Open Thesis / Project

Integrating QUIC into the Contiki OS

Thesis Type

Master Project / Master Thesis

Motivation

QUIC is a new UDP-based, connection-oriented, transport-layer protocol that was originally developed by Google in 2012 and was recently standardised by the IETF as RFC 9000. QUIC improves the performance of connection-oriented Web applications by establishing a number of multiplexed connections between two endpoints using UDP (i.e., it allows multiple streams of data to reach all the endpoints independently), and by providing reliable communication through mechanisms such as flow control, congestion control, and loss detection. QUIC is nearly equivalent to TCP, but with a much-reduced latency and connection overhead: this makes it the protocol of choice for HTTP/3. Another interesting feature of QUIC is that it does not identify a connection with a 4-tuple as TCP does (i.e., `Source_IP`, `Source_Port`, `Dest_IP`, `Dest_Port`), but rather defines connection identifiers that are independent from IP addresses and port numbers. Thanks to this, QUIC can hold a connection when a client or a server changes IP address, supporting handover as explored in this work. Lars Eggert has recently evaluated the feasibility of deploying QUIC on resource-constrained IoT devices, summarizing his results in this paper and publishing the *Quant* QUIC stack open-source on GitHub. *Quant* supports the Particle and RIOT IoT stacks, and is therefore suitable for constrained embedded systems, which are also targeted by the popular Contiki and Contiki-NG operating systems. We aim to use *Quant* as a starting point and enable support for QUIC also in Contiki or Contiki-NG.

Goals and Tasks

Within this context, the student can explore several directions and perform different tasks, such as:

- Getting familiar with the QUIC protocol (see also this video), its features, and its implementations (especially the *Quant* stack);
- Porting *Quant* to Contiki or Contiki-NG, and enable support for QUIC in resource-constrained IoT platforms such as the TI CC2650 or the Nordic Semiconductor nRF52;
- Validating the functionality of the implementation and evaluating the memory footprint and performance in selected use cases.

Target Group

- Students of ICE/Telematics;
- Students of Computer Science;
- Students of Electrical Engineering.

Required Prior Knowledge

- Solid knowledge of networking fundamentals and embedded systems;
- Excellent C programming skills;
- Experience with embedded platforms and Linux systems is of great advantage;
- Ideally, successful completion of the Embedded Internet (VU/LU) course.

Contact Person

- Dipl.-Ing. Elisabeth Salomon
  elisabeth.salomon@tugraz.at
- Assoc.Prof. Carlo Alberto Boano
  cboano@tugraz.at