# Service Discovery Challenges at Scale

Ruslan Nigmatullin

SWE, Dropbox

# Susanin

## Dropbox Service Discovery

# Scale

- Multiple datacenters
- Tens of thousands of hosts per datacenter
- Tens of millions of service discovery clients
- Tens of thousands of state changes per second

# Service Discovery

- Dynamic
- Eventually consistent
- Highly available

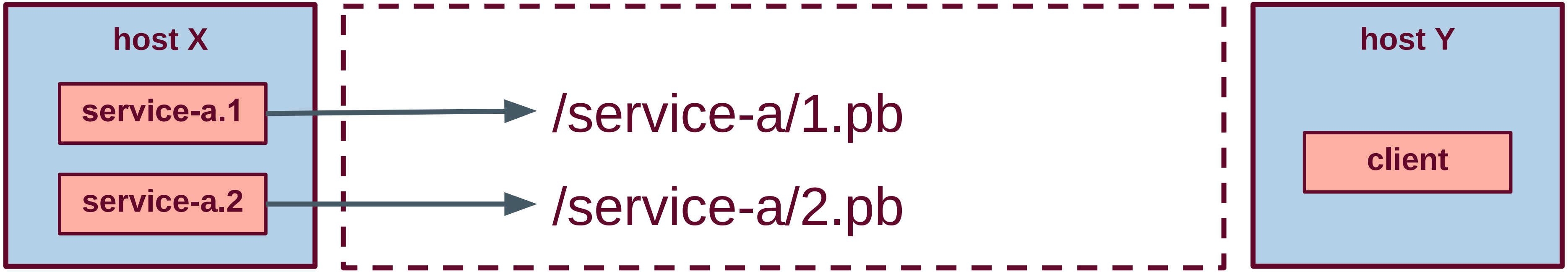
# ZooKeeper Backend

- Operational expertise
- Decent performance
- Read-only mode support

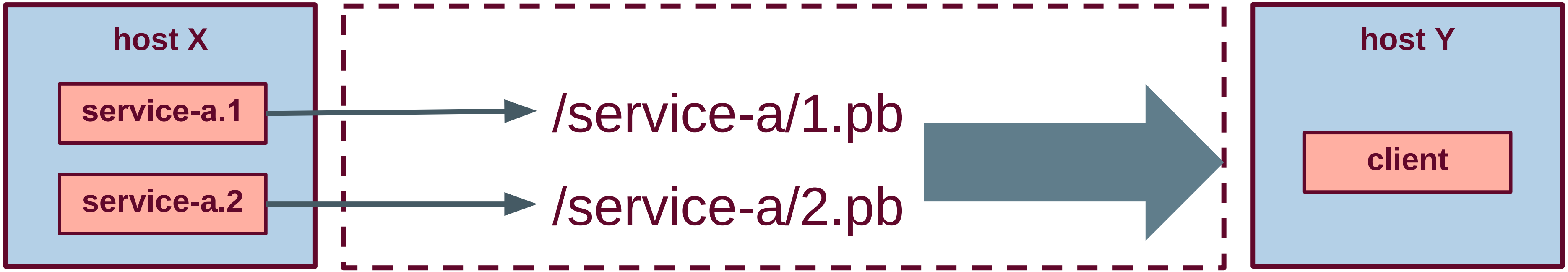
# Scalability


host X
service-a.1
service-a.2
host Y
client


```
# register
zk_cli.create("/service-a/1.pb", zk.EPHEMERAL)
```




```
# register
zk_cli.create("/service-a/1.pb", zk.EPHEMERAL)
zk_cli.create("/service-a/2.pb", zk.EPHEMERAL)
```




```
# register
zk_cli.create(“/service-a/1.pb”, zk.EPHEMERAL)
zk_cli.create(“/service-a/2.pb”, zk.EPHEMERAL)
# resolve
addresses = [
    zk_cli.get(“/service-a/” + child).addr
    for child in zk_cli.children(“/service-a/”)
]
```

# Common approach

Benefits:

- Simple

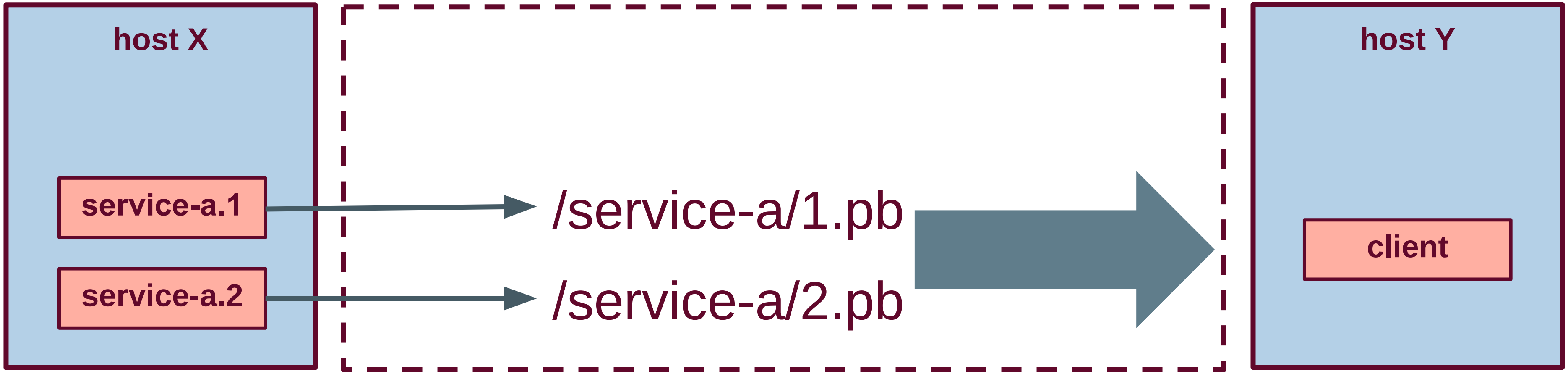
# Common approach

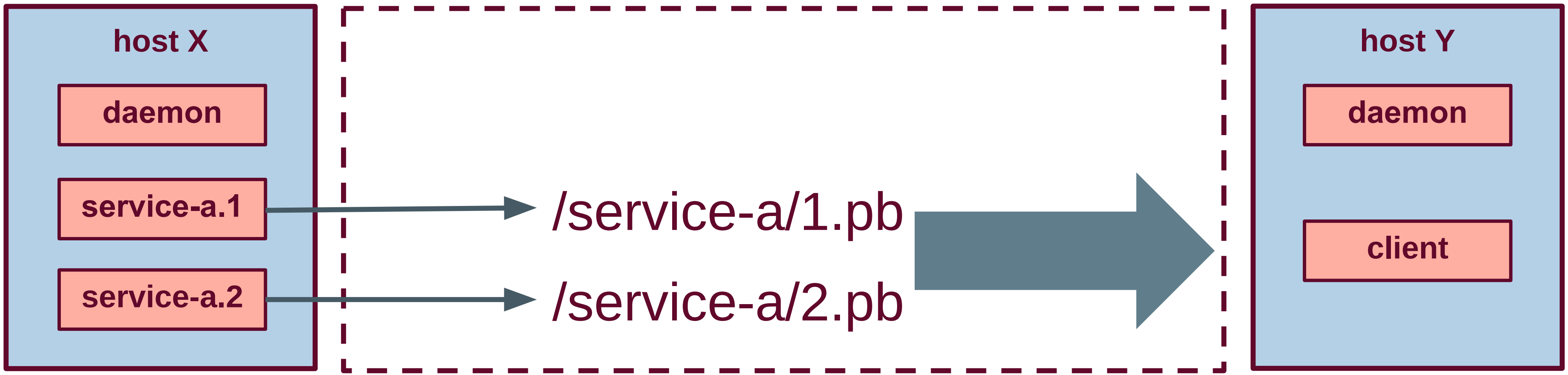
Downsides:

- Load scales up with number of clients
- Thundering herd
- Positive feedback loop
- Susceptible to network outages

# Scaling number of clients

- Session creation/termination is a write operation
- Positive feedback loop in case of overload


host X
service-a.1
service-a.2
/service-a/1.pb
/service-a/2.pb
host Y
client


host X
daemon
service-a.1
service-a.2
/service-a/1.pb
/service-a/2.pb
host Y
daemon
client


host X
daemon
service-a.1
service-a.2
/service-a/1.pb
/service-a/2.pb
host Y
daemon
client

# Summary

Improvements:

- Reduced number of connections
- Per-host local read cache

# Summary

Remaining issues:

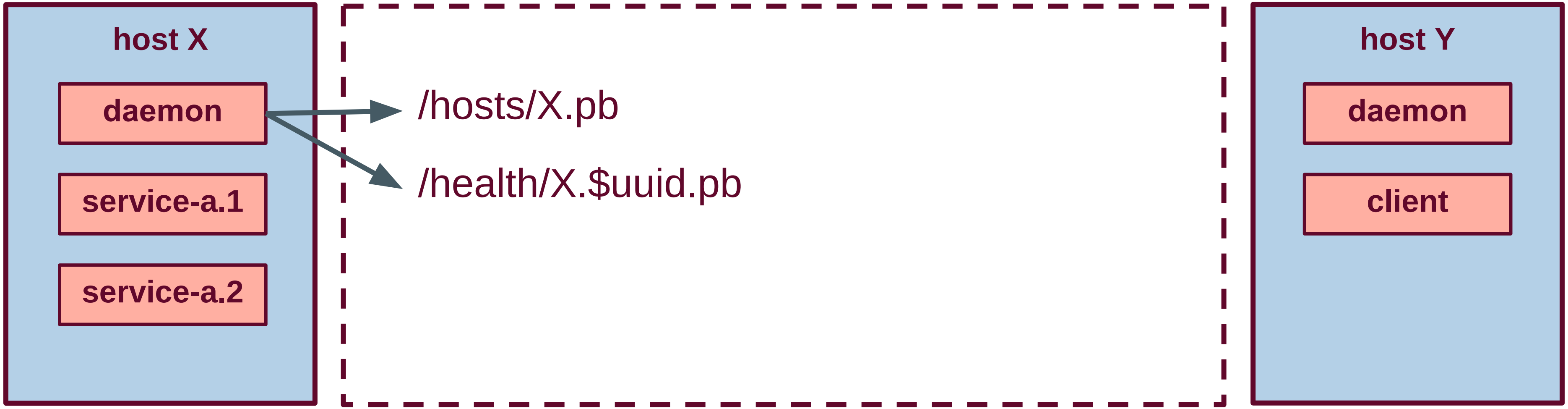
- Positive feedback loop
- Write cost depends on number of backend instances
- Read cost depends on write cost multiplied by number of clients

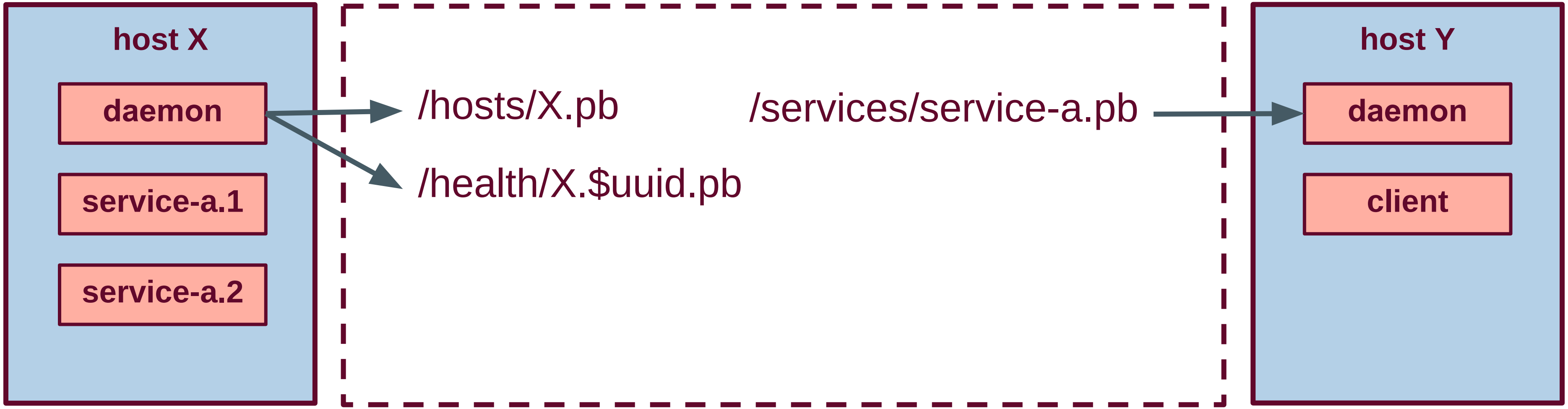

host X
daemon
service-a.1
service-a.2
host Y
daemon
client

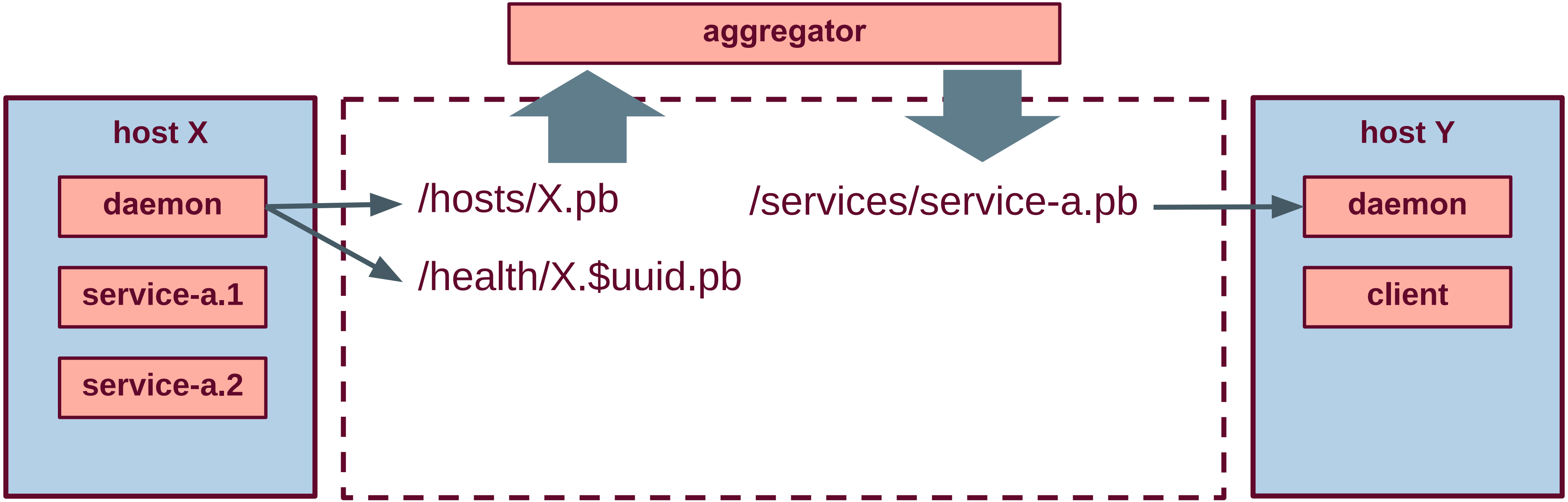

host X
daemon
service-a.1
service-a.2
/hosts/X.pb
host Y
daemon
client

# Scaling number of instances

- Ephemeral files are owned by the session
- Session does not survive process restart
- Health should be indicated by another file


host X
daemon
service-a.1
service-a.2
/hosts/X.pb
/health/X.$uuid.pb
host Y
daemon
client


host X
daemon
service-a.1
service-a.2
/hosts/X.pb
/health/X.$uuid.pb
/services/service-a.pb
host Y
daemon
client


aggregator
host X
daemon
service-a.1
service-a.2
/hosts/X.pb
/health/X.$uuid.pb
/services/service-a.pb
host Y
daemon
client

# Summary

- Separate control and data planes
- Runtime complexity: O(1)
  - Load scales up with number of hosts
  - Load does not scale up with number of clients or rate of updates
- Writes and reads coalescing

# API

# Inspired by DNS

```
# backend
entry = susanin.register(“foo/bar/grpc”, port=5001)
entry.deregister()

# client
addresses = susanin.resolve(“foo/bar/grpc”)
for addresses in susanin.resolve_w(“foo/bar/grpc”):
    pass
```

# Integration with gRPC

```
# backend
server = Server("service-a")
server.serve()

# client
client = Client("service-a")
```

# Courier: Dropbox migration to gRPC

- Integration with Susanin
- Mutual TLS
- Circuit breaking
- Metrics and tracing

More in our tech blog

https://blogs.dropbox.com/tech

# Monitoring and Operations

# Key metrics

Health is defined on per-datacenter basis

- Write availability
- Read availability
- Propagation latency, p95

# Blackbox monitoring

Does the system solve user needs?

Service Discovery:

- Endless register and resolve loop
- Measures latency and availability

ZooKeeper:

- Endless loop of ephemeral writes and reads

# Whitebox monitoring

Is the system correct?

- Introspection API
- Consistency checker

# Whitebox monitoring

Is the system correct?

- Introspection API
- Consistency checker

Found consistency bug in sync protocol in ZooKeeper 3.5

Found bug in leader election implementation

# Disaster Recovery Testing

- Regular leader restart
- Abnormal leader restart
- Leader network shutdown
- Majority network shutdown

# Disaster Recovery Testing

- Regular leader restart
- Abnormal leader restart
- Leader network shutdown
- Majority network shutdown

Found lack of timeouts in ZooKeeper in certain corner cases

# Conclusion

# Lessons learned

- Coalescing helps eliminate feedback loops
- Runtime complexity matters
- Separation of data and control planes allows more design choices
- Verifiable consistency can be used as end-to-end test

# Future work

Sophisticated load balancing:

- Cross-datacenter balancing
- Feedback-driven balancing within datacenter

# Thank you