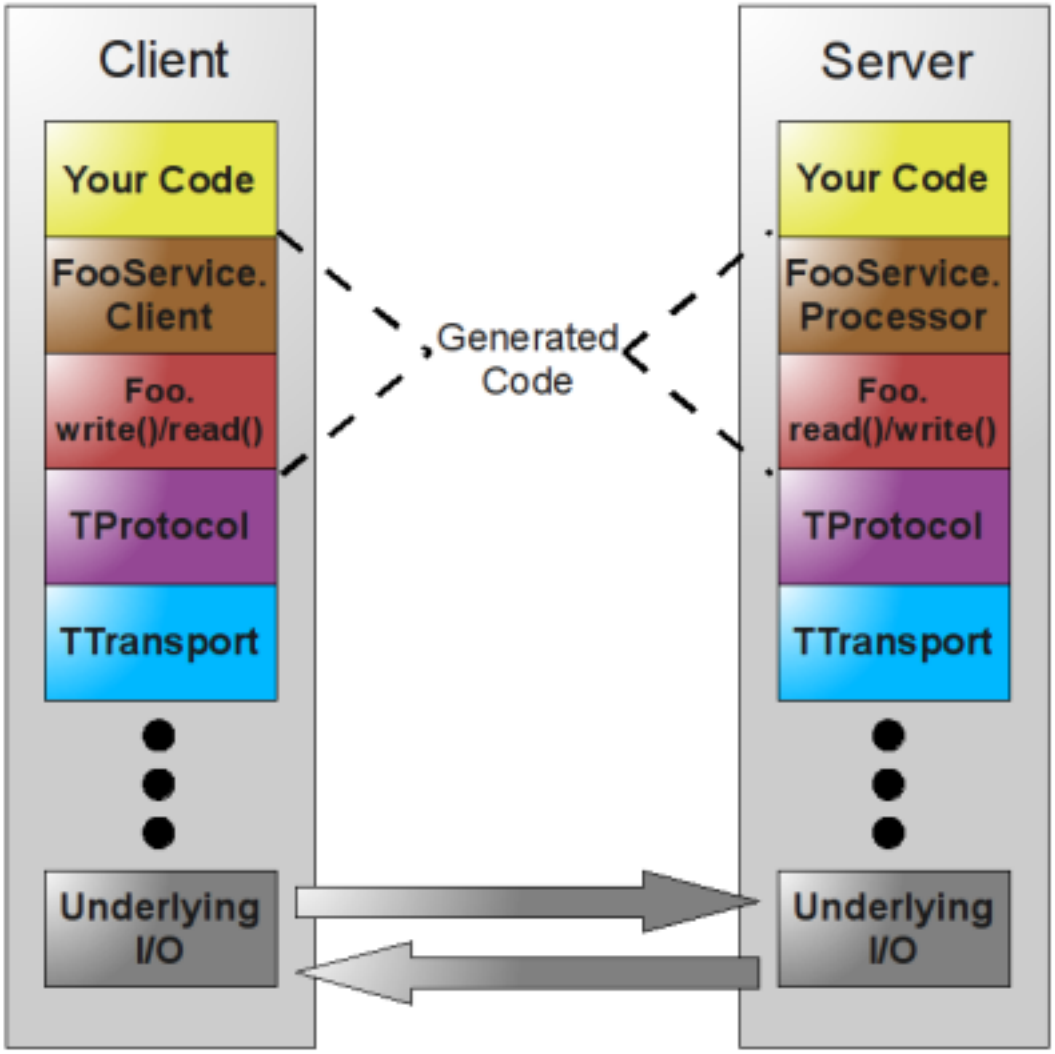
Server內部通訊架構

目前採用Thrift framework，這是一個服務端和客戶端的架構體系，具有自己內部定義的傳輸協議規範(TProtocol)和傳輸數據標準(TTransports)，通過IDL腳本對傳輸數據的數據結構(struct) 和傳輸數據的業務邏輯(service)根據不同的運行環境快速的構建相應的代碼，並且通過自己內部的序列化機制對傳輸的數據進行簡化和壓縮提高高並發、 大型系統中數據交互的成本，下圖描繪了Thrift的整體架構，分為6個部分：1.你的業務邏輯實現(You Code) 2.客戶端和服務端對應的Service 3.執行讀寫操作的計算結果4.TProtocol 5.TTransports 6.底層I/O通信



上圖中前面3個部分是1.你通過Thrift腳本文件生成的代碼，2.圖中的褐色框部分是你根據生成代碼構建的客戶端和處理器的代碼，3.圖中紅色的部分是2 端產生的計算結果。從TProtocol下面3個部分是Thrift的傳輸體系和傳輸協議以及底層I/O通信，Thrift並且提供堵塞、非阻塞，單線程、多線程的模式運行在服務器上，還可以配合服務器/容器一起運行，可以和現有JEE服務器/Web容器無縫的結合。

數據類型

* Base Types：基本類型

* Struct：結構體類型

* Container：容器類型，即List、Set、Map

* Exception：異常類型

* Service： 定義對象的接口，和一系列方法

協議

目前選擇客戶端與服務端之間傳輸通信協議，為節約帶寬，提供傳輸效率所以使用二進制類型的傳輸協議

傳輸層

* TFramedTransport- 使用非阻塞方式，按塊的大小，進行傳輸，類似於Java中的NIO。

服務端類型

* TNonblockingServer – 多線程服務器端使用非堵塞式I/O

異常處理

可以自行定義Exception，在遠端呼叫時發生錯誤，丟出exception，即可透過網路讓local端收到錯誤通知