Desafío gRPC

Para validar tus conocimientos en este tema te proponemos resolver el siguiente desafío:

Tendrás que programar **2 servidores que ejecutan exactamente el mismo código**, estos servidores deberán implementar el siguiente servicio gRPC:

```
service Challenge {
 rpc Ping (Empty) returns (ServiceReply);
 rpc Add (AddRequest) returns (AddResponse);
 rpc Forward (ForwardRequest) returns (ServiceReply);
}
```

Ping

Este método de rpc recibirá una solicitud vacía y deberá responder con un mensaje con el valor: “Pong!”

La solicitud vacía estará definida como:

```
message Empty {}
```

Mientras que la respuesta la tendrás que implementar tú

Add

Este servicio sumará 2 números de 32 bits y responderá usando un número de 64 bits como se especifica en sus solicitudes y respuestas:

```
message AddRequest {
 int32 num1 = 1;
 int32 num2 = 2;
}

message AddResponse {
 int64 result = 1;
}
```

Forward

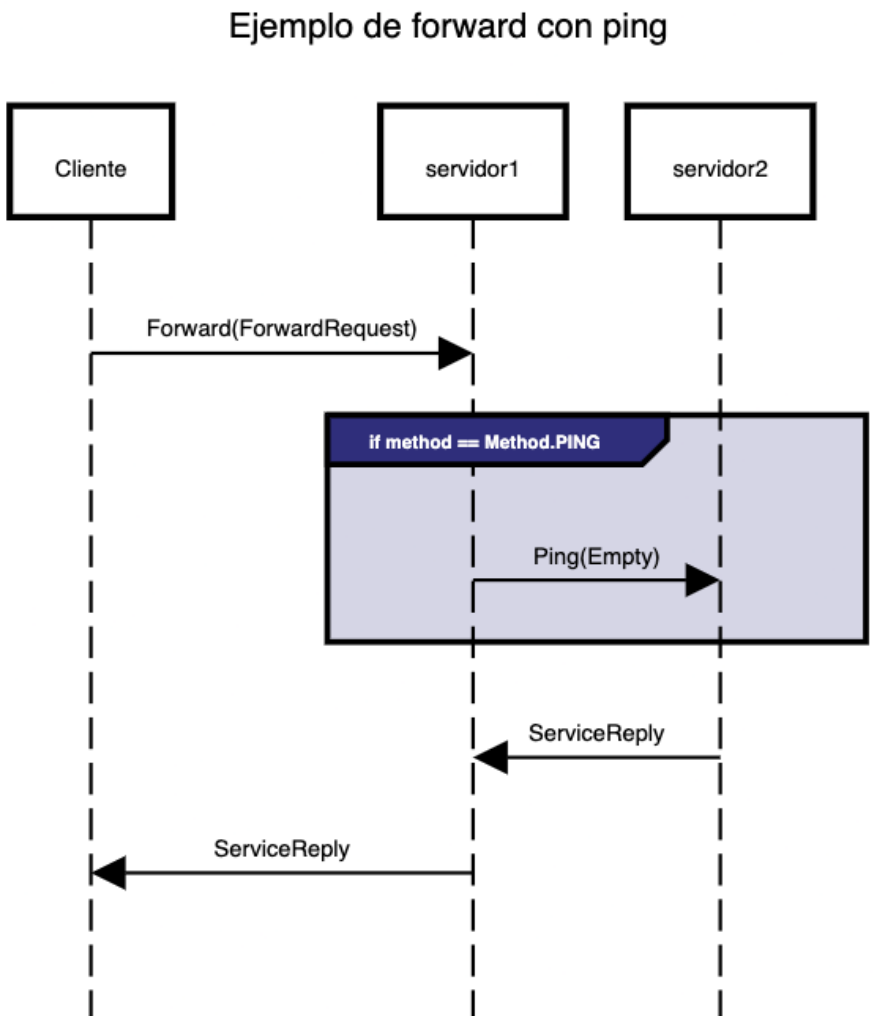
Este método deberá ejecutar una nueva solicitud hacia otro servidor que implemente el mismo servicio “Challenge”. Esta solicitud puede ser de 2 tipos especificado por el enum Method como se muestra a continuación:

```
enum Method {
 PING = 0;
 FORWARD = 1;
}

message ForwardRequest {
 string host = 1;
 Method method = 2;
}
```

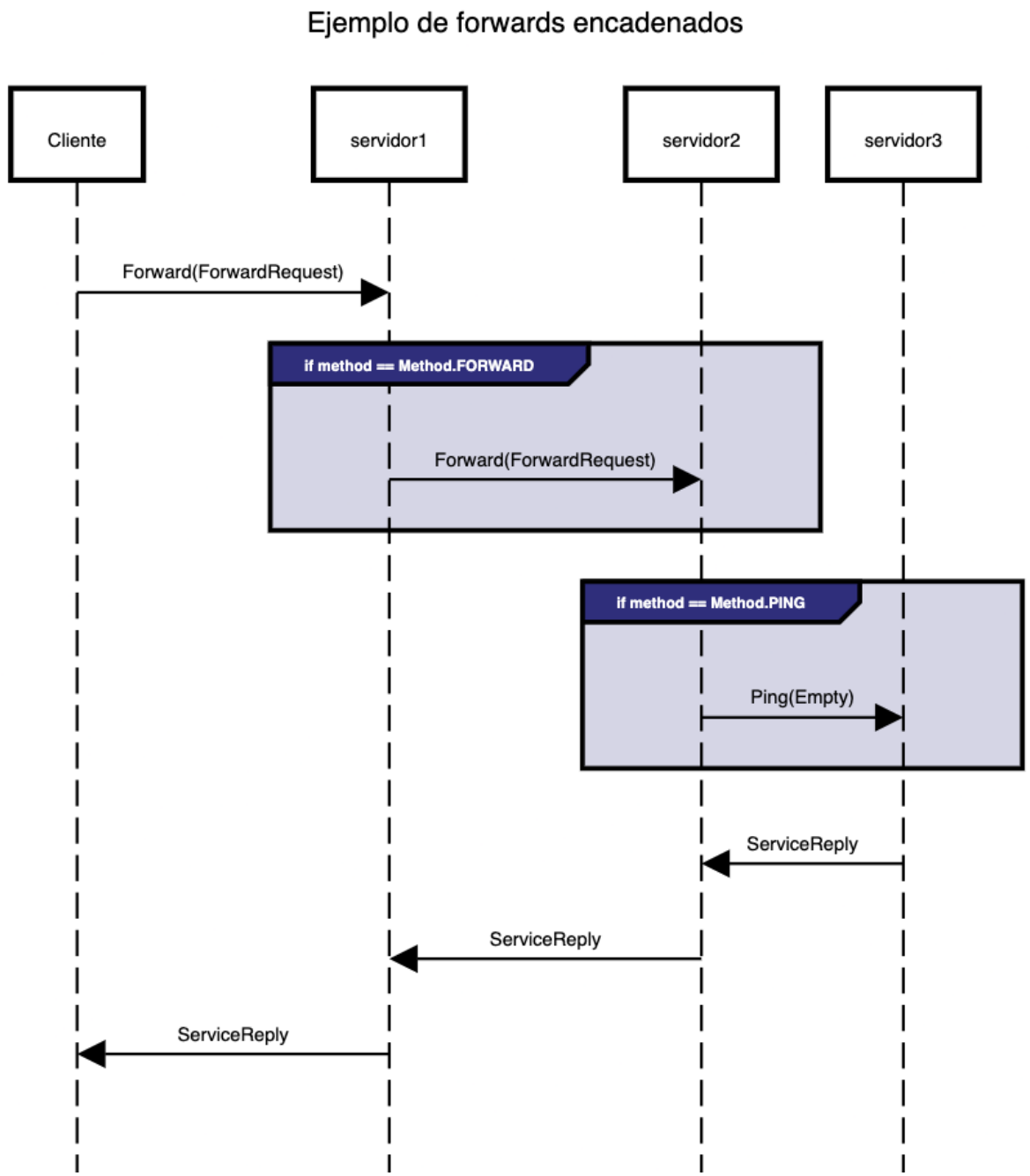
Forward ejecutando ping

En caso de que recibamos una solicitud de tipo forward con un **method = PING** deberemos ejecutar una solicitud ping en el servicio de destino especificado por el campo **host**
(por este motivo el método rpc de Ping y Forward tienen el mismo tipo de respuesta)



Forward ejecutando un nuevo forward

En caso de que el forward especifique ejecutar un nuevo forward y ese nuevo forward corresponda a un ping debemos ejecutarlo y responder con la respuesta del último servicio que llamamos.
Siguiendo esta lógica deberíamos poder soportar múltiples forwards donde siempre alguno termina siendo un ping



Ejercicio

Dado que implementar todo esto desde cero puede superar el alcance de este desafío y siendo que el objetivo final es aprovechar la instancia para practicar tus conocimientos en este tema es que te proponemos usar este repositorio como base para resolver lo siguiente:

1. Agregar los campos necesarios en la respuesta del Ping

```
message ServiceReply {
 //Implementar el o los campos de la respuesta
}
```

2. Implementar el código para responder la solicitud de Ping

```
func (s *ChallengeServer) Ping(ctx context.Context, in *pb.Empty)
(*pb.ServiceReply, error) {
   //Implementar respuesta de ping y cambiar el primer nil por esa
respuesta
   return nil, nil
}
```

3. Implementar el código para ejecutar el método de Add y responder con la suma de los números contenidos en la solicitud

```
func (s *ChallengeServer) Add(ctx context.Context, in *pb.AddRequest)
(*pb.AddResponse, error) {
   //Implementar suma de num1 con num2 y cambiar el primer nil por
esa respuesta
   return nil, nil
}
```

4. Implementar el campo necesario para ejecutar una cadena de Forwards

```
message ForwardRequest {
 string host = 1;
 Method method = 2;
 //Implementar el campo que hace falta para que el servidor de
destino ejecute un nuevo forward
}
```

5. Implementar el código para ejecutar Forwards encadenados.
   (Revisar el switch-case del método Forward ya implementado)

Sugerencias:

- Recomendamos resolver este ejercicio en el lenguaje Go ya que esto va a simplificar la solución al poder utilizar el repo que te brindamos como ayuda
- Puedes usar el repo de ejemplo de gRPC que vimos en clase como ayuda
- Recuerda que puedes usar postman para ejecutar requests gRPC como vimos en clase
- Volver a ver la presentación de gRPC que vimos en clase puede ser de utilidad
- Puedes especificar el puerto en el que quieres ejecutar el servidor de la siguiente forma:

  ```
  go run server/main.go --port 50052
  ```

  Esto ejecutará el servidor en el puerto 50052

Entrega:

Deberás entregar un archivo de **docker-compose.yml** junto con el código que contiene la implementación de ambos servidores.
A su vez deberás agregar el **archivo .proto** donde tienes la definición de los servicios y un **README.md** que contenga los pasos a seguir para ejecutar y validar tu solución.

Criterios de evaluación:

30 puntos si resuelves el método de Ping
30 puntos si resuelves el método de Add
40 puntos si resuelves los Forwards encadenados