A gRPC Based Event Distribution System

Adib Rastegarnia, Douglas Comer
Systems Research Group
Purdue University

Motivation and Problem Statement

The architecture of current software defined management systems exhibits several weaknesses as follows:

- Monolithic and Proprietary
- Lack of a Uniform Set of NB APIs
- Lack of Reusability of Software Modules
- Lack of Scalability and Reliability

SDN Control Plane Disaggregation

To migrate from a monolithic architecture to a microservice architecture for SDN controllers, we need to disaggregate control plane services into a set of cooperative microservices that can communicate with each other via standard APIs.

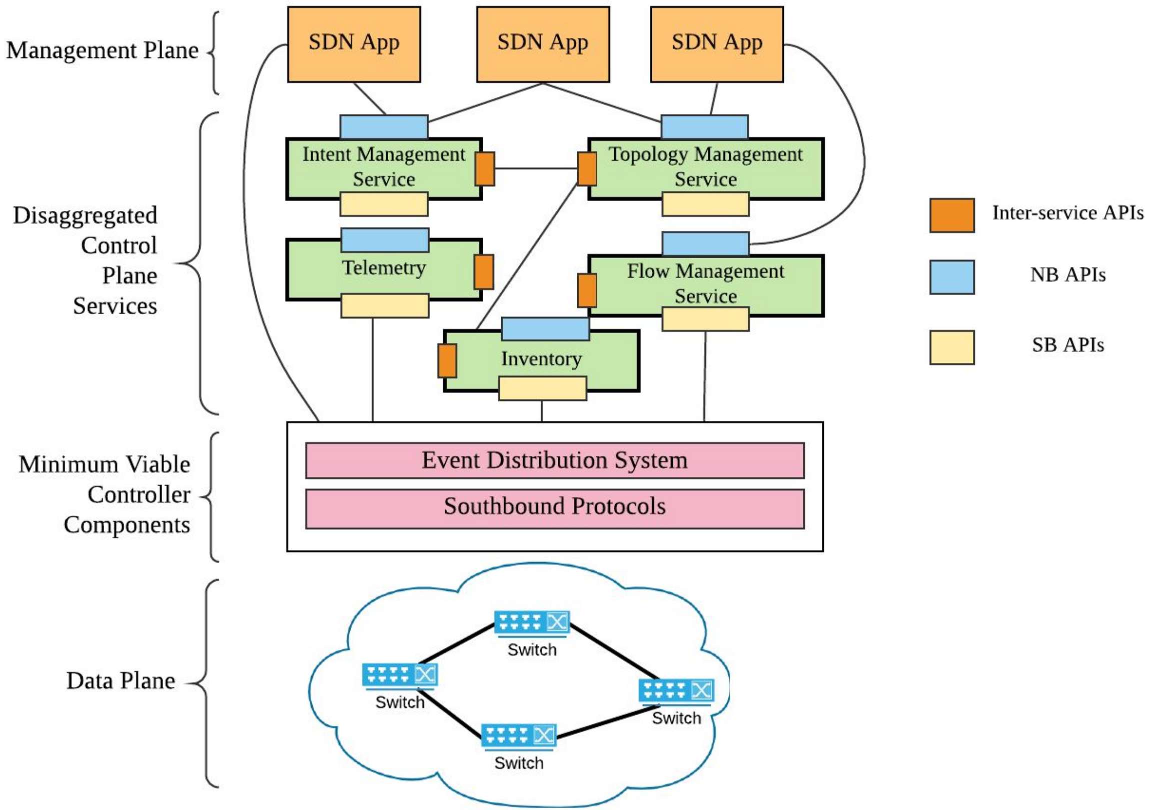
Advantages of a Disaggregated Control Plane

- **Flexibility to scale:**
  - Disaggregation makes it possible to scale a given core service horizontally, independent of other subsystems and services
- **Freedom to choose a programming language**
  - Unlike current designs, disaggregation allows a programmer to choose an arbitrary programming language, programming technology, and third-party libraries when building an SDN management application

Advantages of Disaggregated Control Plane

- **Fault isolation:**
  - Disaggregation means the failure of a given microservice will not affect the execution of other microservices
- **Minimal Built-in Components**
  - Disaggregation minimizes the set of components built into a controller
- **A Disaggregated Codebase**
  - Disaggregation allows the code for services to be independent

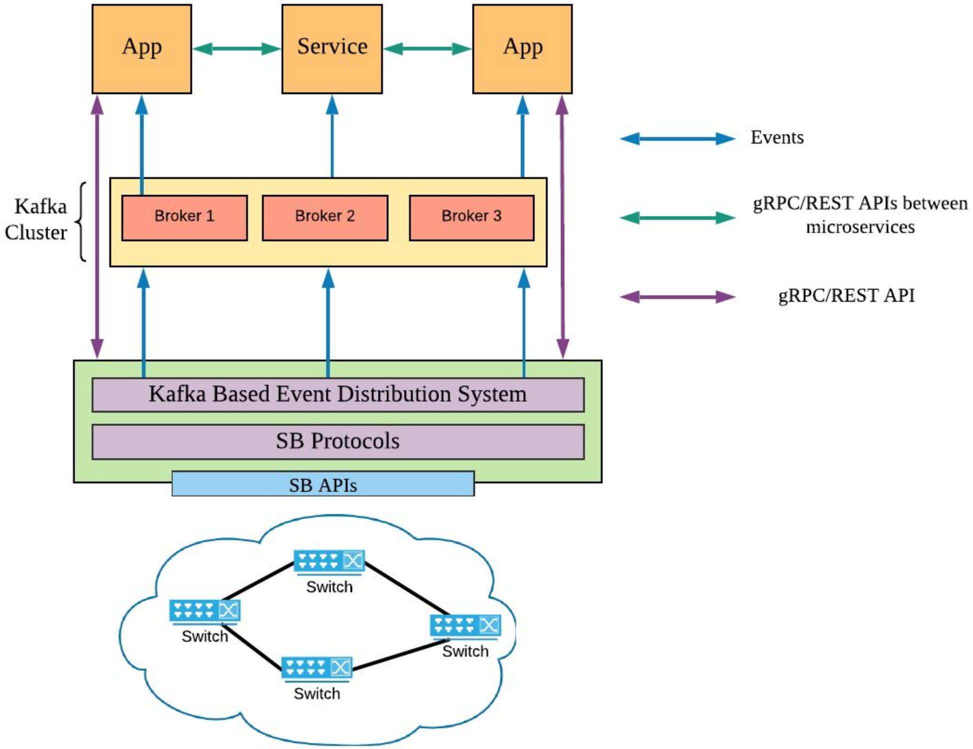
A Disaggregated SDN Control Plane Architecture



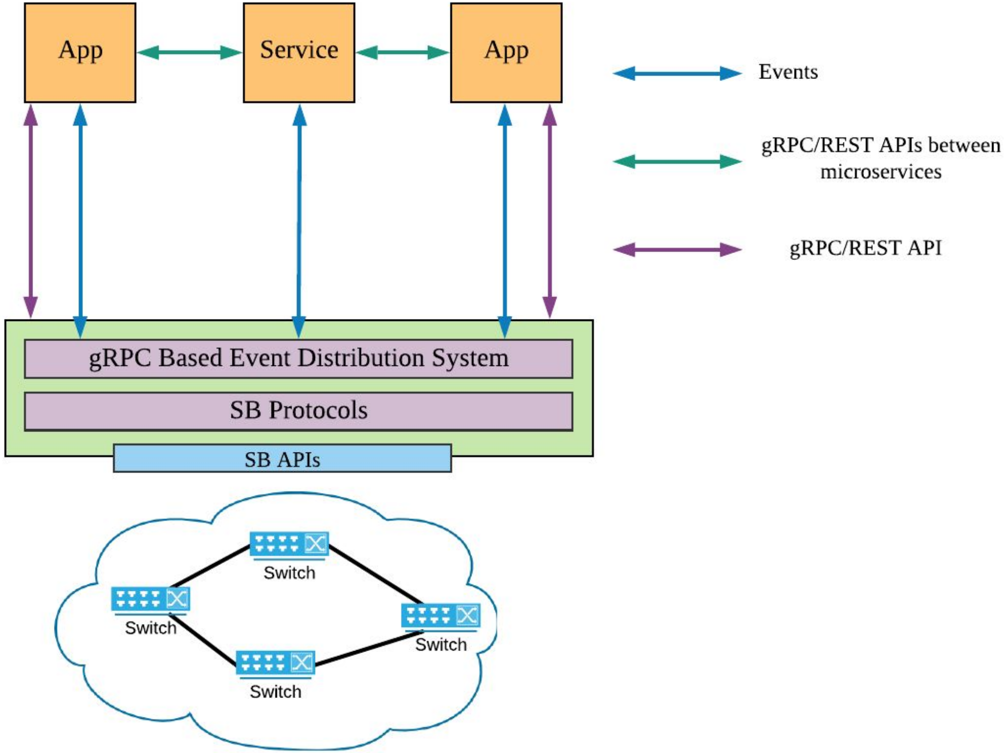
Two Candidate Event Distribution Systems

- An Event Distribution Mechanism that uses a Publish-Subscribe Model
  - Apache Kafka can be used to implement an event distribution mechanism that follows the publish-subscribe model
- An Event Distribution Mechanism that uses a Point-to-Point Model
  - gRPC can be used to implement an event distribution system that follows the point-to-point model

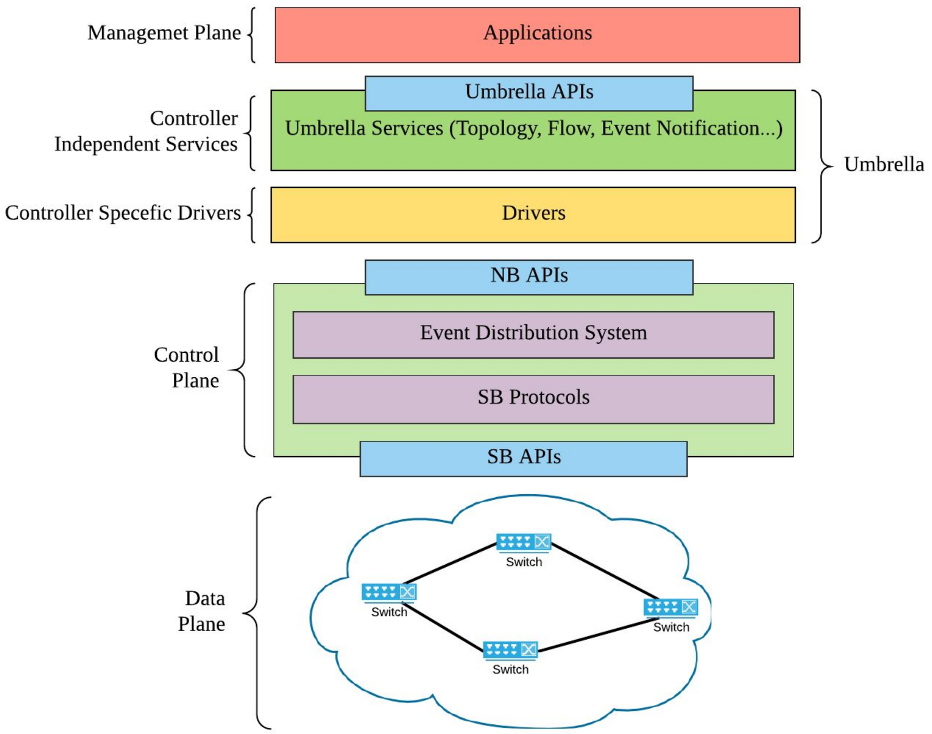
A Kafka-Based Event Distribution System



A gRPC Based Event Distribution System



Umbrella: A Unified SDN Programming Framework



Experimental Setup

- We implemented each of the candidate event distribution systems as an application for ONOS
- To measure the two event distribution mechanisms, we used an SDN testbed
  - Physically, the testbed consists of five OpenFlow switches
  - Logically, the testbed defines ten interconnected sites

Experimental Scenarios

- **Scenario 1**: To understand the the cost of using an event distribution system, compare the amount of time that an external app or service takes to process a packet event with the time it takes to process the same packet event inside the current monolithic version of ONOS
- Basic measure: overall **response time**

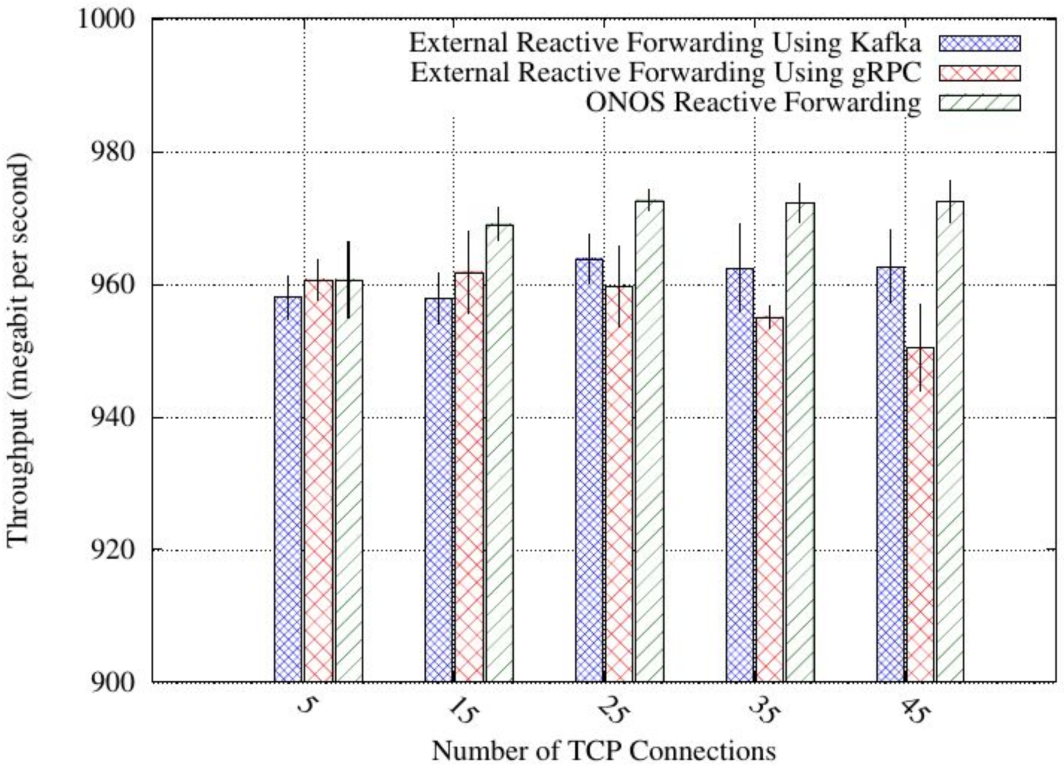
Experimental Results for Scenario 1

- We repeated an experiment 500 times to measure the ping response time between two end hosts in our SDN testbed that are 5 hops apart
- As a baseline, we measured the average response time for processing pings in the current, monolithic version of ONOS, and arrived at an average of 24 ms
- The average response time for a gRPC system is 29 ms
- The average time for a Kafka system is 35 ms

Experimental Scenarios

- **Scenario 2**: To assess the impact of externalized packet processing and the use of a REST API for flow rule installation on **throughput**, we compared two external reactive forwarding applications that use gRPC and Kafka with the same reactive forwarding application compiled into the current monolithic version of ONOS

Experimental Results for Scenario 2



Thank You

Follow Up Links:

systems.cs.purdue.edu