⇒ Proposal: extending the protocol

Since BioMOBY is an XML-based (http://www.w3.org/) protocol, it is possible to incorporate new elements or new attributes for existing elements. This new tags will be ignored by existing synchronous clients and servers and used by the new generation. Only two new attributes are suggested to be incorporated in the mobyData element:

The *async* attribute that will be used by asynchronous clients with:

- value “yes” to request for an asynchronous service (a unique queryID is supplied by client)
- value “poll” to request for task(queryID)-status to an asynchronous service

The *status* attribute will be used by asynchronous servers with:

- value “x%” to inform about the task-status to an asynchronous client, where x% means the percentage of evolution; and specifically 0% means service just launched and 100% (or not *status* attribute) means service complete and the <mobyData> containing the output from the service.

The proposed protocol to manage this new attributes is:

- The client init the session
  - Since synchronous clients are not prepared to use async attribute, thus servers (sync and async) will properly work in current synchronous mode
  - A (new) asynchronous client will uses the attribute async=”yes” to inform that is able to cope with asynchronous services. Rest of the mobyData contains the requested data.
- Server accept client request.
  - Synchronous servers will work properly with synchronous clients, and will not interpret the new attributes from async clients thus, they will proceed synchronously with the service, returning back an output object without status attribute: as previously was established, no status attribute is equivalent to 100% of evolution -the job has end- which is true for synchronous services and the <mobyData> element contains the output (which is also true for synchronous clients).
  - New asynchronous servers will accept the task and return back an object where the <mobyData> will contain the new attribute status=”0%” (service just launched).

At this point both client and server will work in asynchronous mode. Two ways to obtain results have been analysed: notification and polling.

- *Notification* means the service provider taking the initiative to inform to the client about the progress status of a given task. For a notification to be sent, there has to be some form of trigger. Such a trigger is defined as an “Event”, the service informing about a given step has been completed, or the task has finished, or they can also be in time intervals, it’s up to the designer. All of the events for a notification service must be defined in the service.
  - A new functionality needs to be incorporated, both on the client and server side. In the server to behave as a client (initiating communication) and in the client to act as a server to receive the notification. This *Notification Server* is a new system infrastructure component that is separate from and independent of the services and can demand a substantial modification on clients and servers.
  - Client and server must share information to identify univocally the task.
- The client polls the server to periodically request information about the progress status. When the service has ended, the server includes the output object.
  - Asynchronous clients will issue a request (by their own initiative) with attribute async=”poll” to request the task status
  - Asynchronous servers should be able to accept the polling request and inform about the task-status.

- Client must provide an unique task identifier that will be keep by the server and used to identify client requests.
- Request and response are cheaper objects since only contains the URL of the function call and the task ID. It can be estimated a polling request in consuming about 200 bytes of bandwidth. That is not very much at all, even with a thousand clients polling at the same time. Even more, since scheduling module at INB architecture records information about the average elapsed time of services, polling can be issued a couple of times during service execution.
- Strictly speaking there is not need for syntactic modification of bioMOBY protocol; just a new attribute and a new simple functionality in (asynchronous) servers and clients. Asynchronous clients need to take the initiative to get information from the server.
- Including the ability of bioMOBY API to inform when a service is able to work in asynchronous mode would extend the functionality (this fact can be defined during the registration procedure).

The proposal is simple (see Figure 1) and flexible enough to include notification information (i.e. “step 2: sequence alignment done”) to report progress status from the requested service.

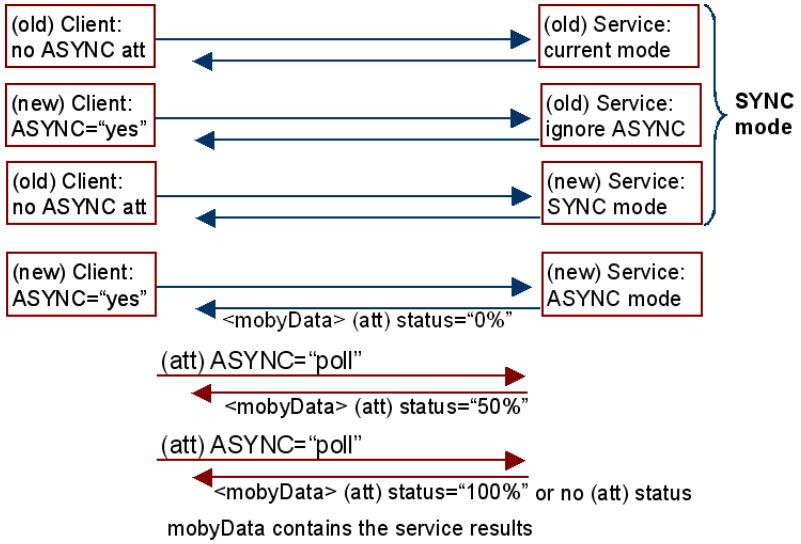


**Figure 1**.- Schematic representation of the proposed asynchronous protocol showing the four different combinations between sync and async clients and servers and the way in which they will behave.