SAF: A Firewall for Serverless Applications

Pedro Escaleira escaleira@ua.pt
Vitor A. Cunha vitorcunha@ua.pt
João P. Barraca jpbarraca@ua.pt
Diogo Gomes dgomes@ua.pt
Rui L. Aguiar ruilaa@ua.pt

universidade de aveiro

30 YEARS instituto de telecomunicações

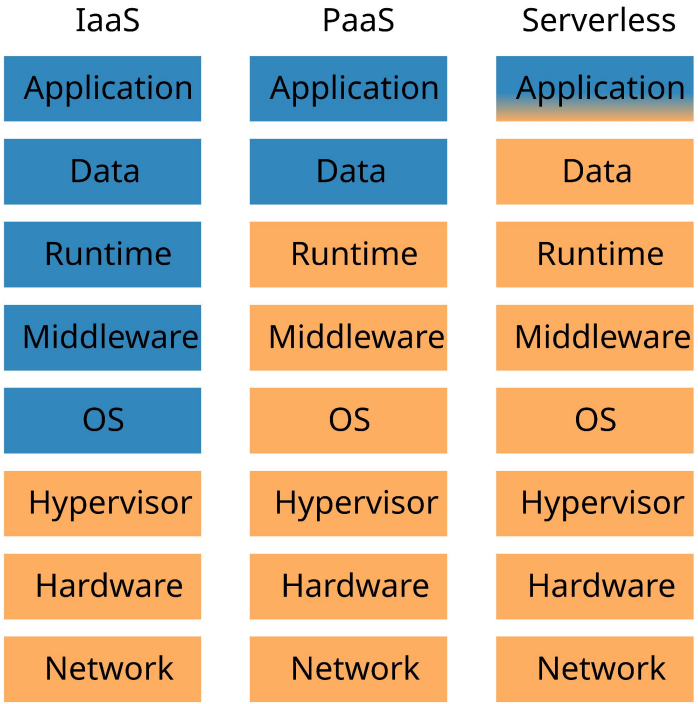
ATNOG

EURO-3C

Problem Statement

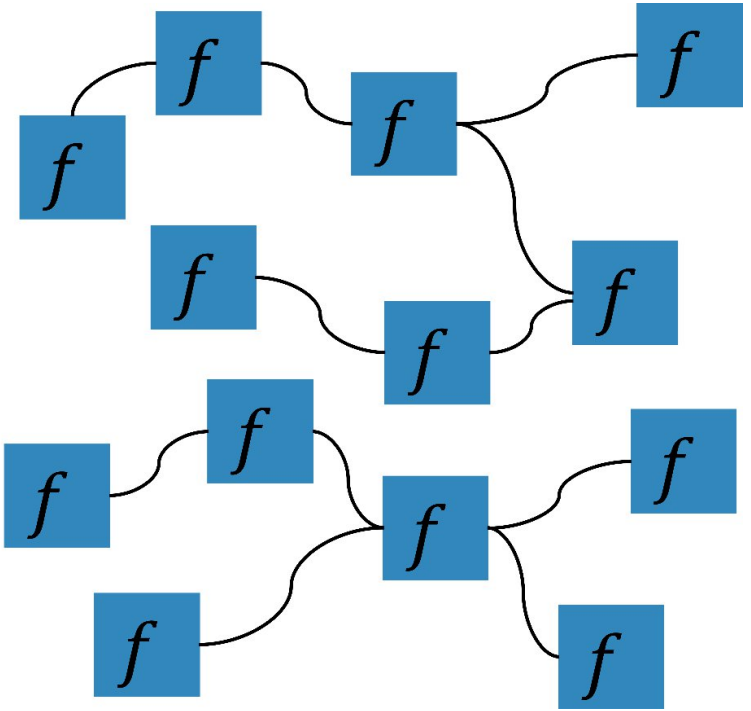
Serverless Computing

Serverless is a paradigm in which the Cloud Service Provider (CSP) ***manages all hardware and software stacks*** *up until the application layer*



Function as a Service

Application



Serverless Invocation Graphs

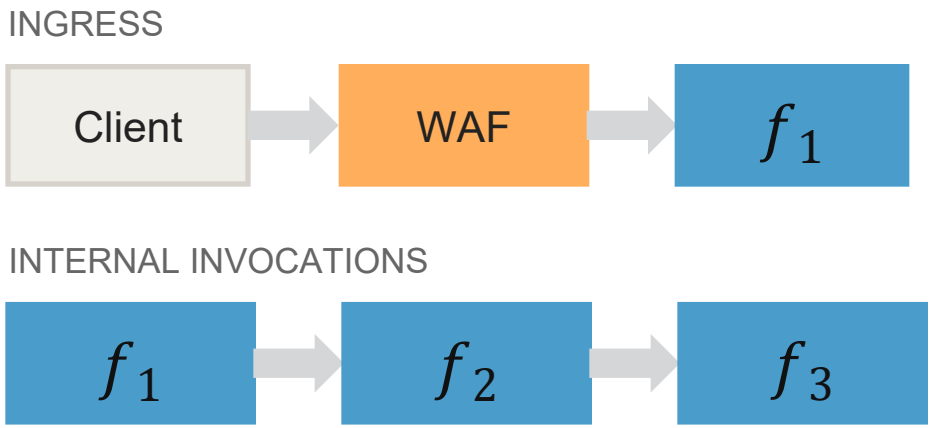
One application, many independently invoked functions.



The application is a graph of function-to-function calls.

WAFs Miss Internal Calls

An ingress WAF filters the first request. It does not inspect calls between functions.

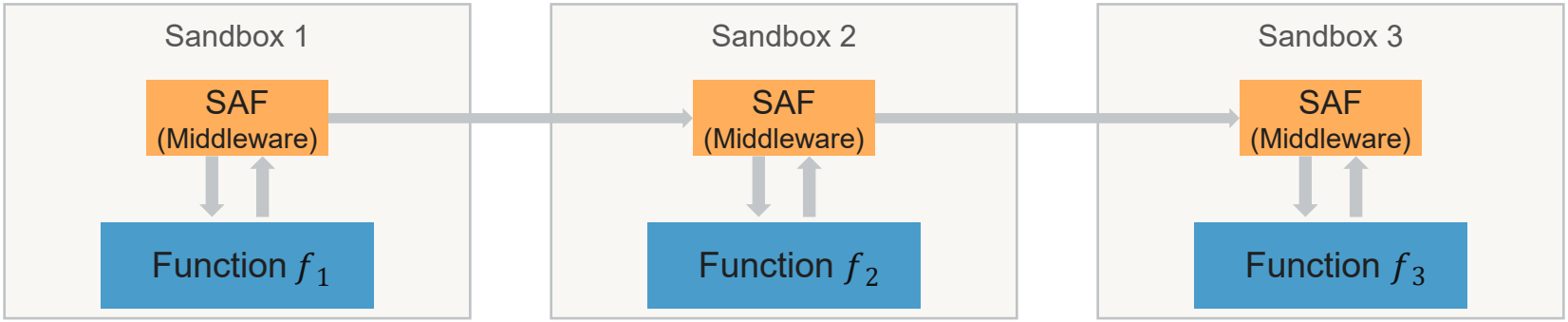


The gap is inside the provider boundary.

Proposal

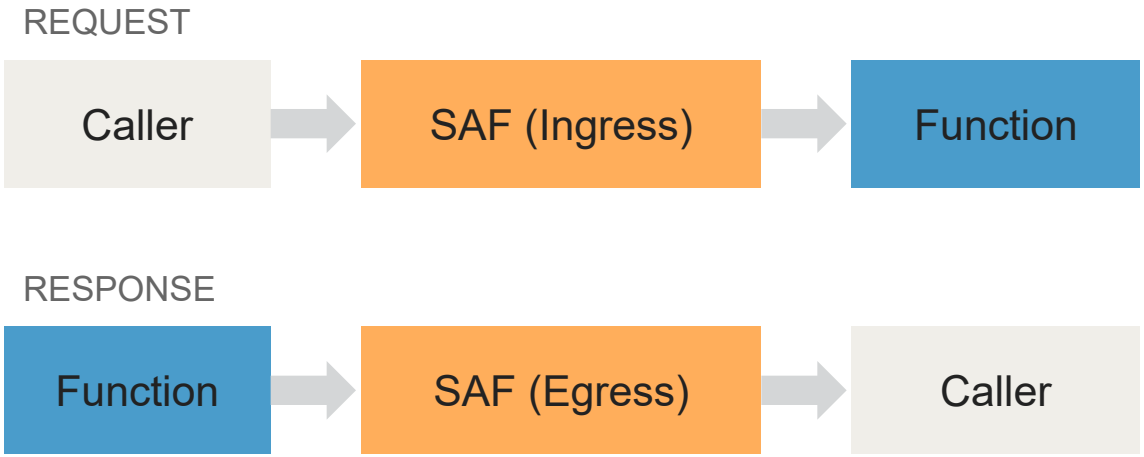
SAF Protects Each Function

Each function runs in its own sandbox with its own SAF.



Two-Way Policy Checks

Each function boundary can filter the request and its response.



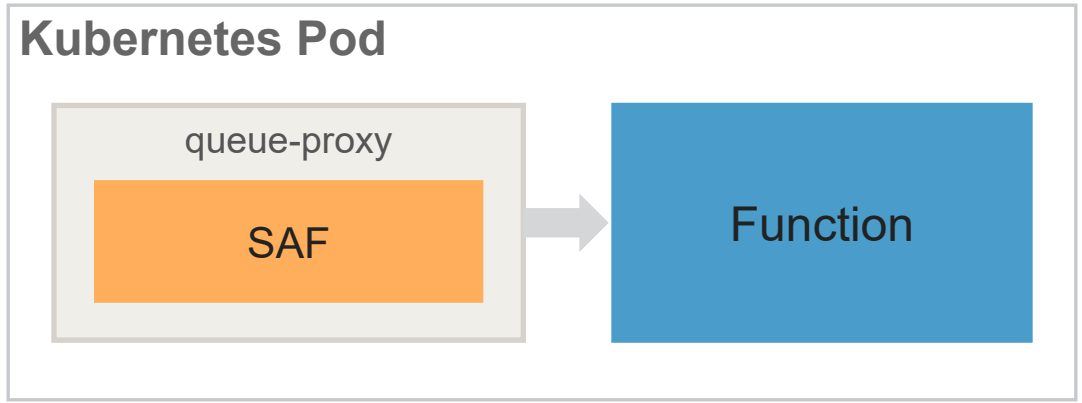
Rules can accept, reject, or drop; unmatched traffic follows the configured default.



Evaluation

SAF in the Managed Runtime

The proof of concept extends Knative queue-proxy.



No changes to function code or function image.

Testbed

A controlled Knative testbed varies policy size and measures the added cost.

CLUSTER

2 VMs

8 vCPU · 16 GB each

ORCHESTRATION

1 VM

4 vCPU · 8 GB

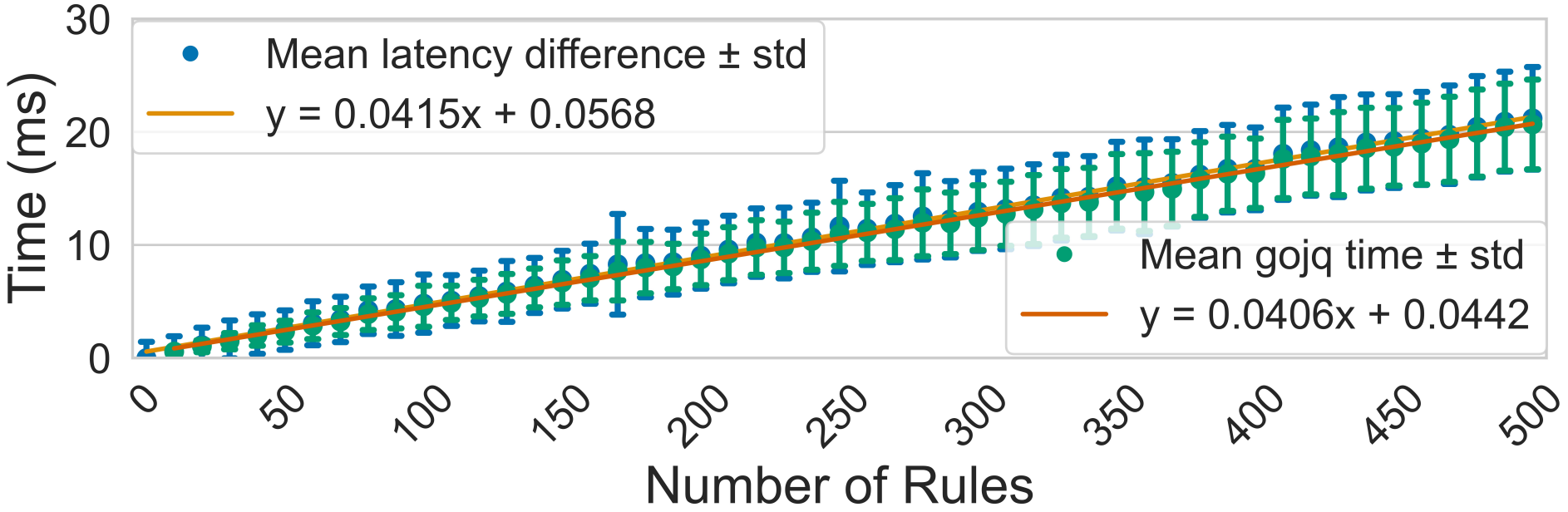
SOFTWARE

K3s 1.35.5

Knative 1.21

550 runs

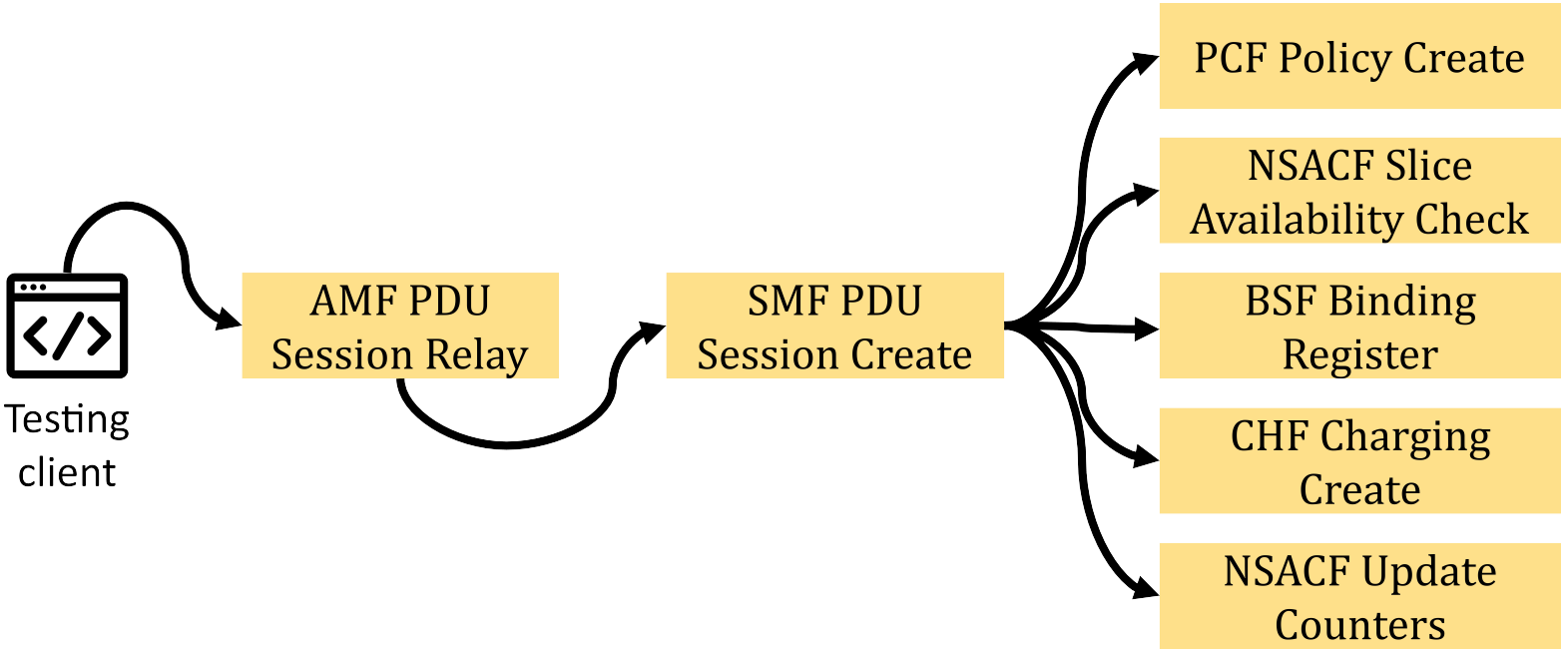
Rule Count Latency Difference



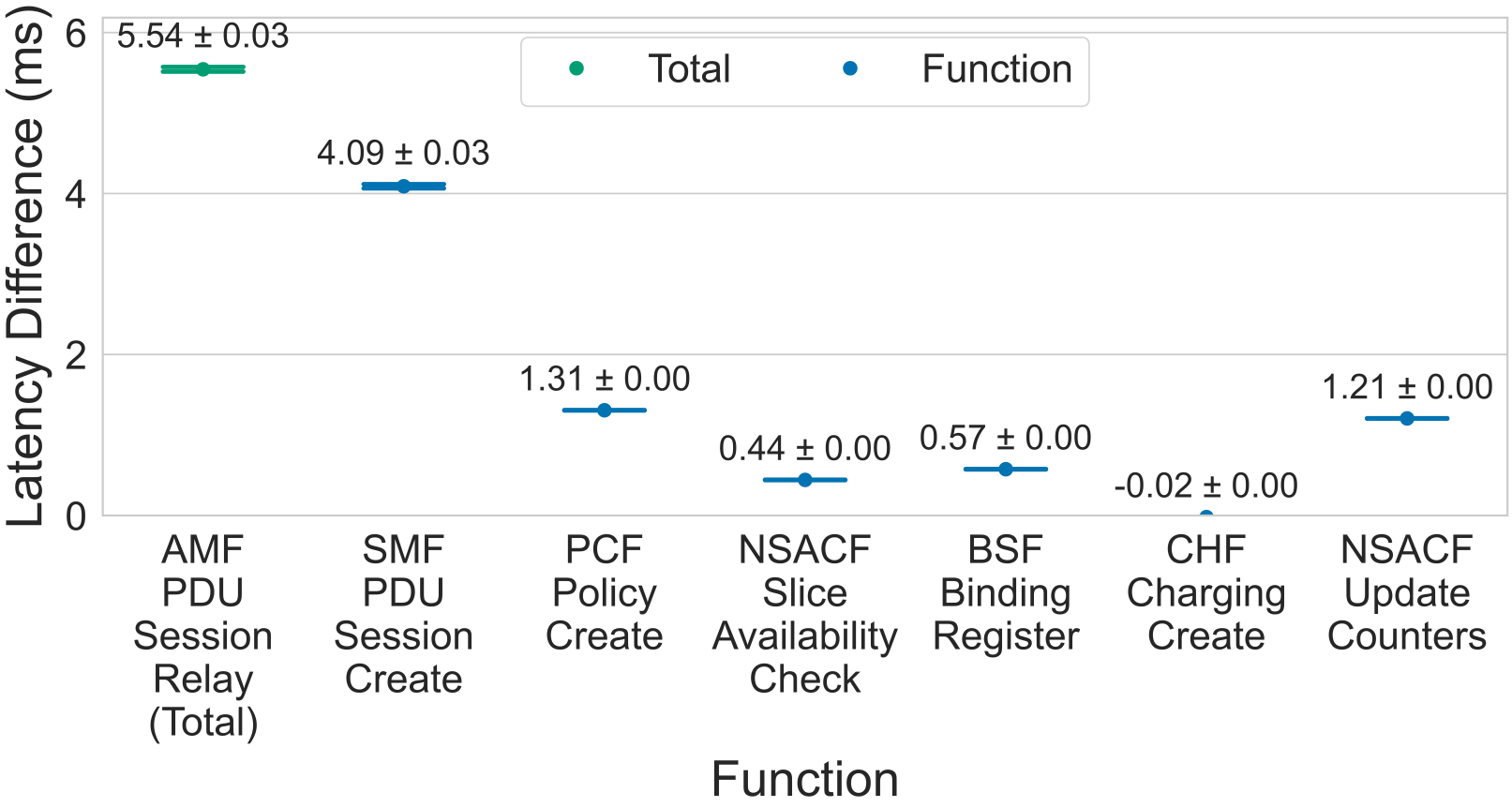
5G PDU Session Workflow

The case study protects a seven-function service path.

55 total rules · 6–10 rules per function



5G PDU Session Latency Difference



Main Takeaway

SAF moves policy enforcement to the function boundary.

Future Work

Change the policy engine from JQ to something with more performance (Coraza for instance)

Automate policy generation

Evaluation in more realistic scenarios

Thank you for your attention

github.com/ATNoG/saf

github.com/ATNoG/serverless5gc-knative