NATS pub-sub daemon packed inside your application

Valdas Petrulis

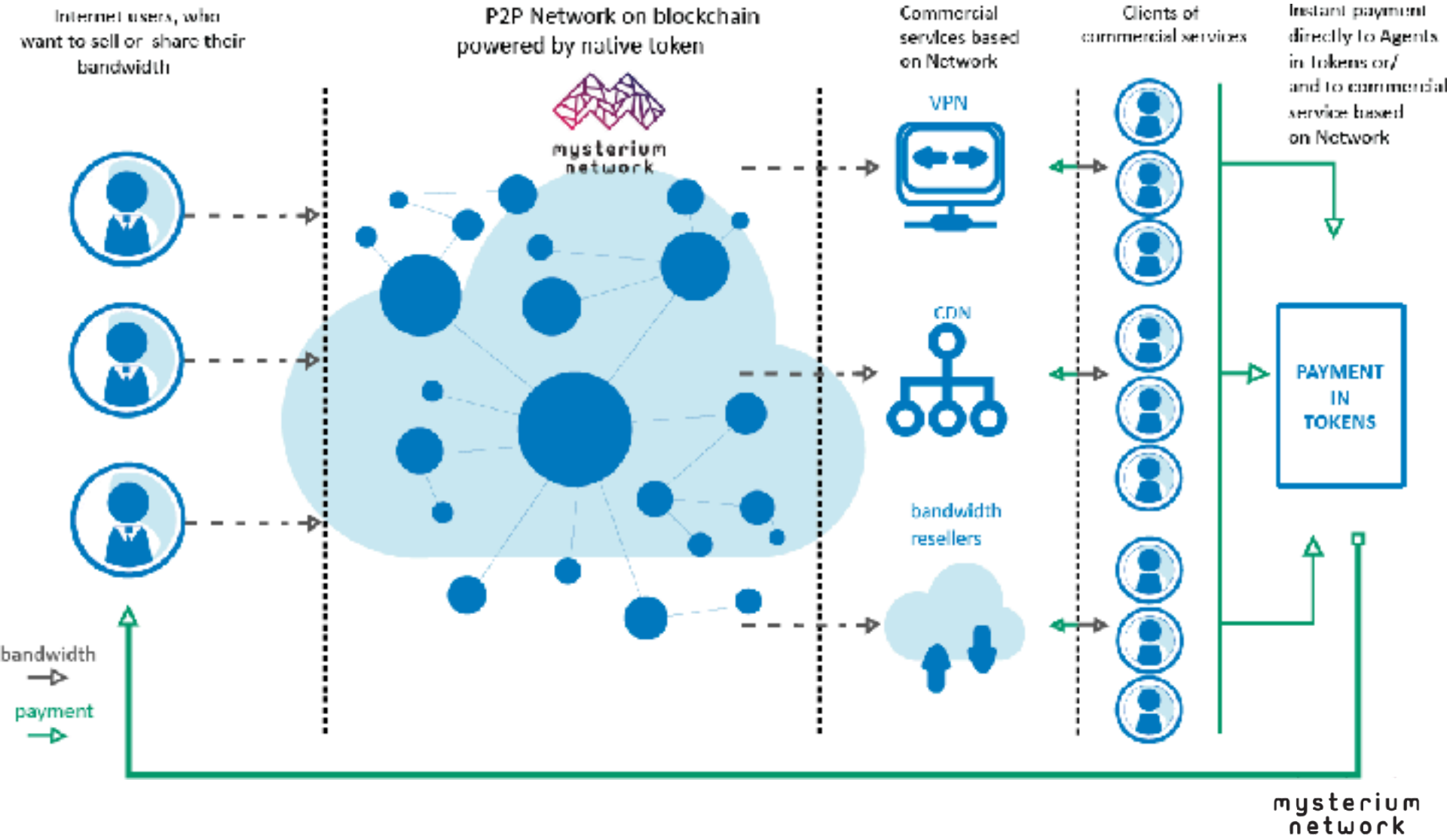
2017-09-26

Valdas Petrulis

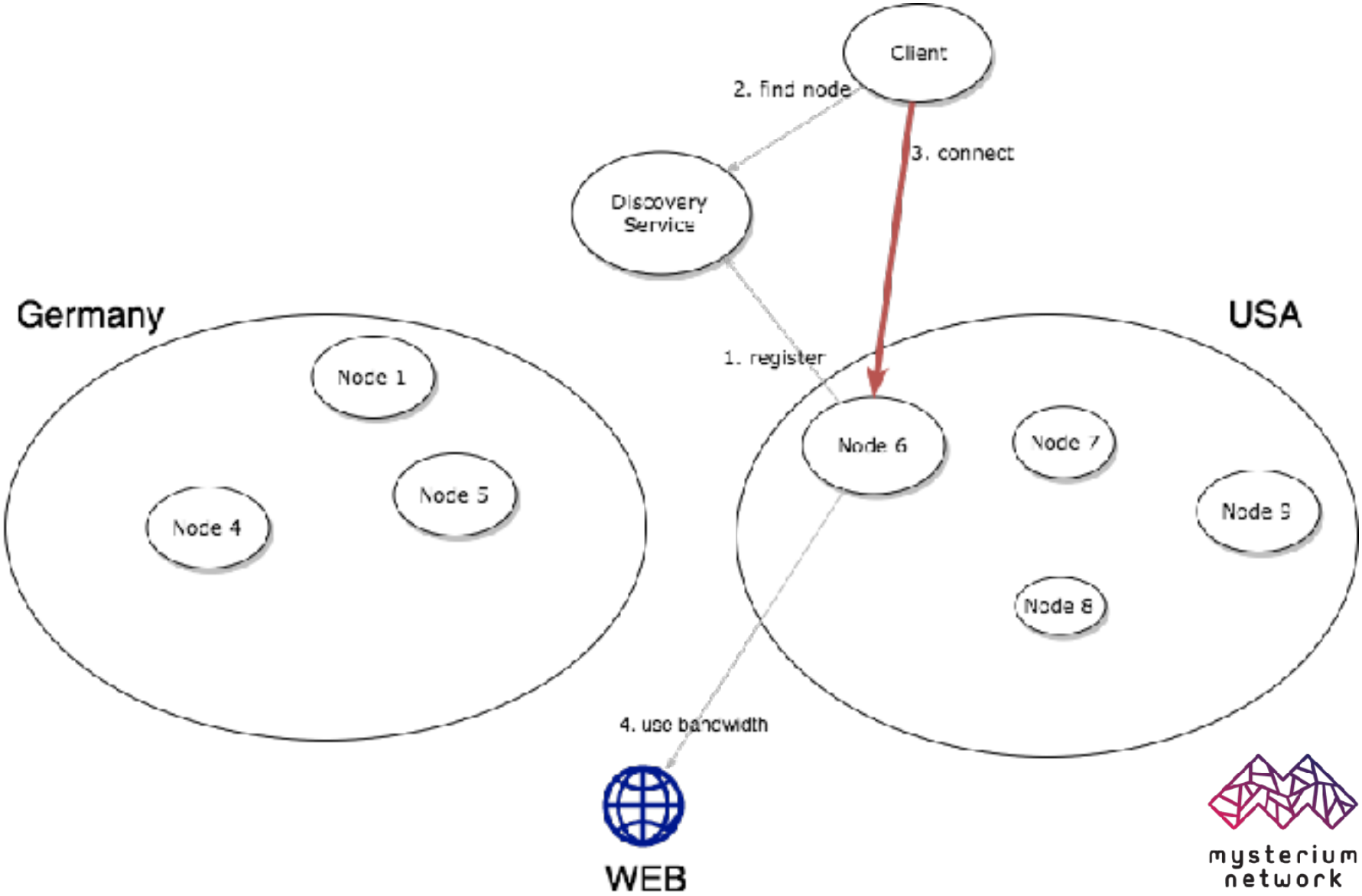
- Golang / PHP / JAVA Engineer
- Backend and Devops
- Past in Fintech, Lamoda.ru, Bigbank.eu
- Now Blockchain Developer at www.mysterium.network

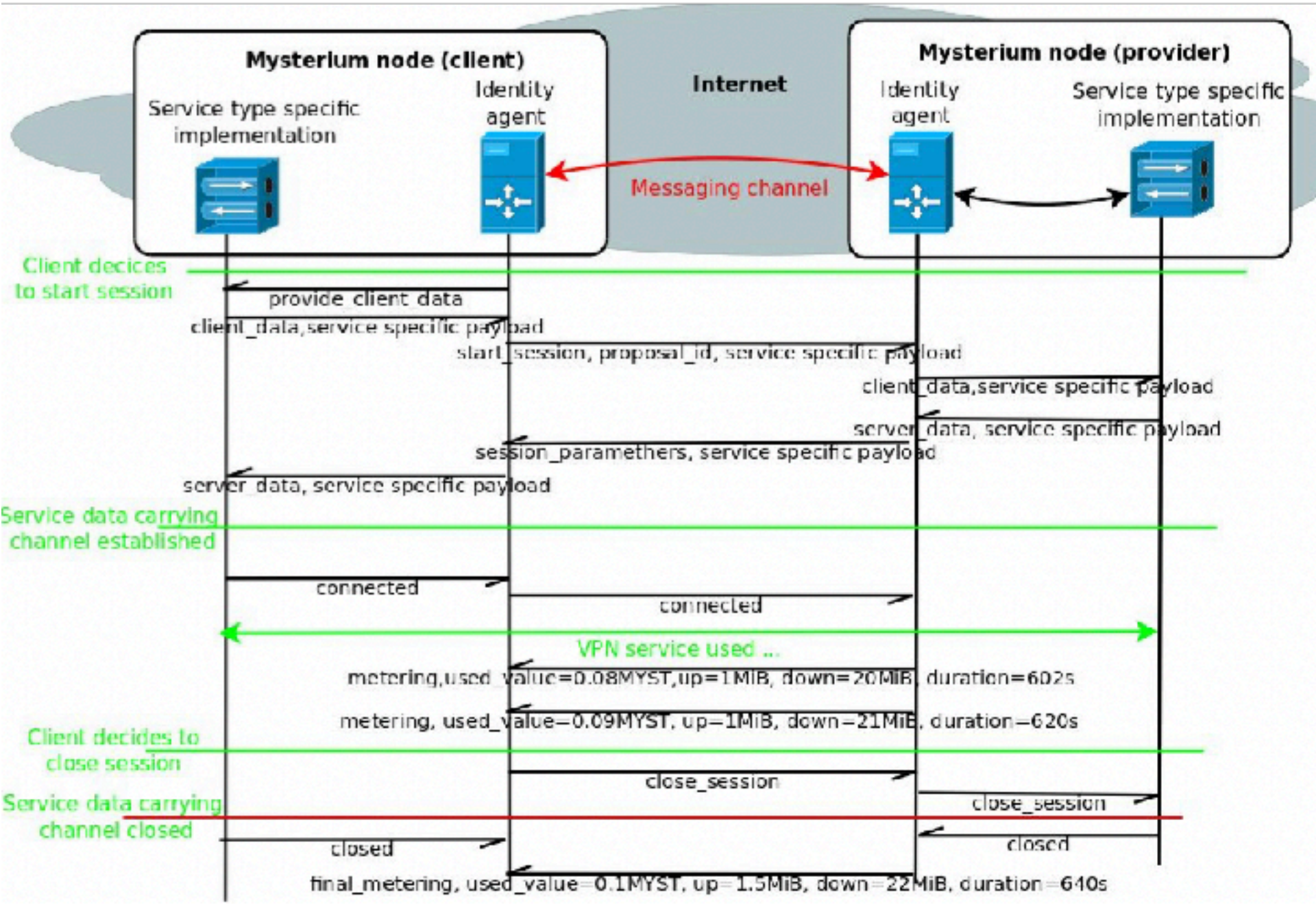
mysterium network

Doing Bandwidth Marketplace



Node Discovery Service





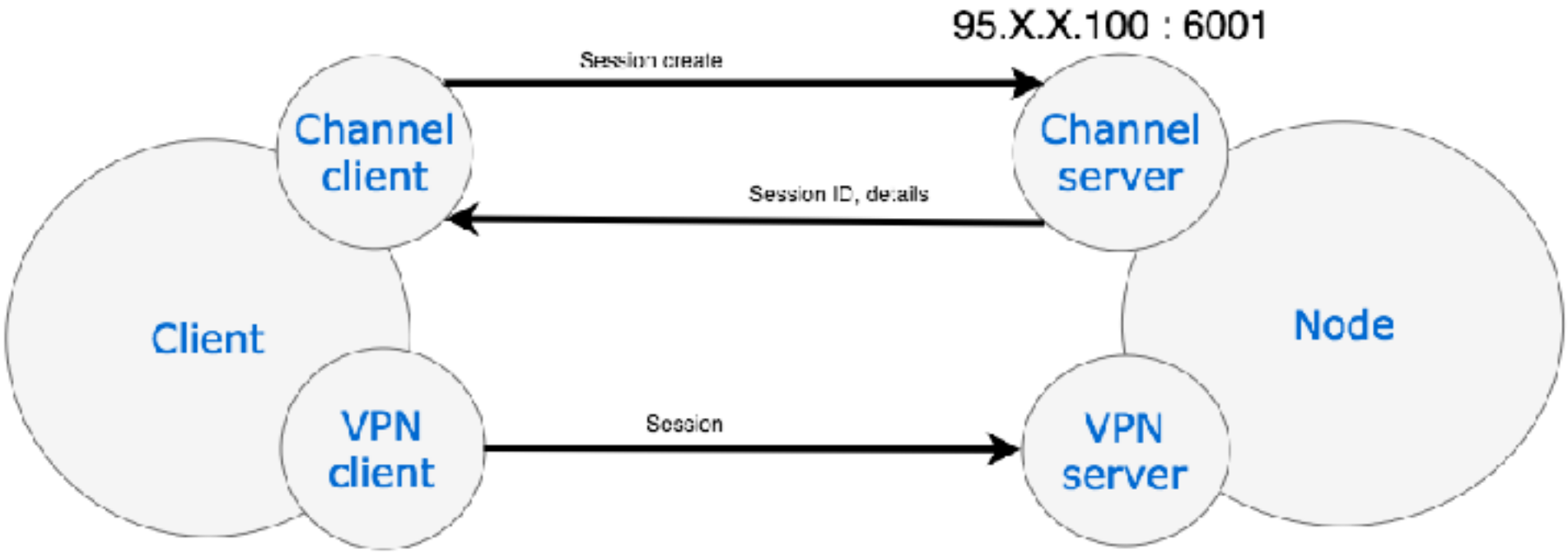
mysterium network

1-1 Communication requirements

- Client -> Node initiates session
- Node -> Client initiates session (later)
- Communication is bidirectional
- Opened channel is 1-1, secret
- Communication contact Node behind router

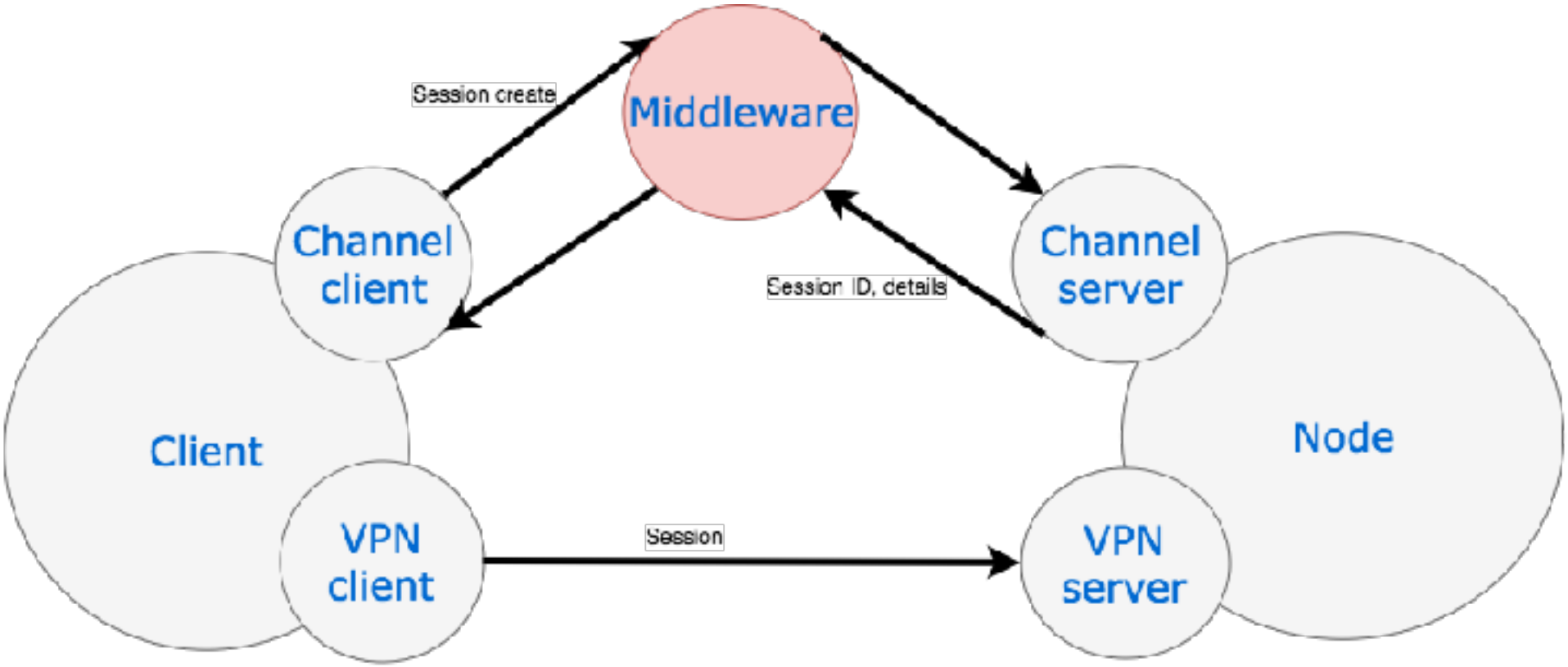
mysterium network

Communication channel (sockets)



mysterium network

Communication channel (middleware)



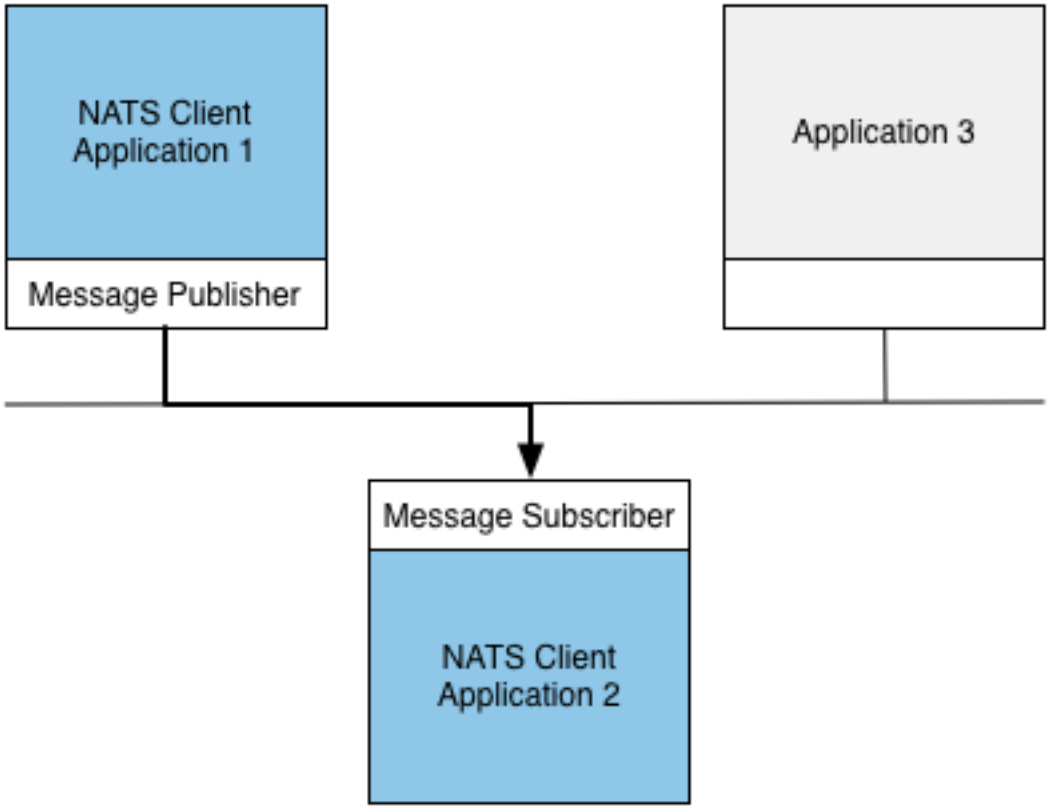
mysterium network

What is NATS?

- Messaging system (middleware)
- Lightweight, high-performance
- Server is written in the Go - nats-io/gnatsd
- Client libraries - nats-io/go-nats
- Microservice frameworks rely as their messaging backbone - Micro, Mainflux, and Hemera

mysterium network

It’s publish - subscribe



mysterium network

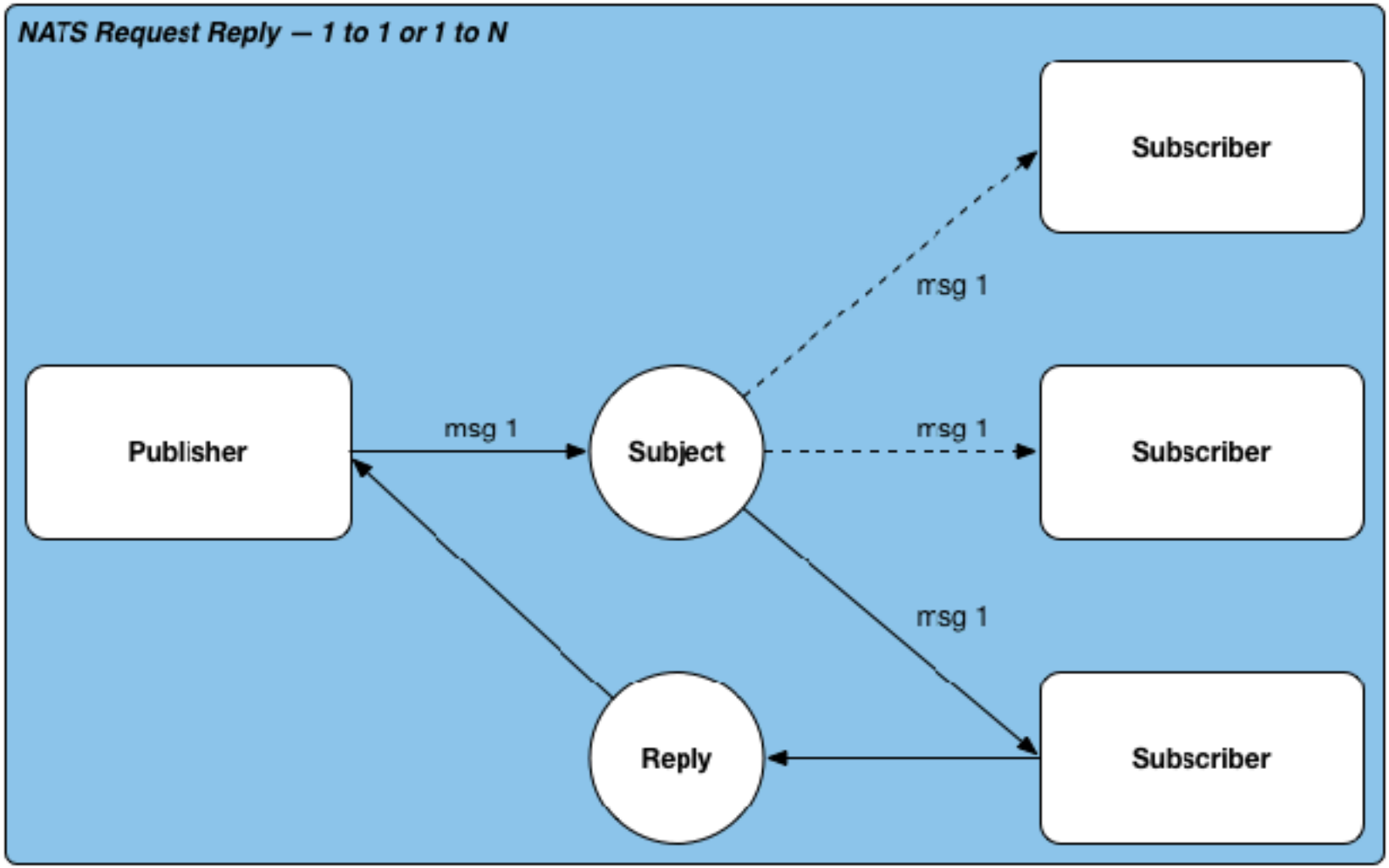
Publish - subscribe example

```
nc, _ := nats.Connect("nats://localhost:4222")

nc.Publish(“101-session-create“, []byte(“100”))
```

```
nc, _ := nats.Connect("nats://localhost:4222")

nc.Subscribe("101-session-create", func(m *nats.Msg) {
    fmt.Printf("Received a message: %s\n", string(m.Data))
})
```

mysterium network

NATS Request Reply



mysterium network

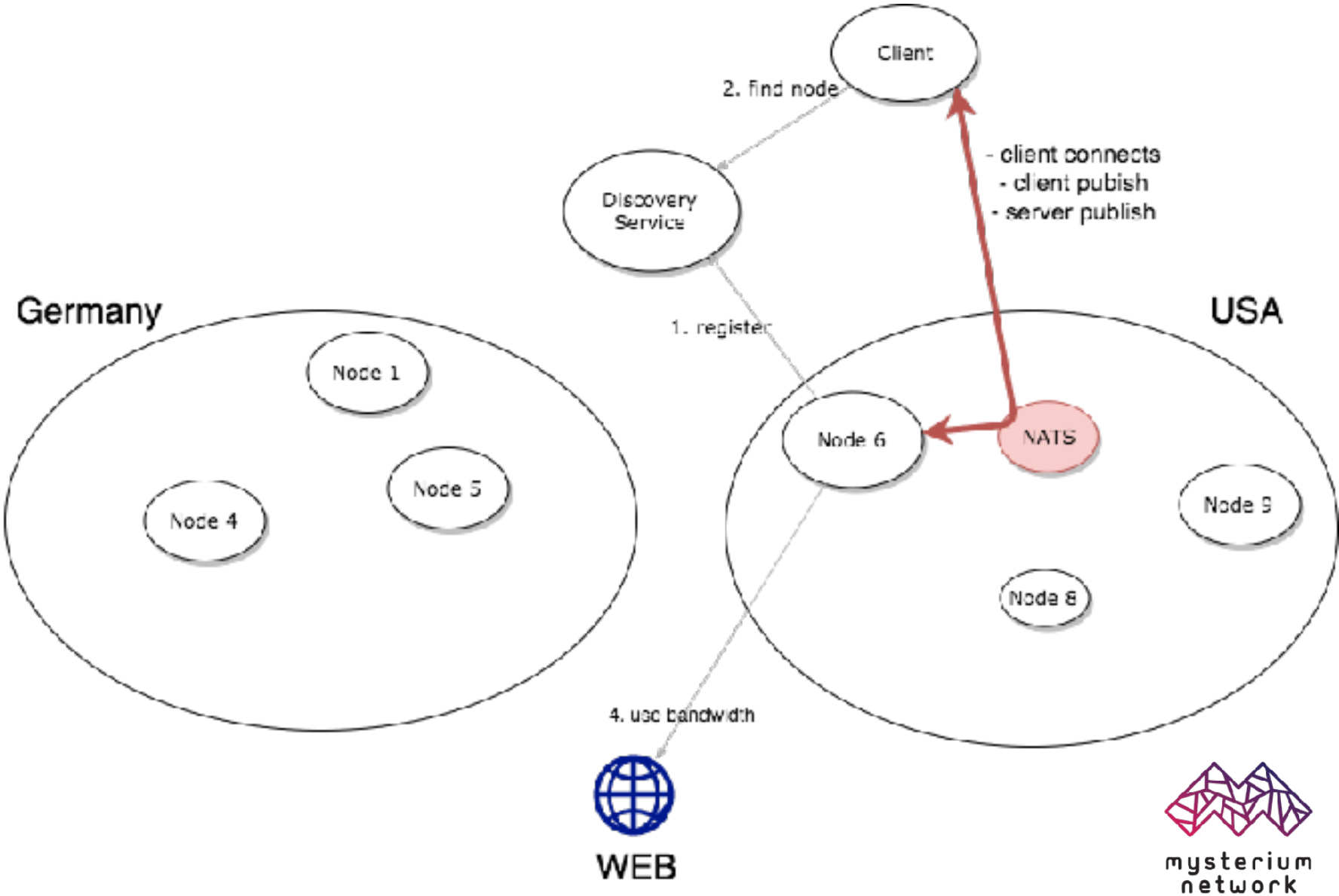
Request Reply example

```
nc, _ := nats.Connect("nats://localhost:4222")

response, err := nc.Request("101-session-create",
[]byte("100"), 500*time.Millisecond)
fmt.Printf("Response: %s\n", string(response.Data))
```

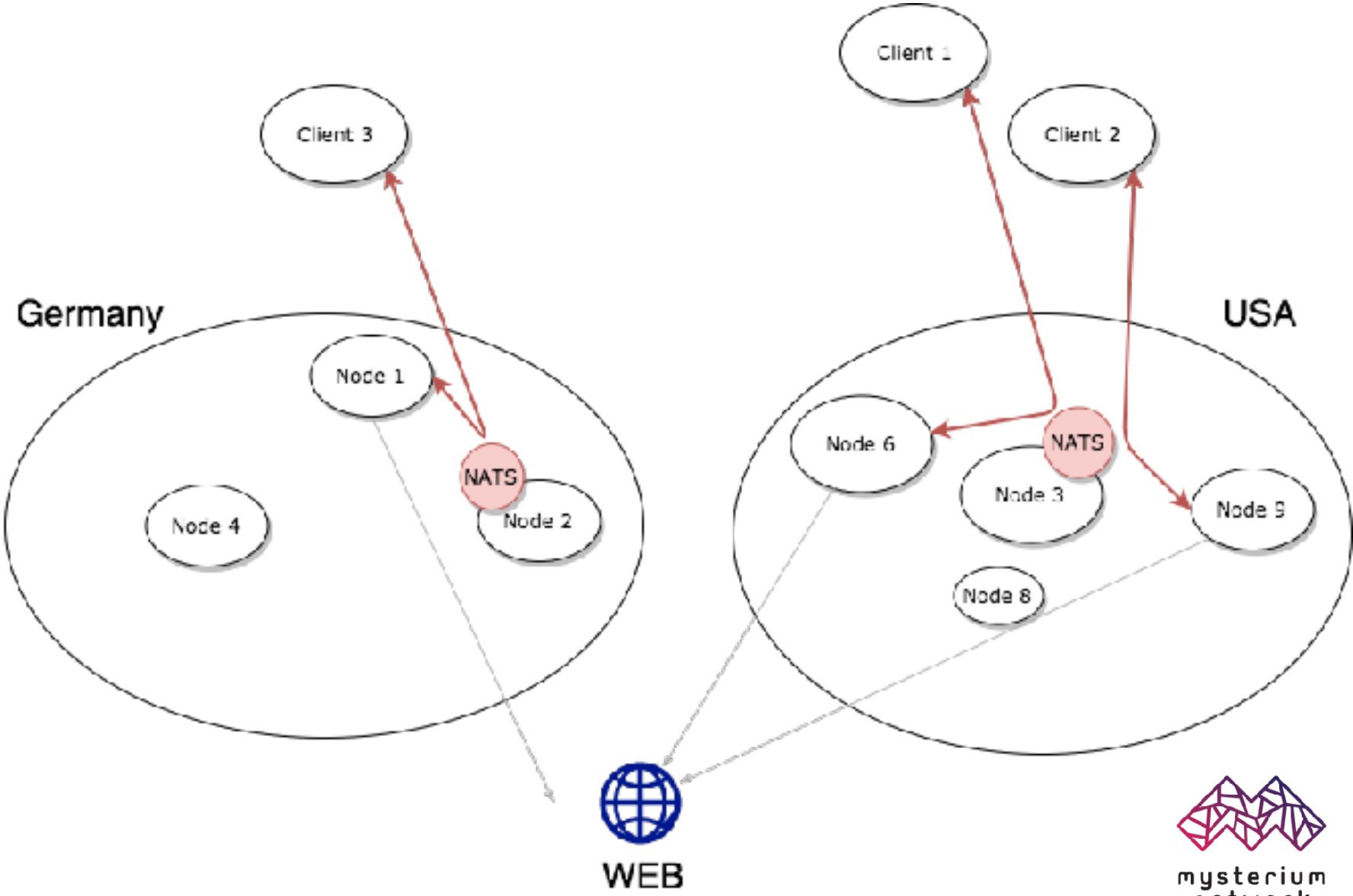
```
nc, _ := nats.Connect("nats://localhost:4222")

nc.Subscribe("101-session-create", func(m *Msg) {
    nc.Publish(m.Reply, []byte("ACCEPTED"))
})
```

mysterium network

Talk thru middleware

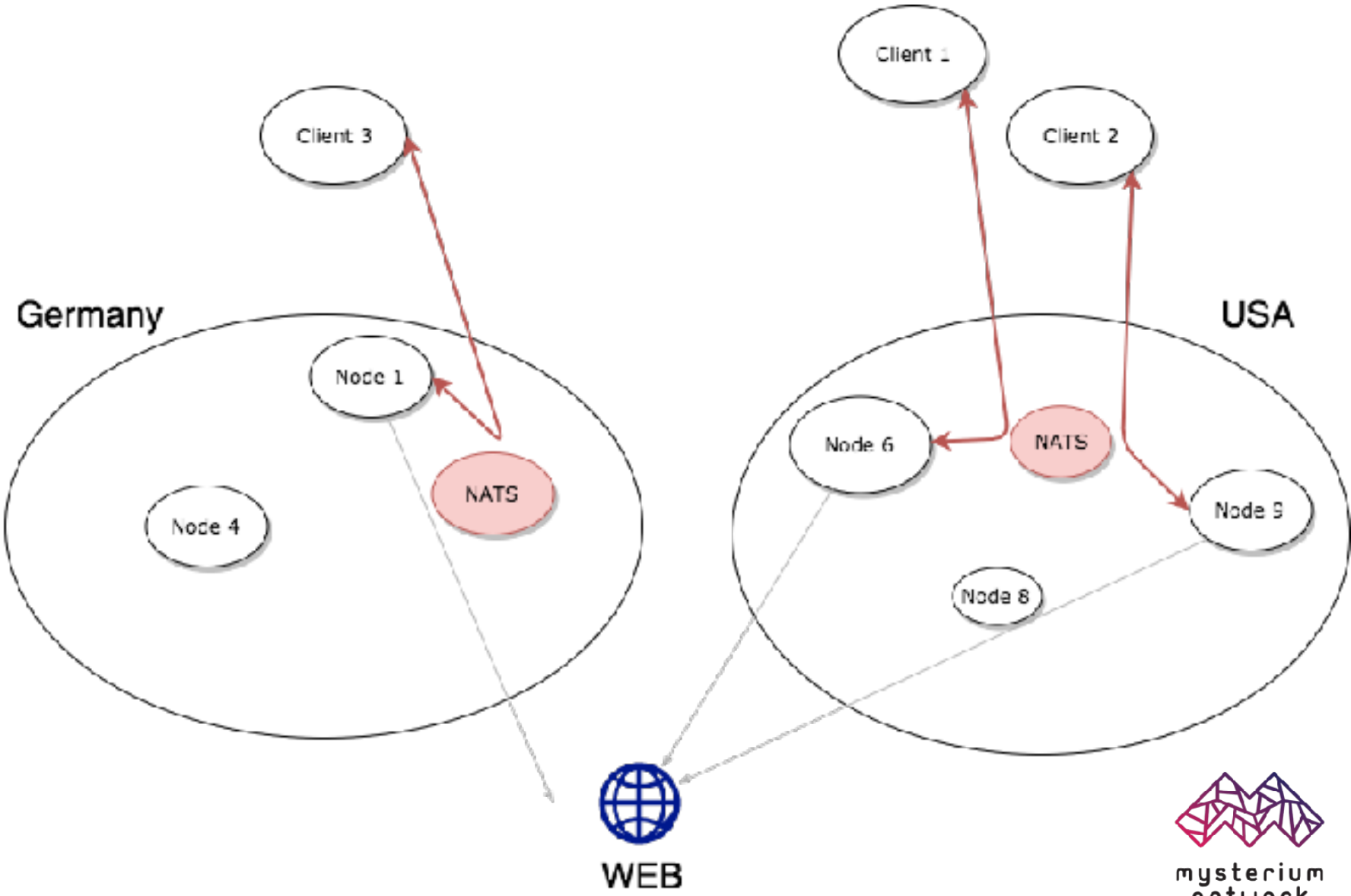


Node runs chat server



mysterium network

Talkative network



mysterium network

gnatsd run natively

```
go get github.com/nats-io/gnatsd
gnatsd
```

gnatsd run with Docker (5.7MB)

```
  node:
    build:
      context: .
      dockerfile: Dockerfile-mysterium-node
    container_name: mysterium-node
    ports:
      - 1194:1194

  client:
    build:
      context: .
      dockerfile: Dockerfile-mysterium-client
    container_name: mysterium-client
    volumes:
      - .:/application

  gateway:
    image: nats
    container_name: mysterium-gateway
    ports:
      - 4222:4222
      - 8222:8222
```

mysterium network

gnatsd run in code

```
import "github.com/nats-io/gnatsd/server"

server := server.New(&server.Options{
    Host:           "localhost",
    Port:           4222,
})

// Run server in Go routine.
go s.Start()
```

mysterium network

gnatsd summary

- Quick win
- Continue on messaging data models
- HTTP-like endpoint, request, response
- RabbitMQ inside your application
- JSON in encoding layer, later binary
- Router problem postponed
- No socket level networking problems
- Still lightweight

mysterium network

Happy1. Golang is multi-thread

- Subservices with channels
  - vpn daemon
  - gnats daemon
  - geth client
- Easy to supervise subservices
- Packaging moves into application

mysterium network

Happy2. Rich core library

- Crypto libraries
- Depending just on 2nd library
  - Logger
  - NATS

mysterium network

Happy3. Golang is cross-platform

- Node application:
  - Linux (Debian, Centos)
  - Docker (community implemented)
  - Windows?
- Client application:
  - Linux (any)
  - OSX
  - Windows (later)

mysterium network

Happy4. Golang is lightweight

- Eats 12-18 MB of RAM
- Run on Arduino (community implemented)
- Run on Router
- IOT enabler (Android)

mysterium network

Questions?

- Github @waldz
- valdas@mysterium.network

mysterium network