P2PChat

Language Abstractions for Concurrent and Parallel Programming (1DL540)

Lucas Arnström

Department of Computer Systems
Uppsala University

2016–12–13

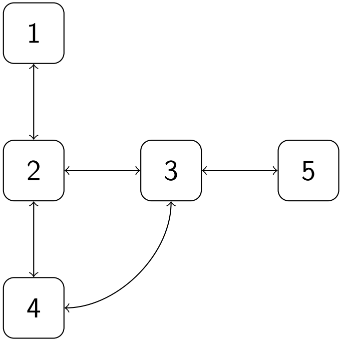
UPPSALA UNIVERSITET

Introduction

- **What is P2PChat?**
  Chat program which utilizes peer-to-peer connections in order to construct a large network of clients whom all participate in a single global group chat.
- **How does it work?**
  - Clients connect to each other and create a network.
  - Clients can then broadcast messages over the network.
  - No tracker sadly; clients have to manually connect to each other.

UPPSALA UNIVERSITET

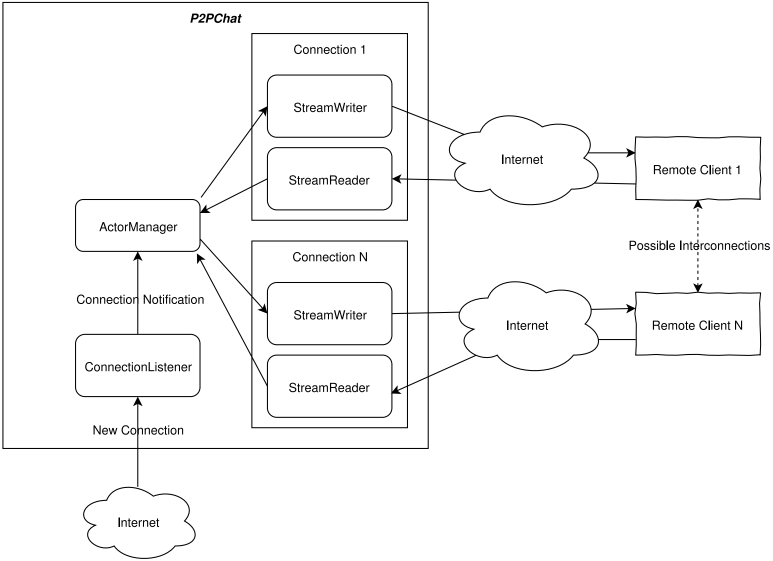
Example Network



Five clients connected to each other forming a simple network. If client 5 broadcasts a message, it will traverse the entire graph and eventually reach all nodes in the network.

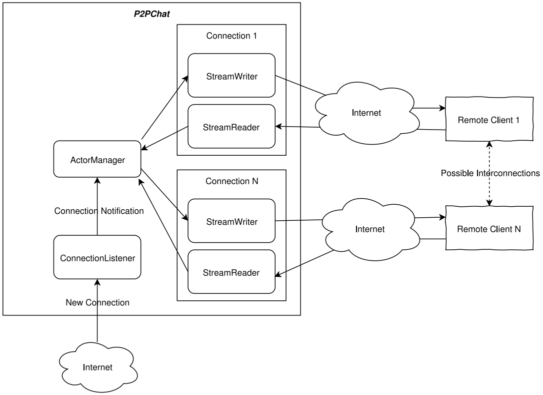
UPPSALA UNIVERSITET

Application Structure



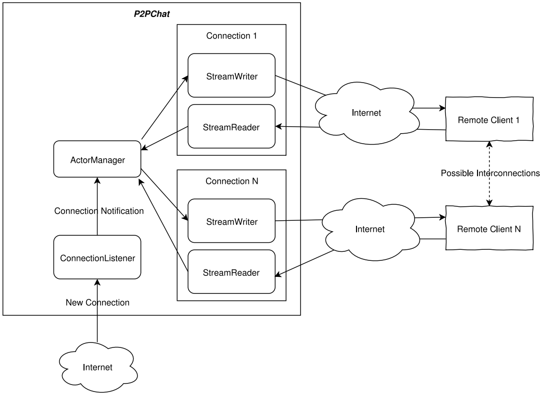
UPPSALA UNIVERSITET

Application Structure



UPPSALA UNIVERSITET

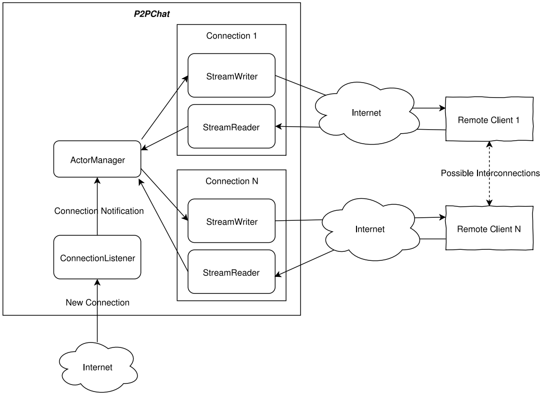
Application Structure



- Rounded rectangles represents actors.

UPPSALA UNIVERSITET

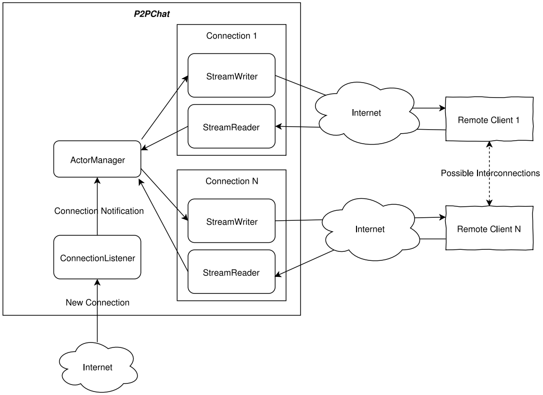
Application Structure



- Rounded rectangles represents actors.
- Arrows inside big rectangle represents channels.

UPPSALA UNIVERSITET

Application Structure



- Rounded rectangles represents actors.
- Arrows inside big rectangle represents channels.
- Arrows outside/across borders represents TCP sockets.

Why Rust?

- Why Rust instead of any other language?

[1] Threads can share data, but under very strict conditions.

UPPSALA UNIVERSITET

Why Rust?

- **Why Rust instead of any other language?**
  - Forces me to reason about memory ownership.

[1] Threads can share data, but under very strict conditions.

UPPSALA
UNIVERSITET

Why Rust?

- **Why Rust instead of any other language?**
  - Forces me to reason about memory ownership.
    - Only one scope can own and use a reference at any given time.

[1]Threads can share data, but under very strict conditions.

UPPSALA
UNIVERSITET

Why Rust?

- **Why Rust instead of any other language?**
  - Forces me to reason about memory ownership.
    - Only one scope can own and use a reference at any given time.
  - Allows easy memory management without the need of a garbage collector.

[1] Threads can share data, but under very strict conditions.

Why Rust?

- **Why Rust instead of any other language?**
  - Forces me to reason about memory ownership.
    - Only one scope can own and use a reference at any given time.
  - Allows easy memory management without the need of a garbage collector.
  - Guarantees memory safety.

[1] Threads can share data, but under very strict conditions.

Why Rust?

- **Why Rust instead of any other language?**
  - Forces me to reason about memory ownership.
    - Only one scope can own and use a reference at any given time.
  - Allows easy memory management without the need of a garbage collector.
  - Guarantees memory safety.
  - Guarantees no data races.

[1] Threads can share data, but under very strict conditions.

UPPSALA UNIVERSITET

Why Rust?

- **Why Rust instead of any other language?**
  - Forces me to reason about memory ownership.
    - Only one scope can own and use a reference at any given time.
  - Allows easy memory management without the need of a garbage collector.
  - Guarantees memory safety.
  - Guarantees no data races.
    - Does so by not allowing any threads to share data.[1]

[1] Threads can share data, but under very strict conditions.

UPPSALA UNIVERSITET

Why Rust?

- **Why Rust instead of any other language?**
  - Forces me to reason about memory ownership.
    - Only one scope can own and use a reference at any given time.
  - Allows easy memory management without the need of a garbage collector.
  - Guarantees memory safety.
  - Guarantees no data races.
    - Does so by not allowing any threads to share data.[1]
    - Need to use channels in order to pass data between threads.

[1] Threads can share data, but under very strict conditions.