Appendix B: Selected sample applications

1. *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. Current Design

Figure 1 shows an overview of the current architecture for this system. It only contains three classes, Client, calculator, and Message. Both the Client and calculator run as separate processes, and maybe even on separate machines. The Client allows the users to type in two words using a simple console interface. Then, it creates an instance of Message class containing these two words and sends it to the calculator. The calculator computes the LD, package that result up in a new instance of Message, and send it back to Client. The UML Sequence Diagram in Figure 2 shows this interaction.

Note, that the interaction is asynchronous from the Client’s perspective. In other words, the Client does not block while waiting for a response to the translation request.

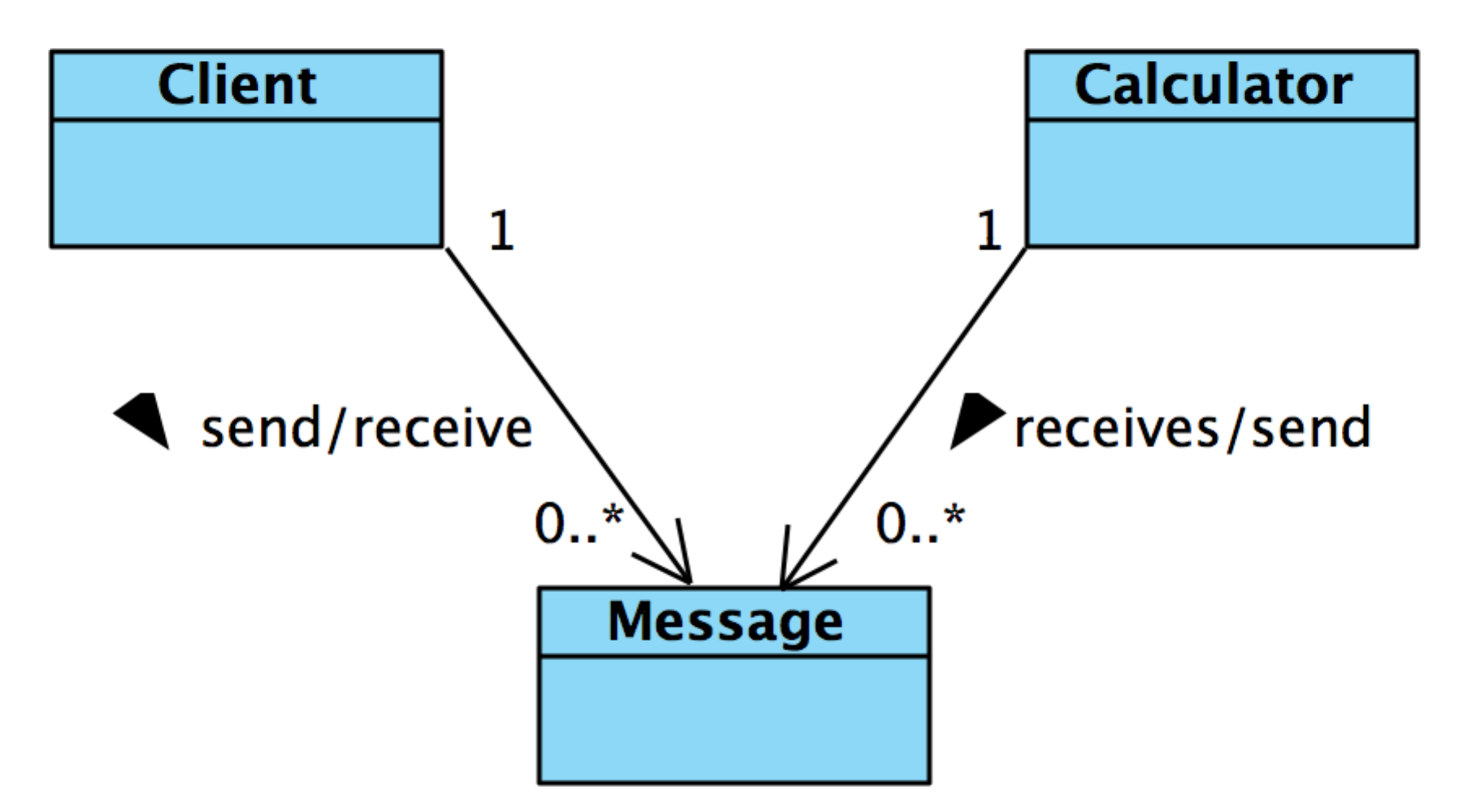


Figure 1: Architecture diagram of Levenshtein Edit-Distance Calculator

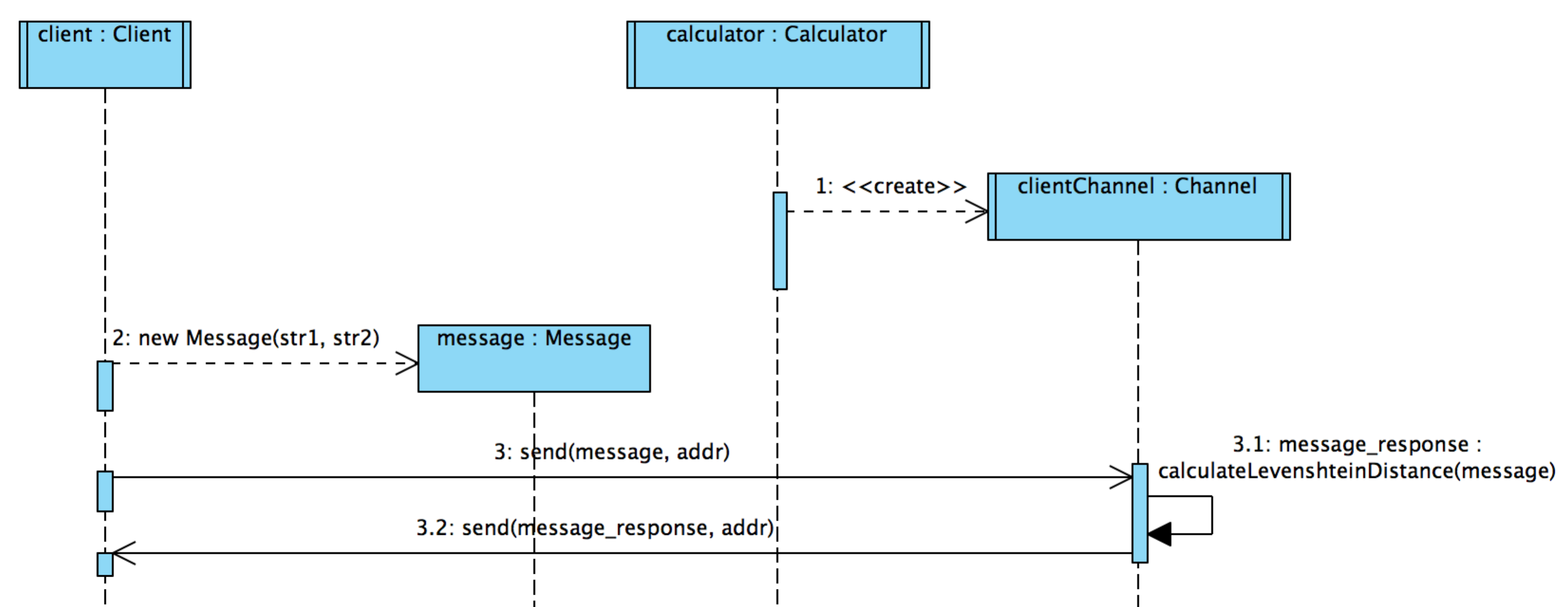


Figure 2: Interaction diagram between Client and Edit-Distance Calculator

1. Measuring Performance
   1. Introduction

It measures some performance related statistics for message-based communications between a sender and receiver.

* 1. Description
* The extension logs following performance related statistics:
  + Total numbers of conversations, which occurred in the system where a conversation can be defined with any combinations of, sends or receives. Different types of conversations are one-way send, one-way receive, request-reply and multi-step conversations
  + Total time for all conversations
  + Average turnaround time for a request to be processed where average turn-around time is the average of a timespan from conversation start time to conversation end time
  + Maximum turnaround time for any conversations
  + Minimum turnaround time for any conversation
* The program logs the time when a conversation starts
* It logs and calculates the above statistics when the conversation ends
* Note that a conversation can be a simple request-reply type exchange of messages or a complex combination of send and receive events. We are defining the conversations for sample applications as follow:
  + *Levenshtein Edit-Distance Calculator:* A conversation is when a client sends a request and receives a response from the calculator
  + *File Transfer Protocol:* A conversation is when a client sends a request for a file download and when it receives the last response of data chunk for that file from the server
  + *Weather Station Simulator:* A conversation is when a receiver sends a request to get weather related data readings and receives the first response from the transmitter
  1. Application Material
* Developers would be provided with the following classes:
* *Stats:* A data structure containing elements to measure performance
* *PerformanceMeasure:* It logs performance measure using sliding window