MeshTest: End-to-End Testing for Service Mesh Traffic Management

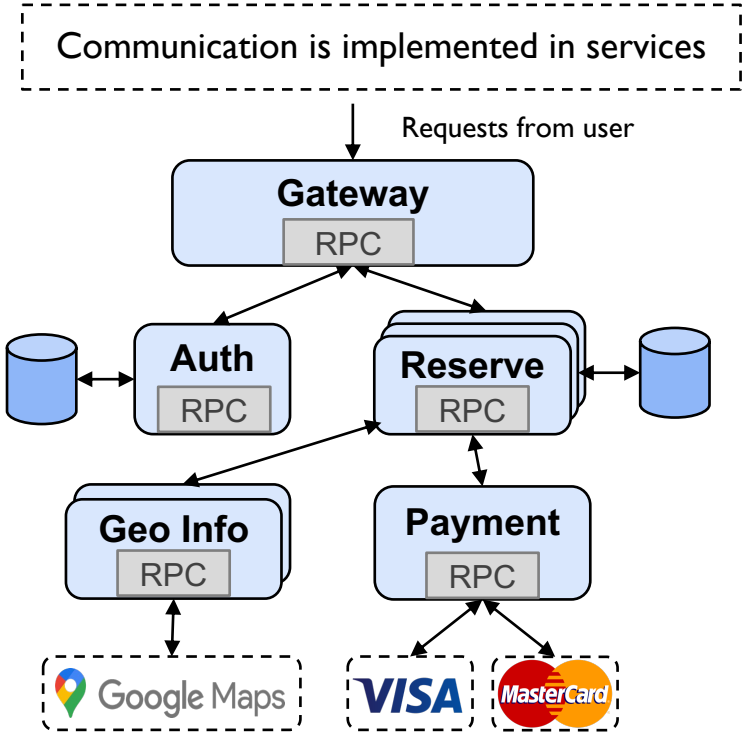
Naiqian Zheng, Tianshuo Qiao, Xuanzhe Liu, Xin Jin

Peking University

北京大學
PEKING UNIVERSITY

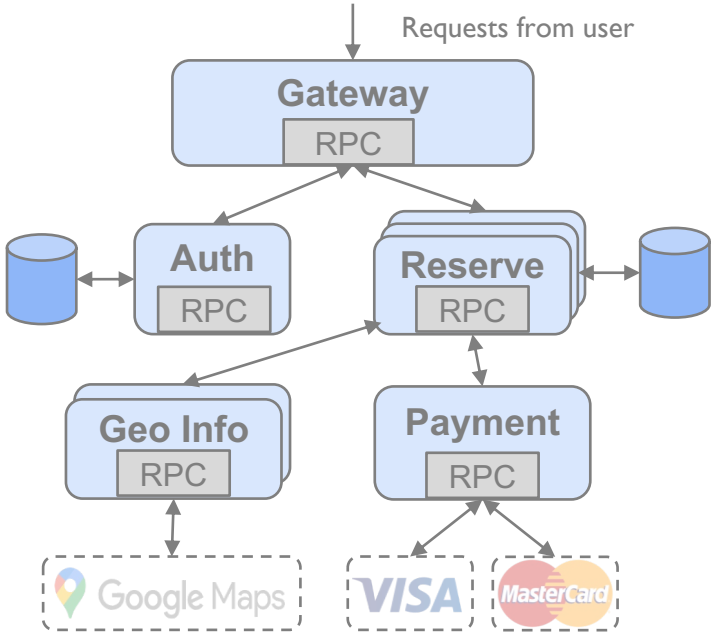
nsdi'25

Service mesh

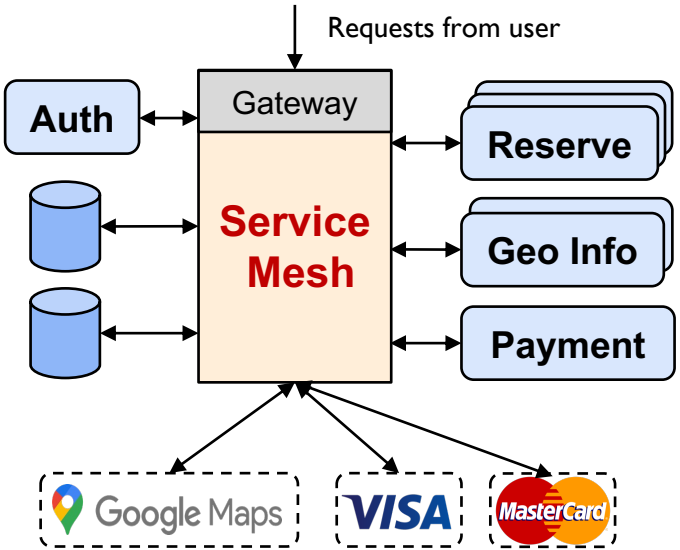


Service mesh

Communication is implemented in services

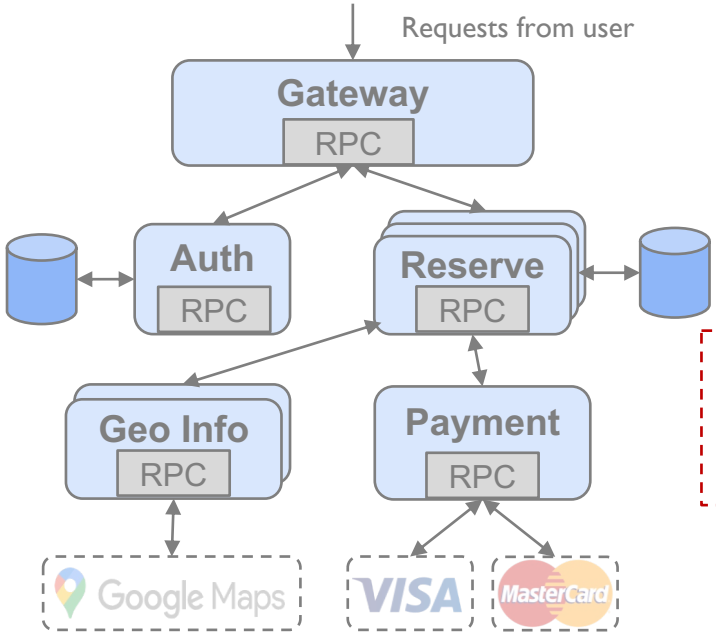


Service mesh takes over communication



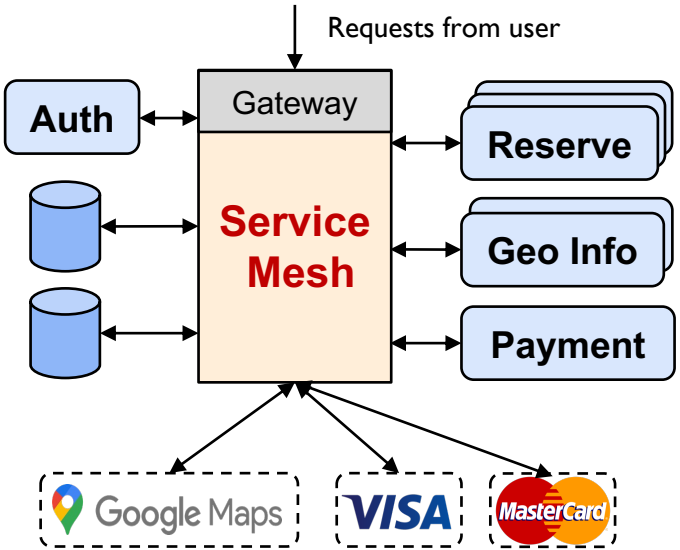
Service mesh

Communication is implemented in services



Benefits:
less maintenance costs
better reliance
better observability

Service mesh takes over communication



Service mesh

- Service mesh is the *"narrow waist"* of microservice communication

popular on GitHub

widely used by industry

integrated in clouds

Service mesh

- Service mesh is the *"narrow waist"* of microservices communication

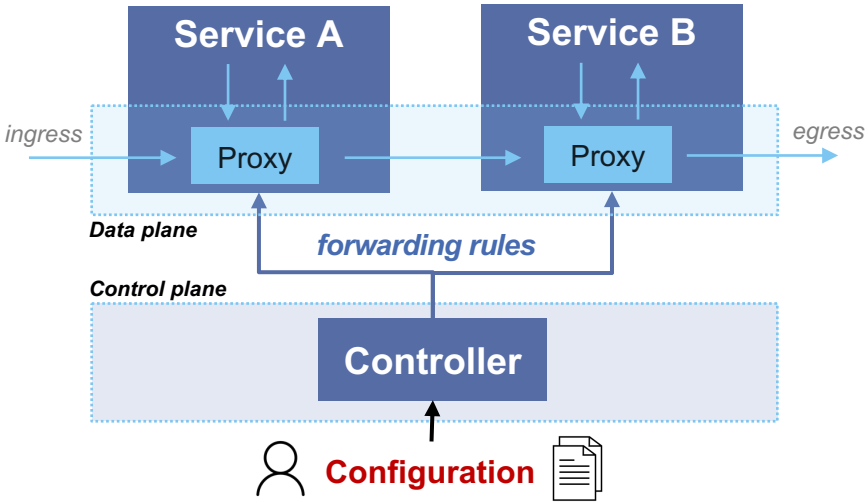
popular on GitHub

widely used by industry

integrated in clouds

- Service mesh functionalities:
  - **Traffic Management**: service routing, load balancing, A/B testing ...
  - Authentication
  - Security
  - Observability

Service mesh is complex



Output: network behavior

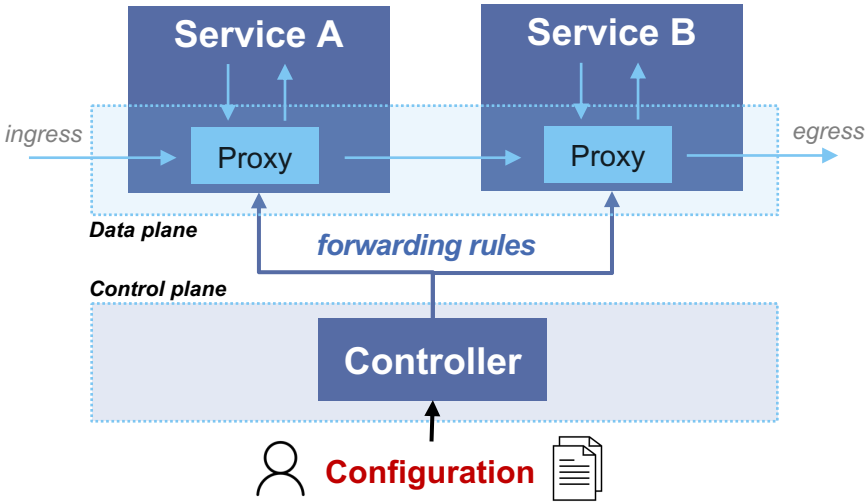
(abstract, logics for arbitrary reqeusts)

Input: communication configuration

(tens of CRDs, millions of options)

Credit to istio: https://istio.io/latest/about/service-mesh/

Service mesh is complex



Code base: extremely complex

(1,000+ components, 300,000+ lines of code)

Output: network behavior

(abstract, logics for arbitrary reqeusts)

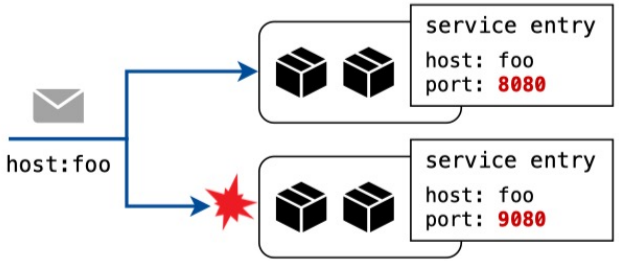
Input: communication configuration

(tens of CRDs, millions of options)

Complexity always brings bugs!

Service mesh is buggy

MeshTest



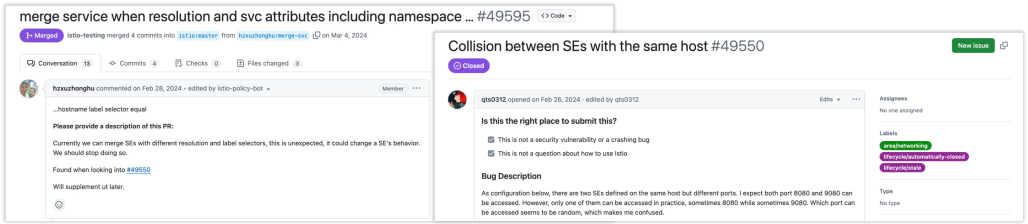
A bug found from istio by MeshTest

- Same host + different port => does not work
- Caused by incorrect rule merge in EDS

```
$ istioctl pc cluster sender-3 --fqdn "www.bookinfo.com"
SERVICE FQDN         PORT     SUBSET     DIRECTION     TYPE     DESTINATION RULE
www.bookinfo.com     8080     -          outbound      EDS
www.bookinfo.com     9080     -          outbound      EDS

$ istioctl pc endpoint sender-3 --cluster "outbound|8080||www.bookinfo.com"
ENDPOINT     STATUS     OUTLIER CHECK     CLUSTER

$ istioctl pc endpoint sender-3 --cluster "outbound|9080||www.bookinfo.com"
ENDPOINT              STATUS       OUTLIER CHECK     CLUSTER
10.244.1.90:9080      DRAINING     OK                outbound|9080||www.bookinfo.com
10.244.2.134:9080     DRAINING     OK                outbound|9080||www.bookinfo.com
```



Reported in https://github.com/istio/istio/issues/49550

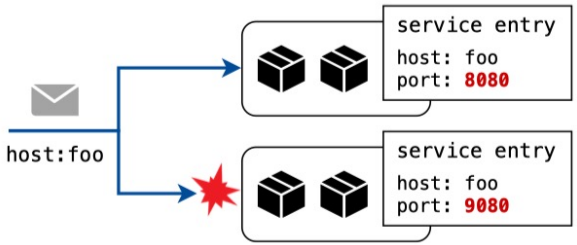
Fixed in https://github.com/istio/istio/pull/49595

Existing tests are not sufficient

- Existing tests
  - A lot of unit tests
    (Istio has 10,000+ unit tests)
  - Very few end-to-end tests
    (Istio has 160 e2e tests, Linkerd has 30 e2e tests)

Existing tests are not sufficient

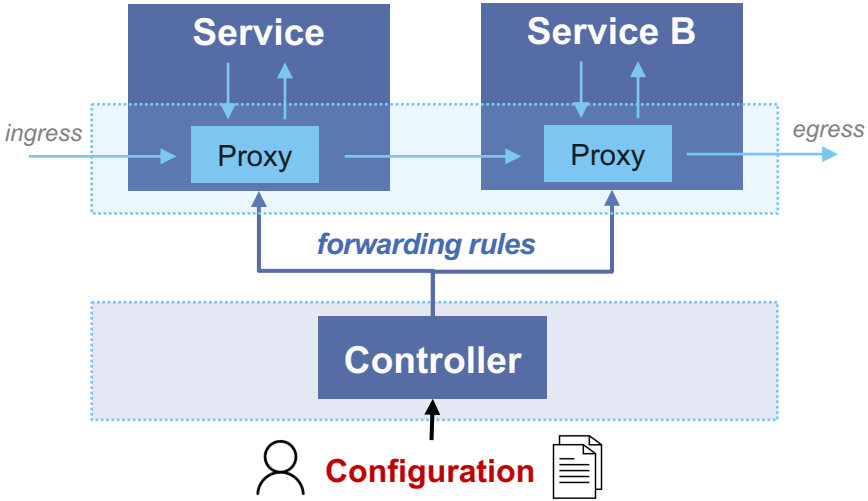
- Existing tests
  - A lot of unit tests
    (Istio has 10,000+ unit tests)
  - Very few end-to-end tests
    (Istio has 160 e2e tests, Linkerd has 30 e2e tests)
- End-to-end testing is effective for the interactions between functions



- Simple in end-to-end testing
- Difficult for unit testing
  Since it is caused by rule merging between two functions

End-to-end testing is challenging

MeshTest



Input

- configurations describing network functions

Output

- network behaviors for requests

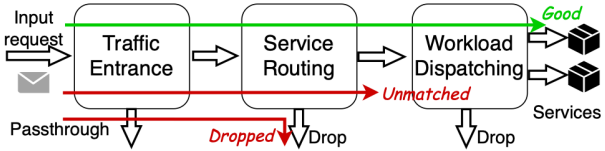
Two steps testing

- Step 1: service mesh configuration generation
- Step 2: network behavior checking

End-to-end testing is challenging

Challenge 1:

The input configurations must be *end-to-end effective*




```
Kind: VirtualService          Kind: DestinationRule
Route:                        Host: example.com
  Host: example.com           Subset:
  Subset: v1 ---------------> v1: workload-1
          v2 ---------------> ✸
```

end-to-end path

broken path (subset:v2 not defined)

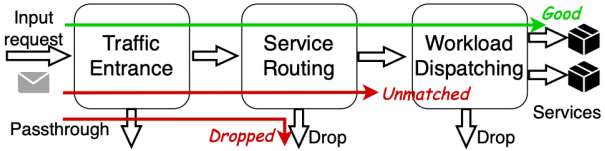
If all requests cannot go to egress service, the input is NOT end-to-end effective.

End-to-end testing is challenging

MeshTest

Challenge 1:

The input configurations must be *end-to-end effective*




```
Kind: VirtualService          Kind: DestinationRule
Route:                        Host: example.com
  Host: example.com           Subset:
  Subset: v1 ---------------> v1: workload-1
          v2 ---------------> ✸
```

end-to-end path

broken path (subset:v2 not defined)

If all requests cannot go to egress service, the input is NOT end-to-end effective.

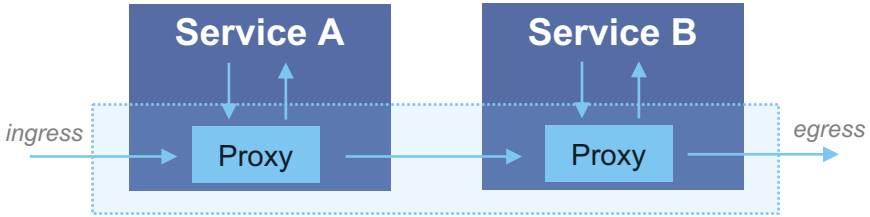
- The configuration must orchestrate functions to compose end-to-end service flow paths
- Each function needs to pass validation rules

- Symbolic execution?
  - 1 million+ options explodes
- Fuzzing?
  - challenging to compose e2e service flow paths
  - not easy to pass constraint validation

End-to-end testing is challenging

Challenge 2:

The output correctness cannot be directly judged

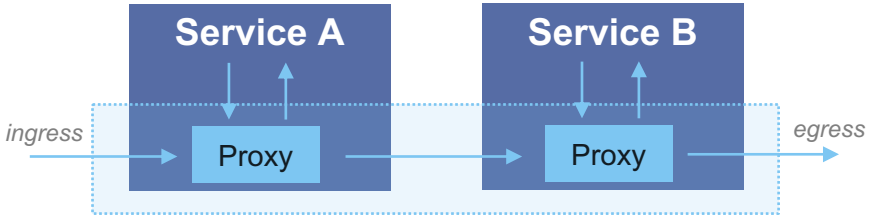


- Correct network behavior means that the service mesh can correctly process any requests
- One request is handled correctly does not mean that all requests will be handled correctly.

End-to-end testing is challenging

Challenge 2:

The output correctness cannot be directly judged



- Correct network behavior means that the service mesh can correctly process any requests
- One request is handled correctly does not mean that all requests will be handled correctly

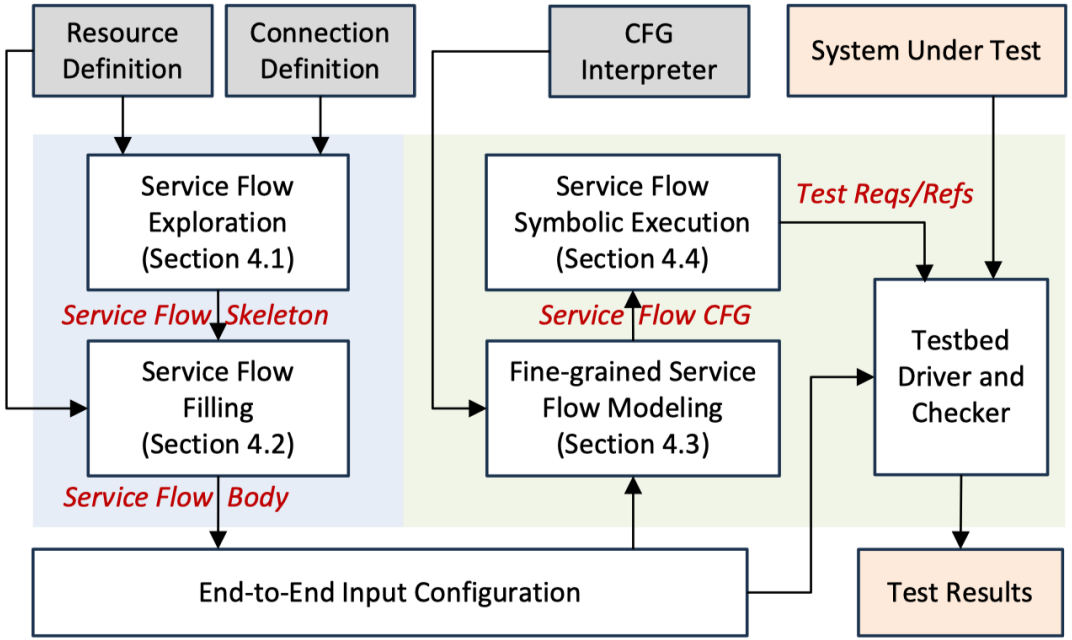
We need to ...

- Choose a comprehensive set of requests that is capable to represent all requests
- Infer the correct processing behaviors of each representative request

The checker should be automatic

MeshTest: end-to-end testing service mesh

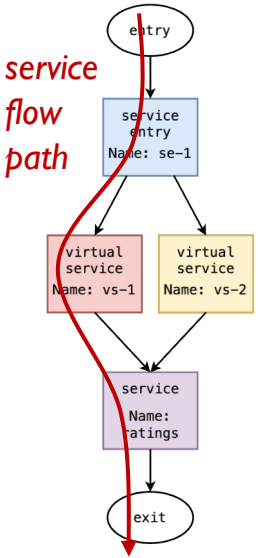
End-to-end configuration generation



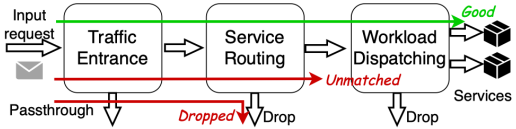
Automatic service mesh oracle checking

Stage 1: service flow exploration

- We start from service flows – the key of end-to-end input configuration



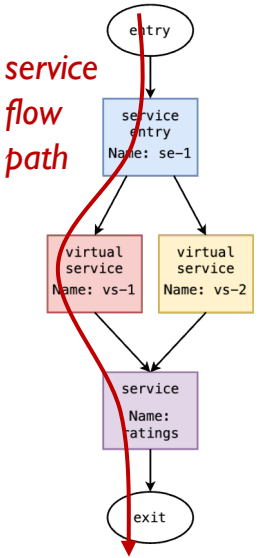
Goal: create e2e service flows



- Domain specific service flow skeleton abstraction
  - Which resources are used in the configuration
  - How resources transmit requests for service flow
- Skeletons reveal interactions between resources
  - priority competition
  - request handover

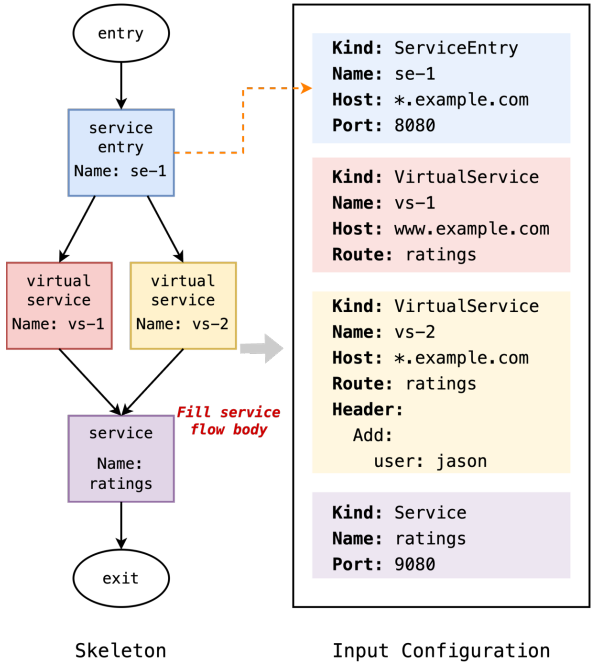
Stage 1: service flow exploration

Goal: create e2e service flows → Goal: create various skeletons



- Insight: vulnerable resource interactions
- Generate skeletons to cover all resource interactions
- Details
  - Start from interaction seed
  - Extend to entry and exit side
  - Compose complete service flow skeleton
- Open to other heuristic seeds

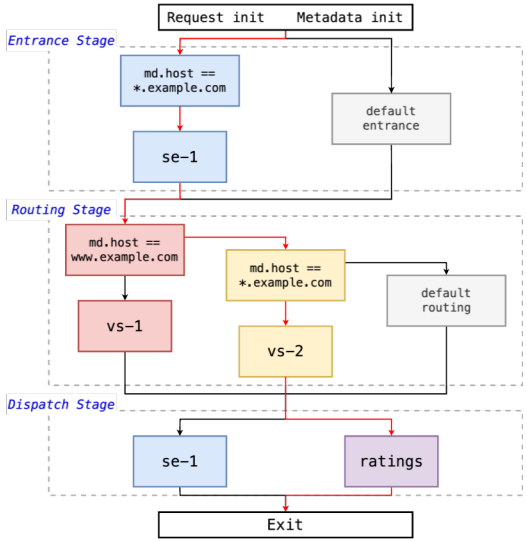
Stage 2: service flow filling



Goal: transform skeleton into complete configuration

- Fill connector fields to realize resource interactions
- Extend configurations with more options
  - by constraint based fuzzing
- More details in out paper...

Stage 3: fine-grained model



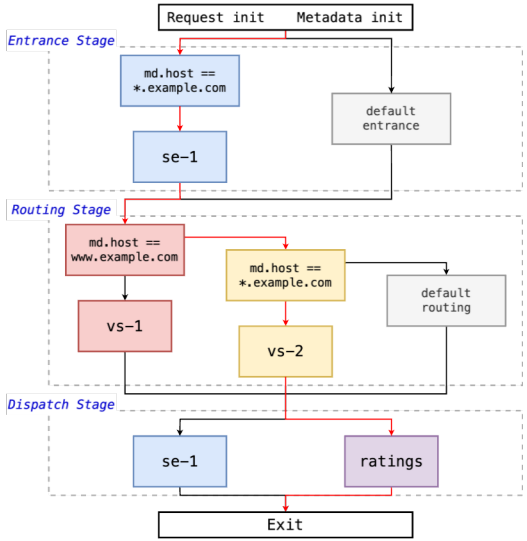
CFG

Goal: select a comprehensive set of requests to check

- Input generator: stage 1 + stage 2
- Oracle: check whether service mesh realizes input configuration correctly

- Stage 3 models accurate behaviors with CFG
- Each path represents a unique request

Stage 3: fine-grained model



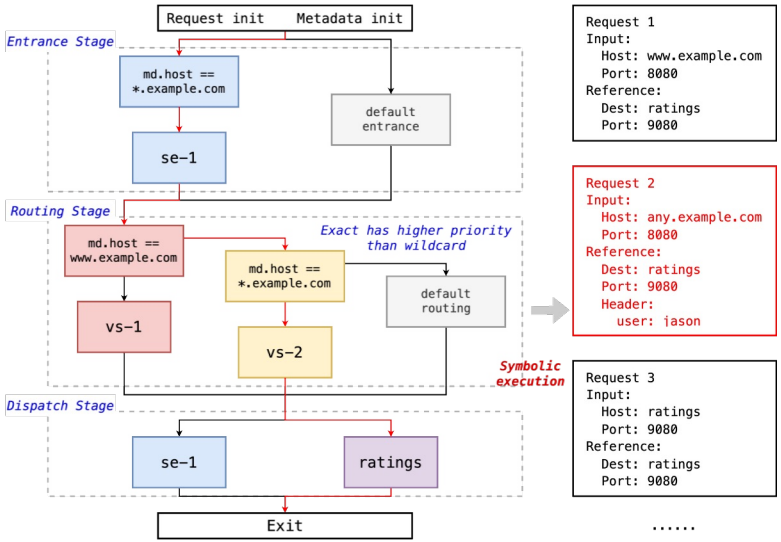
CFG

Goal: select a comprehensive set of requests to check

- Automatic interpreters: configuration => CFG
- The retrofitting effort is less than 2 person-weeks

- We built interpreters for istio and linkerd
- We provide MeshTest CFG APIs for other systems

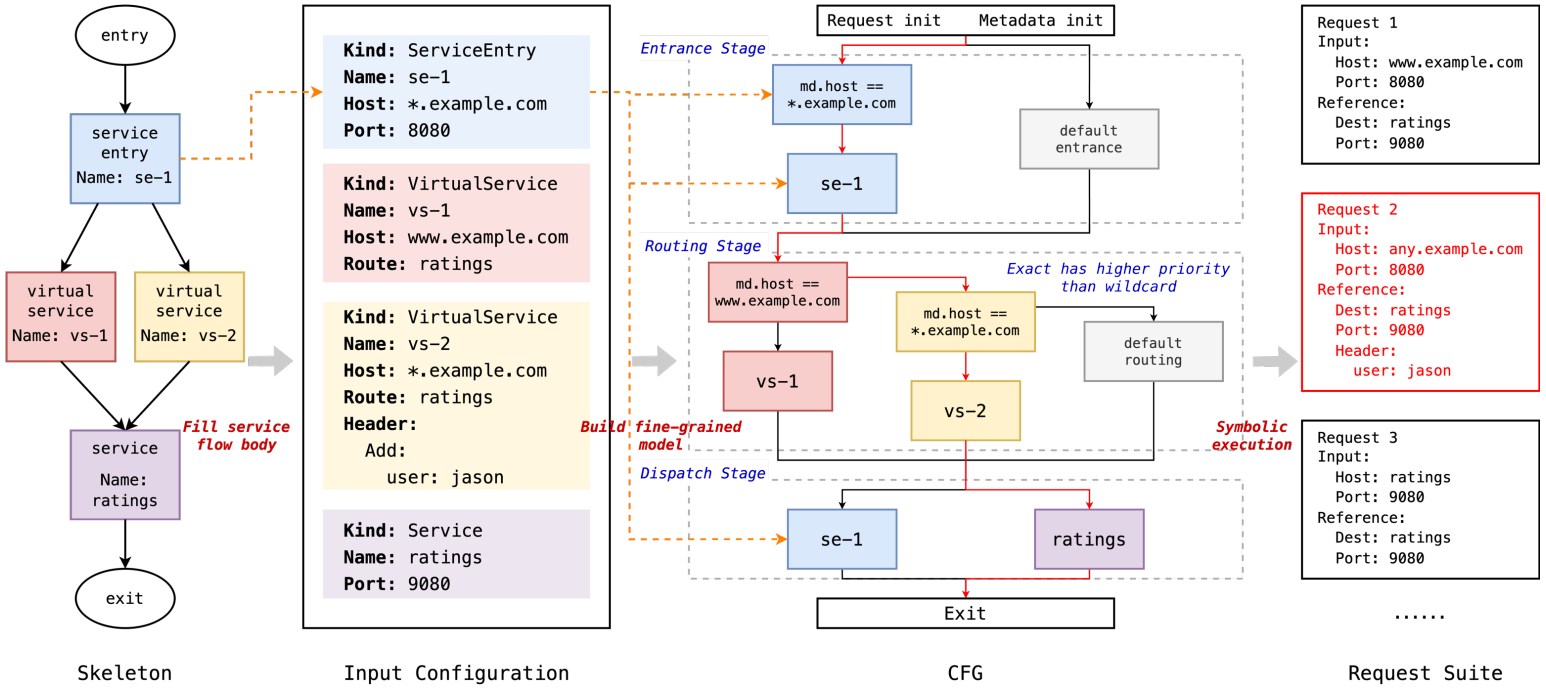
Stage 4: symbolic execution



Goal: check result of requests

- Symbolic execution on CFG
  - solves ingress and reference egress
- Test driver
  - check actual ?= reference
- Properties
  - liveness: no panic or error
  - correctness: consistent with CFG model

MeshTest workflow



Evaluation

Index	Implementation	Bug Description	Status
1 [18]	Istio 1.19–1.21	Empty prefix in specific fields causes an internal error	Fixed
2 [19]	Istio 1.19–1.21/dev	Port 80 is not open by default when Istio gateways are not installed	Reported
3 [20]	Istio 1.19–1.21/dev	Traffic passthroughs cluster when `service entry endpoints` set to an internal IP	Reported
4 [21]	Istio 1.19–1.21/dev	`Service entry` with wildcard host makes traffic skip service routing	Confirmed
5 [22]	Istio 1.19–1.21/dev	`Service entry` defined on port 80 disables `virtual service`	Confirmed
6 [23]	Istio 1.22dev	Routing fails under multiple interleaved resources	Fixed
7 [24]	Istio 1.19–1.21/dev	Traffic is not dropped when port not matched in `virtual service`	Confirmed
8 [25]	Istio 1.19–1.21	`WithoutHeaders` matching fails without target header	Fixed
9 [26]	Istio 1.19–1.21	Delegation influences the priority between `virtual services`	Fixed
10 [27]	Istio 1.19–1.21/dev	Match conditions influence the choice of `virtual service` for gateway	Confirmed
11 [28]	Istio 1.19–1.21/dev	`Service` defined on port 80 disables `virtual service`	Reported
12 [29]	Istio 1.19–1.21	Update on `targetPort` does not trigger update on `EDS`	Fixed
13 [30]	Istio 1.19–1.21/dev	Wildcard matching fails on destination host	Reported
14 [10]	Istio 1.19–1.21	Collision between `service entries` with same host but different workloads	Fixed
15 [31]	Istio 1.19–1.21/dev	`EDS` missing for `service entry` defined on the same host as `service`	Confirmed
16 [32]	Istio 1.19–1.21/dev	`WorkloadSelector` takes effect at wrong place	Confirmed
17 [33]	Istio 1.19–1.21/dev	Header manipulation fails when the value is empty string	Confirmed
18 [34]	Istio 1.19–1.21/dev	Special headers are not ignored in match conditions	Confirmed
19 [35]	Istio 1.19/1.20	Header manipulation fails on pseudo headers	Fixed
20 [36]	Linkerd 2.14	Linkerd extension drives specific pods crash	Fixed
21 [37]	Linkerd 2.14	Routing error under rules with the same matching conditions	Fixed
22 [38]	Linkerd 2.14	Routing error under `http routes` bound on the same `gateway`	Fixed
23 [39]	Linkerd 2.14/dev	Incorrect `hostnames` effects	Confirmed

	Istio	Linkerd	Total
Entrance error	1	0	1
Routing error	9	3	12
Dispatching error	5	0	5
Internal error	1	1	2
Others	3	0	3
Total	19	4	23

MeshTest has found

23 new bugs

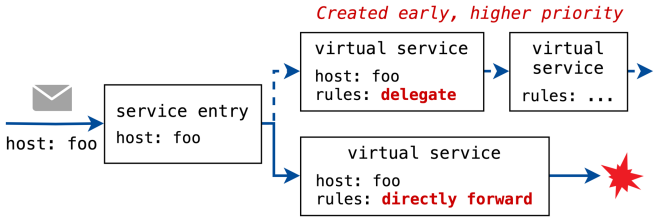
19 confirmed

10 fixed

Evaluation

- Testing coverage
  - 100% coverage on functionalities specified in pairwise resource interactions
  - Istio TM overall: 74.1% (w/o MeshTest) => 78.8% (w/ MeshTest)
  - Istio interaction overall: 70.9% (w/o MeshTest) => 79.4% (w/ MeshTest)
- Efficiency:
  - 2500 configurations per second (input generator)
  - 29 different requests to check one configuration (oracle)

A real bug found by MeshTest



- curl foo/api returns 404 error
- The bug occurs when:
  1. two resources with same host but different rules
  2. higher priority one has a delegation rule

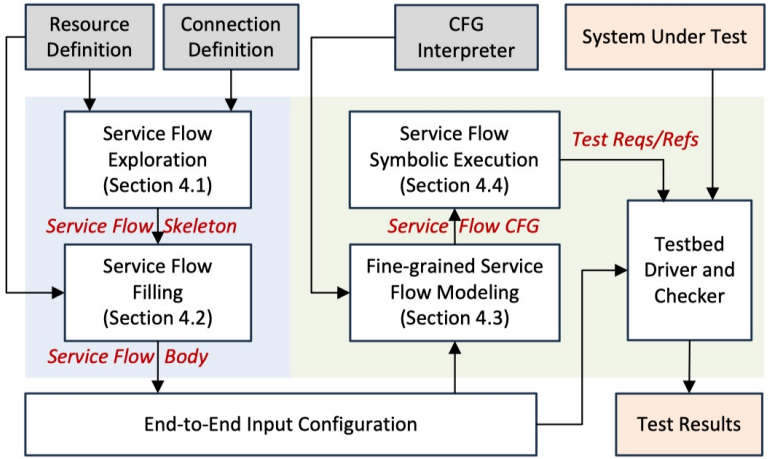
MeshTest found this bug by:

1. generating configuration containing the interaction
2. creating a real request hitting the rule
3. detecting difference between actual and reference

Conclusion

- MeshTest is the first automatic end-to-end testing framework for traffic management of service mesh
- MeshTest is composed by
  - an end-to-end input generator for service mesh
  - a service mesh oracle based on symbolic execution
  - they can work seperately!
- MeshTest has found 23 new bugs (19 confirmed, 10 fixed)
- Available at https://github.com/pkusys/meshtest/

MeshTest: End-to-End Testing for Service Mesh Traffic Management



Thanks!

Naiqian Zheng

www.zhengnq.com

北京大学 PEKING UNIVERSITY

nsdi'25