Application Defined Networking

Ratul Mahajan
University of Washington

ACM India Winter School on “Full-Stack Networking” 2023
From monoliths to microservices
From monoliths to microservices
From monoliths to microservices
Microservices need application networks

Function calls

RPCs
Microservices need application networks

- Service discovery
- Access control
- Load balancing
- Encryption
- Observability
- Fault tolerance

Diagram:
- Frontend
  - Payment
  - Product
- User
- Ad
- RPCs
service mess  /'sɜrvəs mes/
noun

1. the result of spending more compute resources than your actual business logic dynamically generating and distributing Envoy proxy configs and TLS certificates.
Common service mesh datapath
Sidecar overhead

Latency and CPU overhead for the Hotel benchmark
Challenges with the current approach

High overhead

Non-portability

Poor extensibility

How should we design and implement application networks?
Characteristics of application networks

Need rich message processing, not just IP

Connect endpoints of an application, not everyone
Our Approach: Application Defined Networks (ADN)

Developers specify what they want network to do at a high level

- Application-relevant abstractions
- Easy to write, portable

Compiler automatically generates an optimized implementation

- Determine where and how of processing happens (incl. offload)
- Determine message headers/protocols

Meets application-specific needs without a burdened implementation that does it all
Example

S1→S2: RequestRouting→Logging→Compression→FaultInjection(0.1)

Compiler

ADN Processors
Example

S1→S2: RequestRouting→Logging→Compression(zip)→FaultInjection(0.1)
Example

S1→S2: RequestRouting→Logging→Compression(zip)→FaultInjection(0.1)
Example

S1→S2: RequestRouting→Logging→Compression(zip)→FaultInjection(0.1)
Example

$S_1 \rightarrow S_2$: RequestRouting $\rightarrow$ Logging $\rightarrow$ Compression (zip) $\rightarrow$ FaultInjection (0.1)
Example

S1→S2: RequestRouting→Logging→Compression(zip)→FaultInjection(0.1)
Programming abstractions

Graph specification

- RPC processing as a graph of elements
- Each element performs a single network function on RPCs
Programming abstractions

- Element specification
  - Idea 1: Dataflow SQL
    - RPC processing $\approx$ stream processing
    - But not expressive enough
  - Idea 2 (currently pursuing): Match-action
    - Well-understood paradigm for layer 4 processing but layer 7 processing is richer
    - Possible to express common Envoy filters and gRPC middleware
Example

FaultInjection(prob):
match{random(0,1) < prob}
  true => drop
  false => forward

RequestRouting:
match{Look(rpc.user,RouteTable)}
  Some(d) => rpc.dst = d
  None => DROP
Summary

Application networks today have high overhead and are inflexible.

Generality is the root cause of inefficiencies.

Our approach: Application Defined Networking

- Specify desired network functionality in a high-level language.
- Auto-generate optimized, application-specific implementation.