Appendix F: Extended Sample Application Descriptions and Extensions Requirements

1. Connectionless *Levenshtein* Edit-Distance Calculator
   1. Description of Existing Application

This system allows user to enter two word into client console, which then requests a server to compute the Levenshtein Distance, LD, between the two words, where LD is the minimum number of single-character edits (insertion, deletion, substitution) required to change one word into the other. For example, the LD between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

* **k**itten → **s**itten (substitution of "s" for "k")
* sitt**e**n → sitt**i**n (substitution of "i" for "e")
* sittin → sittin**g** (insertion of "g" at the end)
  1. Design Enhancements

This version design is similar to that initial application description. Figure 1 describes the architecture where as Figure 2 describes the interactions between Client and Edit-Distance Calculator. However this version has the following differences from its initial draft:

* Communication between Client and Edit-Distance Calculator happens using connectionless protocol or user datagram protocol (UDP). [***GIVEN***]
* Client will send the request, encapsulated in TranslationRequestMessage.
* Server will send the response, encapsulated in TranslationResponseMessage.
* The message class will be using just MessageID attribute of type UUID instead of RequestID and ResponseID.

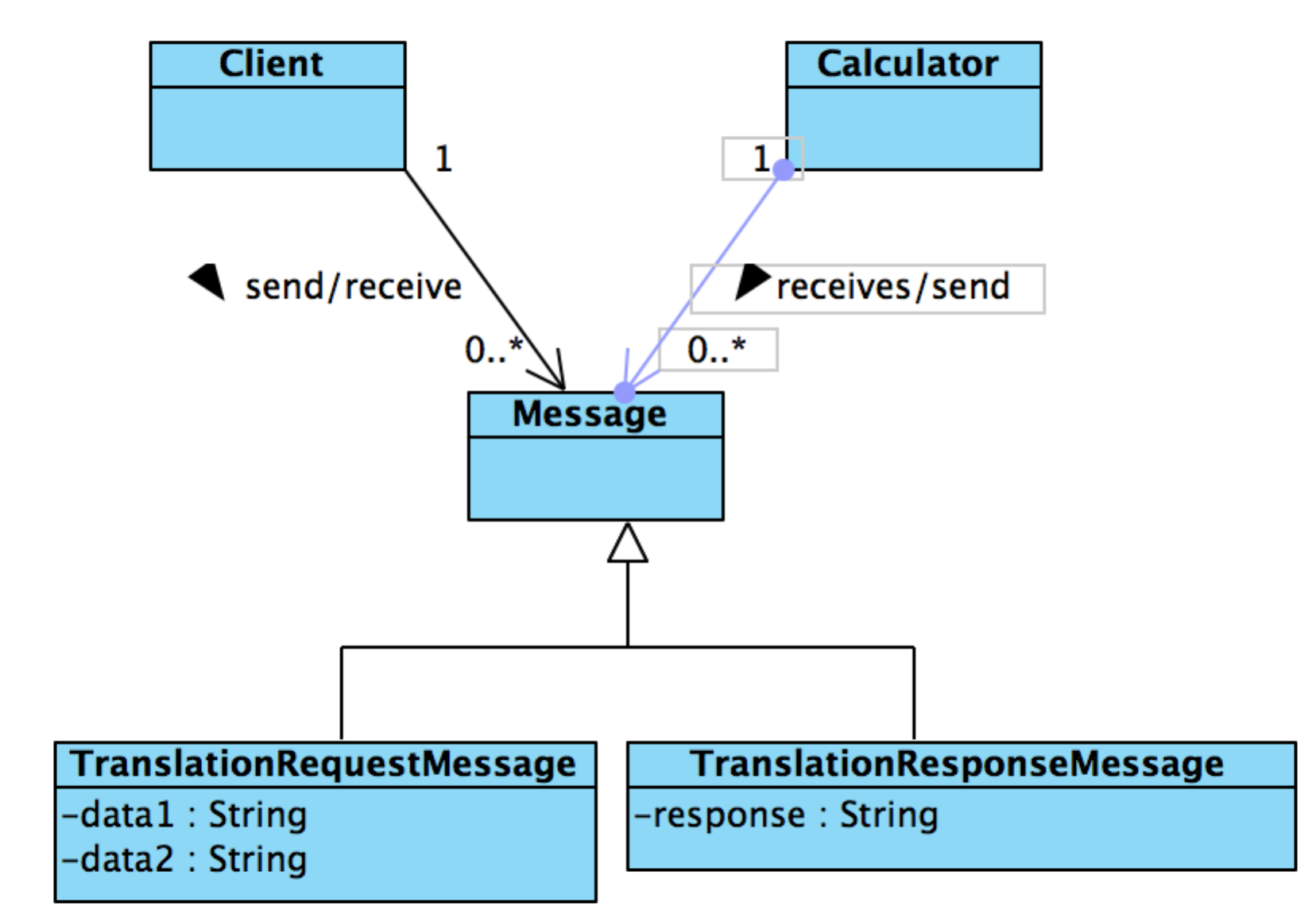


Figure 1: Architecture diagram of Levenshtein Edit-Distance Calculator

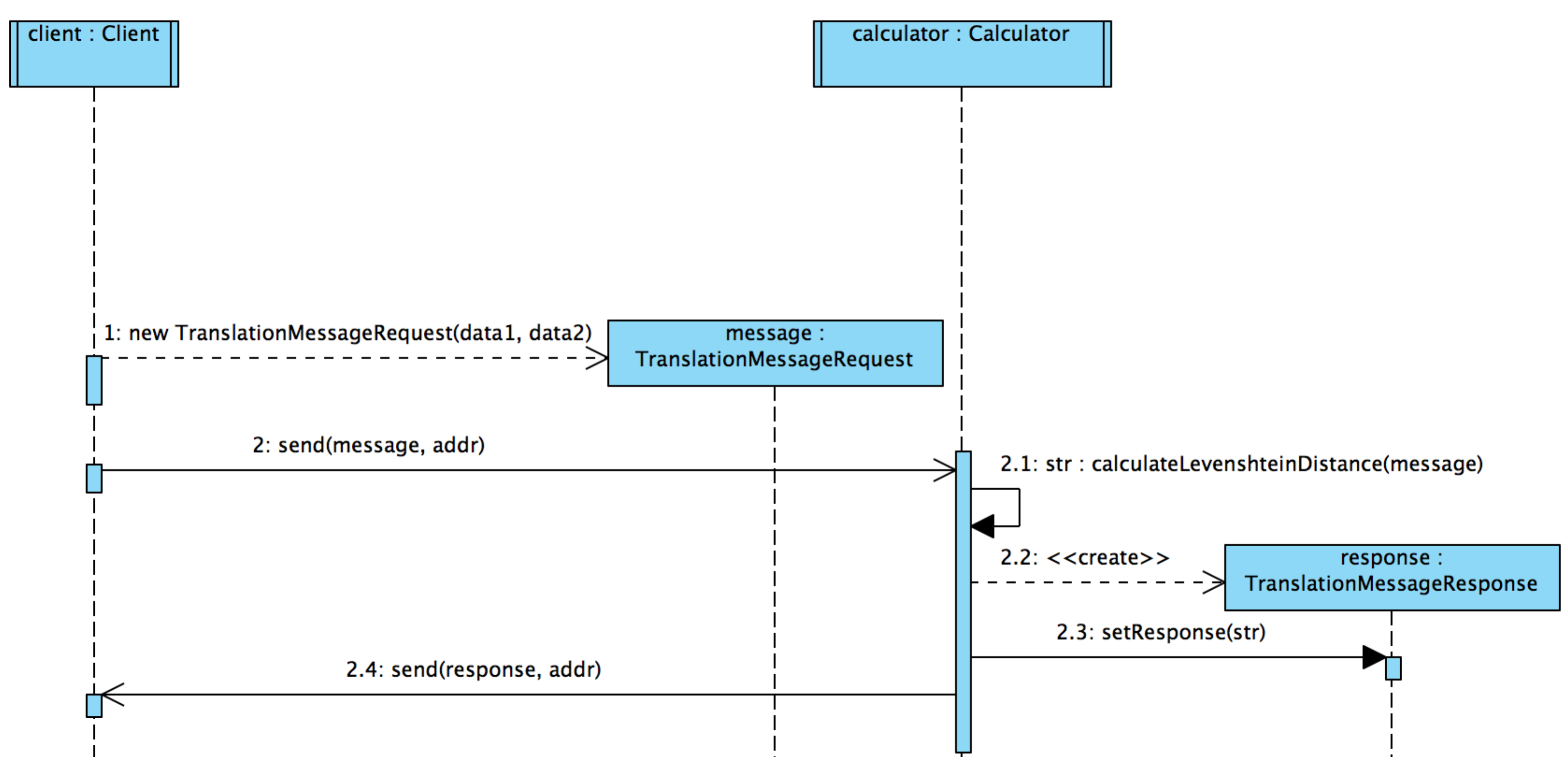


Figure 2: Interaction diagram between Client and Edit-Distance Calculator