CHAIR OF NETWORK ARCHITECTURES AND SERVICES

TECHNICAL UNIVERSITY OF MUNICH

Initial Report

Author: **Thua-Duc Nguyen & Duc-Trung Nguyen**

1 Team Information

- **Thua Duc Nguyen**. **ID:** 03752081
- **Duc Trung Nguyen**. **ID:** 03783118
- **Team Name:** Gossip-7
- **Implementation:** Gossip Protocol

2 Programming Language and Operating System

For this project, we have chosen Go as our programming language. Additionally, we utilize Protocol Buffers for efficient data serialization and gRPC for designing APIs. Employing Go, Protocol Buffers, and gRPC provides a multitude of advantages:

1. **Efficiency in Development:**
   - **GoLang:** Go's simplicity, readability, and built-in concurrency support make it an excellent choice for developing networked applications like VoidPhone. Its compiled nature ensures swift execution and optimal resource utilization, which is essential for a P2P application where performance is paramount.
2. **Protocol Buffers for Data Serialization:**
   - **Efficient Serialization:** Protocol Buffers excel in efficient data serialization. Unlike JSON's text-based approach, they employ a compact binary format. This translates to significant performance enhancements [1]. Benchmarks indicate Protocol Buffers can be up to six times faster than JSON, with messages being 34% smaller and delivery to JavaScript code seeing a 21% speedup [2]. This efficiency makes Protocol Buffers ideal for applications like Voidphone, where data transfer speed and size are crucial.
3. **gRPC for Communication:**
   - **Bidirectional Streaming:** gRPC supports bidirectional streaming [3], facilitating efficient real-time communication between peers.
   - **Strong Typing and Code Generation:** Leveraging Protocol Buffers, gRPC defines service contracts, enabling strong typing and automatic code generation for both client and server. This reduces the likelihood of errors and streamlines the development process.
4. **Security and Reliability:**
   - **TLS Support in gRPC:** gRPC supports Transport Layer Security (TLS) encryption, ensuring secure communication between peers in the VoidPhone network. This helps mitigate potential security threats, safeguarding user privacy and data integrity.

3 Build System

We utilize Go Modules [4], the official dependency management system for Go projects since Go 1.11. Employing Go Modules alongside additional tooling provides an optimal environment for building our VoidPhone project:

- **Simplicity:** The Go build system is seamlessly integrated into the Go toolchain, eliminating the need to install or learn a separate build system. This ensures a streamlined development environment.
- **Native Support:** Go inherently supports compiling Go source code and linking dependencies. Furthermore, official packages for Protobuf generation `protoc-gen-go` [5] and gRPC server/client generation `grpc-gateway` [6] seamlessly integrate with the Go build system.
- **Flexibility:** While lightweight, the Go build system allows for extending functionality through custom build commands and integration with other tools. This flexibility is invaluable for tasks such as generating Protobuf code from '.proto' files and gRPC interfaces.

4 Quality Assurance Measures

To ensure the quality of our software, we implement the following measures:

- **Test Cases:** We employ unit tests for business logic and end-to-end (E2E) tests for APIs.
- **Linter:** We utilize the built-in `glint` linter.
- **Security Scanning Tool:** We employ `securego.io` for security scanning.
- **Static Analysis:** Valgrind is used for static analysis.
- **Logging:** We utilize `uber-go/zap` as our logging library.

5 Support Libraries

5.1 Gossip

- **URL:** `https://pkg.go.dev/github.com/zemnmez/cockroach/gossip`
- **Description:** A library for implementing the gossip protocol allows nodes to efficiently share information in a decentralized network. It is designed to help build scalable, fault-tolerant distributed systems.

5.2 SHA256 Encryption

- **URL:** `https://pkg.go.dev/crypto/sha256`
- **Description:** The `crypto/sha256` package in Go provides implementations of the SHA-256 cryptographic hash function, which is widely used for secure data hashing and verification.

5.3 Config File Parser

- **URL:** `https://github.com/graniticio/inifile`
- **Description:** A library for parsing INI configuration files, allowing easy reading and writing of configuration settings. INI files are simple text files with a structure that is easy to parse.

5.4 API

- **URL:** `https://grpc.io/`
- **Description:** gRPC is a high-performance, open-source framework for building remote procedure call (RPC) APIs. It uses HTTP/2 for transport and supports features like bidirectional streaming and strong typing.

5.5 Code Generation

- **URL:** `https://pkg.go.dev/google.golang.org/protobuf/cmd/protoc-gen-go`
- **Description:** The `protoc-gen-go` plugin for Protocol Buffers generates Go source files from `.proto` definitions. This facilitates the use of Protocol Buffers in Go applications by automating the creation of data structures and serialization code.

5.6 P2P

- **URL:** `https://libp2p.io/`
- **Description:** libp2p is a modular network stack that enables the development of peer-to-peer (P2P) applications. It provides tools for creating decentralized networks, handling peer discovery, and managing secure communications.

5.7 Testing

- **URL:** `https://pkg.go.dev/testing`
- **Description:** The `testing` package in Go supports automated testing of Go code. It includes tools for writing and running tests, benchmarks, and example code to ensure code correctness and performance.

5.8 Build

- **URL:** `https://go.dev/blog/using-go-modules` & `https://taskfile.dev/`
- **Go Modules:** Go's official dependency management system, introduced in Go 1.11. It simplifies dependency management and versioning for Go projects.
- **Taskfile:** A task runner for Go projects that allows defining tasks in a YAML file. It helps automate development workflows, such as building, testing, and

6 License

We decide to use the **MIT** License for this project because of the following reason:

- **Permissive:** Allows unrestricted use, modification, and distribution for any purpose.
- **Simple:** Clear and concise terms make it easy to understand and comply with.
- **Compatible:** Works well with other open-source licenses, facilitating collaboration and derivative works.
- **Minimal restrictions:** Encourages collaboration by imposing few limitations on usage.
- **Legal clarity:** Provides clear legal protection for creators and users of the code.
- **Encourages commercial use:** Attractive to businesses as it allows incorporation into proprietary products without open sourcing.

7 Team Expertise

- **Thua-Duc Nguyen:** Experience in Software Development and Current Go back-end developer
- **Duc-Trung Nguyen:** Experience in Software Development and Understanding of blockchain systems and security,

8 Planned Workload Distribution

- **Thua-Duc Nguyen**
  - Developing and testing Gossip API
  - Implementing security measures
- **Duc-Trung Nguyen**
  - Designing and implementing the Gossip P2P architecture
  - Quality assurance of the P2P implementation

Bibliography

[1] S. K. et al. “A hardware accelerator for protocol buffers”. In: (2021).

[2] “Beating JSON Performance with Protobuf”. In: (). Accessed: 2024-05-19. URL: https://auth0.com/blog/beating-json-performance-with-protobuf/.

[3] “gRPC Core Concepts”. In: (). Accessed: 2024-05-19. URL: https://grpc.io/docs/what-is-grpc/core-concepts/.

[4] “Using Go Modules”. In: (). Accessed: 2024-05-19. URL: https://go.dev/blog/using-go-modules.

[5] “protoc-gen-go”. In: (). Accessed: 2024-05-19. URL: https://pkg.go.dev/google.golang.org/protobuf/cmd/protoc-gen-go.

[6] “grpc-gateway”. In: (). Accessed: 2024-05-19. URL: https://github.com/grpc-ecosystem/grpc-gateway.