Elective exercise using Go and RPC

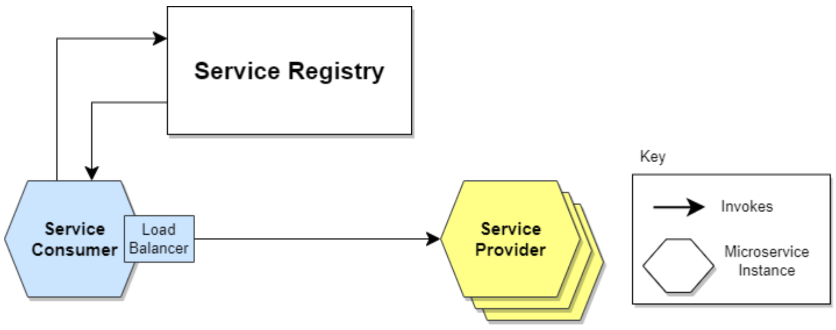
Corso di Sistemi Distribuiti e Cloud Computing
A.A. 2025/26

Valeria Cardellini

Laurea Magistrale in Ingegneria Informatica

Elective exercise using Go and RPC

- Realize a distributed application using client-side service discovery and a service registry



- Requirements:
  – Use either Go and RPC or Go and gRPC
  – Organize properly your code into separate files
  – 1 student per team (2 students: optional part is mandatory)

Overview: architecture

- Replicated servers
  - Multiple servers that execute the same services
  - A stateless service of your choice
  - A stateful service of your choice (state must to be kept consistent across multiple replicas)
- The client queries the service registry to get a list of available servers
- The client selects a server using a client-side load balancing algorithm and sends an RPC request
- The selected server executes the service and sends back a response
- The client caches the server list for future use within the same session, avoiding multiple requests to the service registry

Overview: architecture

- Each server
  - Registers with the service registry
  - When it shuts down, it deregisters itself from the service registry
- Stateless service
  - Simple, no state
- Stateful service
  - Maintains state across requests
  - How? Multiple solutions you can consider, including
    - Duplicate state across multiple servers using a consistency protocol (e.g., quorum-based, primary-backup)
    - Store state externally (e.g., key-value store, database)

Application setup and assumptions

- Single machine: all nodes (clients, servers, service registry) can execute on the same machine
  - IP address = localhost
- Simplifying assumptions
  - No failures: clients and servers do not fail during computation
  - The service registry remains available
- There can be multiple clients accessing the servers concurrently

Optional

- Two client-side load balancing algorithms
- Containerize your application
  - Use the official Go image https://hub.docker.com/_/golang
  - Containerize each application component
  - Use Docker Compose for orchestration
- 2 students per team: mandatory

Delivery

- When
  - By January 21, 2026
- What
  - Your code, including a README with instructions to run it
  - Optional: very short report describing the architecture of your distributed application and its main features
- How
  - By email
  - Use as mail subject: [SDCC] consegna esercizio in Go