Application Defined Networks

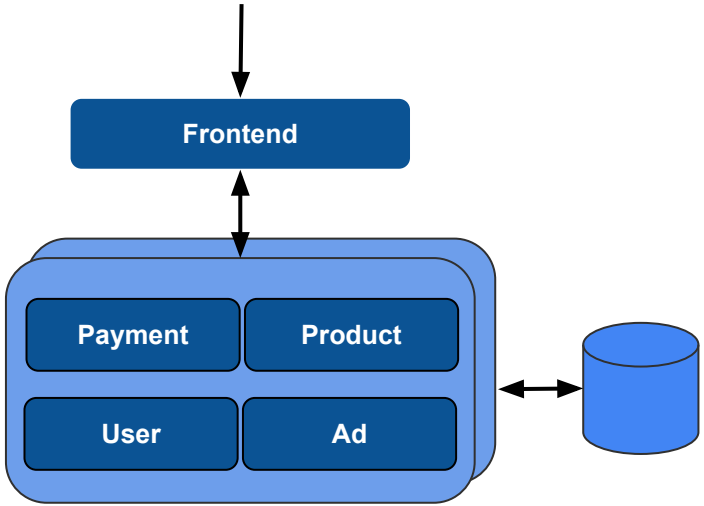
Xiangfeng Zhu, Weixin Deng, Banruo Liu, Jingrong Chen, Yongji Wu,
Thomas Anderson, Arvind Krishnamurthy, Ratul Mahajan, Danyang Zhuo

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

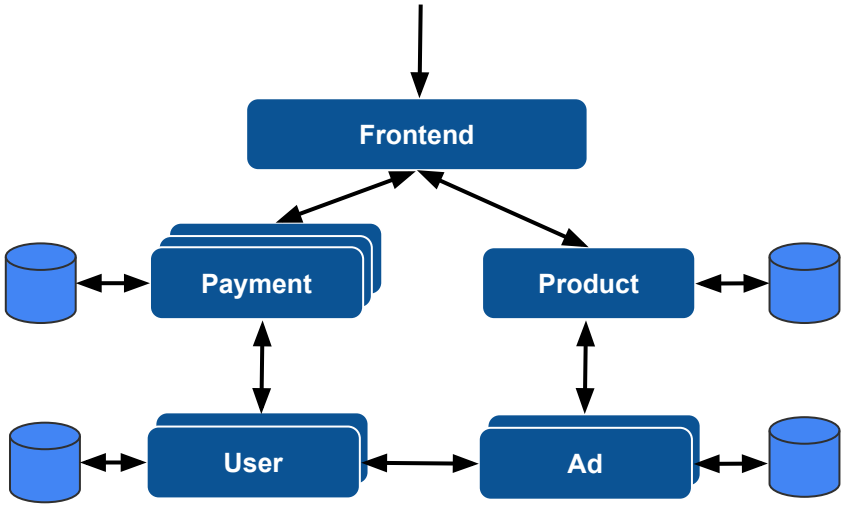
Duke
UNIVERSITY

清華大學
Tsinghua University

From Monolith to Microservices

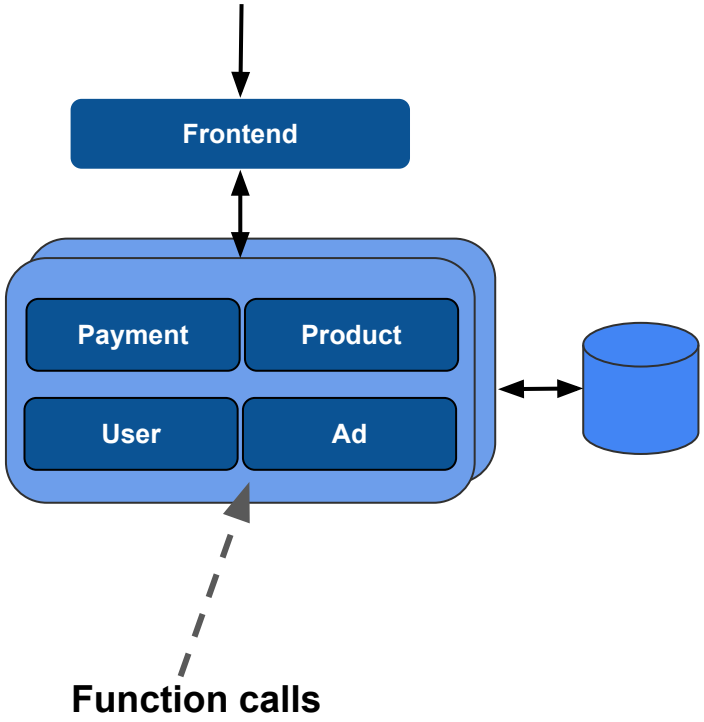


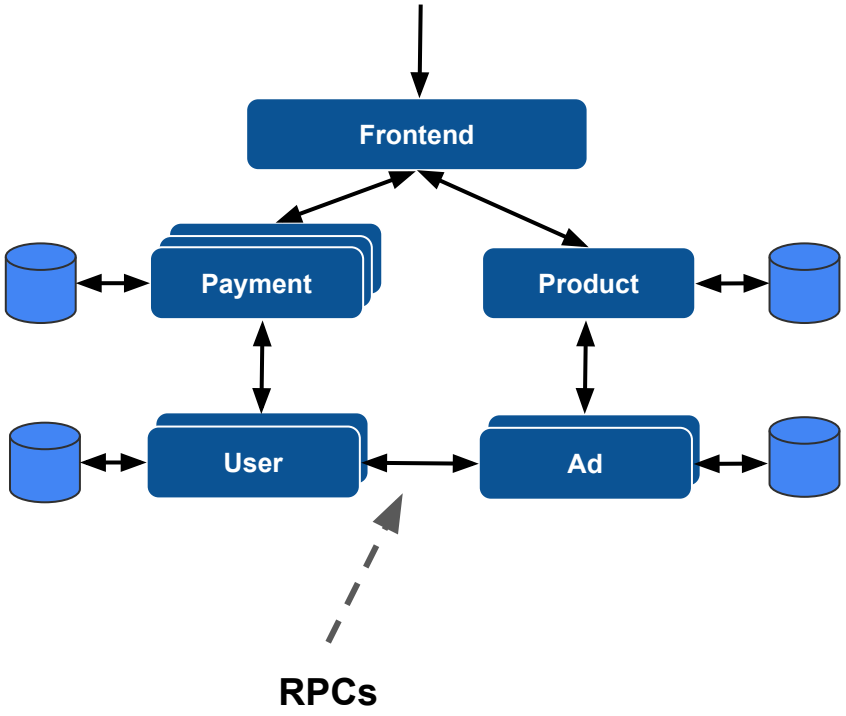
Monolithic Application



Microservices

From Monolith to Microservices

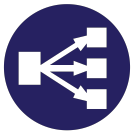




Microservices Need Application Networks

Service Discovery

Access Control

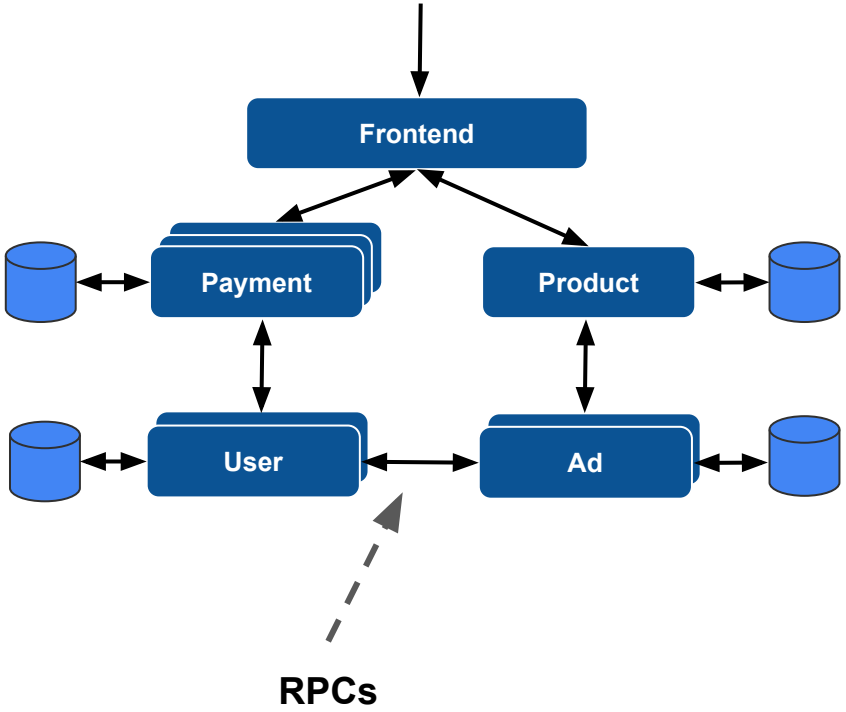


Load Balancing

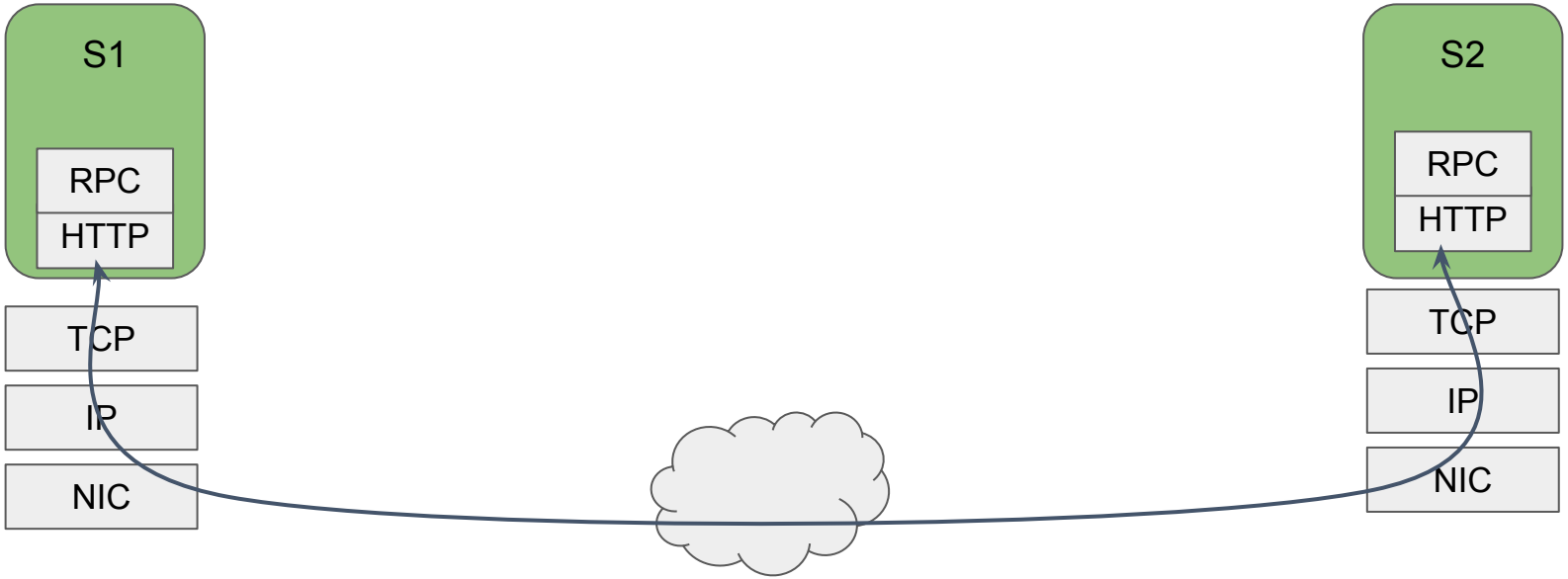
Observability

Encryption

Fault Tolerance

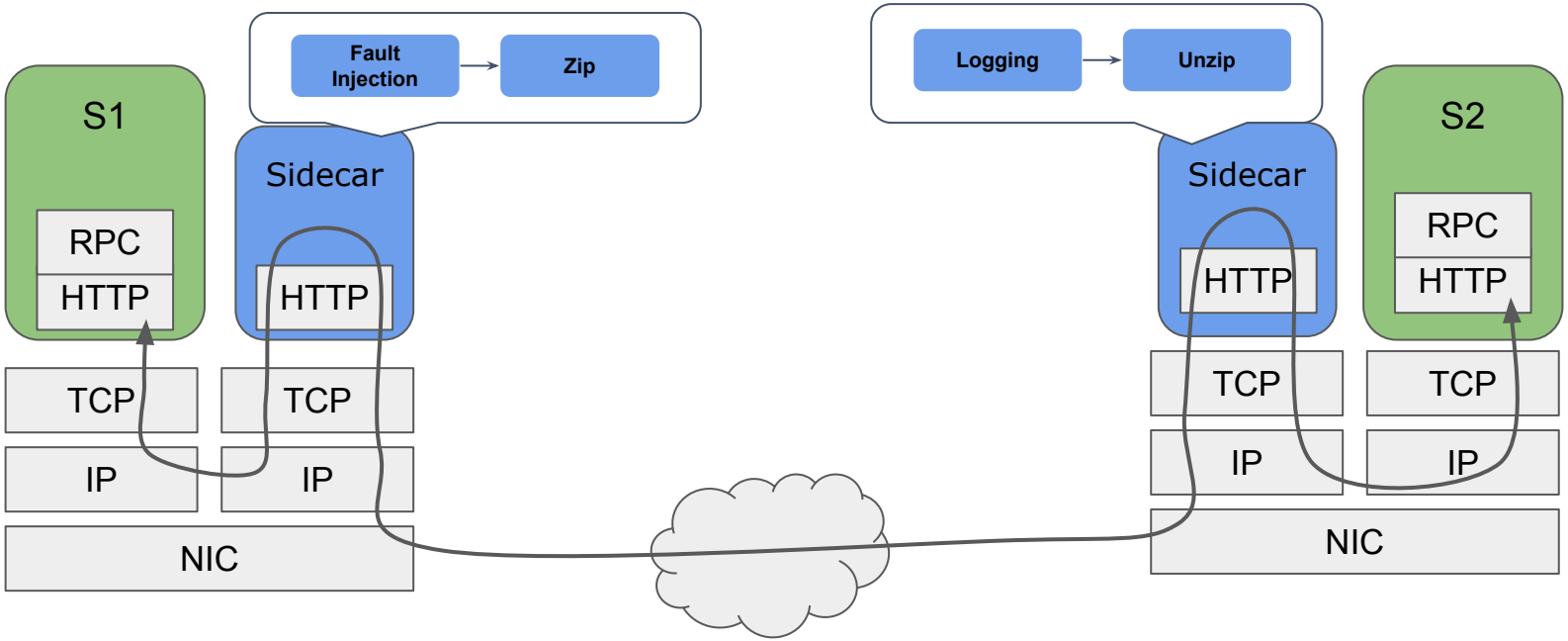


Communication between Microservices



Example Message Processing

Status Quo: Black-box Approach on General Protocols



Service Meshes with Sidecar Pattern

Challenges with the General and Black-box Approach

- High Overheads*
  - Throughput/Latency/CPU Usage
- Non-portability
  - Difficult to offload to kernel and hardware
- Non-optimizability
  - Difficult to reorder or consolidate elements

*Dissecting Overhead of Service Mesh Sidecars, *SoCC '23*

Our Approach: Application Defined Networks (ADN)

Developers specify what the network should do at a high level

- ❏ Application-relevant abstractions
- ❏ Easy to write, expressive and portable

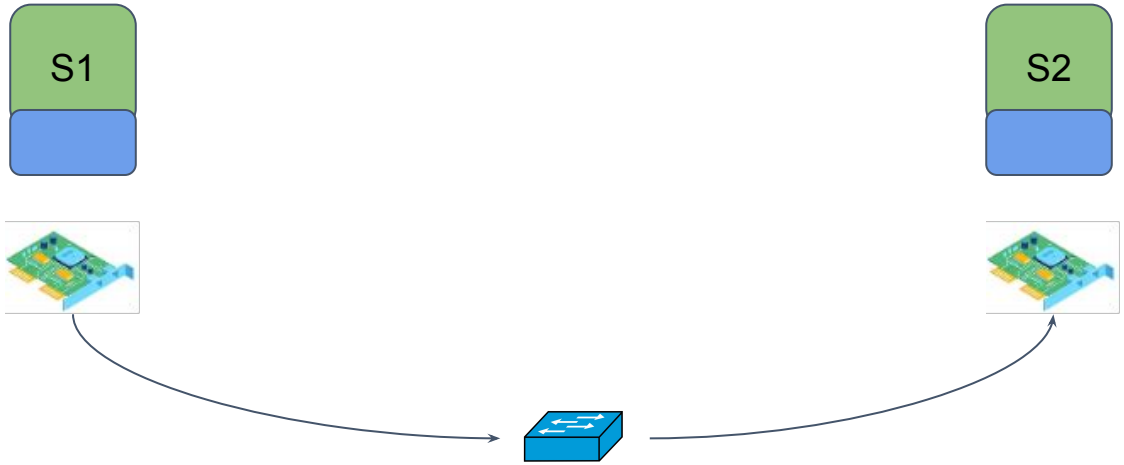
A compiler auto-generates an optimized, application-specific implementation

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

Goal: Build Custom Network for each Application

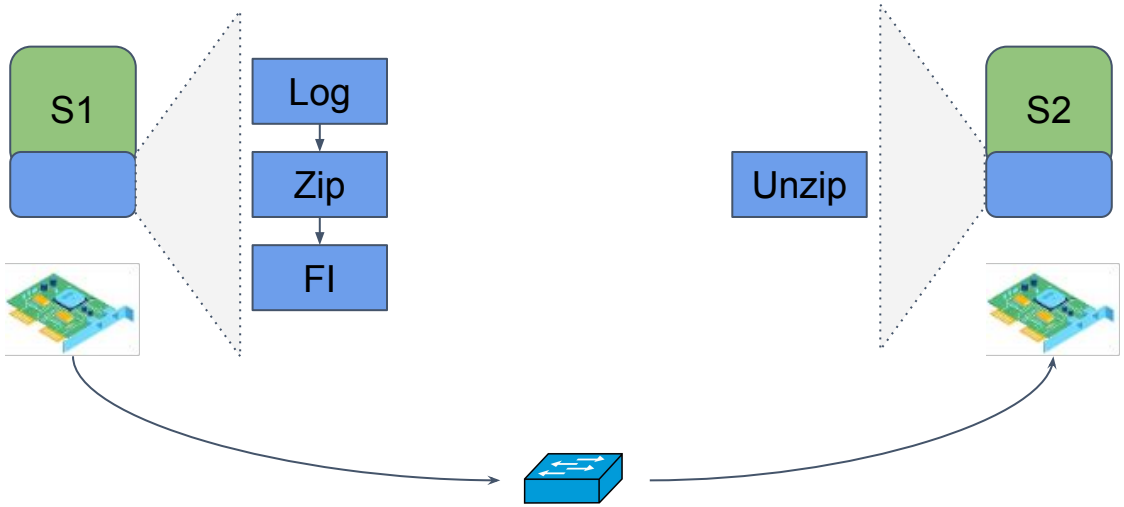
Example

```
S1→S2: Logging→Compression(zip)→FaultInjection(0.1)
```



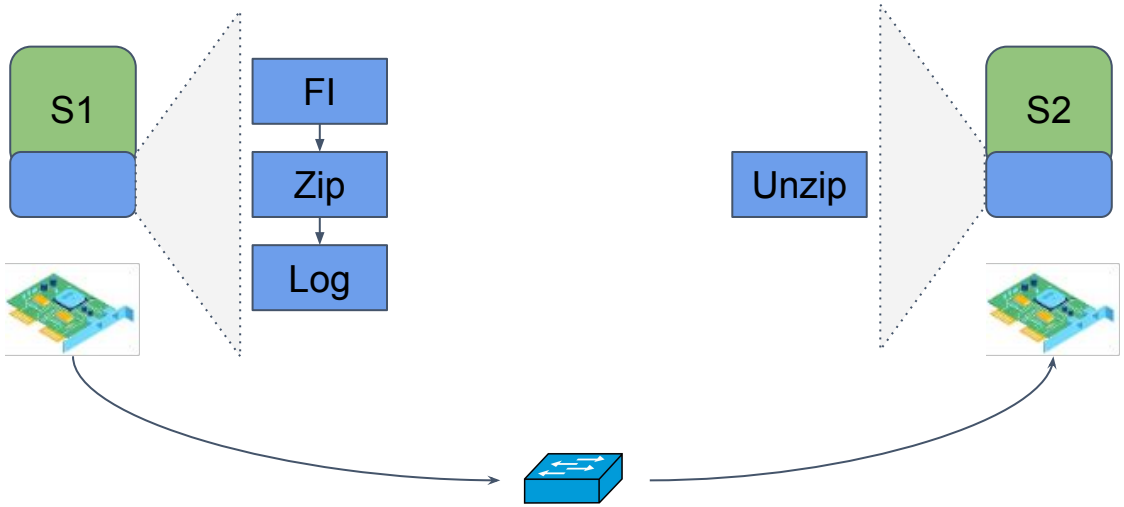
Example

```
S1→S2: Logging→Compression(zip)→FaultInjection(0.1)
```



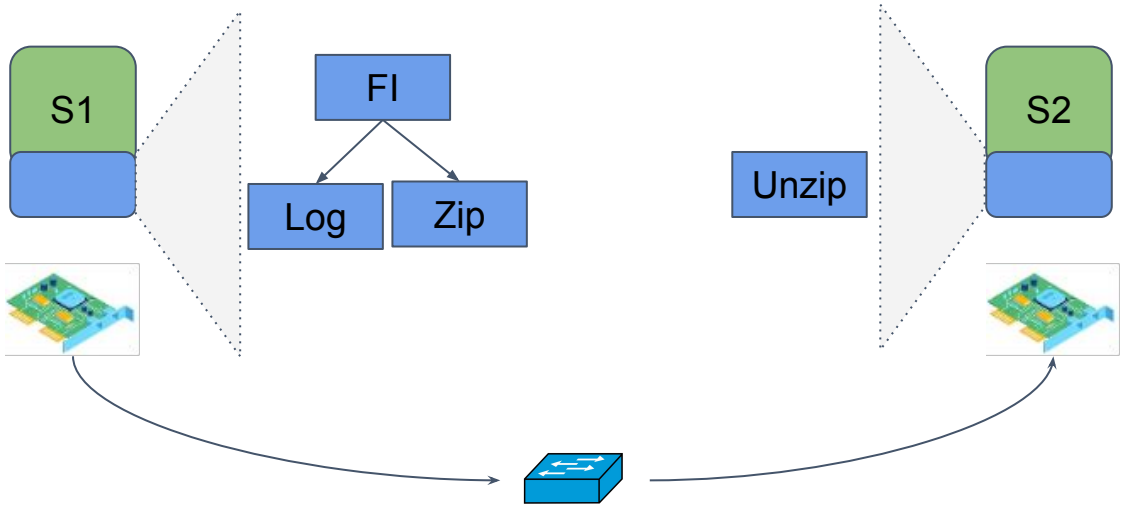
Example

```
S1→S2: Logging→Compression(zip)→FaultInjection(0.1)
```



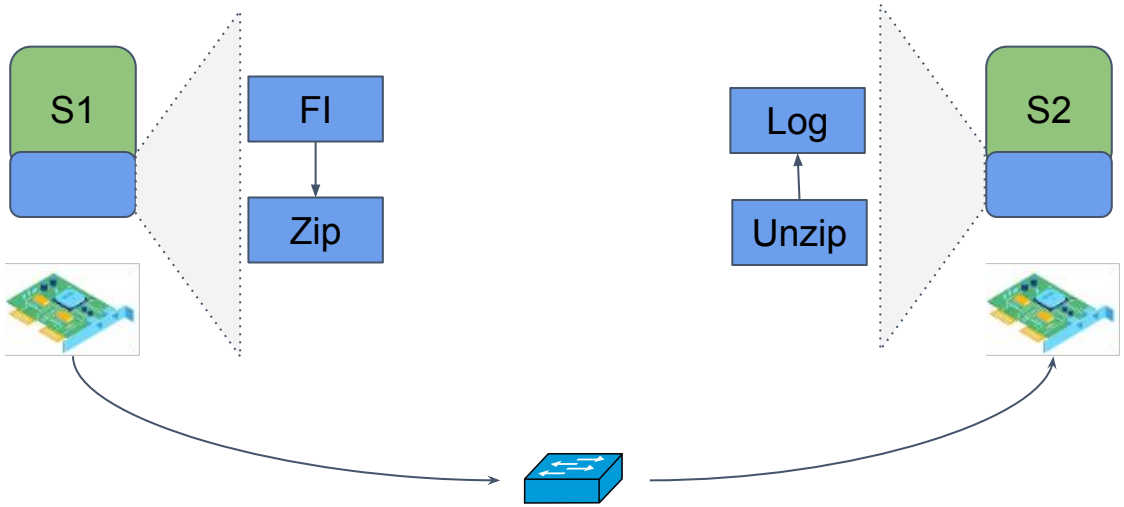
Example

```
S1→S2: Logging→Compression(zip)→FaultInjection(0.1)
```



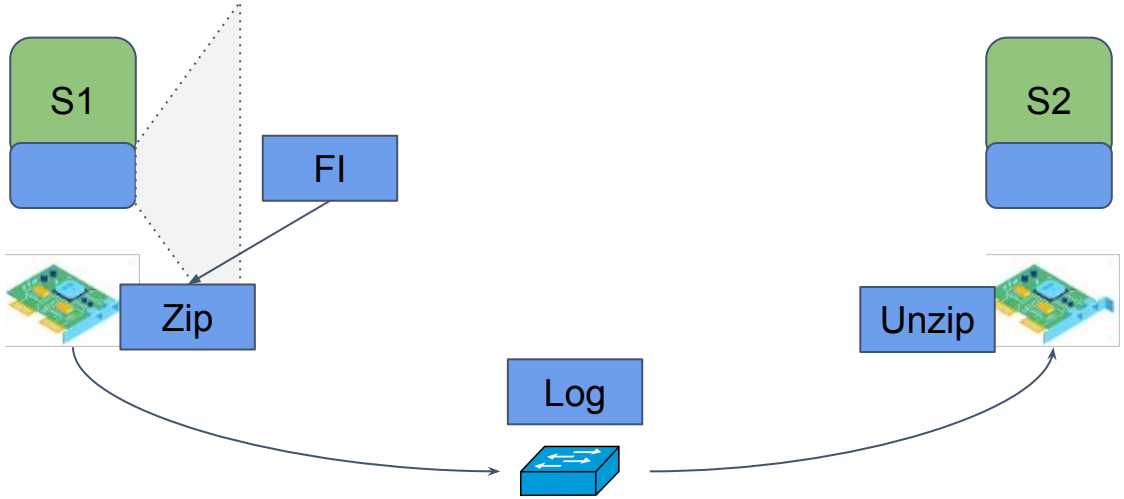
Example

```
S1→S2: Logging→Compression(zip)→FaultInjection(0.1)
```



Example

```
S1→S2: Logging→Compression(zip)→FaultInjection(0.1)
```



Programming Abstraction

- Graph Specification
  - RPC processing as a graph of elements
  - Each element perform a single network function on RPCs
- Element Specification
  - Dataflow-like SQL
    - RPC processing ~= stream processing
    - Easy to infer element properties
    - Enable element reuse

Conclusion

- Application networking is on the rise
- Generality is the root cause of inefficiencies
- Our solution: Application Defined Networks
  - Developer specify the desired network functionality using a high-level language
  - ADN auto-generates optimized and customized implementation for each application