1 Einleitung

Im Internet werden Nachrichten hauptsächlich über die Transportprotokolle TCP und UDP übertragen. TCP gewährleistet eine zuverlässige, geordnete und verlustfreie Übertragung, kann aber durch seinen aufwendigen Verbindungsaufbau, eingeschränkte Erweiterungsmöglichkeiten und begrenzte integrierte Sicherheitsmechanismen in modernen, latenz- und sicherheitskritischen Anwendungen an seine Grenzen stoßen. Das QUIC-Protokoll wurde entwickelt, um diese Limitierungen zu überwinden und moderne Anforderungen wie geringe Latenz, zuverlässige Übertragung und integrierte Verschlüsselung zu erfüllen.
Diese Ausarbeitung gibt zunächst eine Einführung in die grundlegenden Konzepte und Eigenschaften des QUIC-Protokolls. Dabei werden zentrale Mechanismen wie Verbindungsaufbau, Multiplexing und integrierte Verschlüsselung erläutert. Darauf aufbauend werden anschließend die drei in Java verwendbaren QUIC-Frameworks Kwik, Quiche4J und Netty miteinander verglichen. Der Vergleich betrachtet unter anderem den Funktionsumfang, den Abstraktionsgrad sowie die Einsatzgebiete, für die sich die jeweiligen Frameworks am besten eignen.
Im letzten Teil der Arbeit wird eine eigene Implementierung vorgestellt, die auf dem Framework Kwik basiert. Dabei werden Designentscheidungen, die konkrete Umsetzung sowie gemachte Erfahrungen und Herausforderungen beschrieben. Ziel ist es, sowohl einen praxisnahen Einblick in die Arbeit mit QUIC in Java zu geben als auch die Eignung von Kwik für reale Anwendungsfälle zu evaluieren.

2 Quic vs TCP

2.1 Verbindungsaufbau

Beim Aufbau einer Verbindung über TCP erfolgt zunächst ein dreistufiger Handshake (SYN, SYN-ACK, ACK), um eine zuverlässige, geordnete Verbindung herzustellen. Dieser Handshake verursacht mindestens eine Round-Trip-Time (RTT), bevor Daten übertragen werden können. Wird zusätzlich eine Verschlüsselung wie TLS eingesetzt, muss über der bereits etablierten TCP-Verbindung ein weiterer Handshake stattfinden, der in der Regel ein bis zwei zusätzliche RTTs benötigt, bis die fertige Verbindung Aufgebaut ist. In Szenarien mit vielen kurzen Verbindungen summieren sich diese Verzögerungen und können die Performance erheblich beeinträchtigen.
Das QUIC-Protokoll verwendet UDP als Transportbasis und baut darauf eine eigene Verbindungsschicht auf. Bereits beim ersten Verbindungsaufbau werden die Verschlüsselungsschlüssel ausgehandelt, sodass dafür in der Regel eine RTT erforderlich ist. Bei einem Wiederaufbau einer bereits bekannten Verbindung (Reconnect) kann der Handshake sogar ohne zusätzliche RTTs (0-RTT) erfolgen. Durch diese Integration von Transport- und Sicherheitsmechanismen reduziert QUIC die Latenz beim Verbindungsaufbau erheblich und eignet sich besonders für Anwendungen mit häufigen oder kurzlebigen Verbindungen.