Poster #61: Composing SDN Controller Enhancements with Mozart

Zhenyu Zhou\textsuperscript{1,*} and Theophilus A. Benson\textsuperscript{2}
\textsuperscript{1}Duke University, \textsuperscript{2}Brown University, *Presenter

1 Background

Cloud providers employ Software Defined Networking to
\begin{itemize}
  \item simplify network management
  \item configure networking infrastructure using higher level abstractions
\end{itemize}

Decoupling SDN
\begin{itemize}
  \item SDNApps: Networking functionality
  \item SDNEnhancements: Optimizations
\end{itemize}

Two fundamental questions remain not answered

Research Questions

What is the right interface for enabling principled interactions between SDNApps and SDNEnhancements?

What abstractions are required to systematically include SDNEnhancements into the SDN ecosystem?

3 Design

Main Idea

- Developers simply specify the class of transformations that are tolerable, or not.
- No requirements to understand all SDNEnhancements.

Analogy to Compiler Optimization:

Compilers for SDNs

- SDN assembly code: low-level control messages
- "Code block": policies among a certain group of hosts
- Compilation: SDNEnhancement function
- SDN compiler flags: SDN-Flags

4 Evaluation

Performance Improvement

- Proactive SDNApp (Hedera): Saves 24.8\% reduction in aggregate bandwidth introduced by TCAMOptimizer and decreases TCAM usage saving from 57.5\% to 18.2\%.

- Reactive SDNApp (RtFlow): Flows get activated 7.8 times faster regarding to time to recovery.

Mozart Overhead

- Sublinear.
- Only increases 1.58\% to latency.