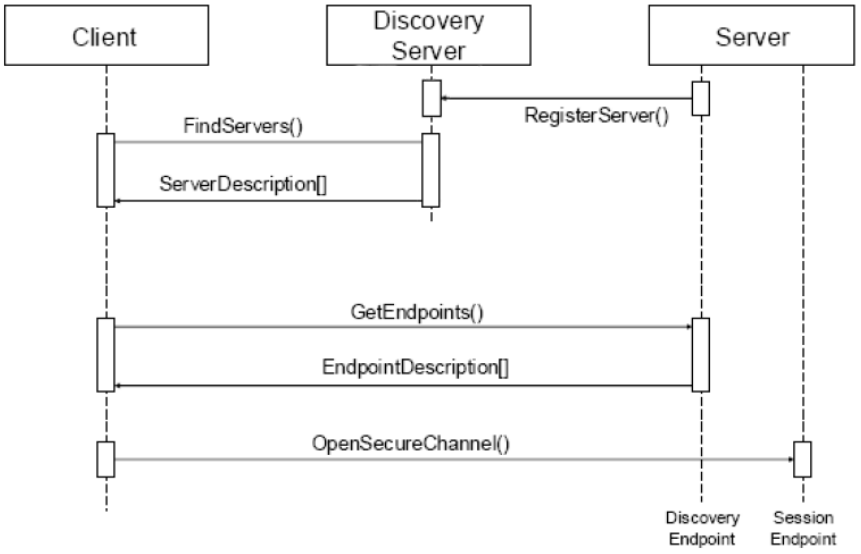
OPCUA Local Discovery Server Example

Bonanno Antonino
Biuso Mario

Obiettivo:
Realizzare un Local Discovery Server (LDS) che si interfaccia con un client e diversi server, utilizzando Stack e linguaggi diversi.

Un client conoscendo l'indirizzo del LocalDiscoveryServer, chiamerà il servizio FindServers sul LDS per ottenere un elenco di server e i loro DiscoveryUrl. Il client a questo punto può chiamare il servizio GetEndpoints per uno dei server restituiti. Il processo di rilevamento per questo scenario è illustrato nella seguente figura.



Stack Utilizzati:

- per **LDS**: node-opcua
- per il **Client**: UA-Java-Legacy
- per i **Server**:
    - node-opcua
    - UA .Net Standard
    - open62541

Abbiamo anche provato senza successo i seguenti Stack:

- python-opcua
- Eclipse Milo

Local Discovery Server (LDS)

Per l'implementazione del Local Discovery Server abbiamo deciso di utilizzare lo stack in node.js, ovvero node-opcua, perchè implementa i servizi di discovery.

Nello stack è presente la classe *OPCUADiscoveryServer*. Una volta studiato e capito il meccanismo di discovery siamo riusciti a configurare opportunamente la classe e quindi il LDS, gestendo i certificati attraverso il *OPCUACertificateManager*.

ServerNode

Dopo aver implementato il Local Discovery Server abbiamo deciso di implementare un server in node.js utilizzando lo stack node-opcua, per poter registrare il nuovo server al LDS.

La realizzazione del server consiste nella configurazione della classe *OPCUAServer*, in particolare per poter registrare il server al LDS abbiamo aggiunto due attributi alla configurazione del server, ovvero ***registerServerMethod*** impostandolo come *opcua.RegisterServerMethod.LDS* e ***discoveryServerEndpointUrl*** settando l'endpoint del LDS.

Inoltre abbiamo gestito i certificati, come nel caso precedente abbiamo utilizzato la classe *OPCUACertificateManager* e abbiamo aggiunto un'*address space* con una variabile per la lettura della memoria libera sul server.

ServerCs

Per realizzare il server in C# abbiamo utilizzato un esempio di server presente nello stack UA-.NETStandard.

In questo stack, per impostazione predefinita, ha configurato tutti i server di esempio per la registrazione con un Local Discovery Server (LDS).

Aggiungendo l'endpoint del LDS come configurazione, lo stack gestisce automaticamente la registrazione al LDS.
Anche in questo caso abbiamo gestito opportunamente i certificati.

ServerC

Utilizzando lo stack open62541 abbiamo realizzato un server in c/c++.
Anche in questo caso la gestione della registrazione del server ad un LDS è abbastanza trasparente.

Dopo aver realizzato e configurato un semplice server è bastato richiamare la funzione ***UA_Server_addPeriodicServerRegisterCallback*** passando l'endpoint del LDS per effettuare la richiesta di registrazione al LDS.

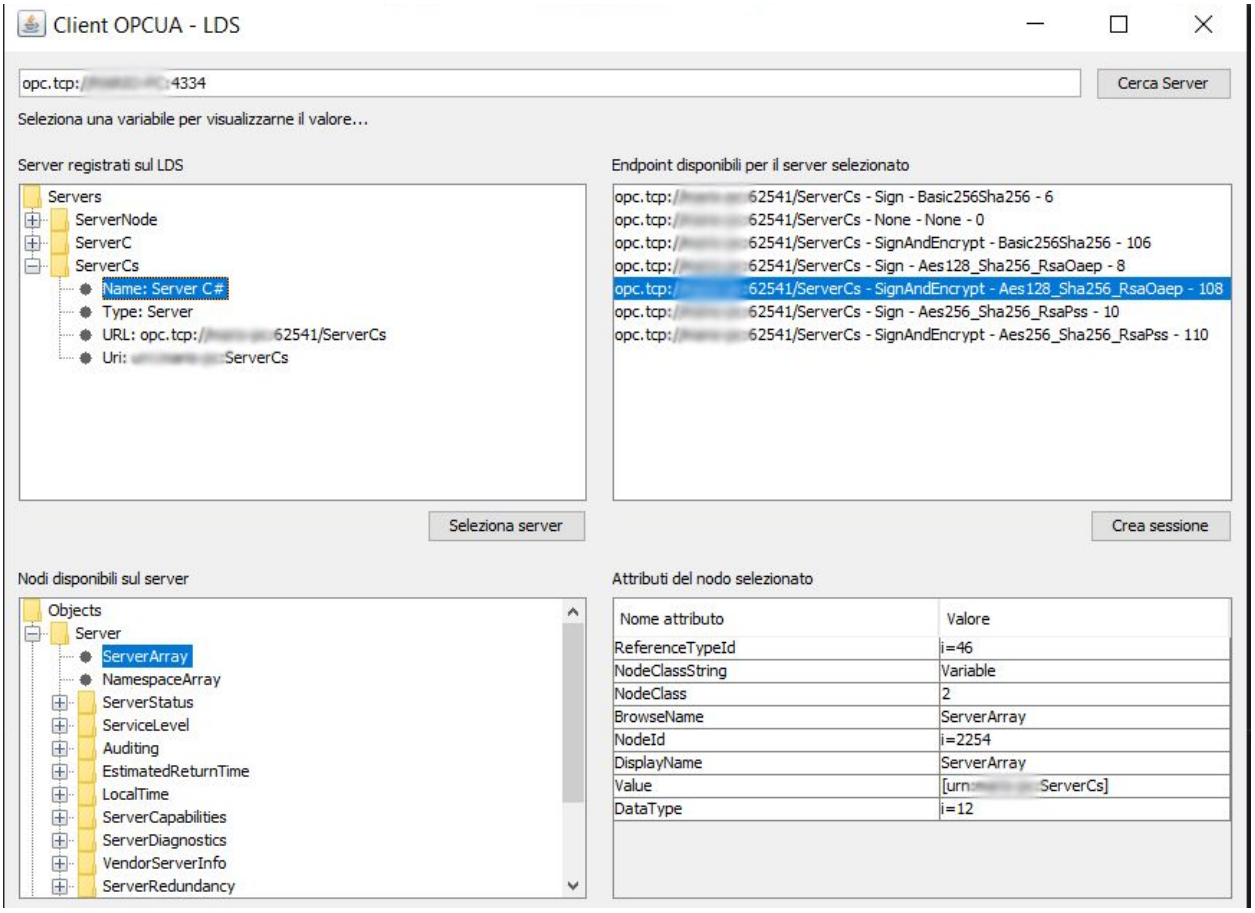
ClientOPCUA_LDS

La realizzazione del client è stata fatta utilizzando lo stack UA-Java-Legacy scritto in Java.

Abbiamo iniziato realizzando la classe CientOPCUA implementando dei metodi per le varie operazioni da eseguire al LDS e ai server. In particolare abbiamo realizzato:

- **findServers**: recupera attraverso l'endpoint del LDS i server che si sono registrati
- **discoverEndpoints**: recupera tutti gli endpoint disponibili del server selezionato
- **createSession**: crea una sessione sicura con il server selezionato
- **closeSession**: metodo richiamato per la chiusura della sessione stabilita con il server
- **getBrowse**: permette di effettuare la browse sul server connesso specificando uno specifico nodo
- **getAttributes**: effettua una read su un nodo specifico del server connesso

Questi metodi vengono utilizzati da una GUI opportunamente realizzata.



Inserendo l'url del LDS nella parte alta si va a richiamare il metodo ***findServers***, permettendo il recupero e la visualizzazione dei server registrati nel LDS.

Una volta selezionato un server, viene richiamata la funzione ***discoverEndpoints*** che restituisce la lista degli endpoints disponibili del server.

Scelto l'endpoint desiderato, è possibile stabilire la sessione con il server e l'endpoint selezionato, utilizzando il metodo ***createSession***. Contemporaneamente viene richiamato anche il metodo ***getBrowse***, per permettere di recuperare l'ObjectsFolder visualizzando così i nodi disponibili.

Cliccando uno specifico nodo, vengono richiamati i metodi ***getBrowse*** e **getAttributes**, con lo scopo di visualizzare i sottonodi e gli attributi del nodo selezionato.

La chiusura della sessione viene gestita alla chiusura del client o quando si cambia server/endpoint.

Python-opcua & Eclipse Milo

Abbiamo provato anche a realizzare dei server con gli stack python-opcua & eclipse milo, ma sfortunatamente i servizi di discovery non sono momentaneamente implementati.

I dettagli per l'installazione sono riportati nel file README.md