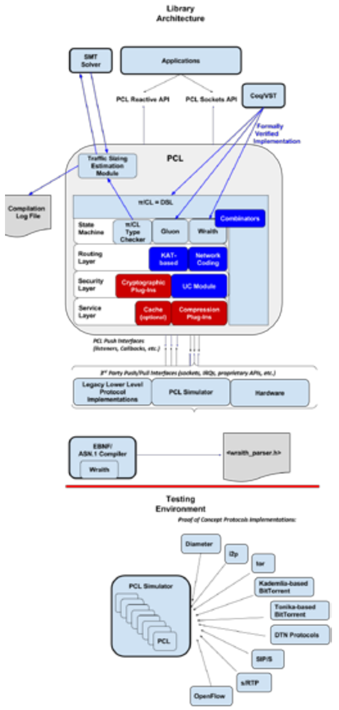
Programming and Formal Verification of Network Communication Protocols Implementations

Birkbeck
UNIVERSITY OF LONDON

Department of Computer Science and Information Systems

Research Student
Maxim Belov

Supervisors
Carsten Fuhs
Trevor Fenner



Verified Software Toolchain

Research Aims

The goal of the research is to explore ways how to build formally verifiable and efficient implementations of network protocols for industrial mission-critical systems by using modern declarative programming principles. The ultimate goal of the research is to develop a programming framework, which would allow Software Engineers to develop any kind of **formally verifiable** communication protocol stacks. Such a framework will be implemented as a library with a working name PCL (Protocol Combinators Library), which can be linked with programs developed in an industrial programming language, such as C11 or C++14. One of the goals of the research is to keep it relevant in the context of the modern embedded software development industry, hence the choice of programming languages.

Research Methodology

Each network protocol specification addresses two large topics:

1. *Protocol messages processing*. Message processing usually includes: parsing, validation, encryption, compression.
2. *Higher level semantics definition of the protocol*. The semantics of the protocol can be represented as some kind of reactive system or a finite state automaton, which can react on the messages of the protocol in a correct way and which, ideally, cannot be put into an undefined state by an incorrect message or by an illegal message sequence. The set of valid behaviours includes: authentication and authorization mechanisms, routing of the messages, higher level dialogs implementation, concurrency issues.

We are going to implement PCL using latest results achieved in each of these fields and additionally, as the library name suggests, we are going to use PCL as a test platform to research the problem of formal verification of protocol combinations, e.g. communication subsystems, which use composed communication protocols or protocol stacks. The critical parts of the PCL will be formally verified themselves by using the Coq proof assistant and the Verified Software Toolchain. The currently chosen implementation language of the library is C11, because it makes it easy to reuse a wealth of already existing verification tactics for C programs already developed by the Coq community.

Research Approach

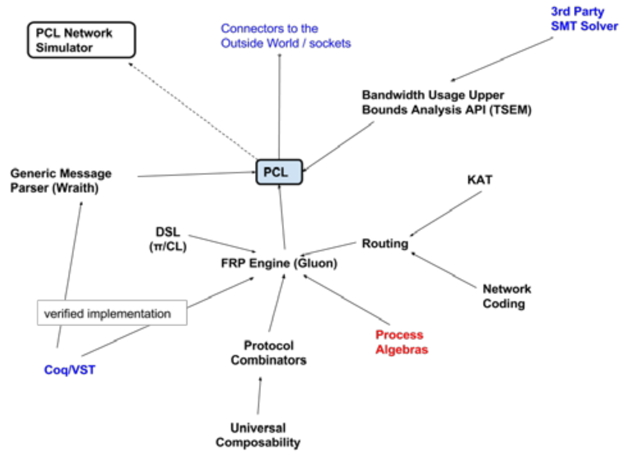


Figure 1. Concept Map of the PCL project

The following problems are being addressed during our work on the Protocol Combinator Library:

1. Implementation of a generic top-down message parsing DSEL (domain-specific embedded language) suitable for the use with C11 (short name Wraith, because its main inspiration is `Boost.Spirit` from the C++ universe).
2. Implementation of the declarative reactive DSL, which would allow programming the high level logic of communication protocols as well as combining protocol definitions into protocol stacks. The DSL's semantics will be based on the results from the fields of Process Algebras, mainly π-calculus and Session Types, and Functional Reactive Programming (FRP).
3. For research purposes and in an attempt to make the library as advanced as possible, the latest results from the following fields will be used in order to implement PCL and its DSL: NetKAT, Network Coding, Universal Composability Framework.
4. The new idea of using SMT solvers in order to estimate upper bounds of the network bandwidth used by traffic generated by the software implemented using PCL will be tested as a part of the research. This idea was inspired by the KoAT project.