Automated Analysis with Broker

Matthias Vallentin, CEO
Dominik Charousset, CTO

BroCon '18

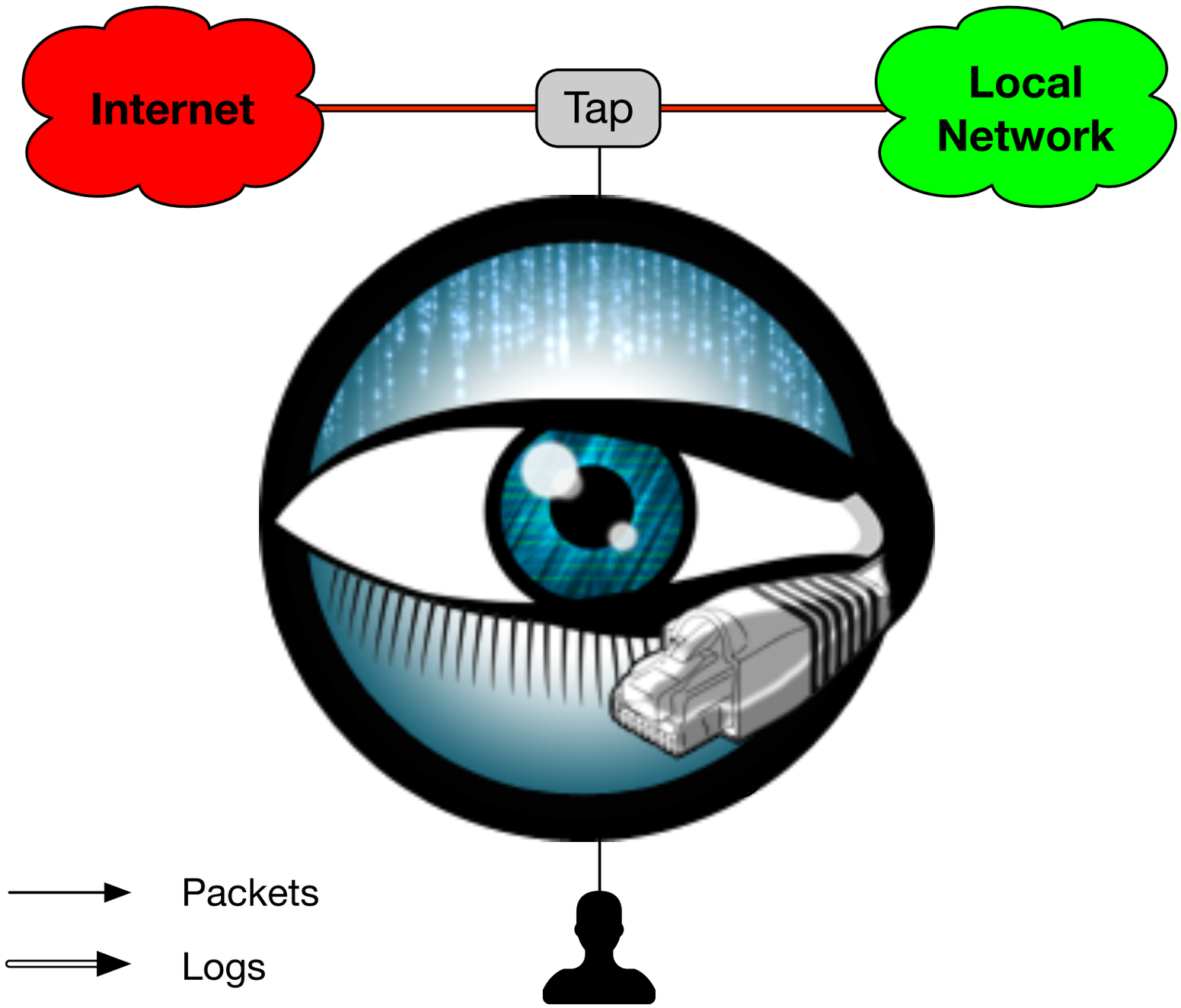
Automated Analysis with Zeeker?

Matthias Vallentin, CEO
Dominik Charousset, CTO

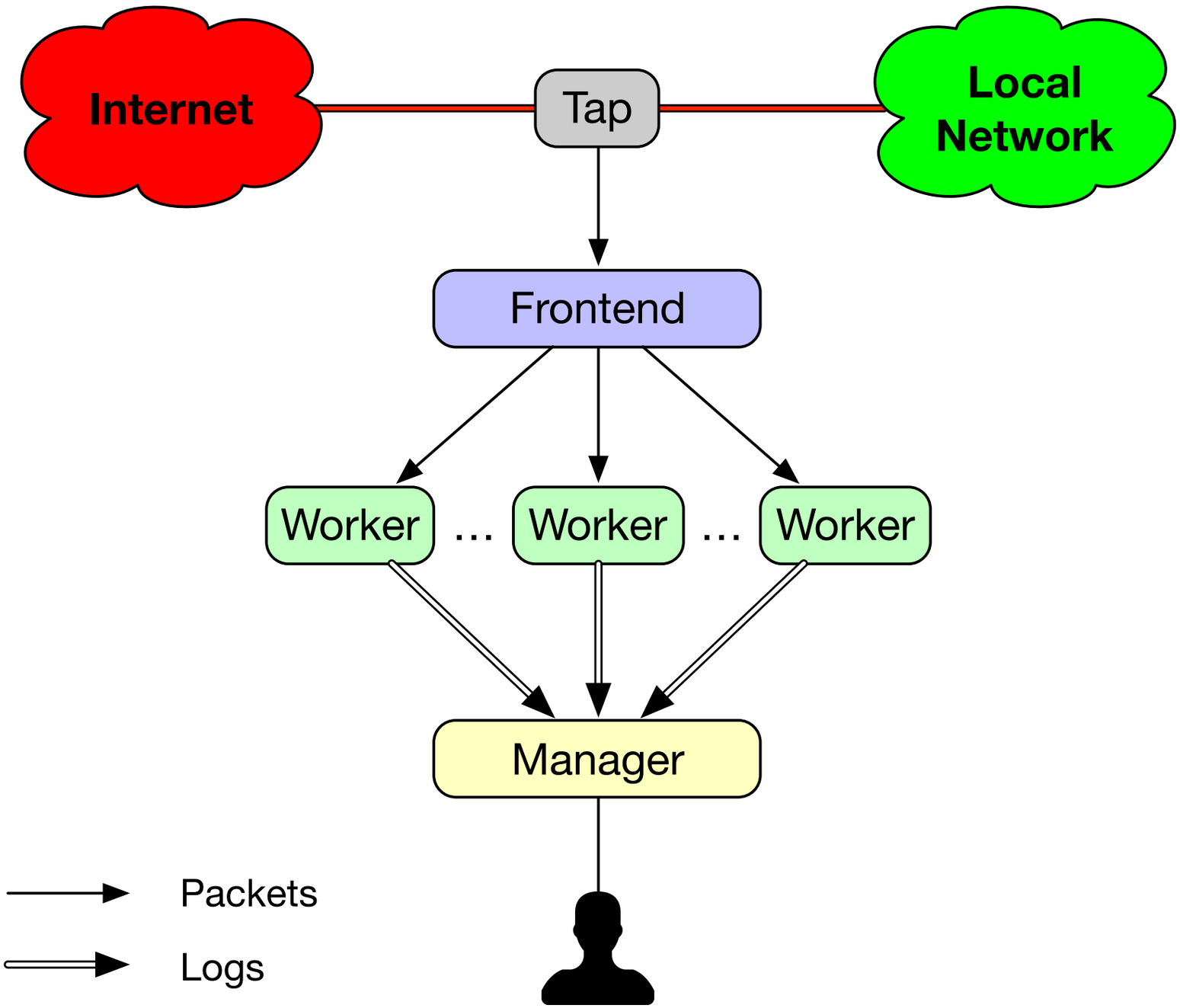
BroCon '18

What is Broker again?

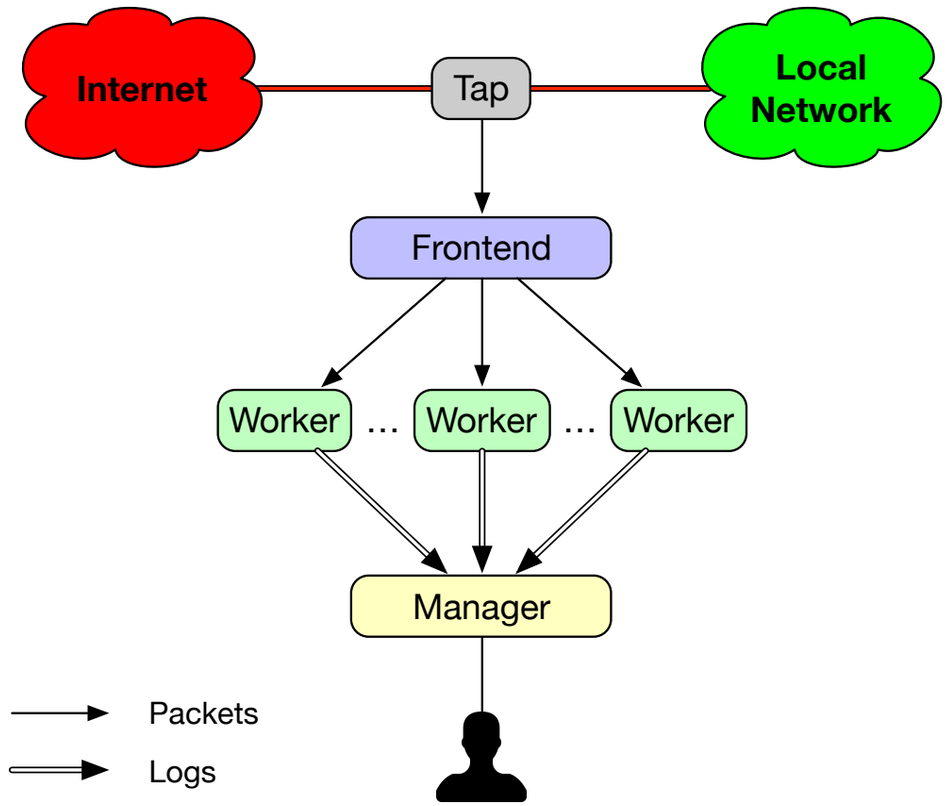
Zeek Communication



Zeek Communication

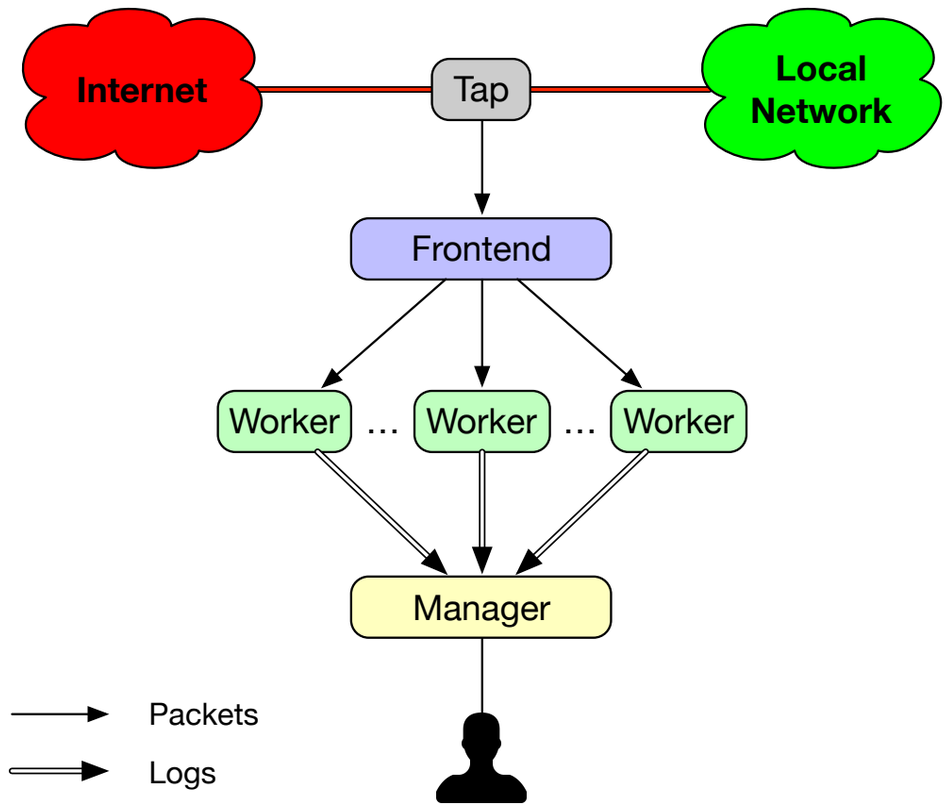


Zeek Communication



- **Manager and workers** exchange valuable insights

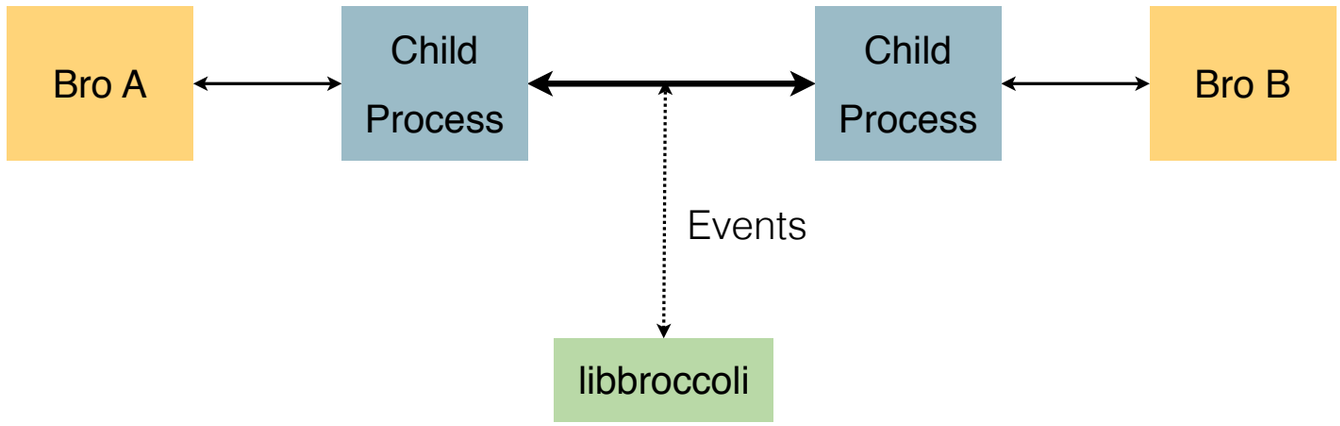
Zeek Communication



- **Manager and workers** exchange valuable insights
- Tapping into this **resource** enables powerful analytics

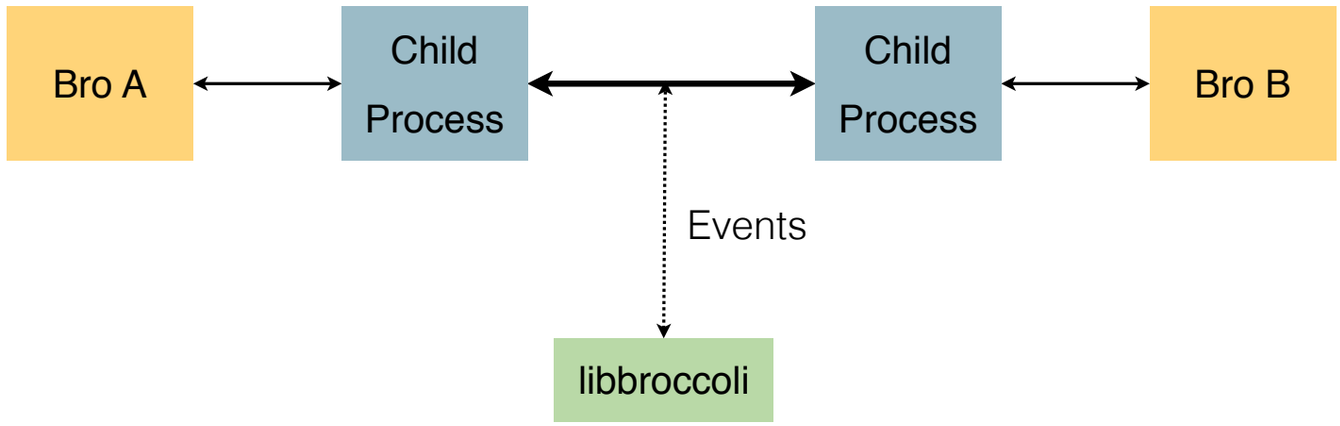
Historic Perspective

Historic Perspective



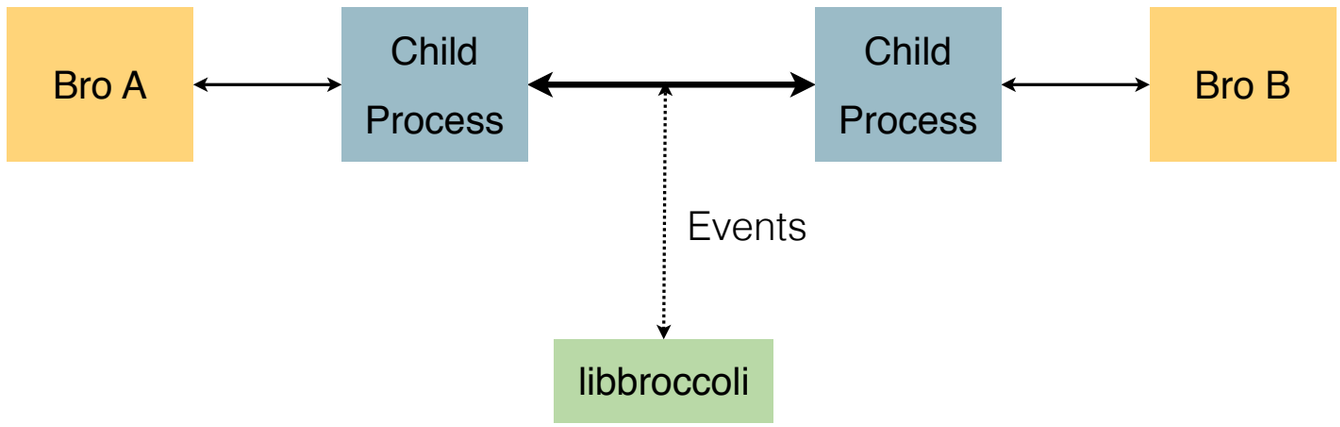
- **Second Bro process** for communication

Historic Perspective



- **Second Bro process** for communication
- Broccoli: **re-implementation** of Bro's protocol

Historic Perspective



- **Second Bro process** for communication
- Broccoli: **re-implementation** of Bro's protocol
- **No persistent state** and limited control over data flow

Introducing Broker

Introducing Broker

- Unify **event access** (C++, Python, Zeek Scripts)

Introducing Broker

- Unify **event access** (C++, Python, Zeek Scripts)
- Provide **persistent key/value stores**

Introducing Broker

- Unify **event access** (C++, Python, Zeek Scripts)
- Provide **persistent key/value stores**
- Leverage **flexible publish/subscribe** communication

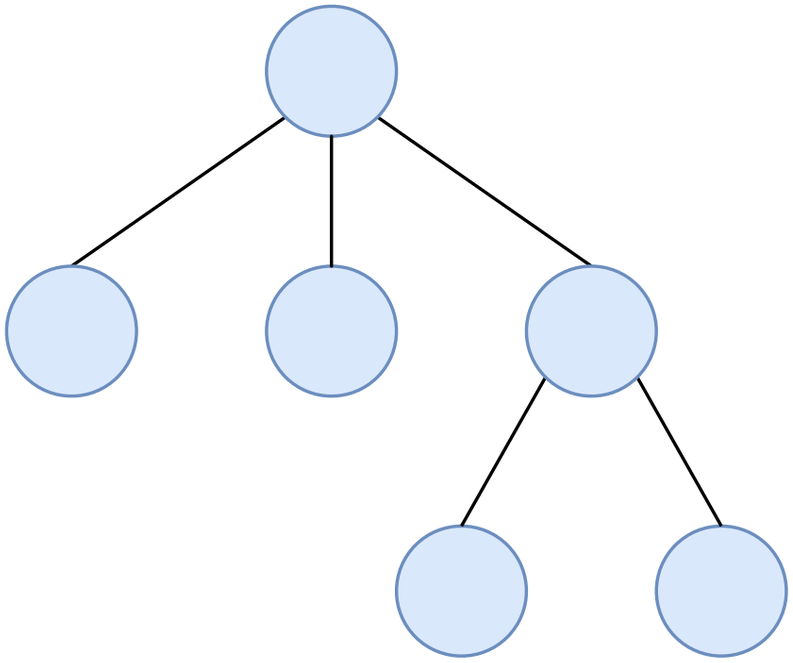
Introducing Broker

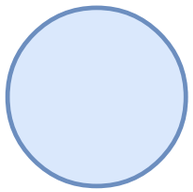
- Unify **event access** (C++, Python, Zeek Scripts)
- Provide **persistent key/value stores**
- Leverage **flexible publish/subscribe** communication
- Enable **cross-correlation** and automated analysis

Broker in a Nutshell

Broker in a Nutshell

- **Endpoints peer** with each other (tree topology without loops)

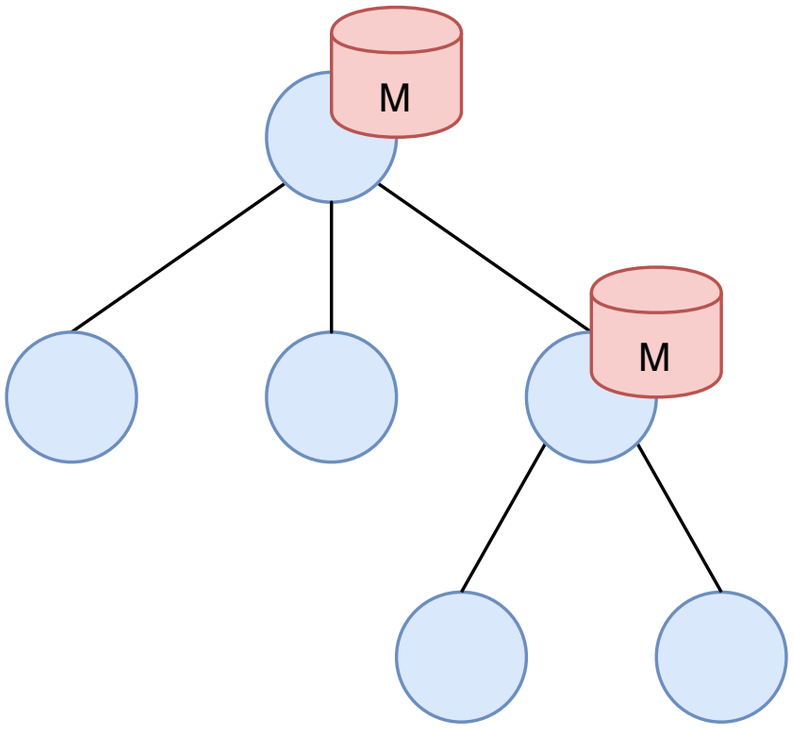


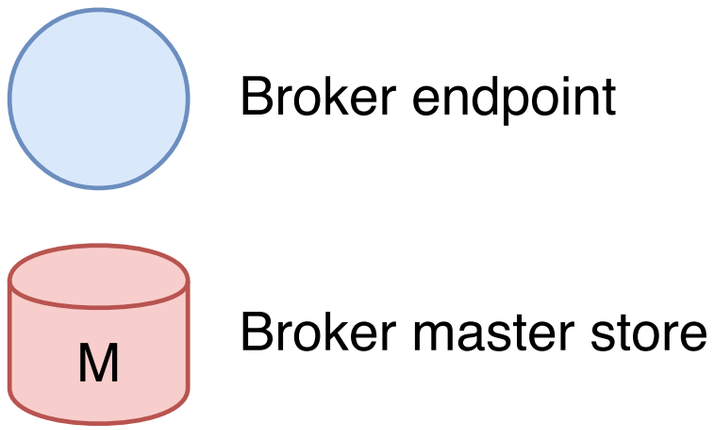


Broker endpoint

Broker in a Nutshell

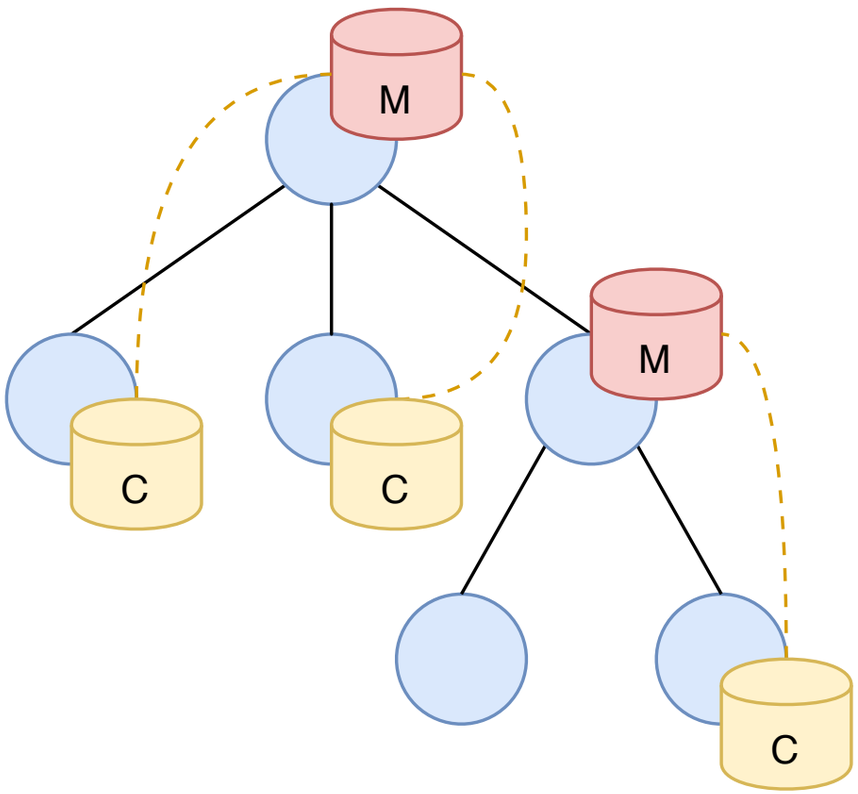
- **Endpoints peer** with each other (tree topology without loops)
- Users can attach **key-value data stores** to endpoints

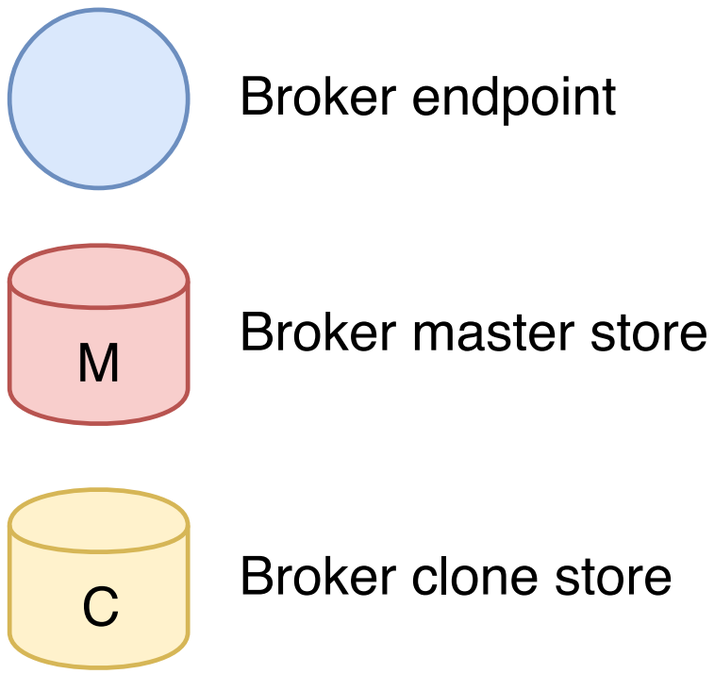




Broker in a Nutshell

- **Endpoints peer** with each other (tree topology without loops)
- Users can attach **key-value data stores** to endpoints
- **Clones act as caches** for faster access



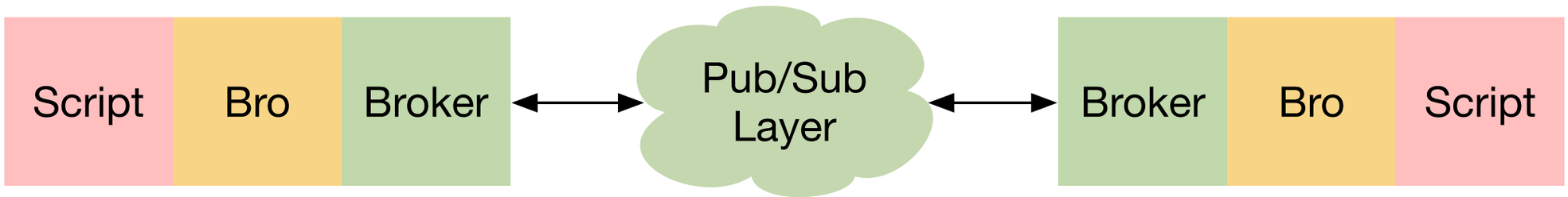


Flexible Zeek Messaging

Flexible Zeek Messaging

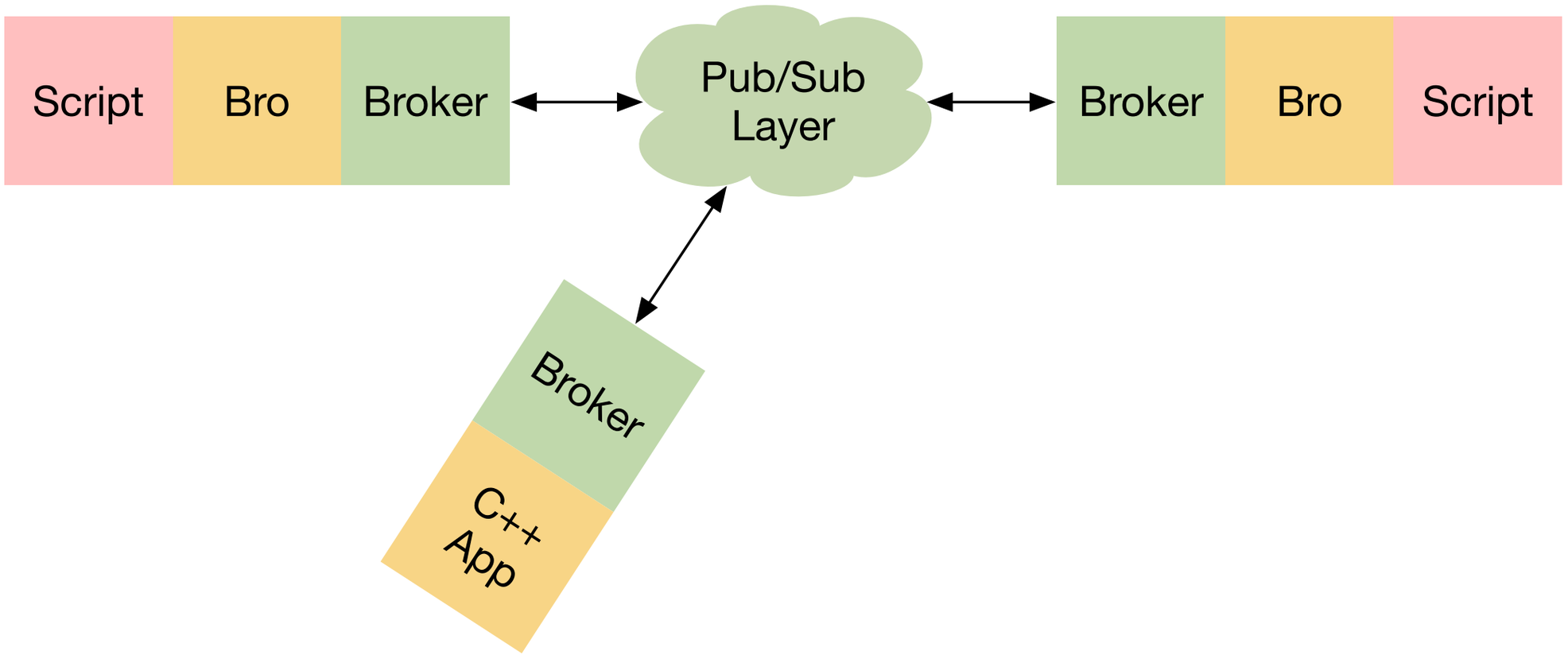
Zeek 2.6

Flexible Zeek Messaging



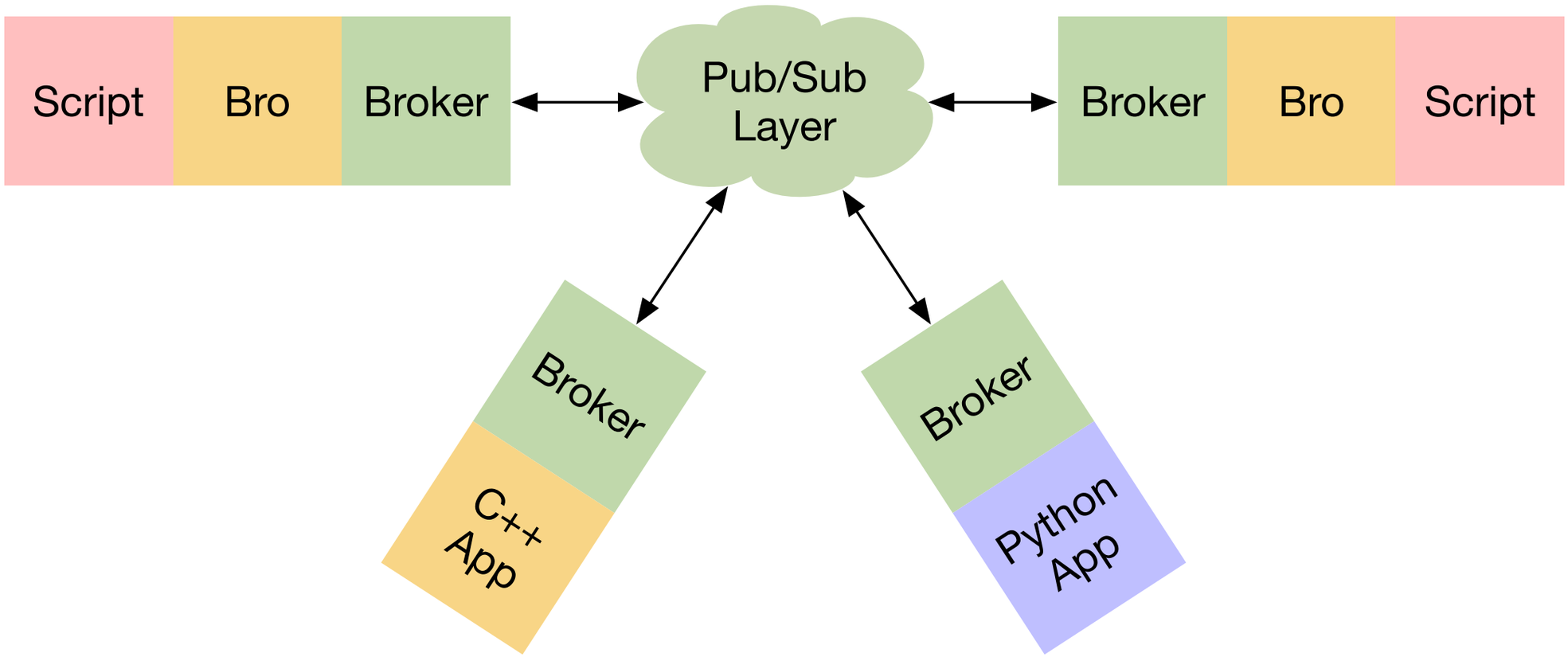
Zeek 2.6

Flexible Zeek Messaging



Zeek 2.6

Flexible Zeek Messaging



Zeek 2.6

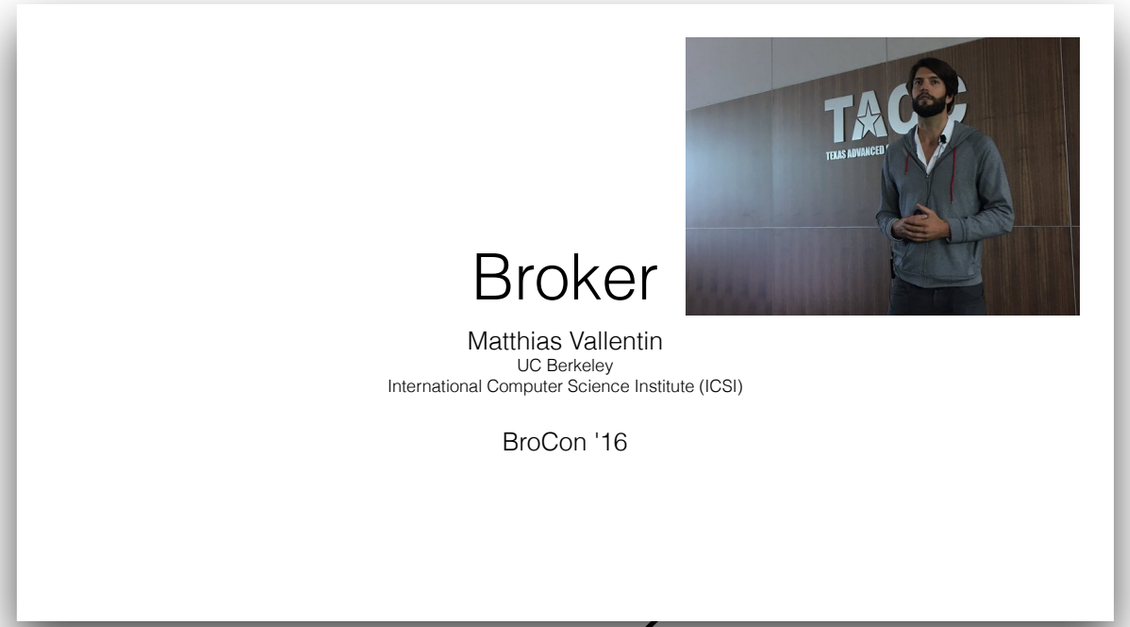
Broker Timeline

Broker Timeline

First Prototype

2015

Broker Timeline

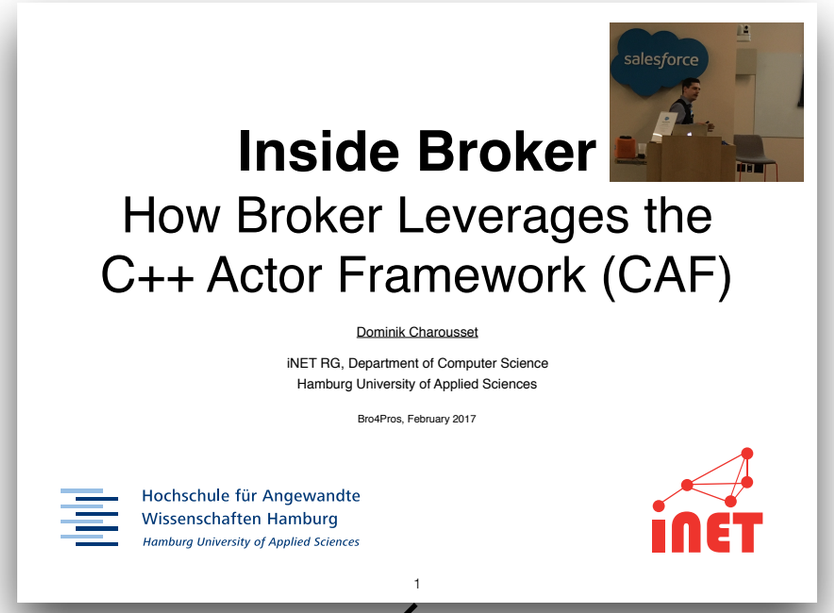


First Prototype

Nonblocking API

2015 **2016**

Broker Timeline

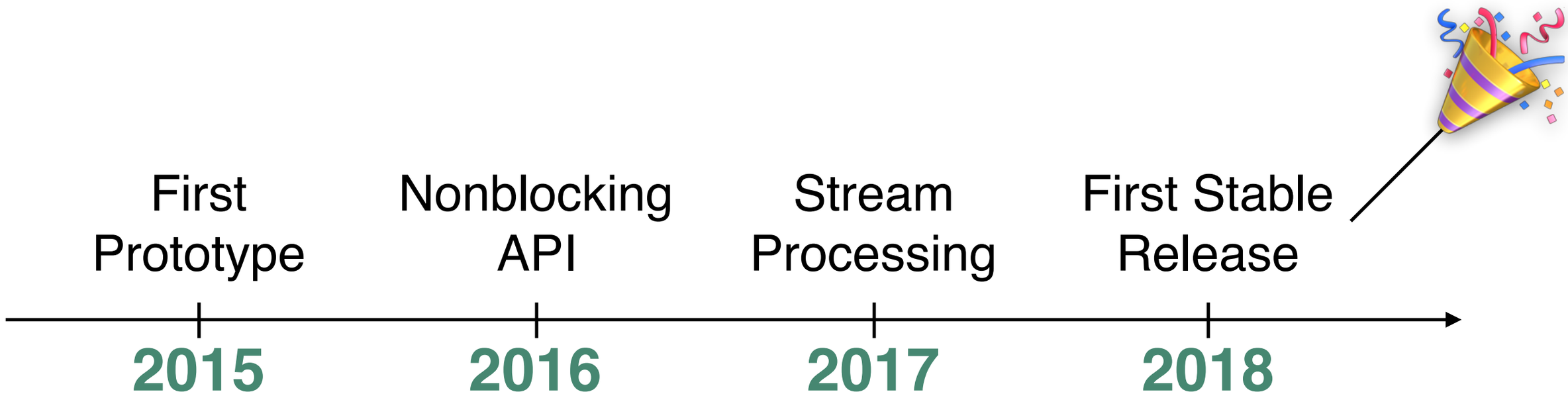


First Prototype — 2015

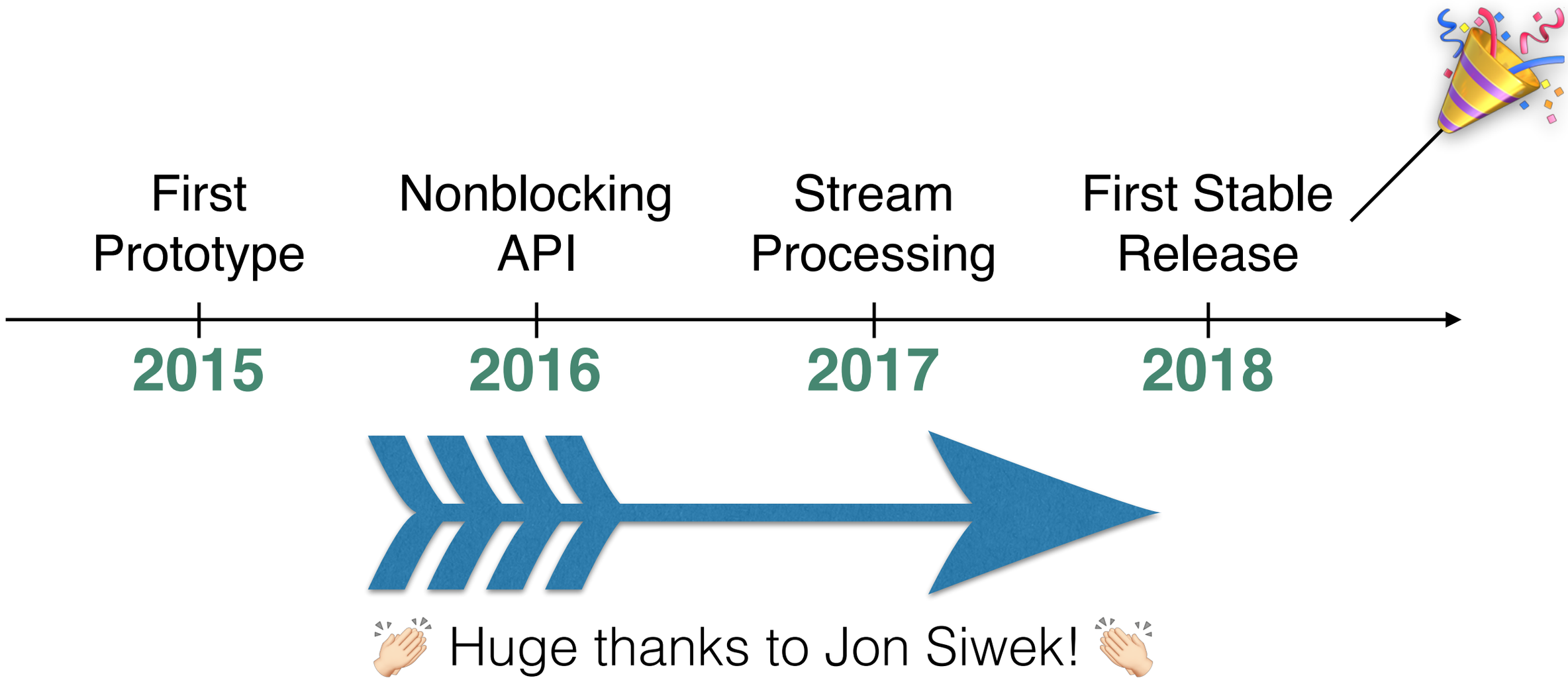
Nonblocking API — 2016

Stream Processing — 2017

Broker Timeline



Broker Timeline



👏 Huge thanks to Jon Siwek! 👏

State of Broker

State of Broker

✅ Stable: tested and merged to master

State of Broker

- ✅ Stable: tested and merged to master
- ✅ Native C++ API and Python bindings

State of Broker

- ✅ Stable: tested and merged to master
- ✅ Native C++ API and Python bindings
- ✅ Topic-based pub/sub topologies

State of Broker

- ✅ Stable: tested and merged to master
- ✅ Native C++ API and Python bindings
- ✅ Topic-based pub/sub topologies
- ✅ Key/value stores with configurable backends

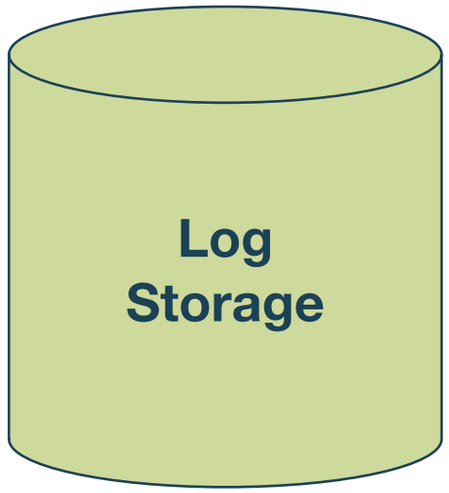
State of Broker

- ✅ Stable: tested and merged to master
- ✅ Native C++ API and Python bindings
- ✅ Topic-based pub/sub topologies
- ✅ Key/value stores with configurable backends
- 🚫 No access to Zeek logs (yet)

What can I do with Broker?

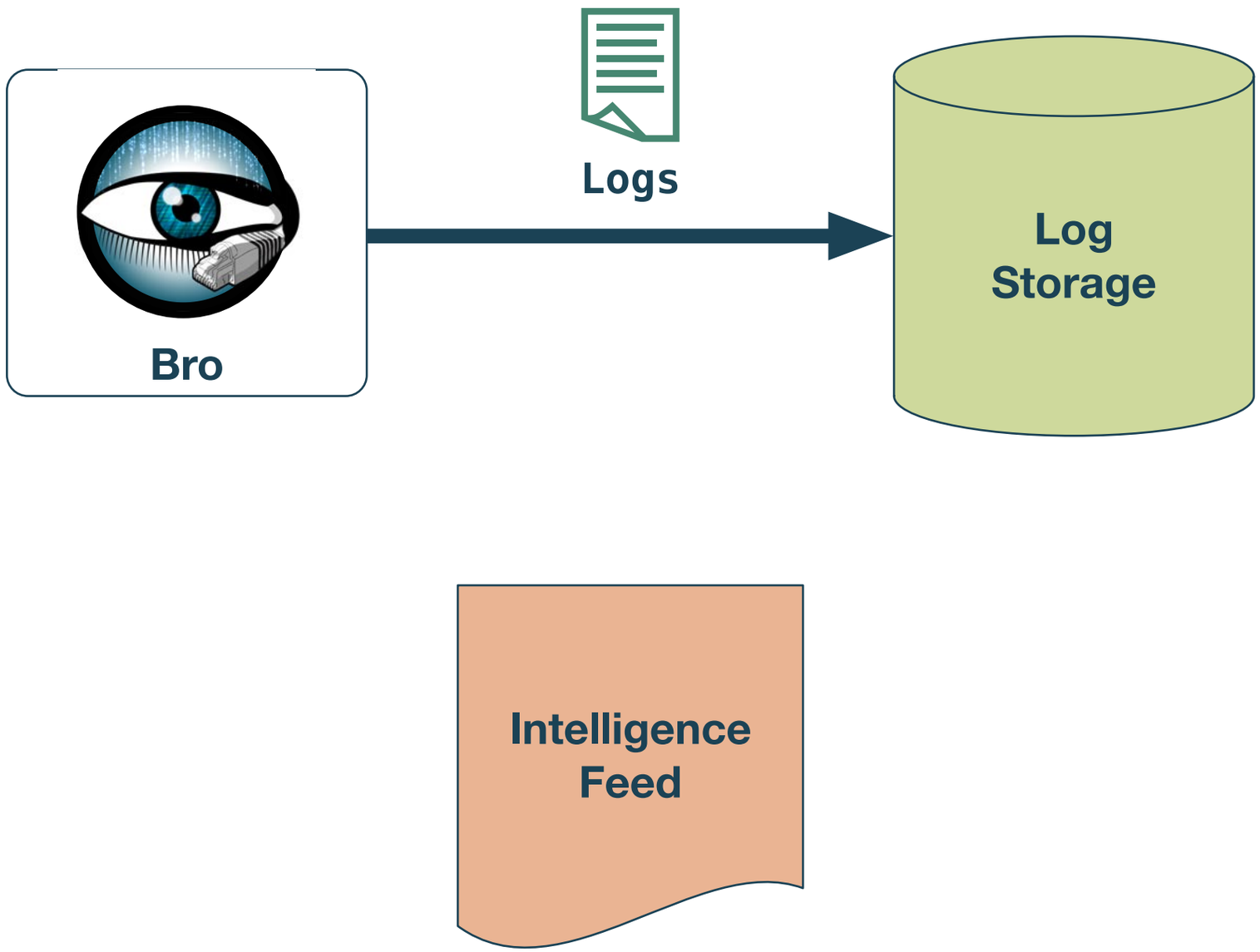
Scenario

Scenario

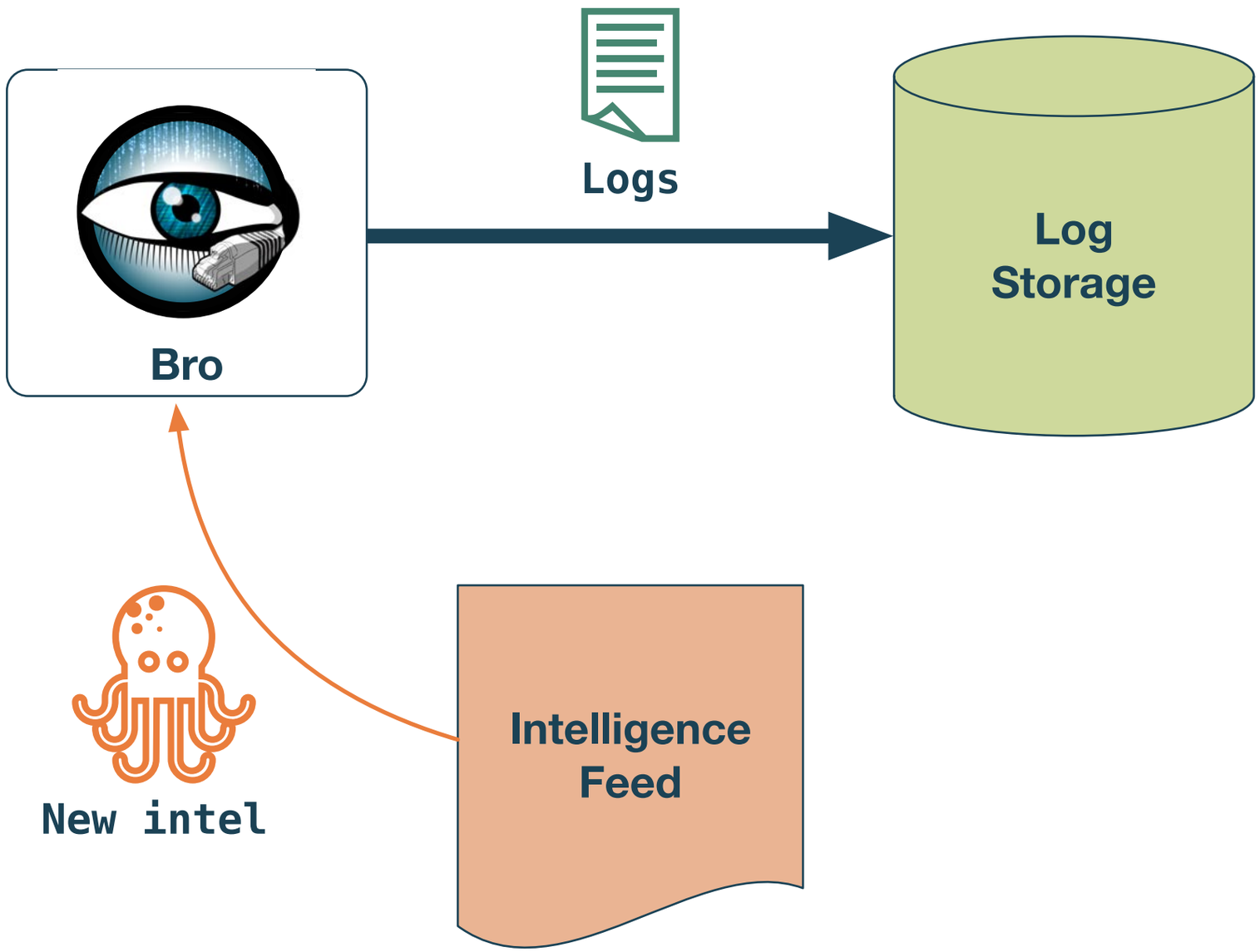


Scenario

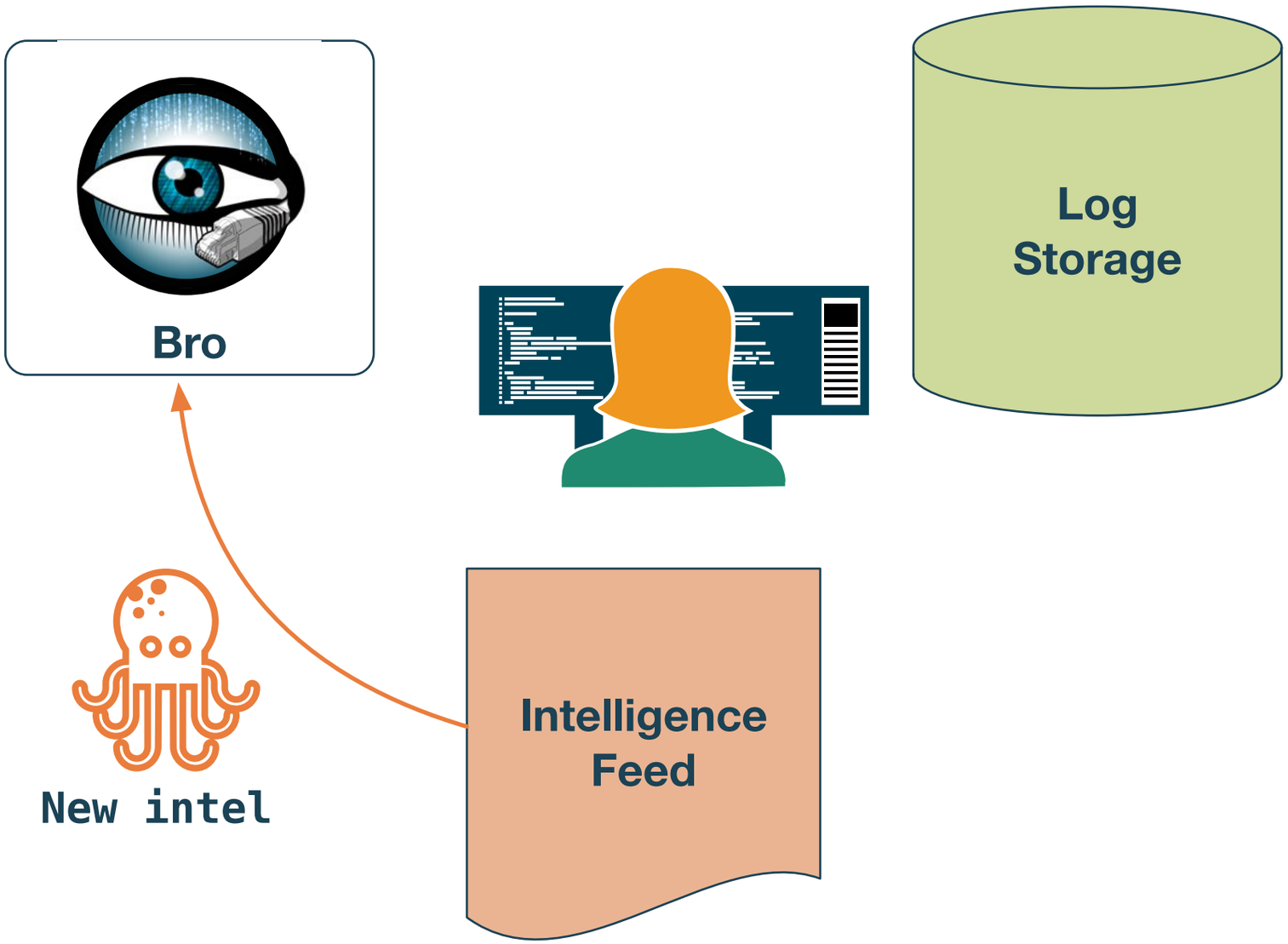
Scenario



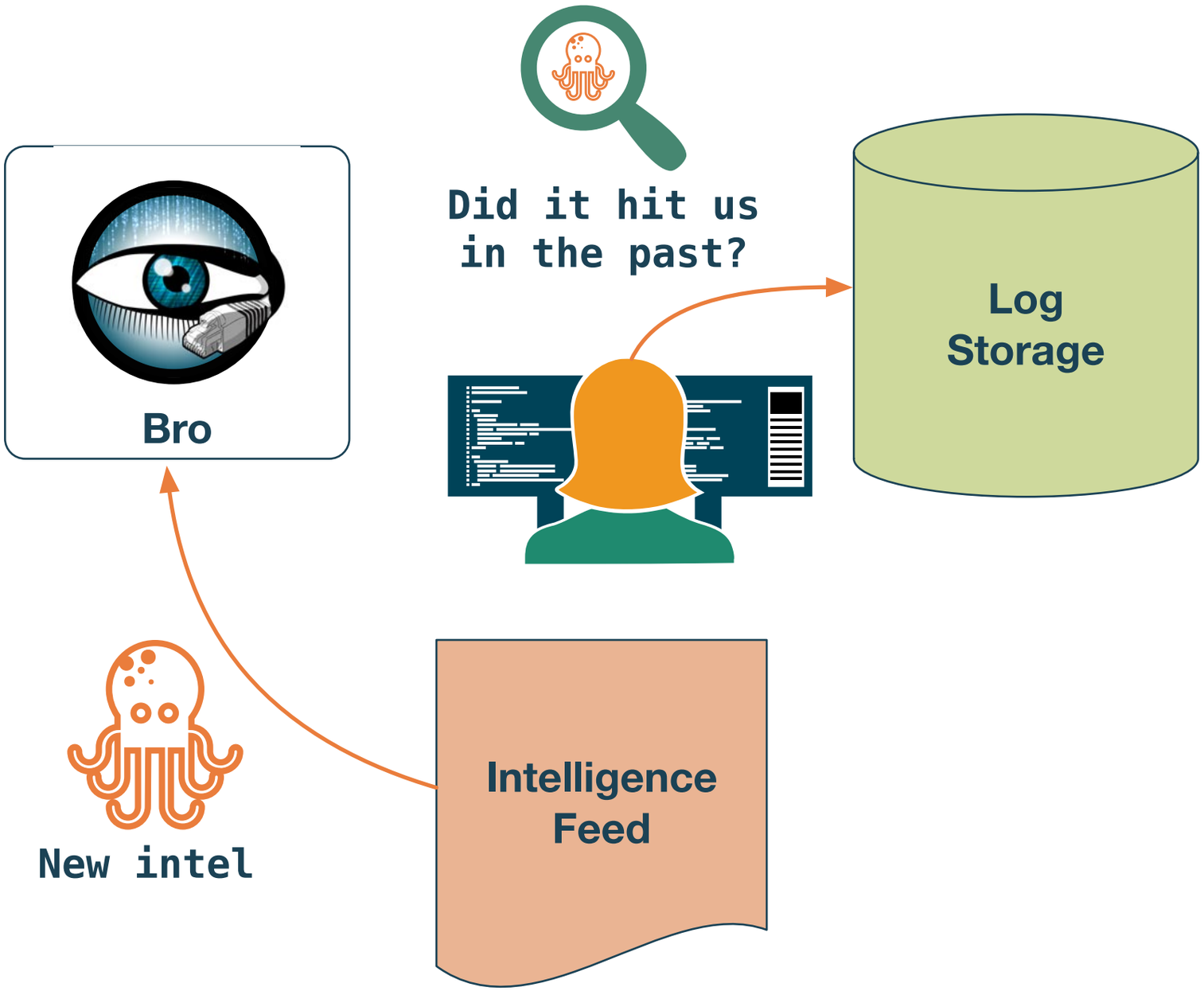
Scenario



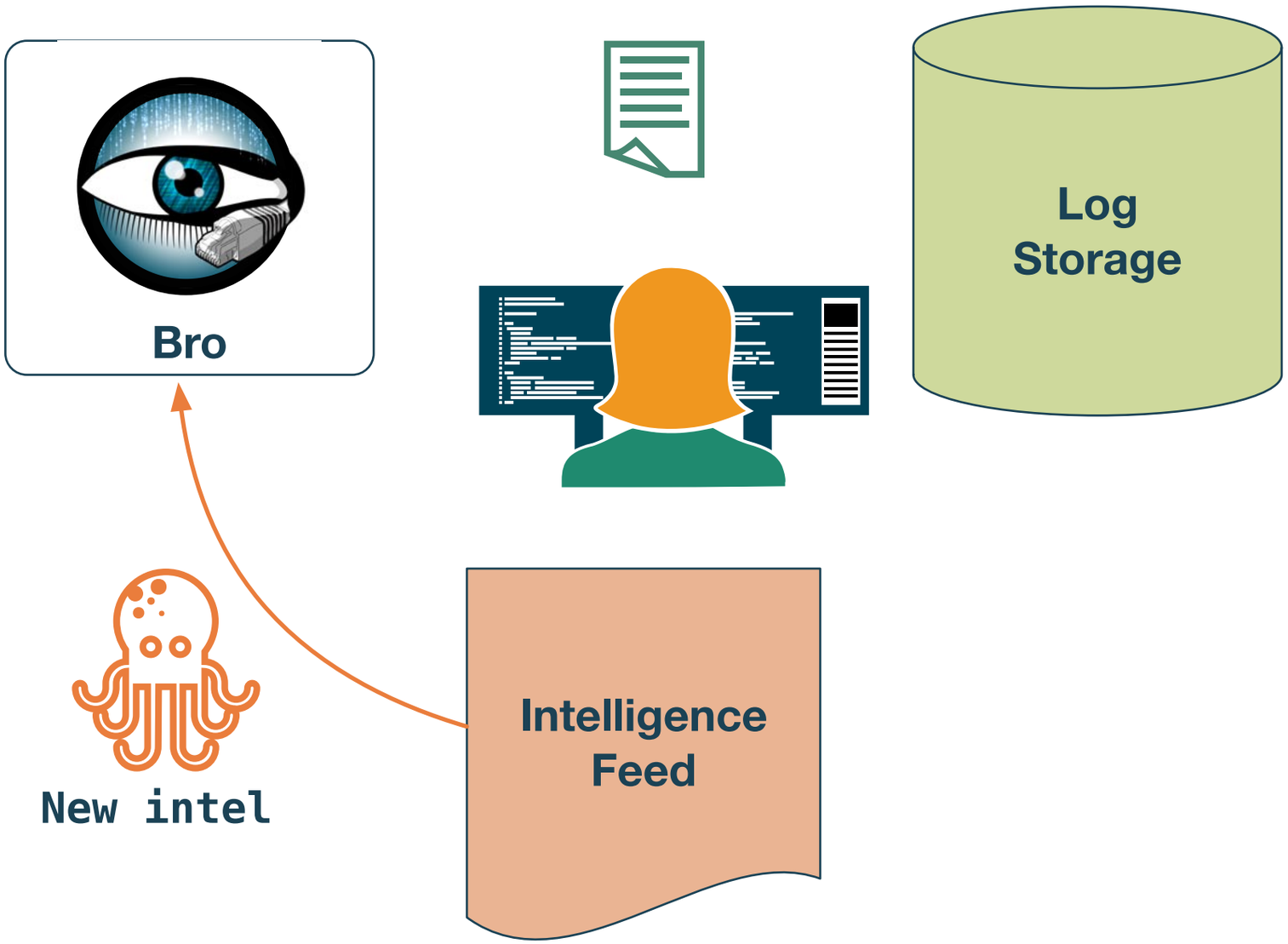
Scenario



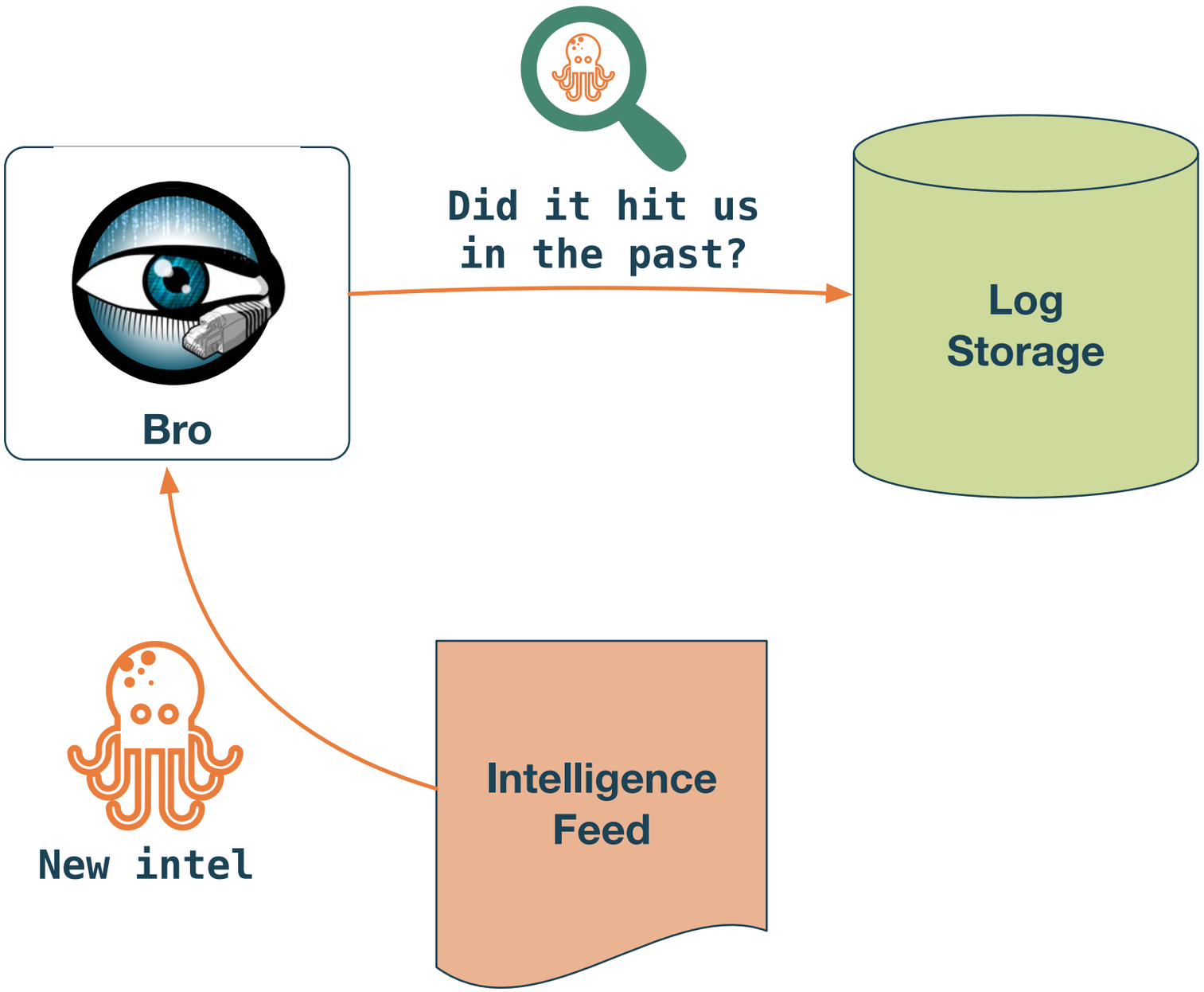
Scenario


Bro
Did it hit us
in the past?
Log
Storage
Intelligence
Feed
New intel

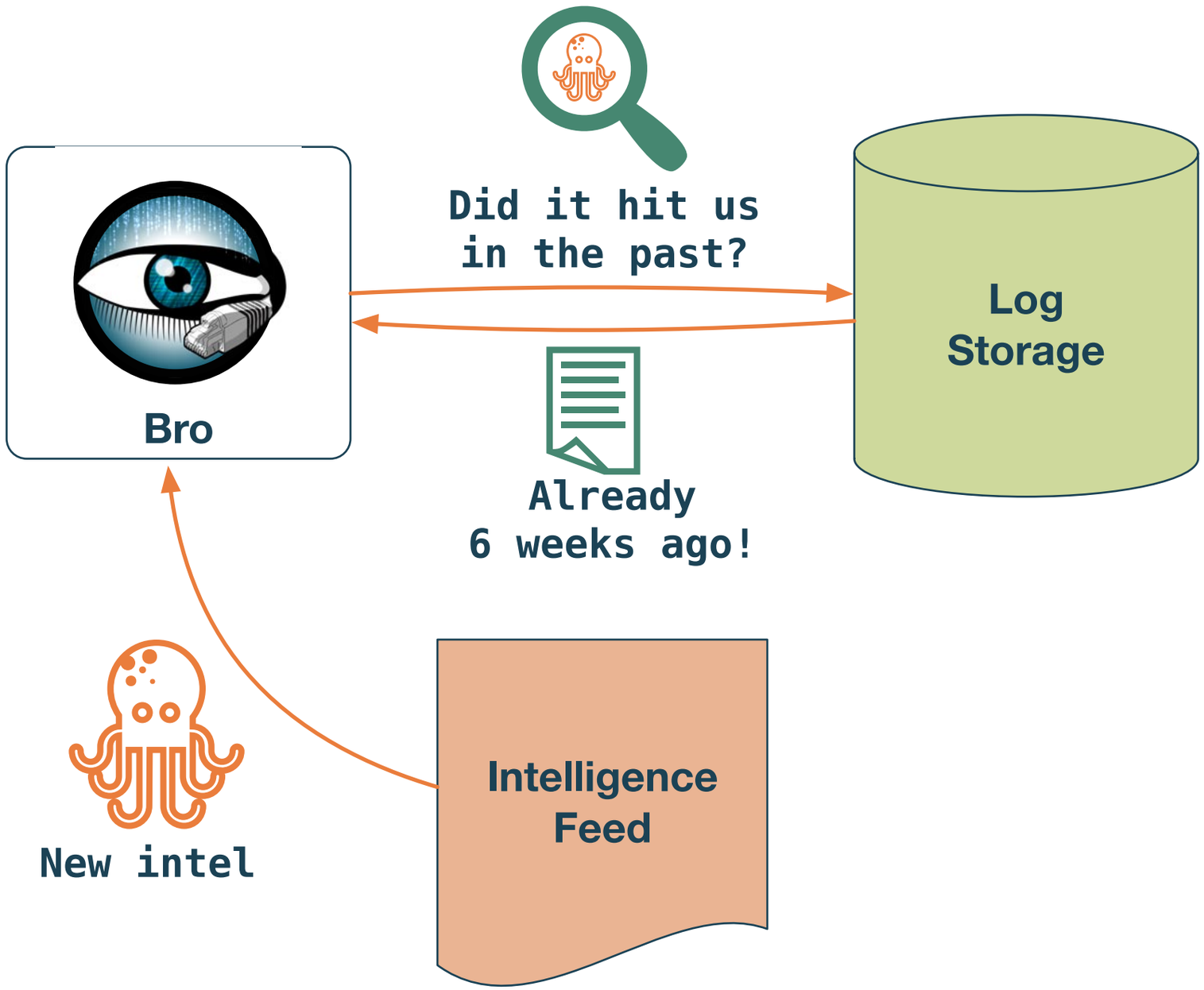
Scenario


Already
6 weeks ago!
Bro
Log
Storage
New intel
Intelligence
Feed

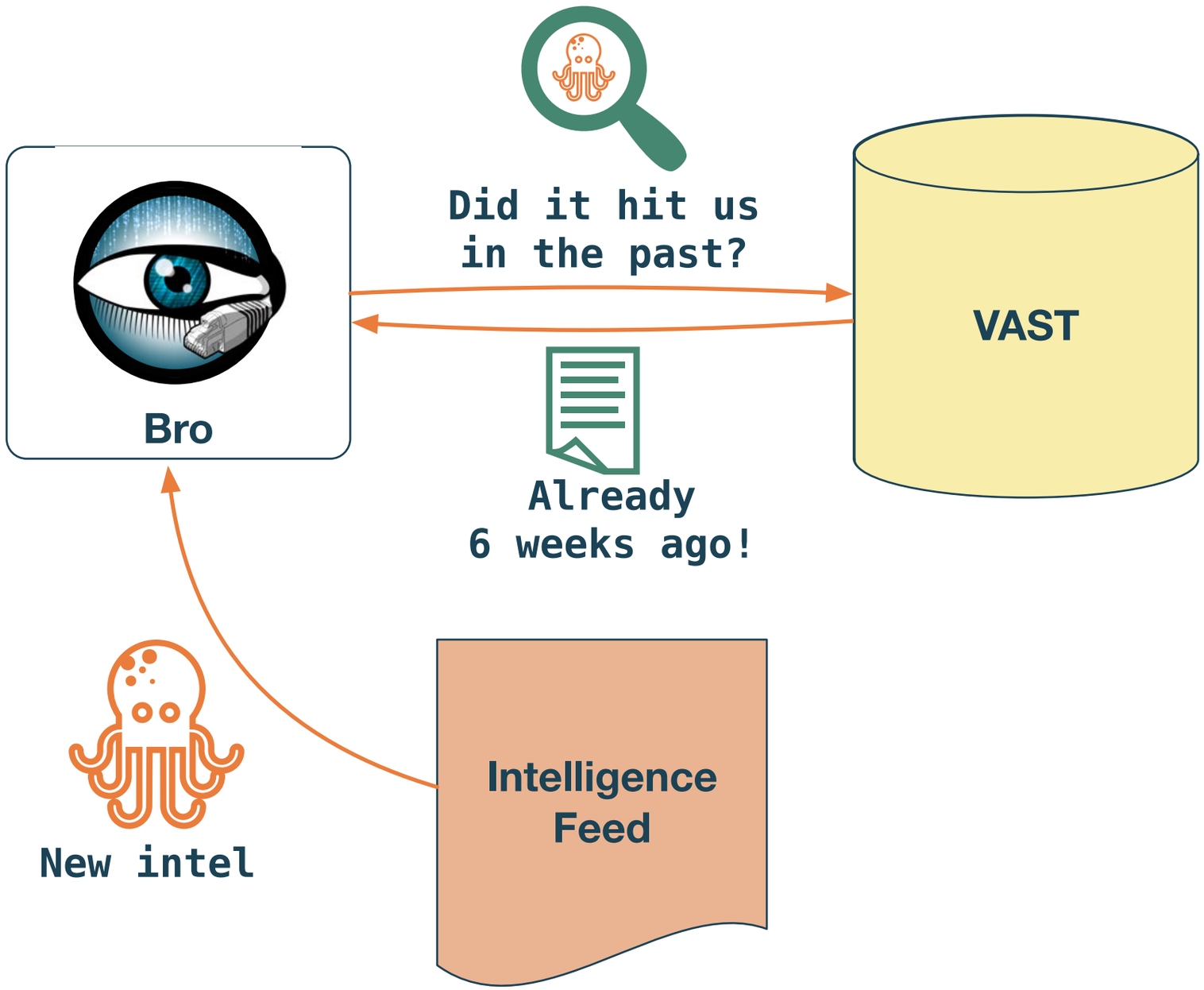
Scenario



Scenario


Did it hit us
in the past?
Log
Storage
Bro
Already
6 weeks ago!
New intel
Intelligence
Feed

Scenario


Did it hit us
in the past?
VAST
Bro
Already
6 weeks ago!
New intel
Intelligence
Feed

VAST

Visibility Across Space and Time

VAST

*under development

VAST

- Scalable open-source **data plane** for network forensics

*under development

VAST

- Scalable open-source **data plane** for network forensics
- Features
  - **Interactive search** in typed language
  - **Native support for Zeek** & PCAP import and export
  - Integration with **R, Python/Pandas, Spark***

*under development

VAST

- Scalable open-source **data plane** for network forensics
- Features
  - **Interactive search** in typed language
  - **Native support for Zeek** & PCAP import and export
  - Integration with **R, Python/Pandas, Spark***
- We are zeeking for alpha testers. Come talk to us!

*under development

VAST

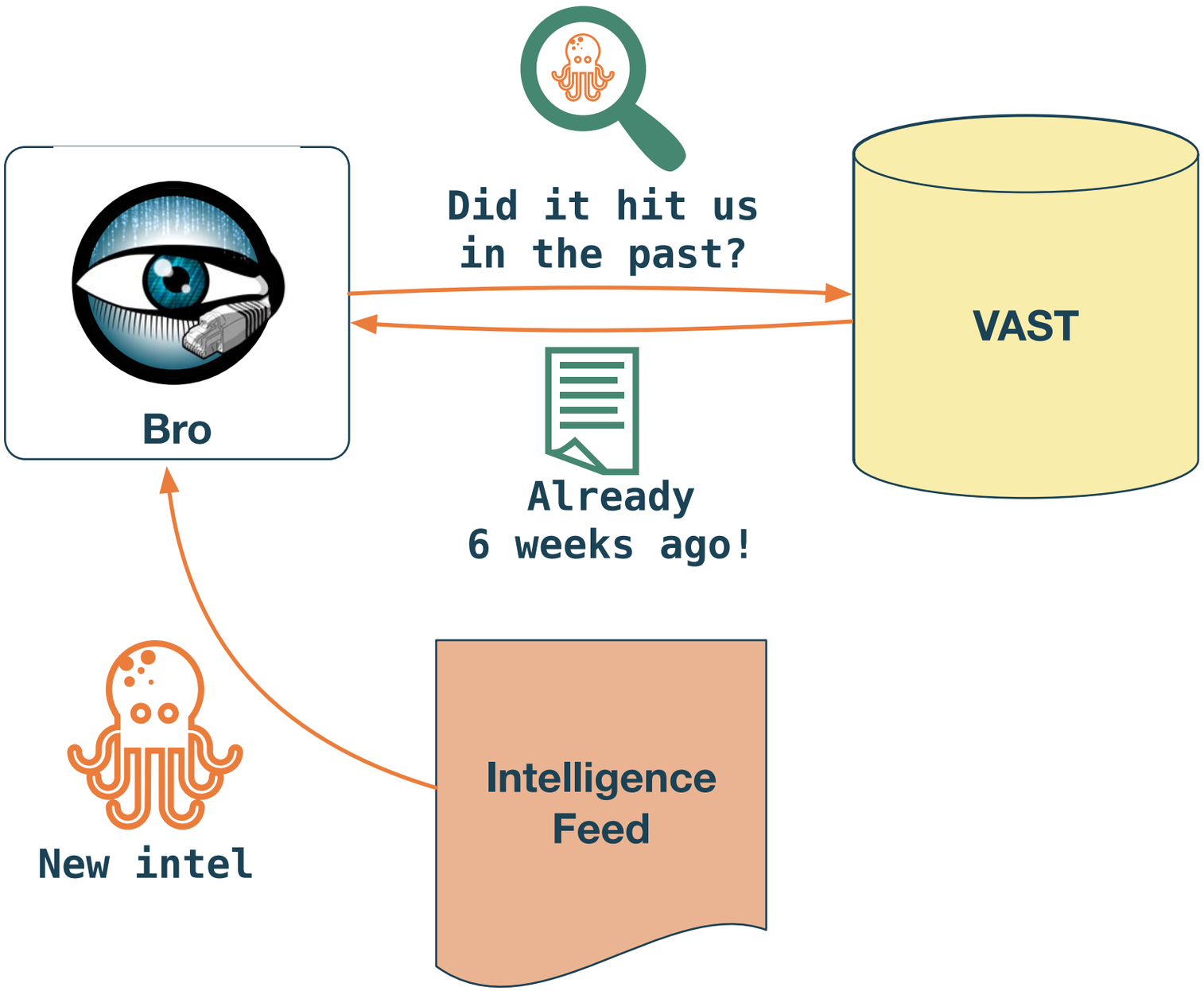
developed by

- Scalable open-source **data plane** for network forensics
- Features
  - **Interactive search** in typed language
  - **Native support for Zeek** & PCAP import and export
  - Integration with **R, Python/Pandas, Spark***
- We are zeeking for alpha testers. Come talk to us!

*under development

Scenario

Scenario


Did it hit us
in the past?
Bro
VAST
Already
6 weeks ago!
New intel
Intelligence
Feed

Scenario

Let’s build this!

(with Broker)

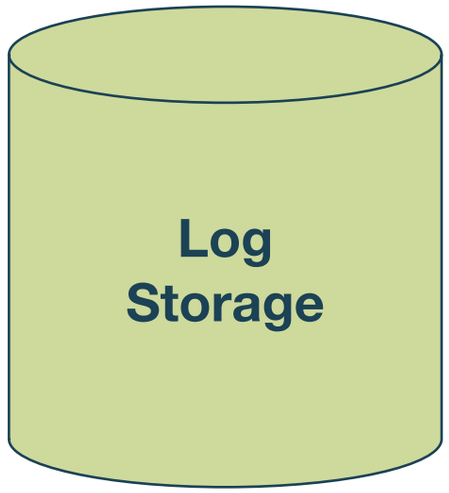
Let’s build this!

(with Broker)

tenzir/events

Version 1:
Stub →|| Stub

Version 1:
Stub →|← Stub

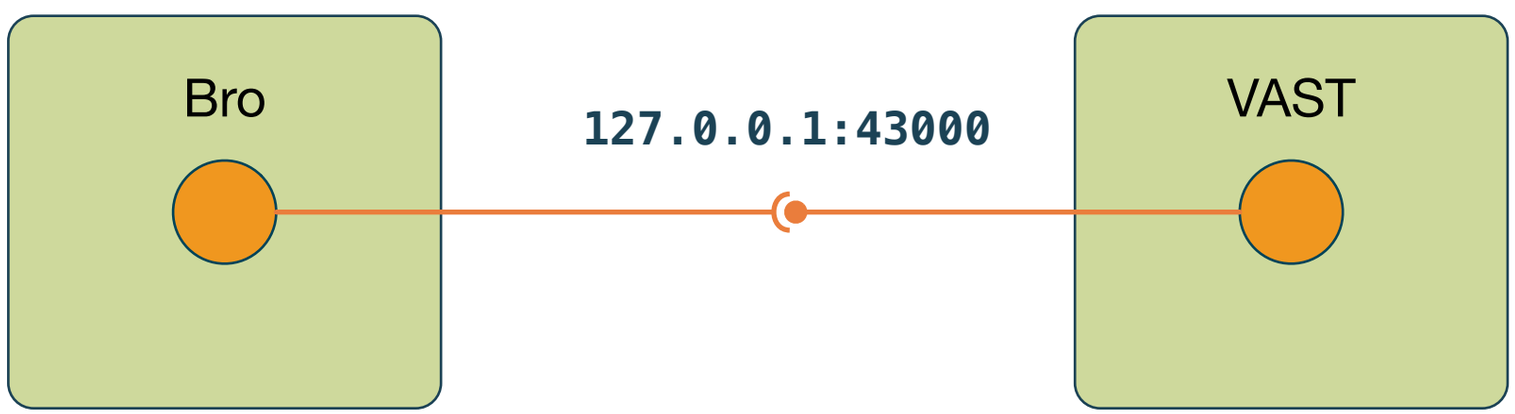


Version 1:
Stub →|| ← Stub

Version 1:
Stub →|← Stub

Version 1:
Stub →|← Stub

Version 1:
Stub →|← Stub



Version 1:
Stub →|← Stub

Version 1:
Stub →|← Stub

/vast/control

Version 1:
Stub →|← Stub

`/vast/data` `/vast/control`

Version 1:
Stub →|← Stub

/vast/control **/vast/data**

Version 1:
Stub →|← Stub

`/vast/control`

`/vast/data -> Bro`

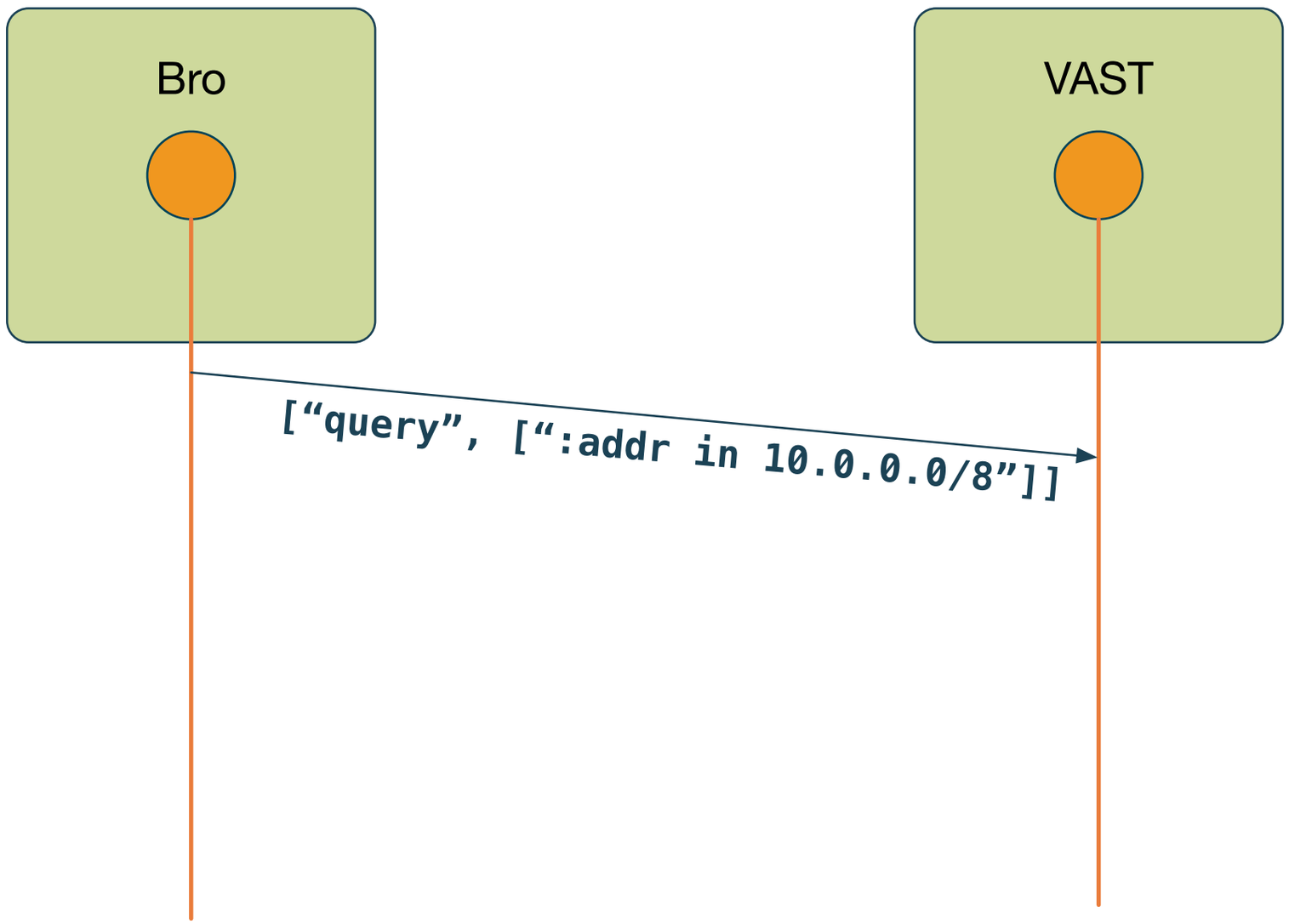
Version 1:
Stub →|← Stub

`/vast/control -> VAST`

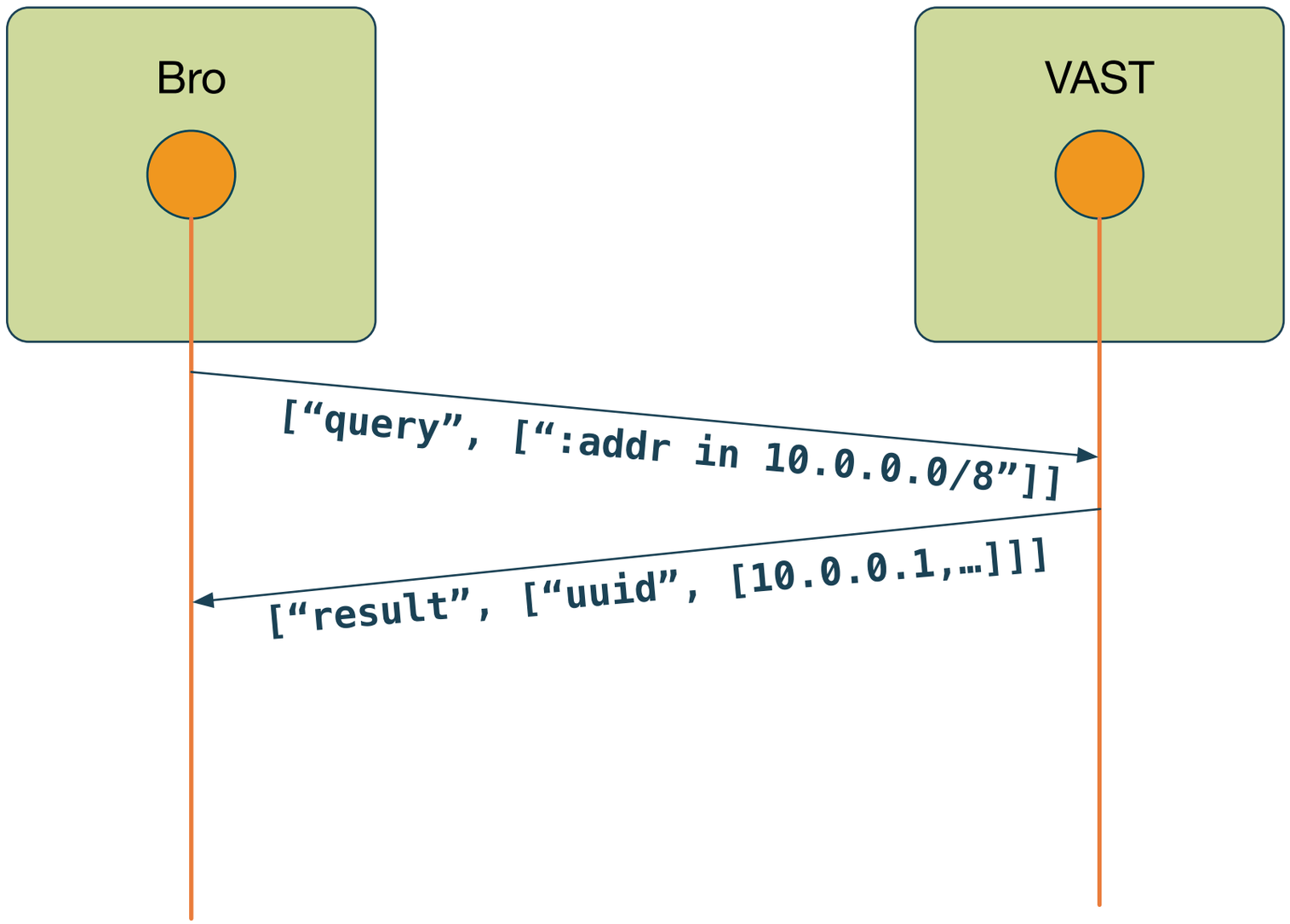
`/vast/data -> Bro`

Version 1:
Stub →|← Stub

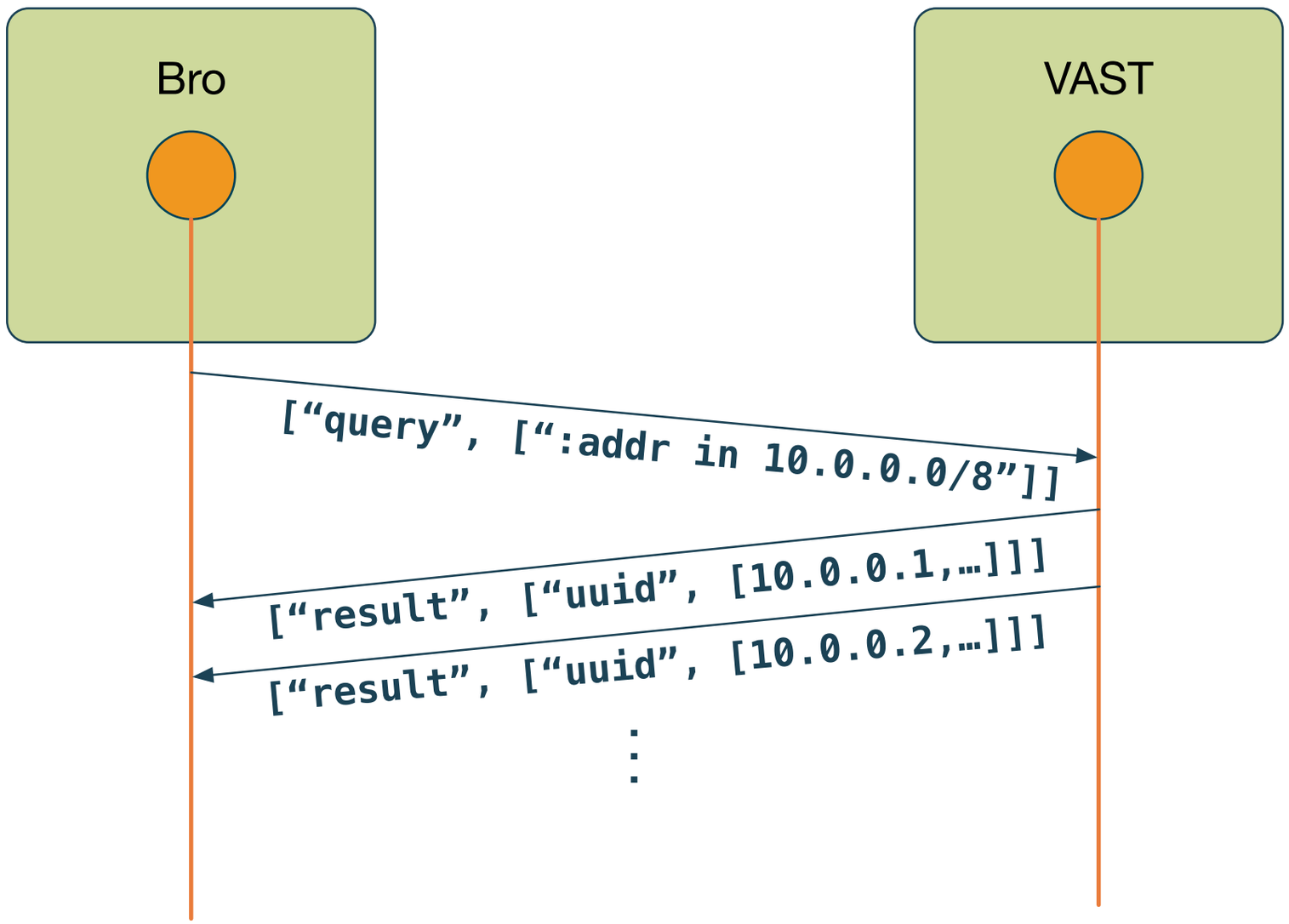
Version 1:
Stub →|← Stub



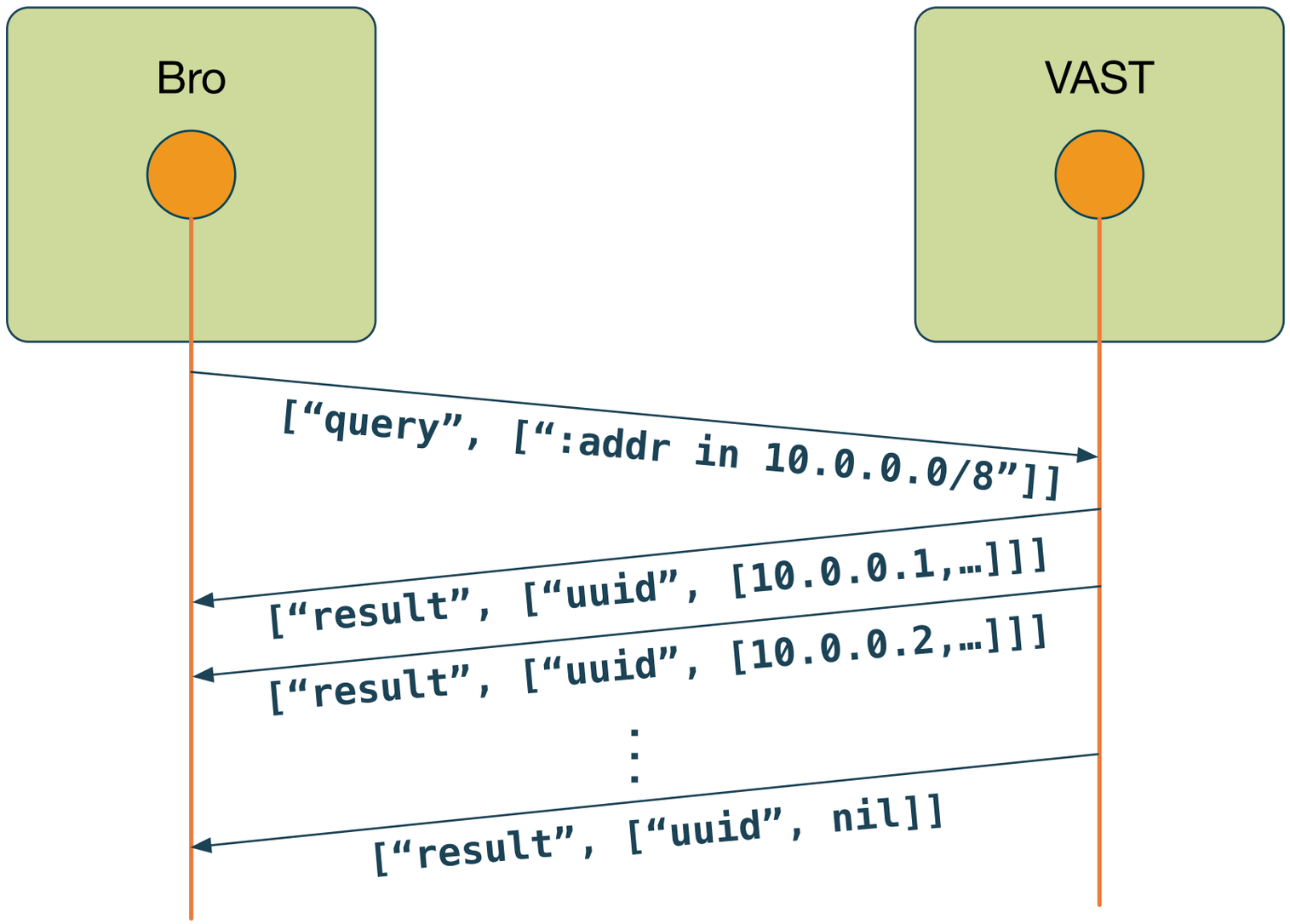
Version 1:
Stub →|← Stub



Version 1:
Stub →|← Stub



Version 1:
Stub →|← Stub



VAST Python Stub

VAST Python Stub

```
# Create endpoint & subscribers.
endpoint = broker.Endpoint()
subscriber = endpoint.make_subscriber(["/vast/control"])
endpoint.listen("127.0.0.1", 43000)
```

VAST Python Stub

```
# Create endpoint & subscribers.
endpoint = broker.Endpoint()
subscriber = endpoint.make_subscriber(["/vast/control"])
endpoint.listen("127.0.0.1", 43000)

# Loop until peering established successfully.
while True:
    print("waiting for commands")
    (topic, data) = subscriber.get()
    event = broker.bro.Event(data)
    (qid, expression) = event.args()
```

VAST Python Stub

```
# Create endpoint & subscribers.
endpoint = broker.Endpoint()
subscriber = endpoint.make_subscriber(["/vast/control"])
endpoint.listen("127.0.0.1", 43000)

# Loop until peering established successfully.
while True:
    print("waiting for commands")
    (topic, data) = subscriber.get()
    event = broker.bro.Event(data)
    (qid, expression) = event.args()

    # Answer the query with dummy data
    make_result_event = lambda *xs: broker.bro.Event("VAST::result", qid, *xs)
    generate_data = lambda x: [x, time.ctime(), [IPv4Address("10.0.0.1")]]
    for x in map(generate_data, range(10)):
        endpoint.publish("/vast/data", make_result_event(x))
    endpoint.publish("/vast/data", make_result_event(None))
```

Zeek Python Stub

Zeek Python Stub

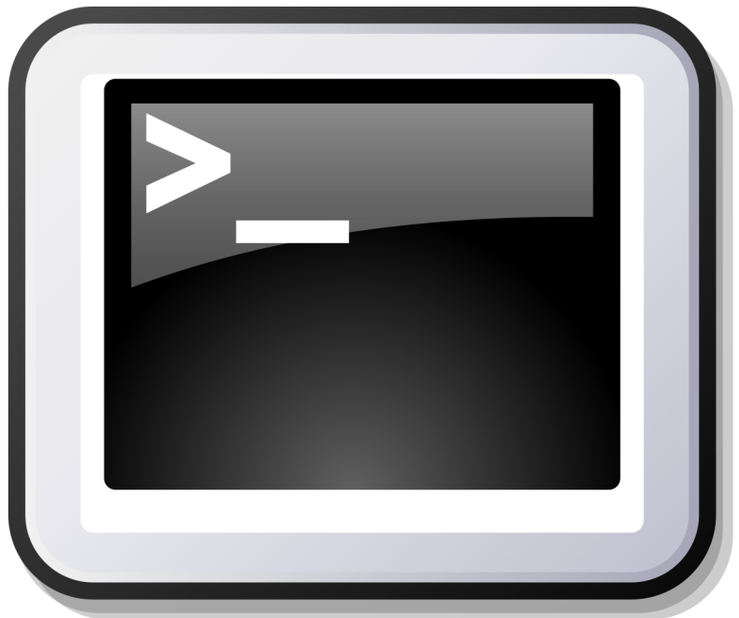
```
# Create endpoint & subscribers.
endpoint = broker.Endpoint()
subscriber = endpoint.make_subscriber(["/vast/data"])
endpoint.peer("127.0.0.1", 43000)
```

Zeek Python Stub

```
# Create endpoint & subscribers.
endpoint = broker.Endpoint()
subscriber = endpoint.make_subscriber(["/vast/data"])
endpoint.peer("127.0.0.1", 43000)

# Create a query event and send it to VAST.
query_id = str(uuid.uuid4()) # random value
event = broker.bro.Event("VAST::query", query_id, ":addr in 10.0.0.0/8")
endpoint.publish("/vast/control", event)
```

Zeek Python Stub

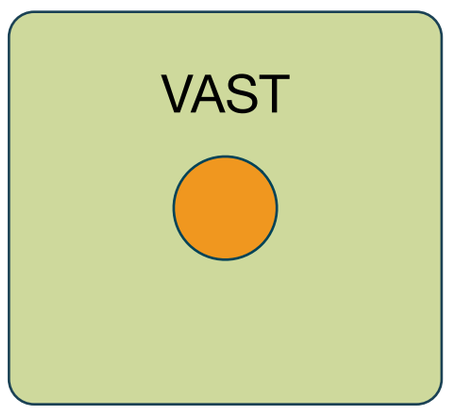
```
# Create endpoint & subscribers.
endpoint = broker.Endpoint()
subscriber = endpoint.make_subscriber(["/vast/data"])
endpoint.peer("127.0.0.1", 43000)

# Create a query event and send it to VAST.
query_id = str(uuid.uuid4()) # random value
event = broker.bro.Event("VAST::query", query_id, ":addr in 10.0.0.0/8")
endpoint.publish("/vast/control", event)

# Loop until we got all results.
while True:
    topic, data = subscriber.get()
    event = broker.bro.Event(data) # parse Broker data as Bro event
    print(topic, event.args())
    (qid, result) = event.args()
    if result is None:
        break; # we're done
```



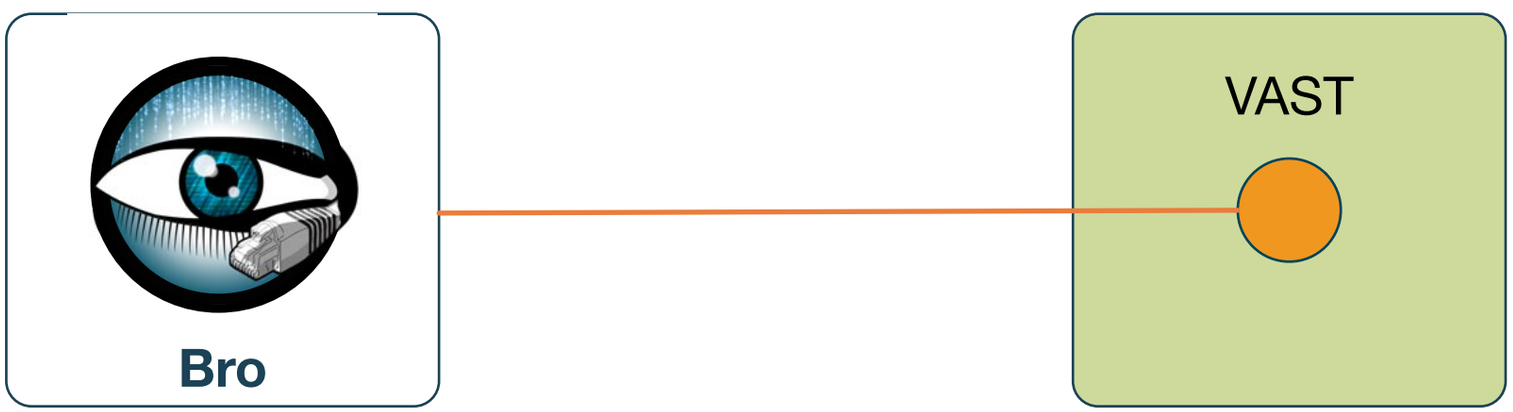
Demo

Version 2:
Zeek →|| Stub

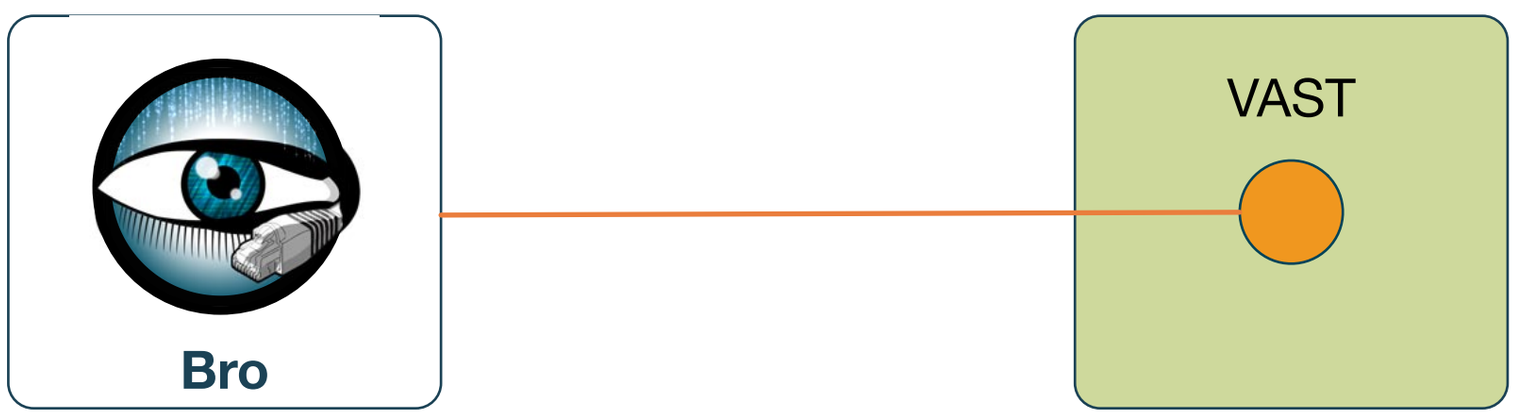
Version 2:
Zeek →|| ← Stub



Version 2:
Zeek →|| Stub



Version 2:
Zeek →|| Stub



(Protocol same as before)

Zeek Script

Zeek Script

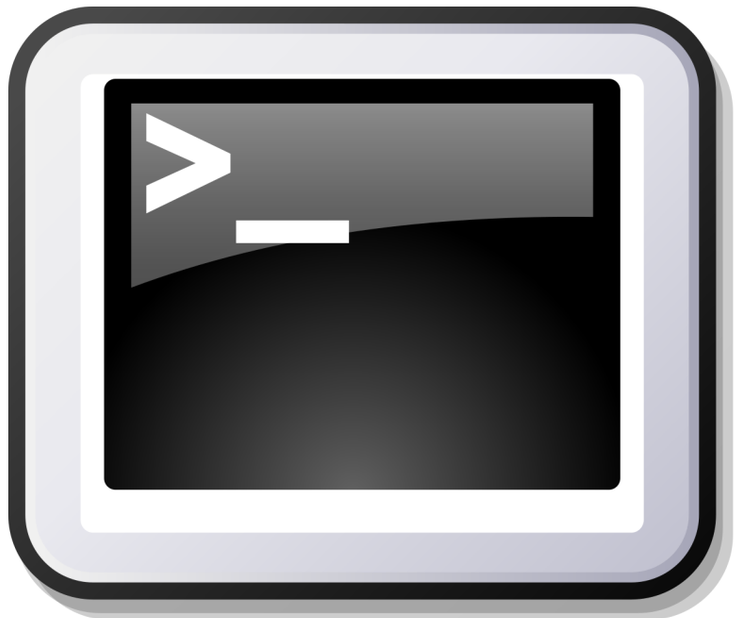
```
event bro_init() {
  Broker::subscribe("/vast/data");
  Broker::peer("127.0.0.1", 43000/tcp);
}
```

Zeek Script

```
global query: event(id: string, expression: string);

event Broker::peer_added(endpoint: Broker::EndpointInfo, msg: string) {
  local query_id = random_uuid();
  local e = Broker::make_event(query, query_id, expression);
  Broker::publish("/vast/control", e);
}
event bro_init() {
  Broker::subscribe("/vast/data");
  Broker::peer("127.0.0.1", 43000/tcp);
}
```

Zeek Script

```
event result(uuid: string, data: any) {
  switch (data) {
    default:
      terminate(); # harsh way to signal that we're done
      break;
    case type vector of any as xs:
      print xs;
      break;
  }
}
global query: event(id: string, expression: string);
event Broker::peer_added(endpoint: Broker::EndpointInfo, msg: string) {
  local query_id = random_uuid();
  local e = Broker::make_event(query, query_id, expression);
  Broker::publish("/vast/control", e);
}
event bro_init() {
  Broker::subscribe("/vast/data");
  Broker::peer("127.0.0.1", 43000/tcp);
}
```

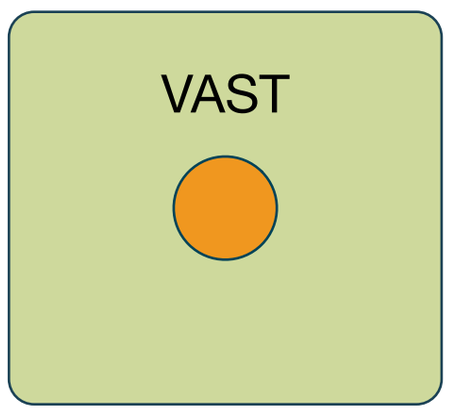


Demo

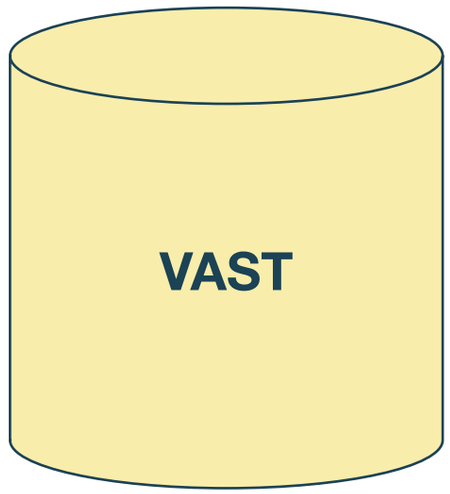
Version 3:

Zeek →|| VAST

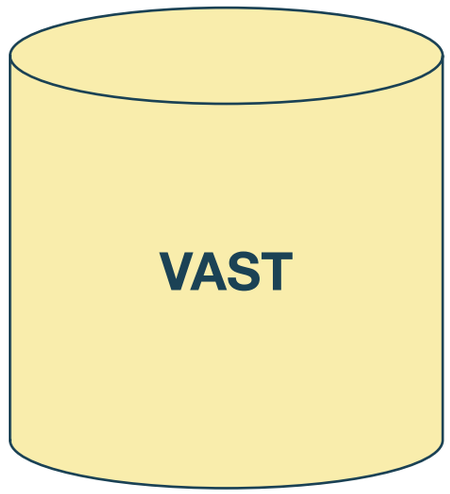
Version 3:
Zeek →|← VAST



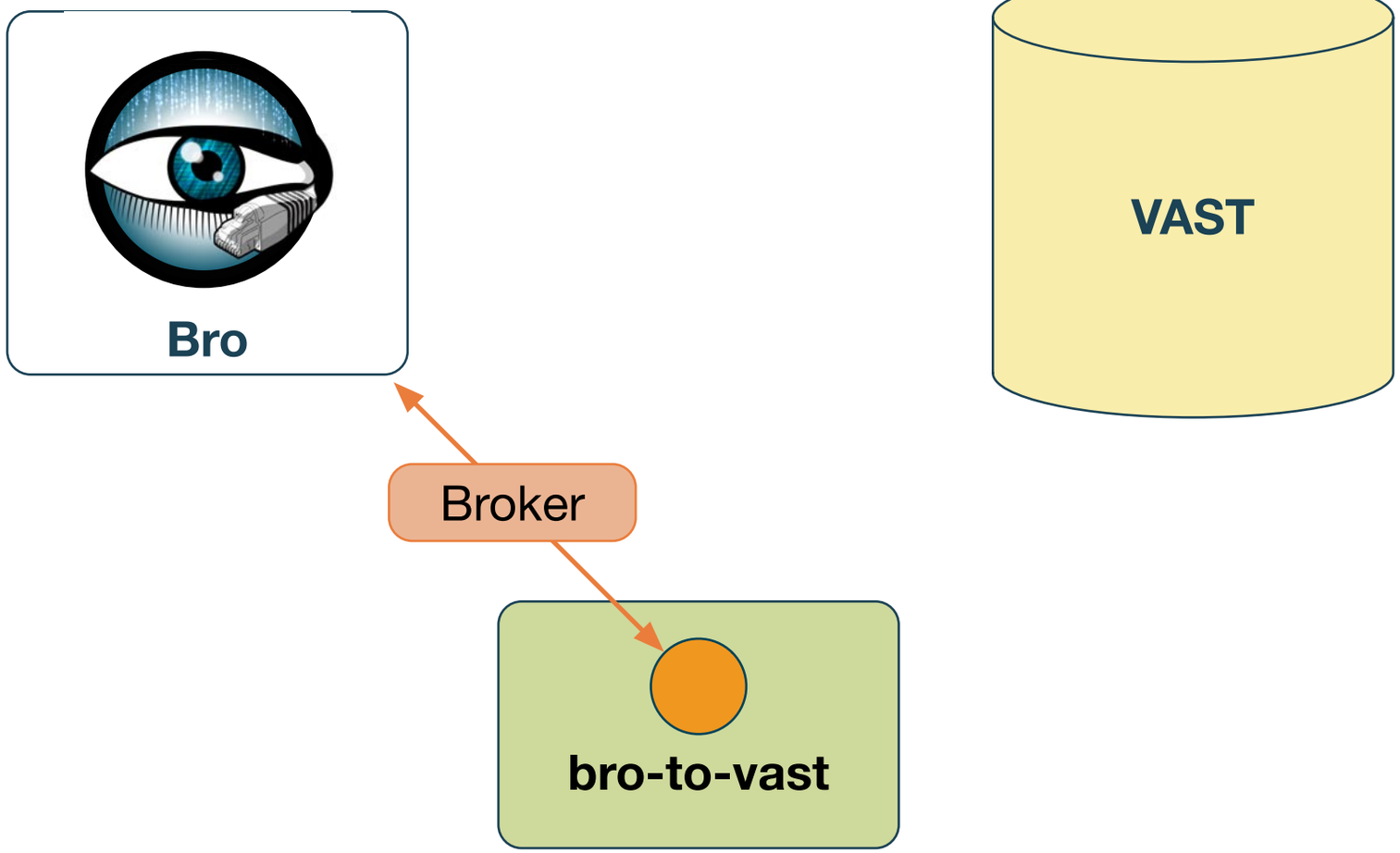
Version 3:
Zeek →|← VAST



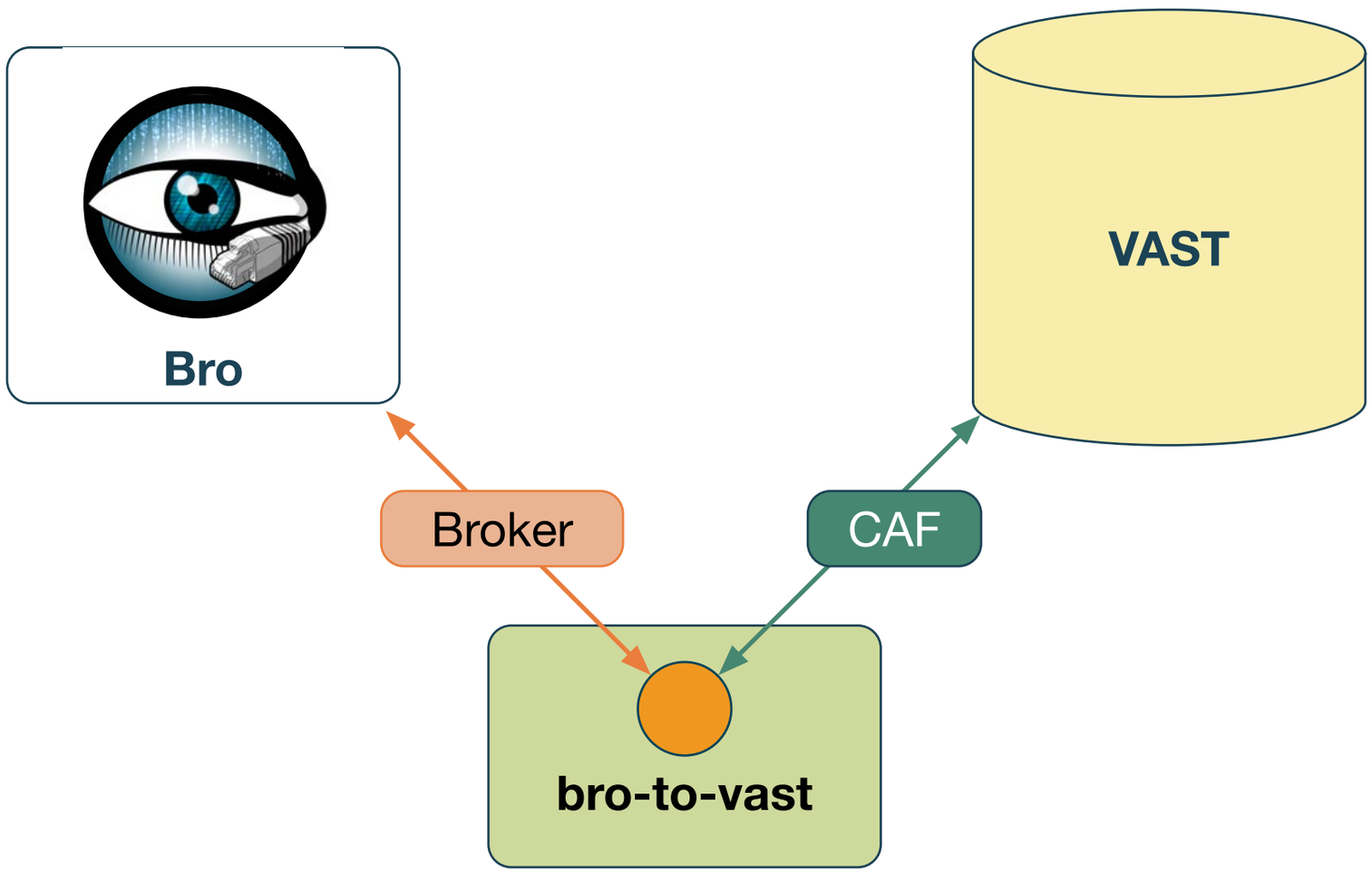
Version 3:
Zeek →|← VAST



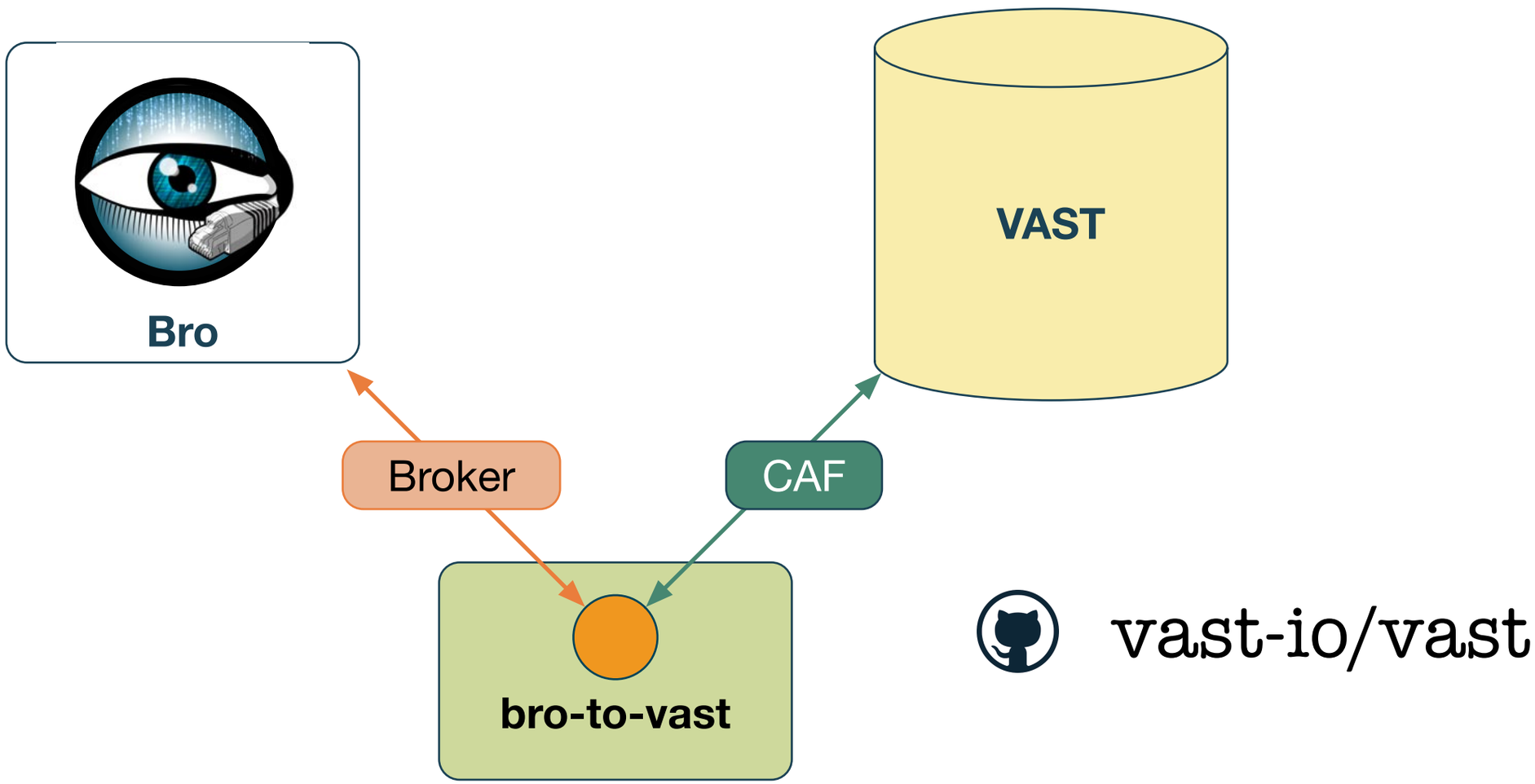
Version 3:
Zeek →|← VAST



Version 3:
Zeek →|← VAST

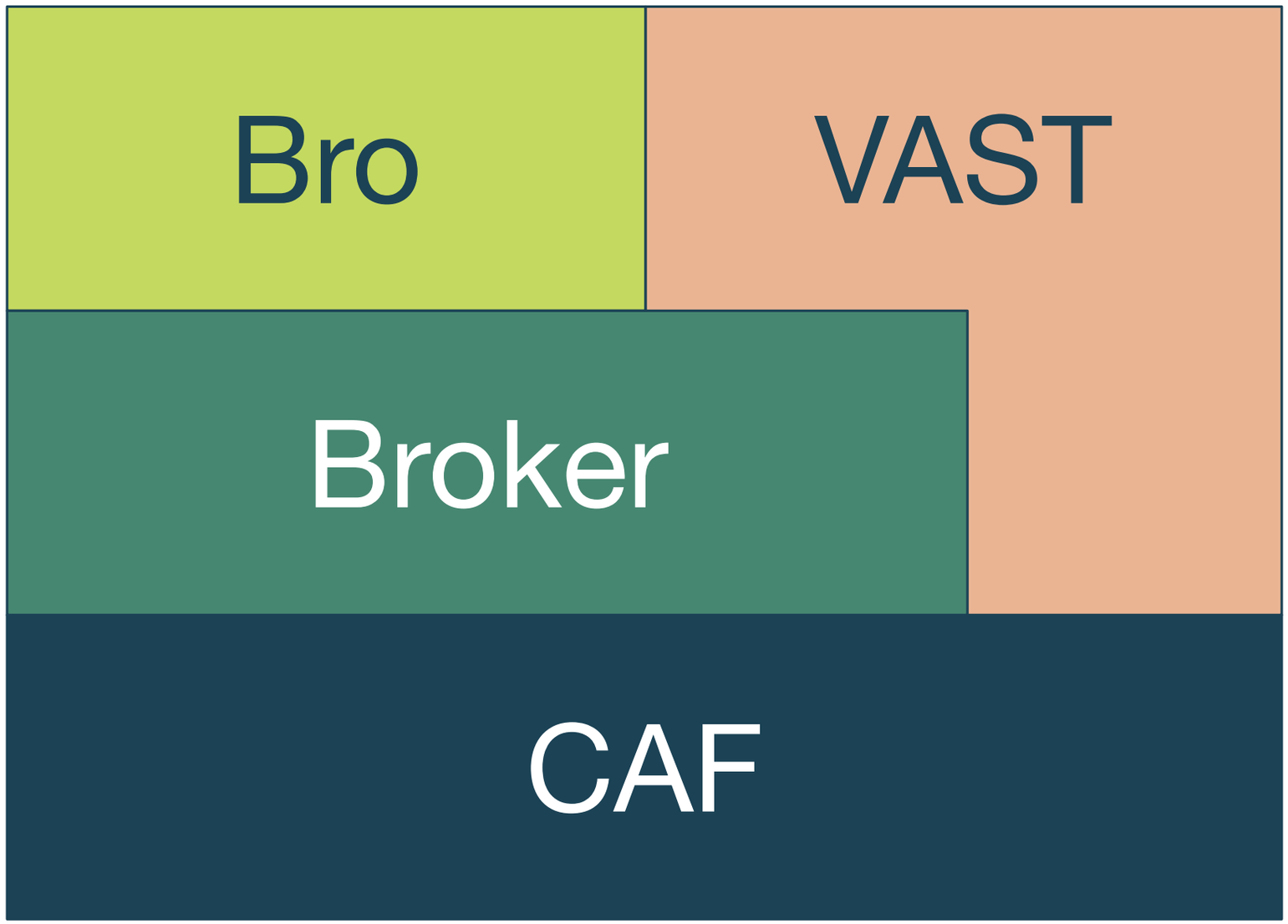


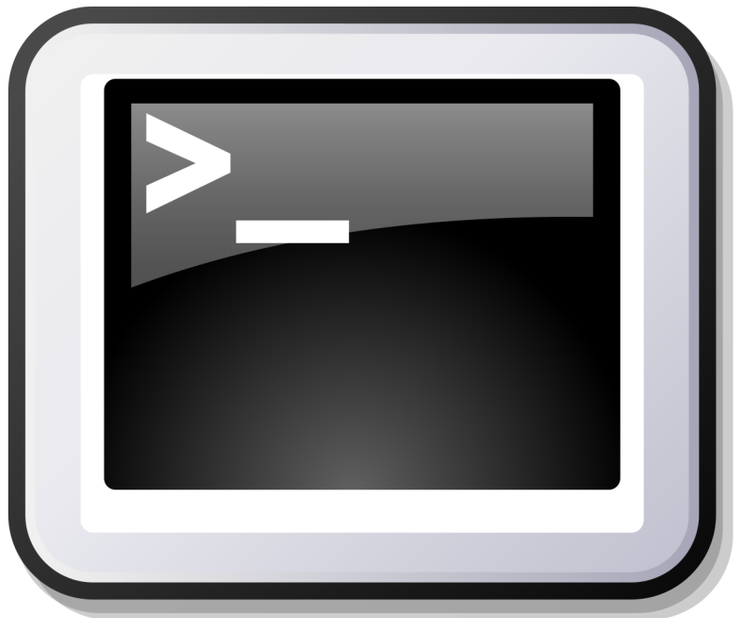
Version 3:
Zeek →|← VAST



vast-io/vast

Architecture



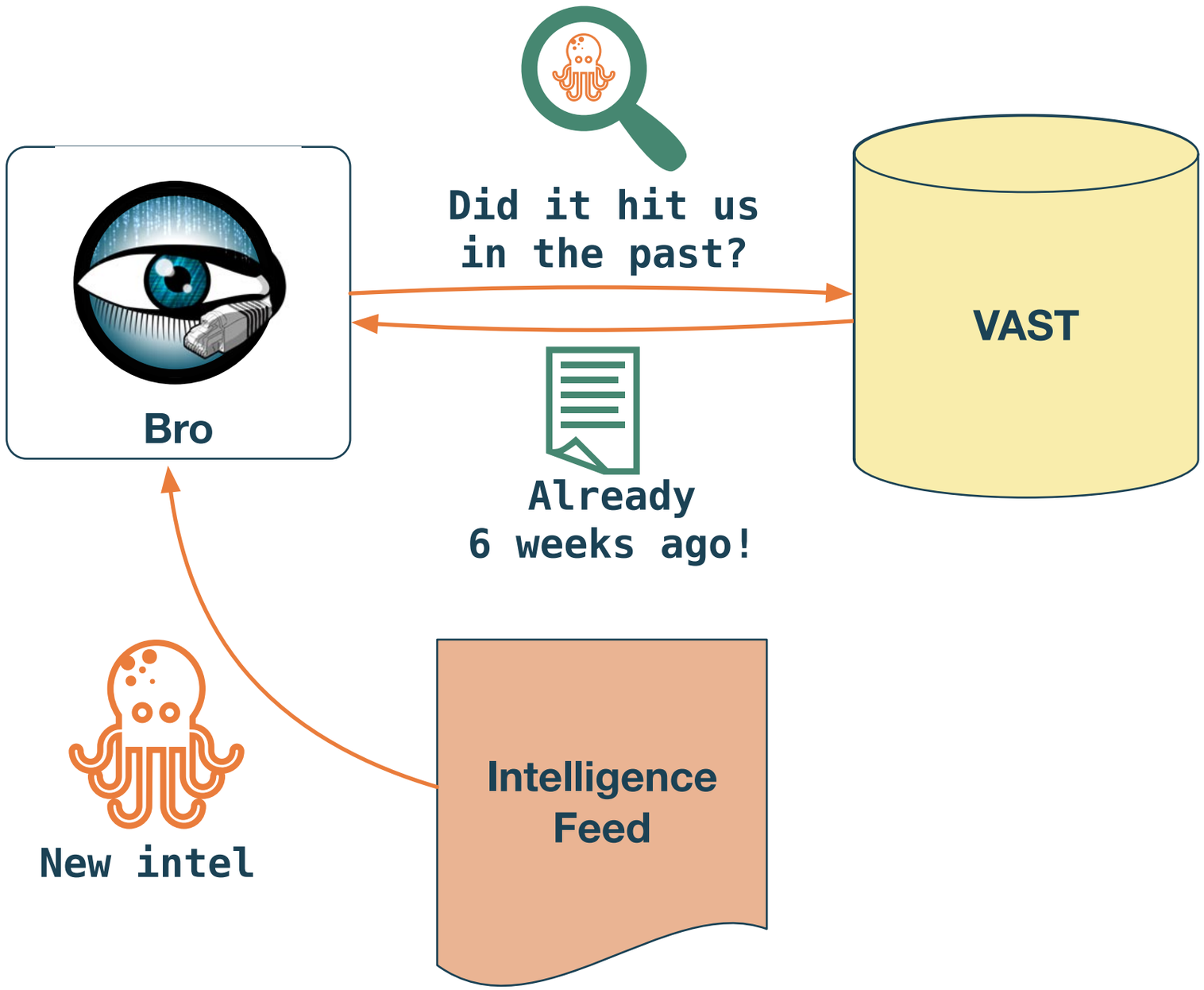


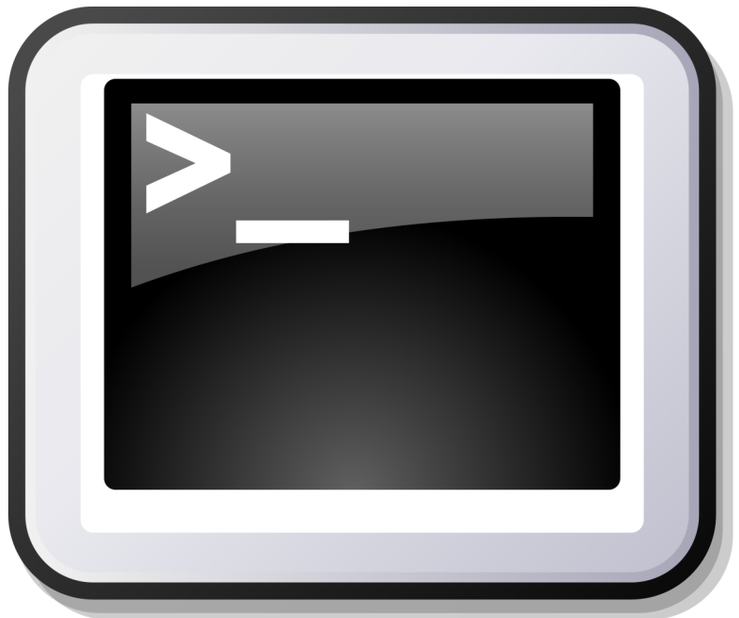
Demo

Scenario

Scenario

Scenario


Did it hit us
in the past?
Bro
VAST
Already
6 weeks ago!
New intel
Intelligence
Feed



Demo

Bro Package

tenzir/bro-vast

```
> bro-pkg install bro-vast
```

How does Broker perform?

A Closer Look

A Closer Look

- Our demo highlighted **two communication patterns**:

A Closer Look

- Our demo highlighted **two communication patterns**:
  - **Request-response**: lookup and answer

A Closer Look

- Our demo highlighted **two communication patterns**:
  - **Request-response**: lookup and answer
  - **Streaming results** to the query issuer

A Closer Look

- Our demo highlighted **two communication patterns**:
  - **Request-response**: lookup and answer
  - **Streaming results** to the query issuer
- How does Broker perform in **microbenchmarks**?

A Closer Look

- Our demo highlighted **two communication patterns**:
  - **Request-response**: lookup and answer
  - **Streaming results** to the query issuer
- How does Broker perform in **microbenchmarks**?
  - **Latency** between lookup and answer

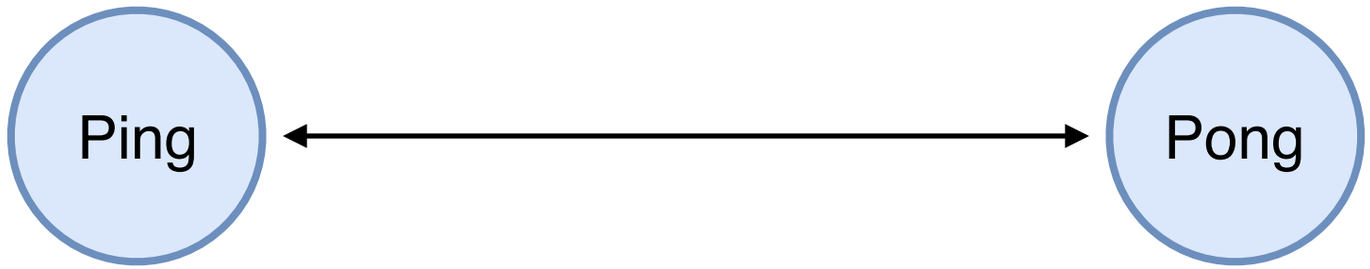
A Closer Look

- Our demo highlighted **two communication patterns**:
  - **Request-response**: lookup and answer
  - **Streaming results** to the query issuer
- How does Broker perform in **microbenchmarks**?
  - **Latency** between lookup and answer
  - **Throughput** of result stream

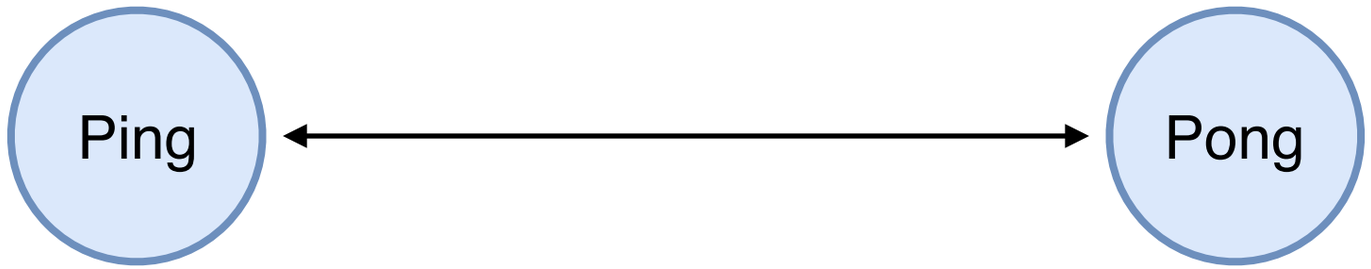
Testing Setup

- AMD Opteron 6376, 2.3 GHz, 500GB RAM
- 2 CPUs, 16 cores each
- 64 logical processors (hyper-threading)
- OpenSUSE Leap 42.3
- Localhost communication only (microbenchmarking)

Roundtrip Latency Setup

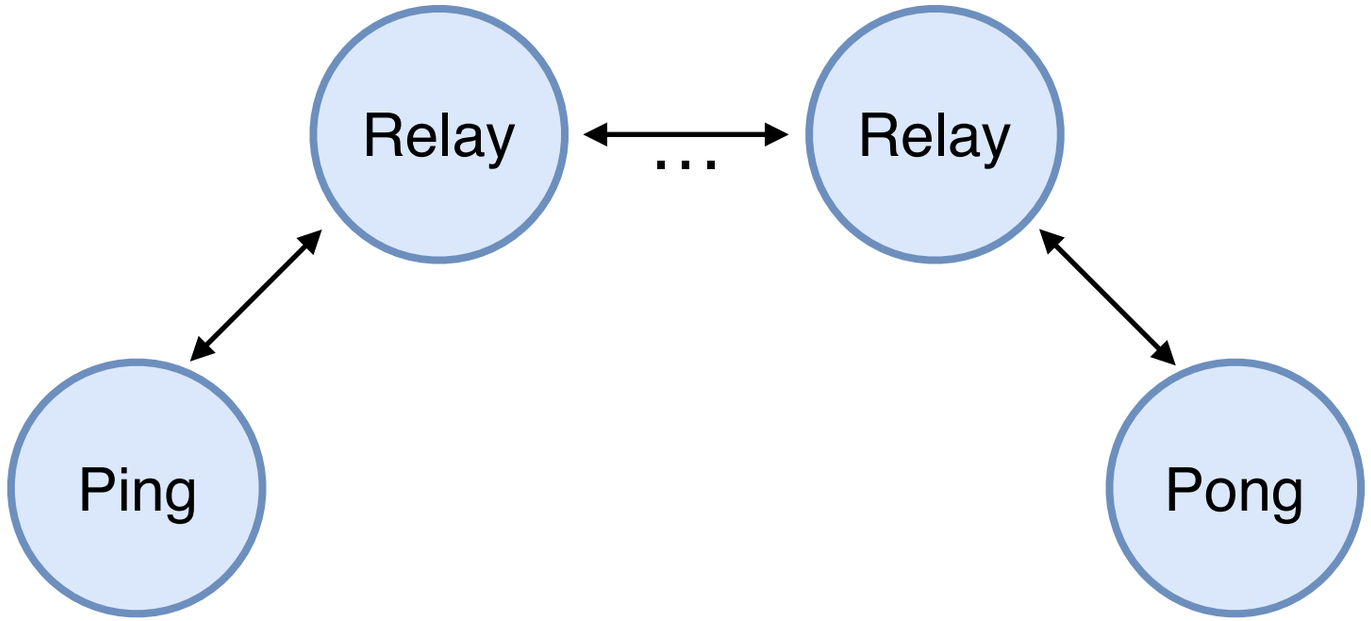


Roundtrip Latency Setup



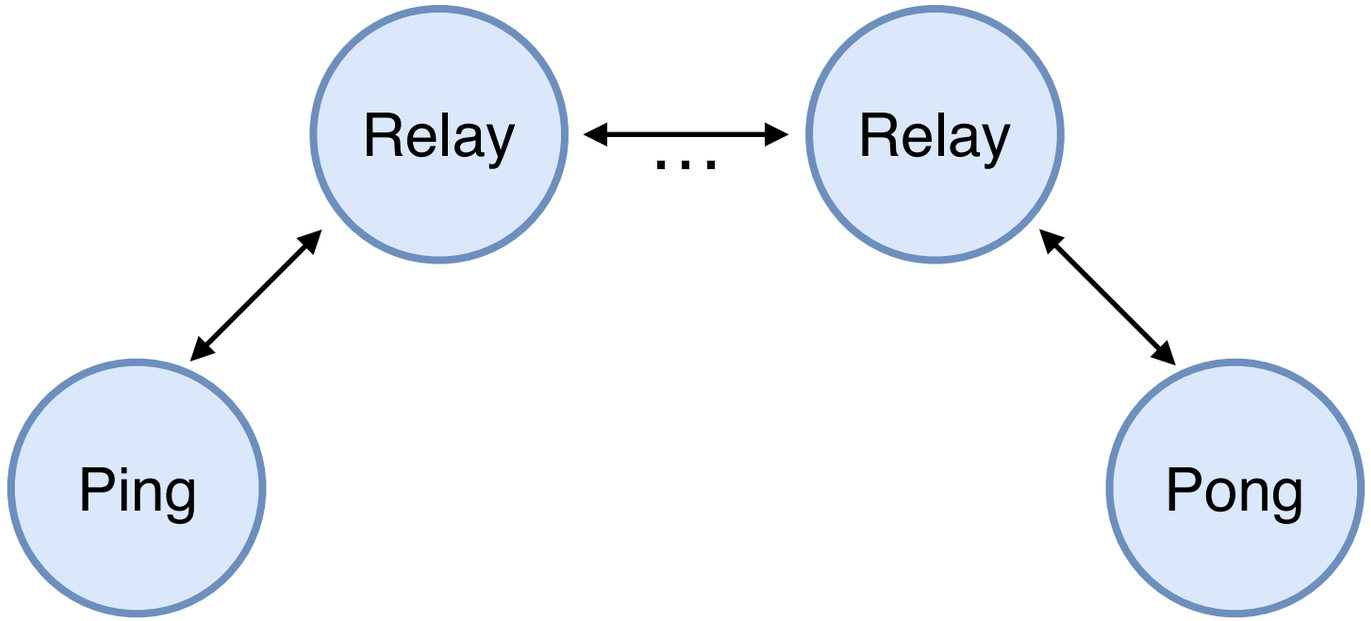
- **Measured**: time between send and receive on *Ping*

Roundtrip Latency Setup



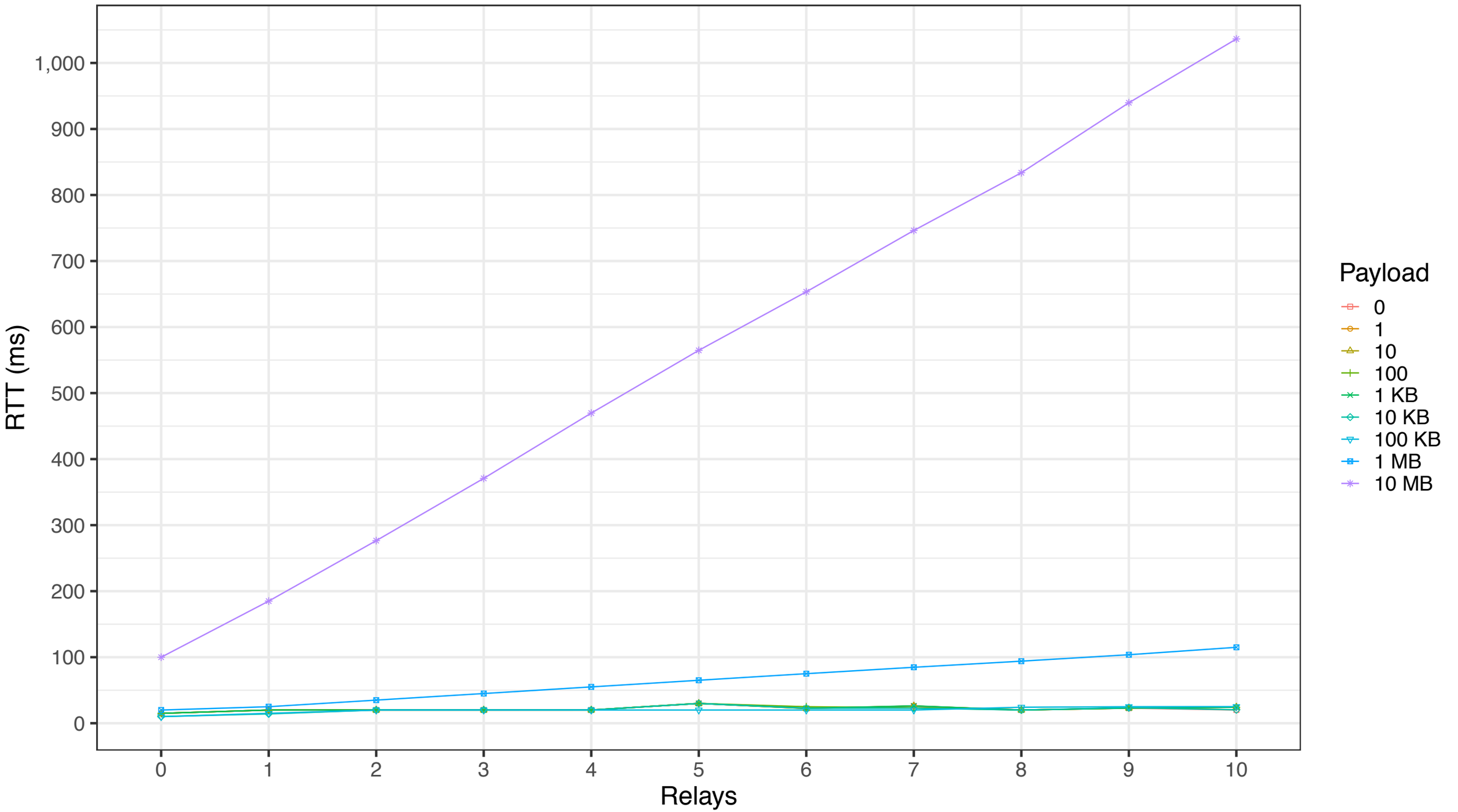
- **Measured**: time between send and receive on *Ping*
- Varying **payload sizes** and varying **number of relays**

Roundtrip Latency Setup

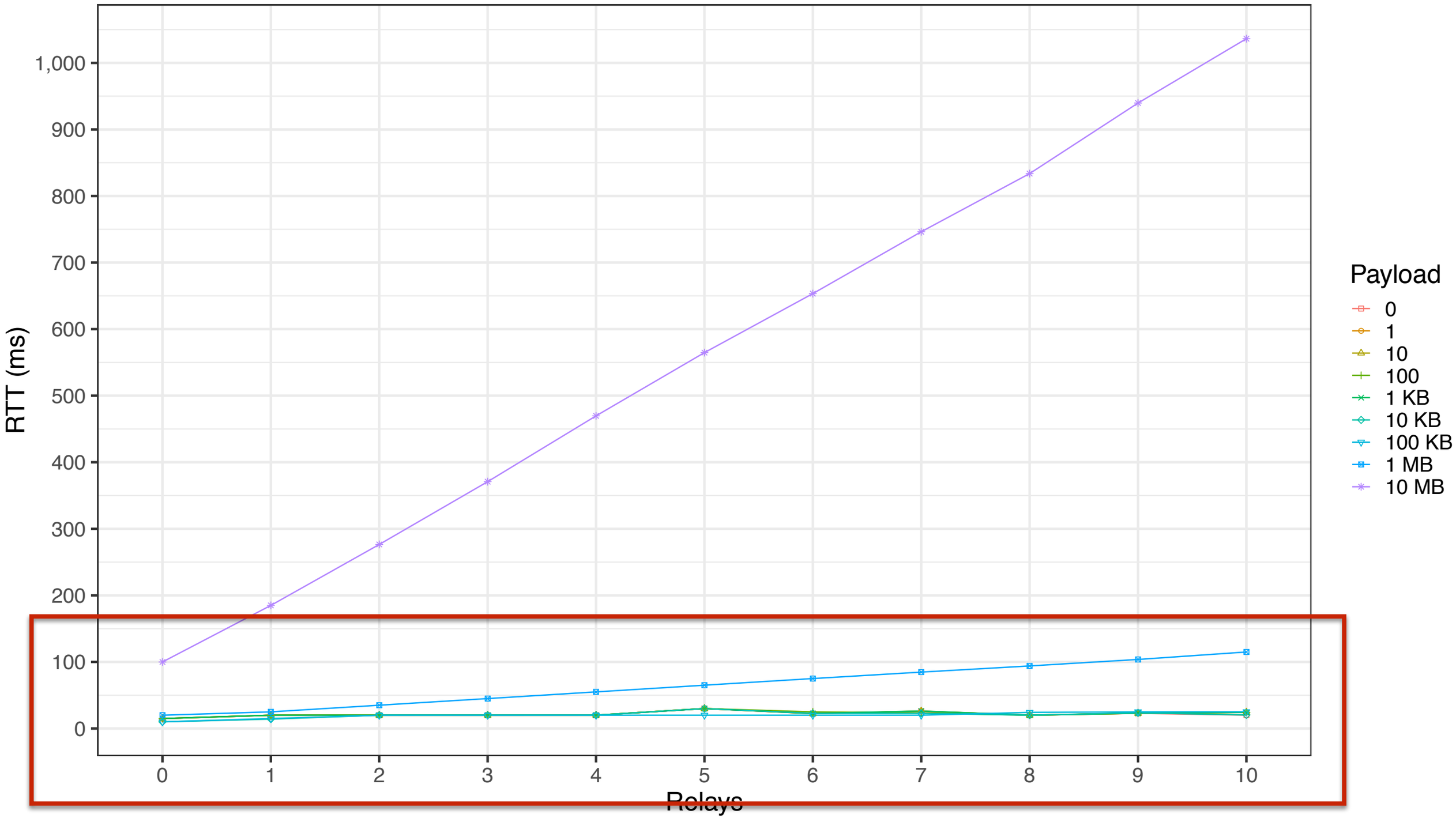


- **Measured**: time between send and receive on *Ping*
- Varying **payload sizes** and varying **number of relays**
- **Expectation**: linear latency increase with number of relays

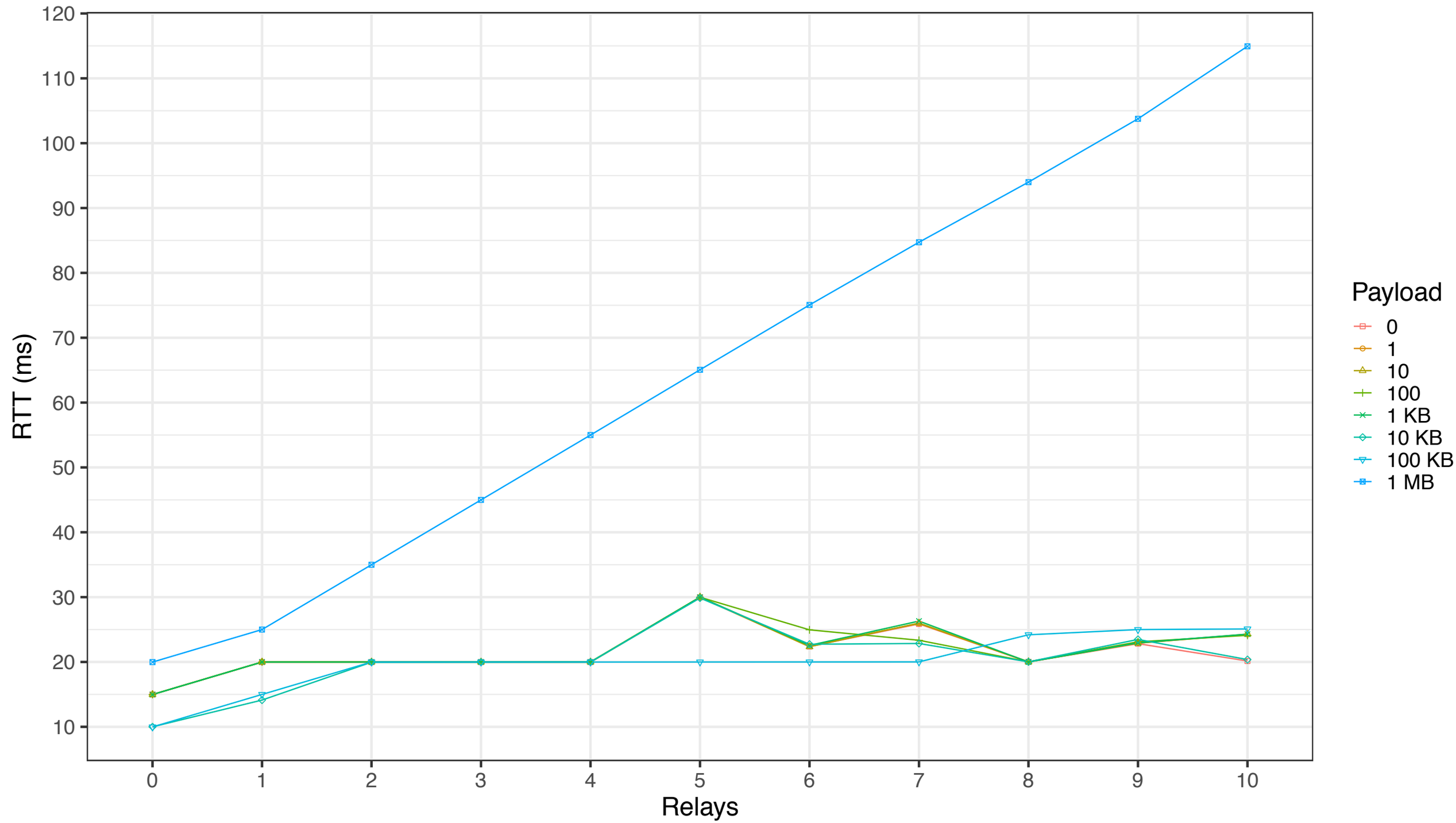
Roundtrip Latency



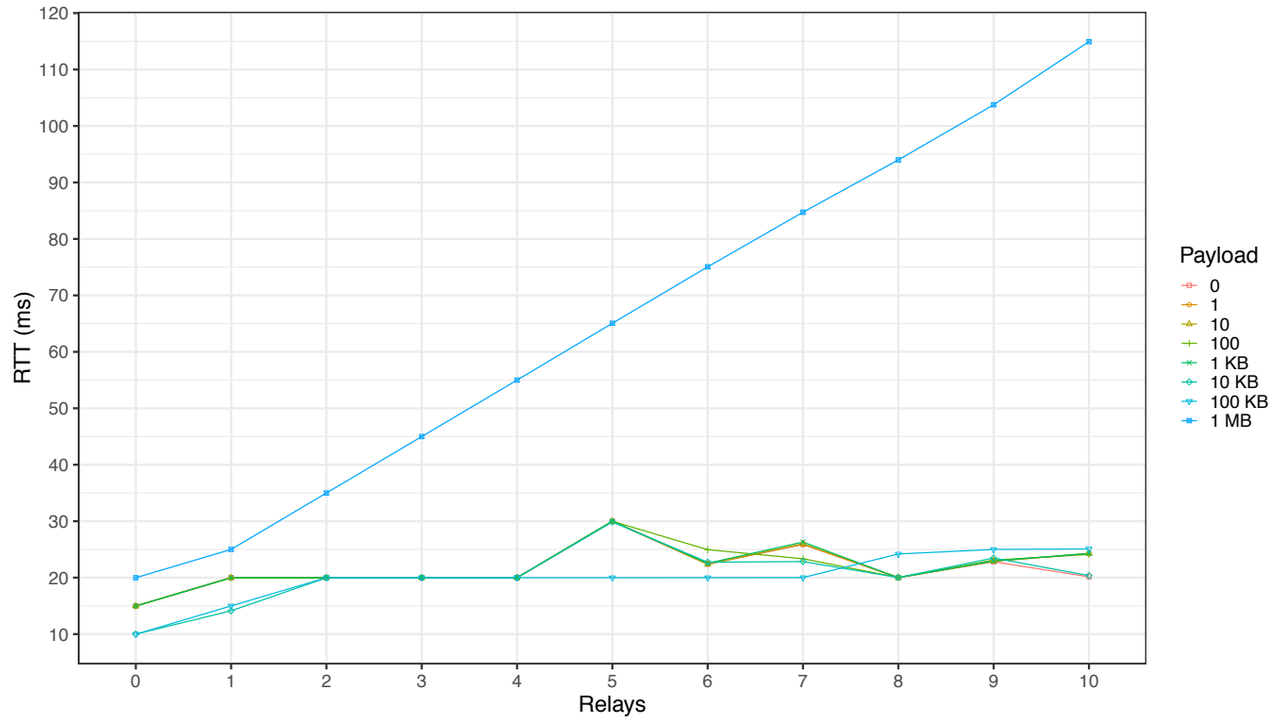
Roundtrip Latency



Roundtrip Latency

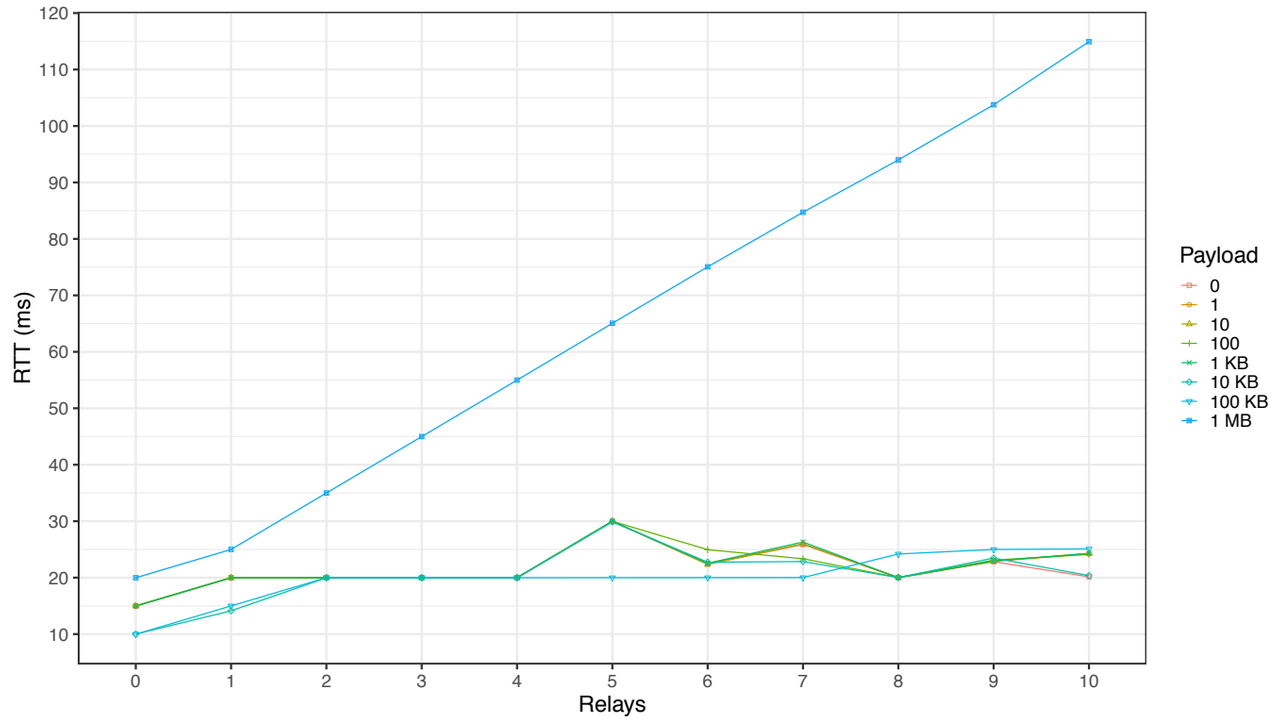


Roundtrip Latency



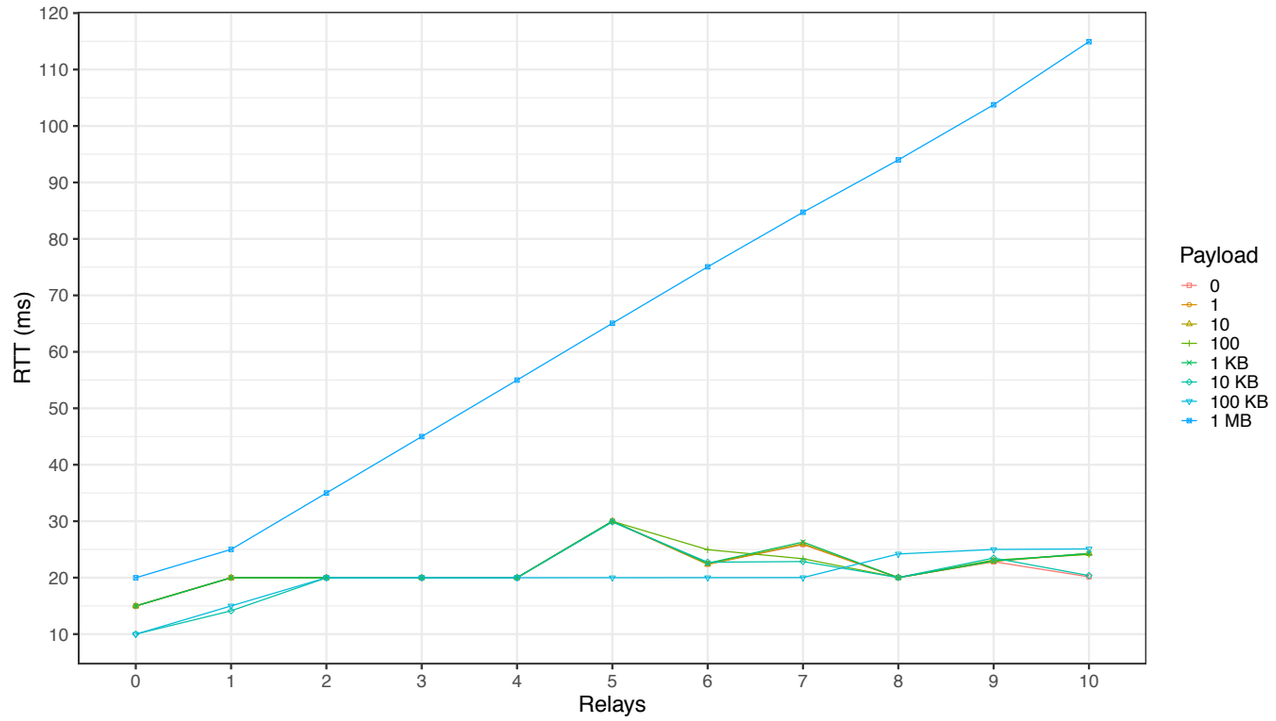
- Payloads **≤ 100 KB** behave similarly

Roundtrip Latency



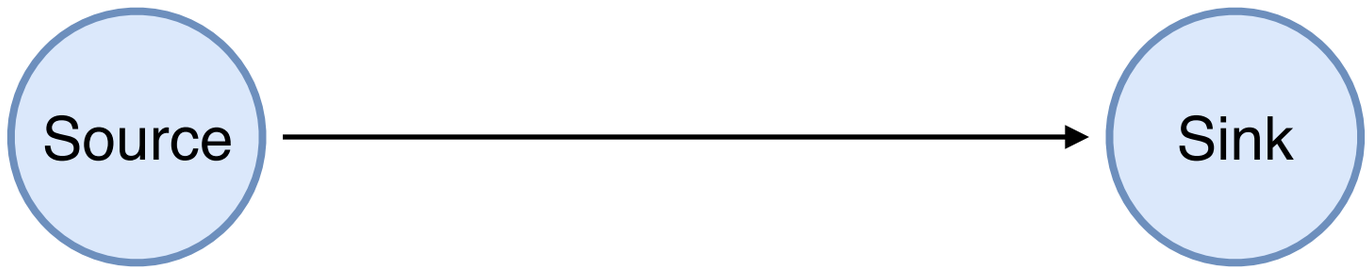
- Payloads **≤ 100 KB** behave similarly
- **Copying overhead** takes over at 1MB

Roundtrip Latency

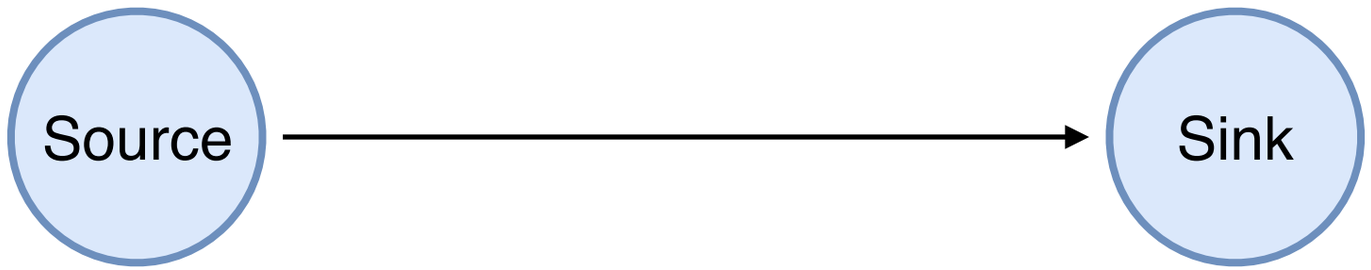


- Payloads **≤ 100 KB** behave similarly
- **Copying overhead** takes over at 1MB
- **Small penalty** for adding additional relays

Throughput Setup

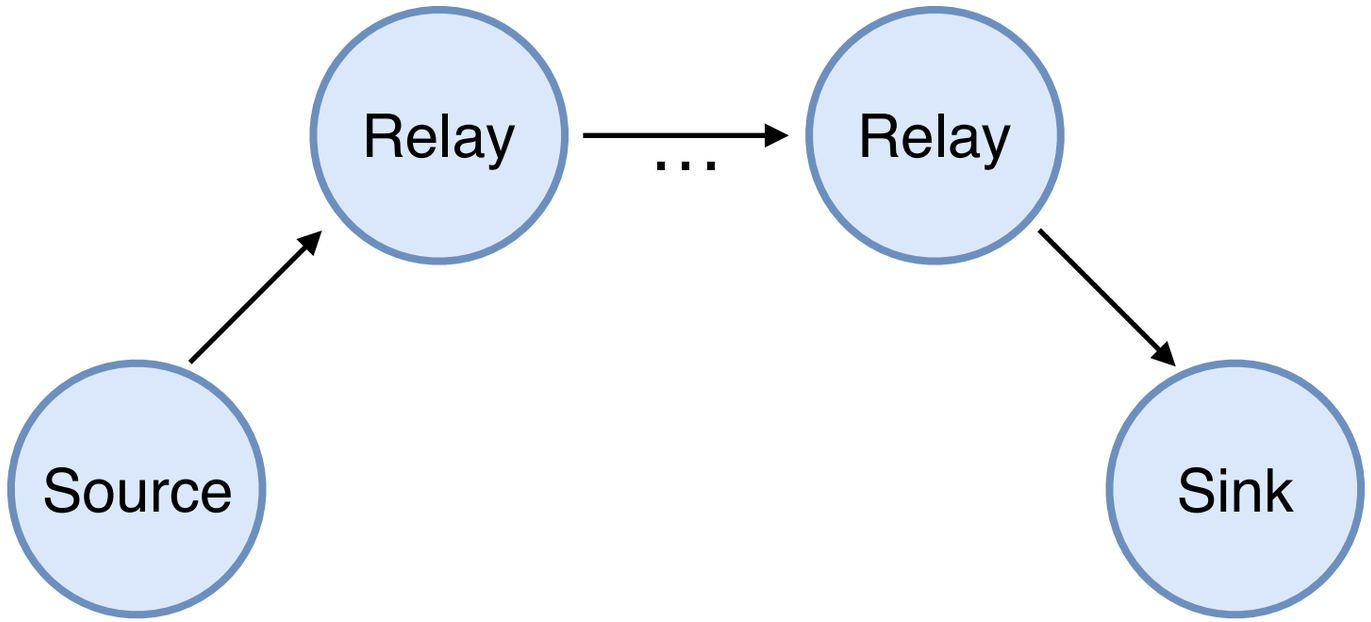


Throughput Setup



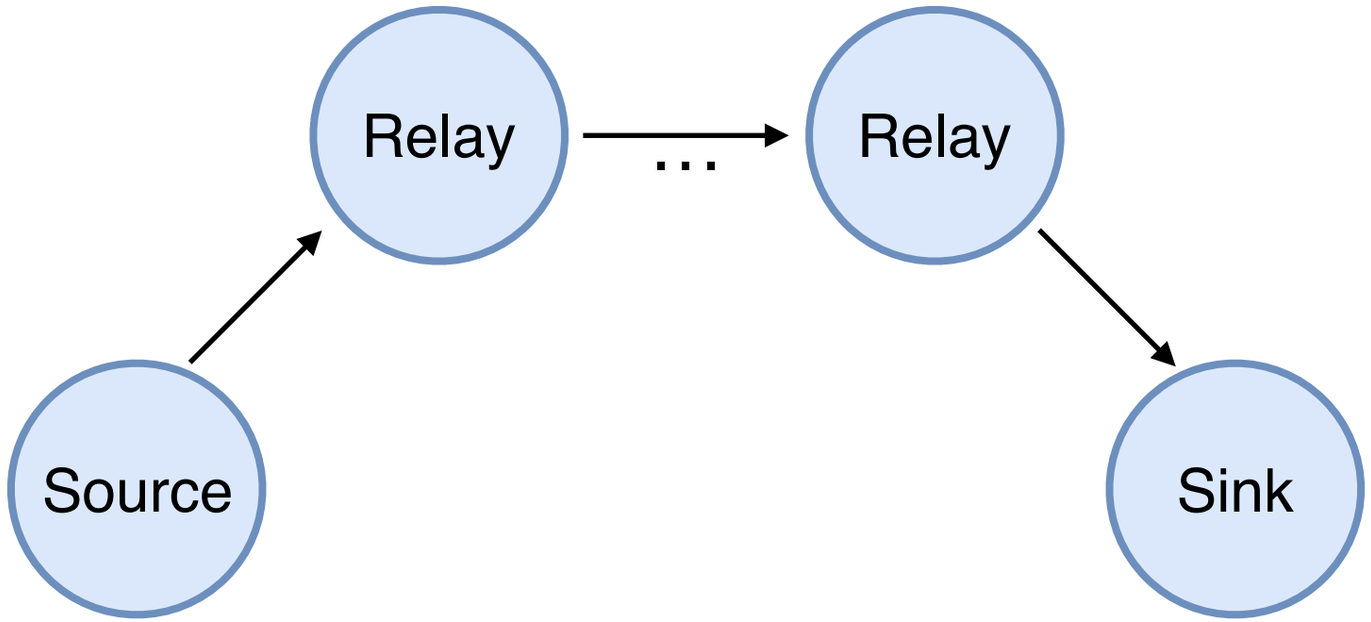
- **Measured**: messages per second received by *Sink*

Throughput Setup



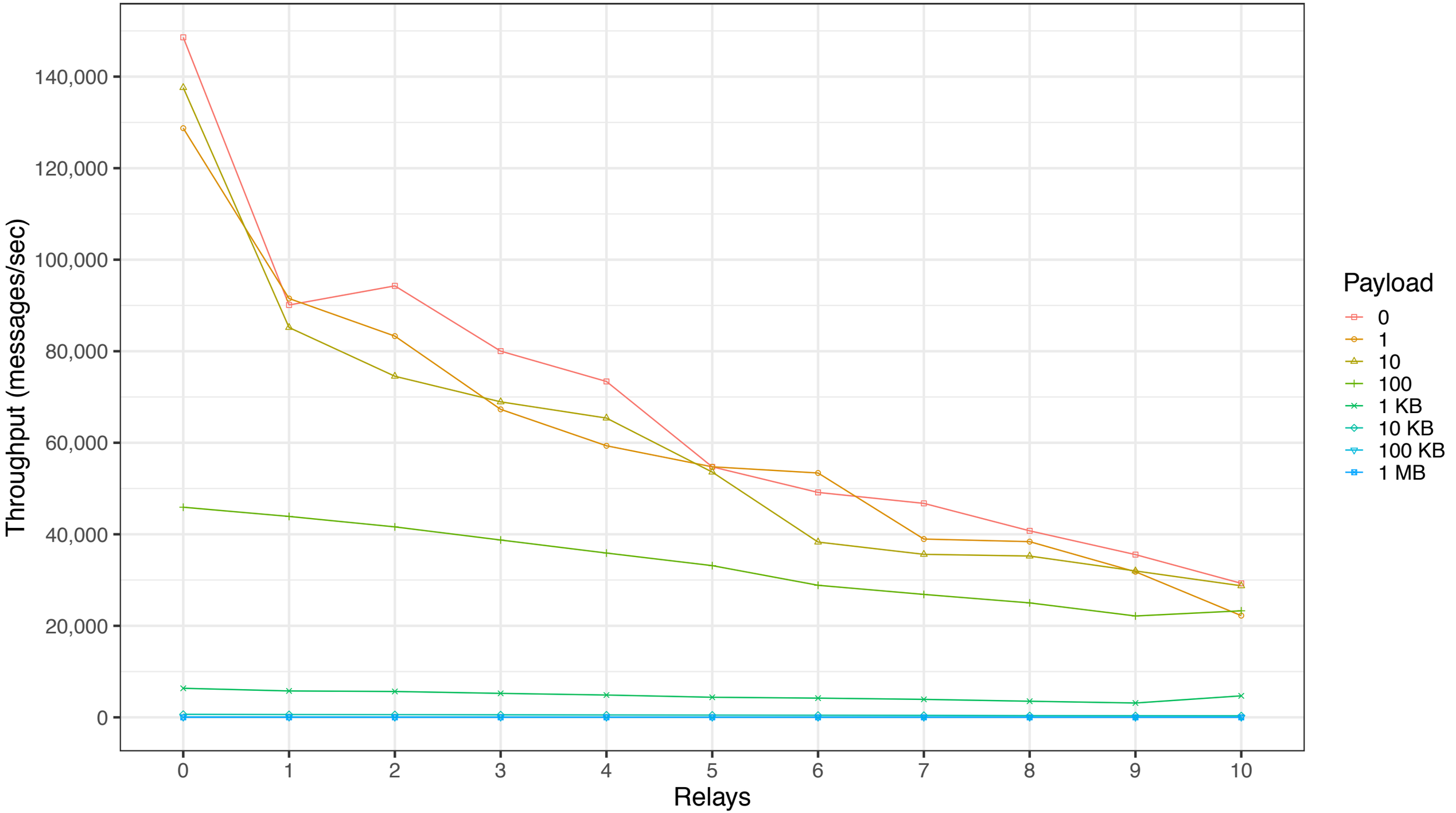
- **Measured**: messages per second received by *Sink*
- Varying **payload sizes** and varying **number of relays**

Throughput Setup

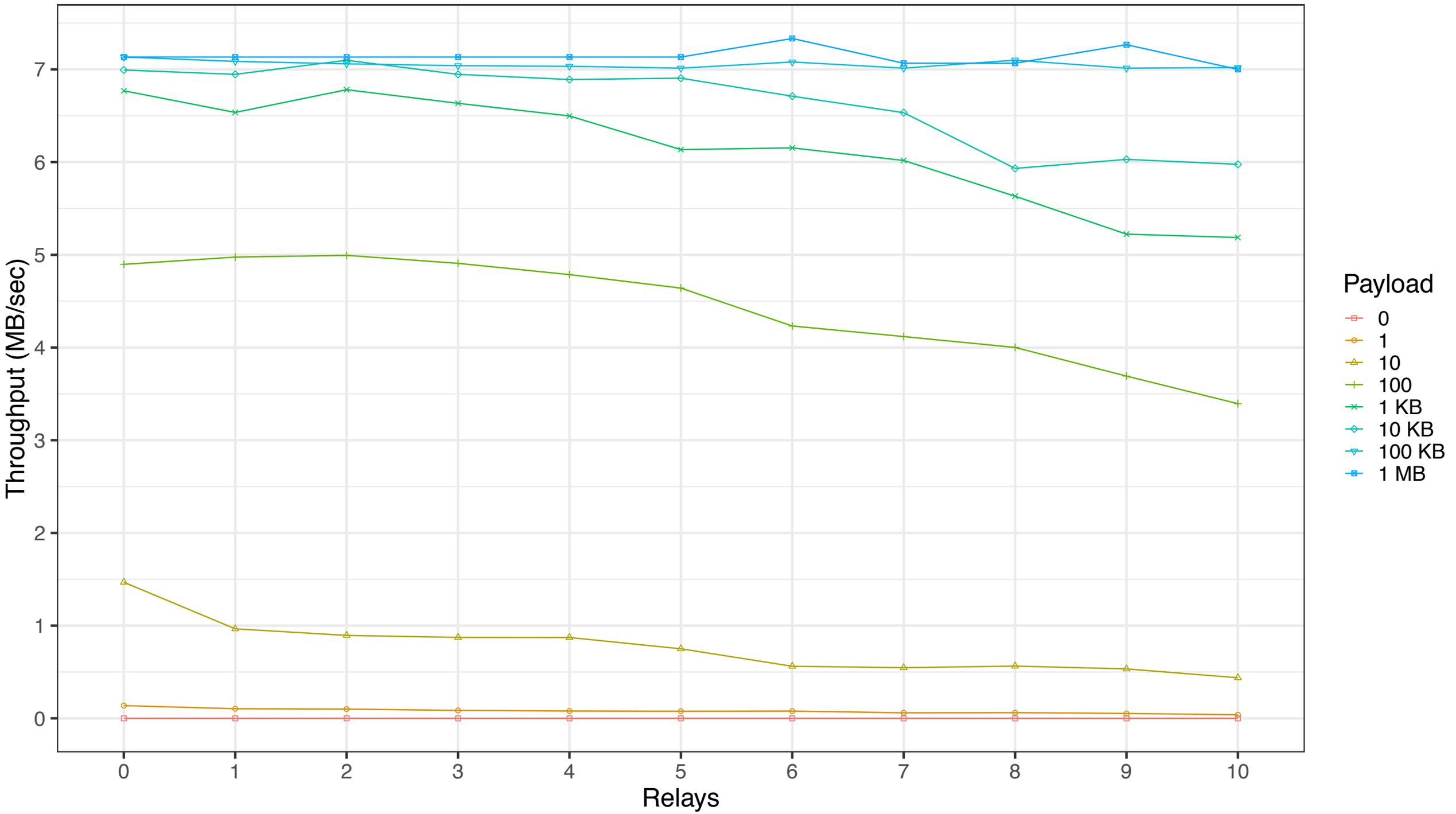


- **Measured**: messages per second received by *Sink*
- Varying **payload sizes** and varying **number of relays**
- **Expectation**: throughput should remain constant

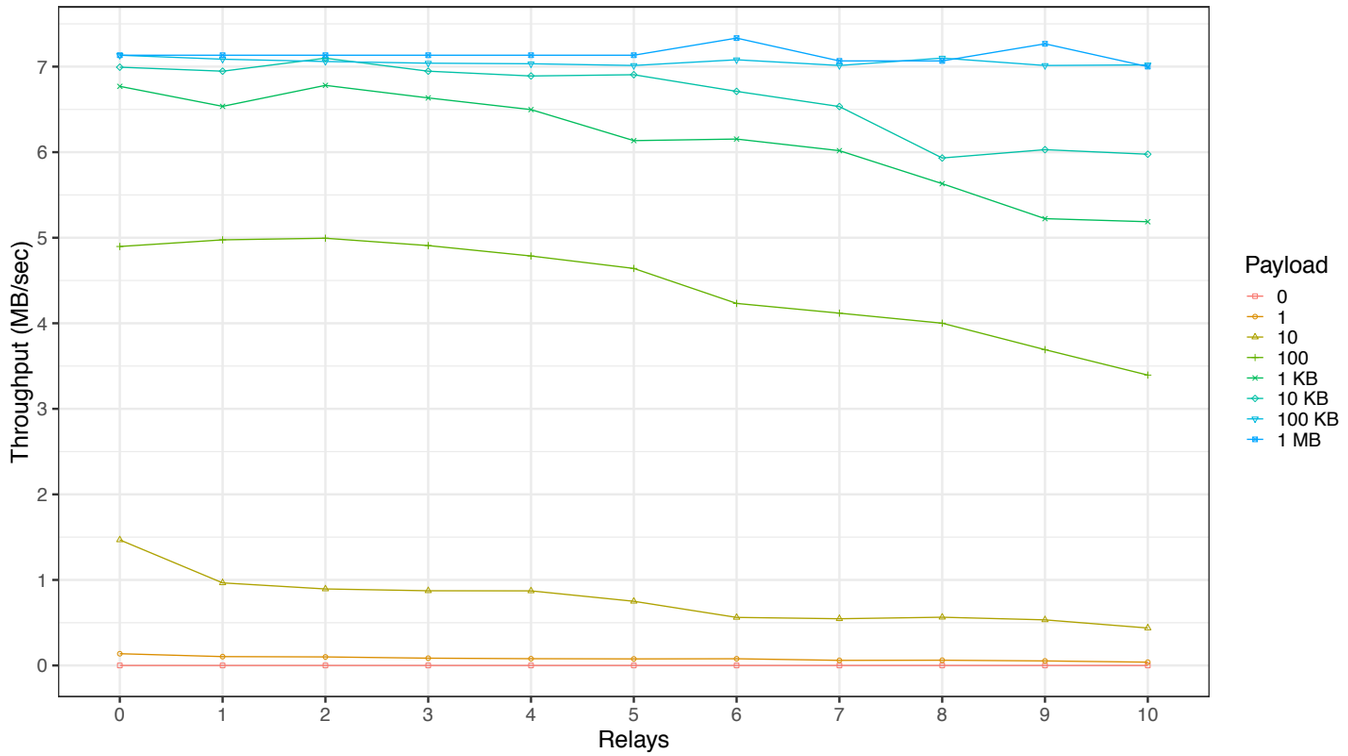
Throughput



Throughput

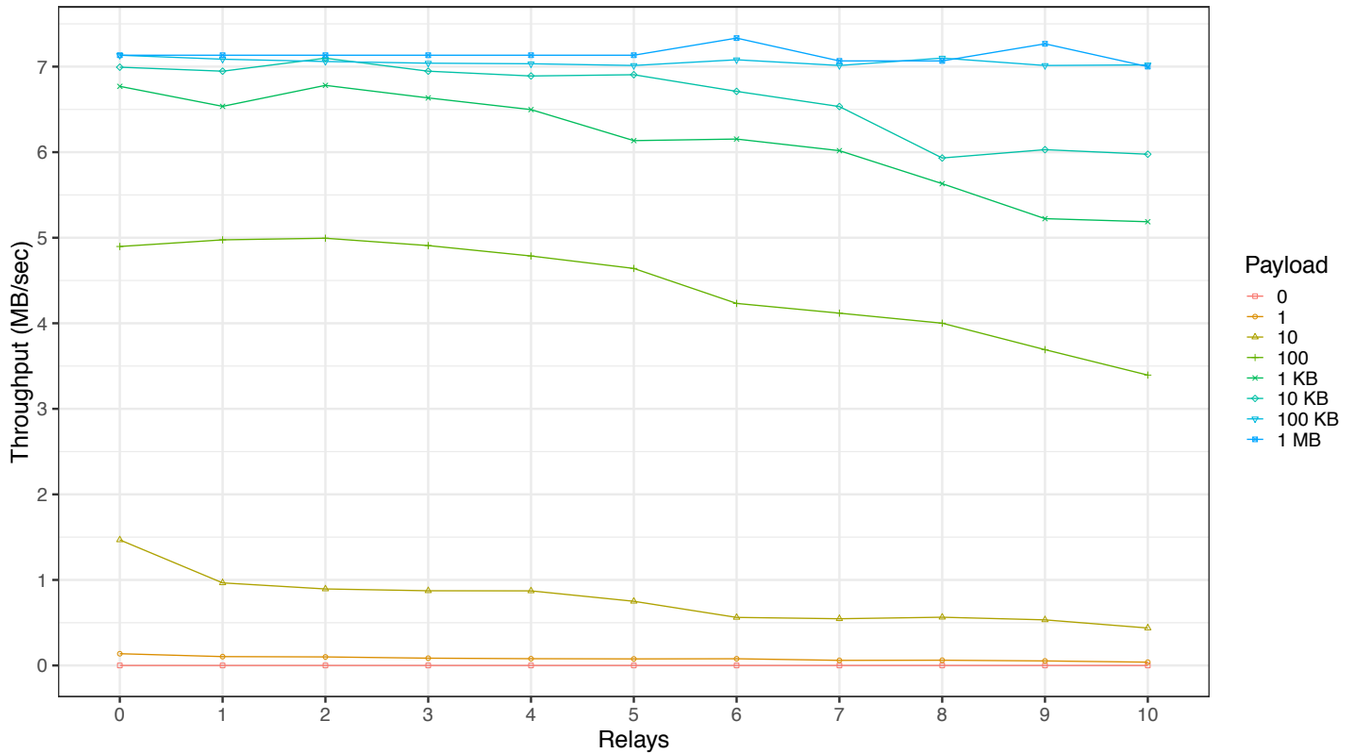


Throughput



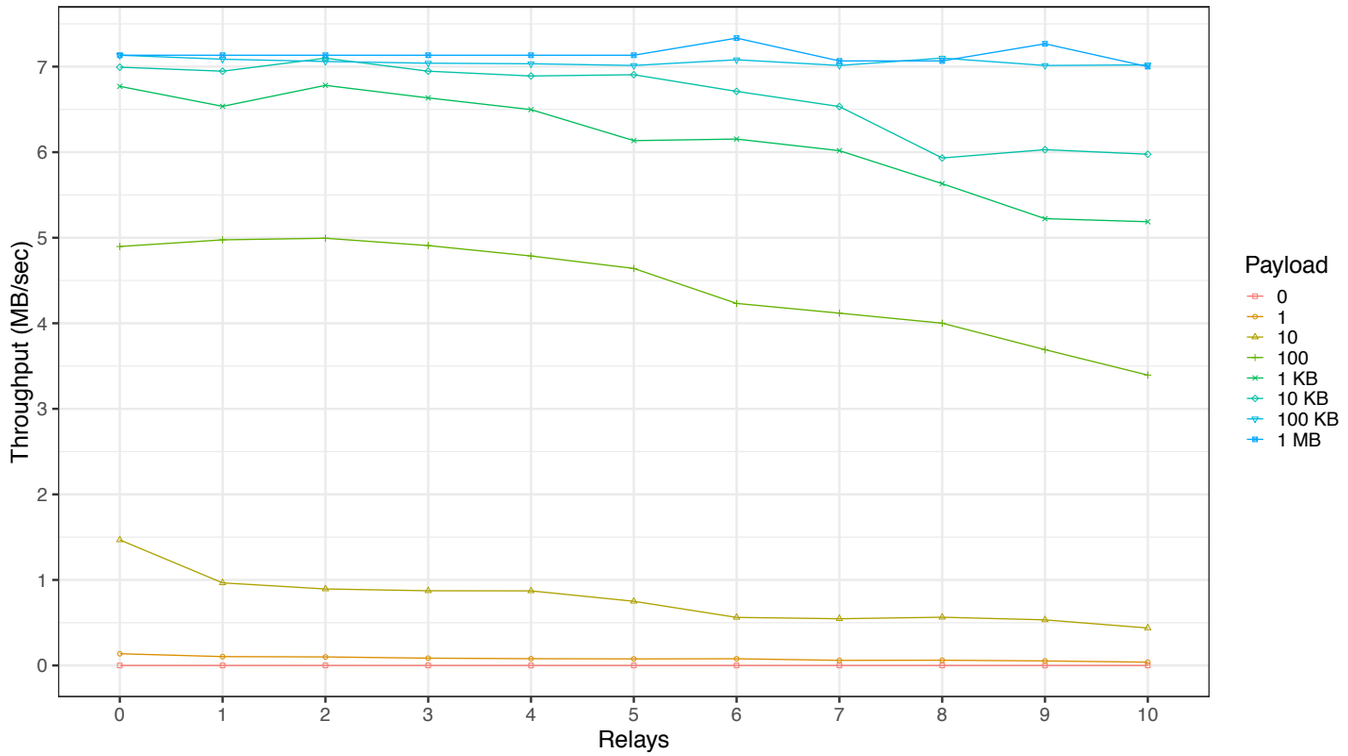
- Payloads **≤ 100 Bytes** behave similarly

Throughput



- Payloads **≤ 100 Bytes** behave similarly
- **Pub/sub layer** not designed for transferring large data

Throughput



- Payloads **≤ 100 Bytes** behave similarly
- **Pub/sub layer** not designed for transferring large data
- System **behavior not ideal** (more relays = less throughput)

Recap

Recap

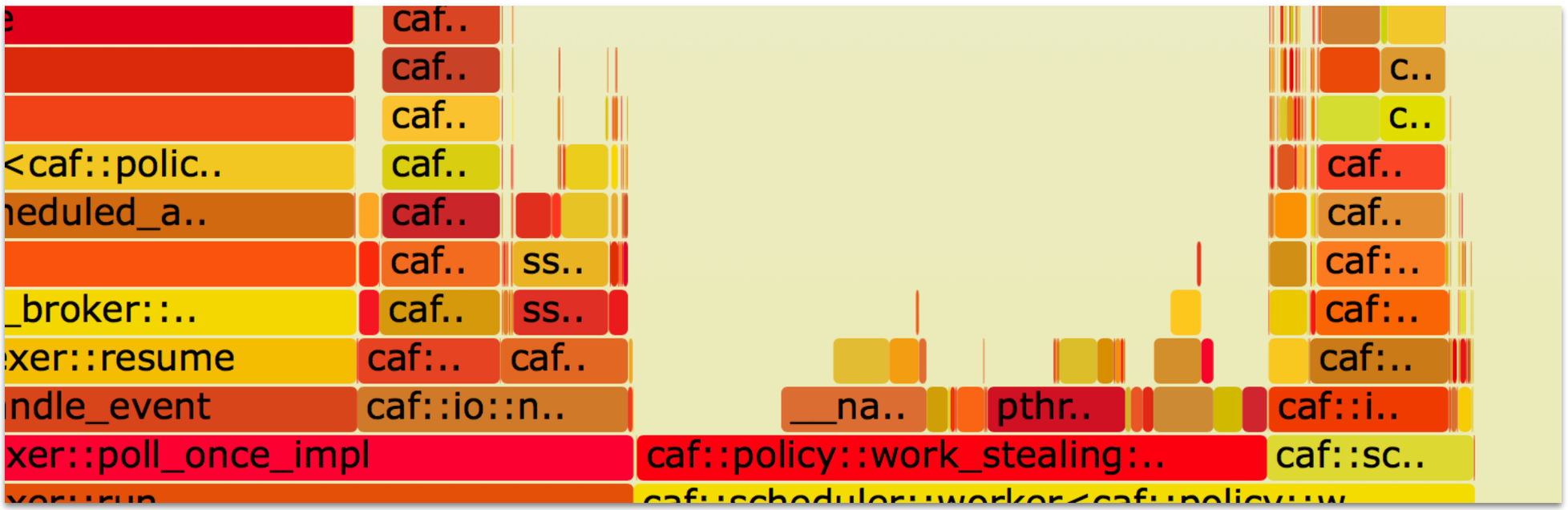
- **Latency** only increases for large payloads 👍

Recap

- **Latency** only increases for large payloads 👍

- **Throughput** affected by increasing number of relays 🤔

Inside Broker



Benchmark Findings

Benchmark Findings

- **Inefficient code** during serialization

Benchmark Findings

- **Inefficient code** during serialization
  - We already have a patch! 👨🏻‍💻

Benchmark Findings

- **Inefficient code** during serialization
  - We already have a patch! 👨‍💻
  - No longer shows up in flame graph 🔥

Benchmark Findings

- **Inefficient code** during serialization
  - We already have a patch! 👨‍💻
  - No longer shows up in flame graph 🔥
- Endpoint **bottlenecked by copying** overhead

Benchmark Findings

- **Inefficient code** during serialization
  - We already have a patch! 👨‍💻
    - No longer shows up in flame graph 🔥
- Endpoint **bottlenecked by copying** overhead
  - Reproduced in mockup, throughput stable with fix ✅

Benchmark Findings

- **Inefficient code** during serialization
  - We already have a patch! 👨‍💻
    - No longer shows up in flame graph 🔥
- Endpoint **bottlenecked by copying** overhead
  - Reproduced in mockup, throughput stable with fix ✅
  - No trivial patch, might affect Broker's API 😕

What did we learn?

Conclusion

Conclusion

- Broker enables **new use cases and integrations**

Conclusion

- Broker enables **new use cases and integrations**
  - **Automated workflows**, e.g., intel triggers historic query

Conclusion

- Broker enables **new use cases and integrations**
  - **Automated workflows**, e.g., intel triggers historic query
  - Straightforward integration with tools like VAST

Conclusion

- Broker enables **new use cases and integrations**
  - **Automated workflows**, e.g., intel triggers historic query
  - Straightforward integration with tools like VAST
- Easy **rapid prototyping** with Python

Conclusion

- Broker enables **new use cases and integrations**
  - **Automated workflows**, e.g., intel triggers historic query
    - Straightforward integration with tools like VAST
- Easy **rapid prototyping** with Python
- **Latency looks good** in microbenchmarks

Conclusion

- Broker enables **new use cases and integrations**
  - **Automated workflows**, e.g., intel triggers historic query
    - Straightforward integration with tools like VAST
- Easy **rapid prototyping** with Python
- **Latency looks good** in microbenchmarks
- Throughput has **room for improvement**

Thanks for Listening!

bro/broker

tenzir/events

tenzir/bro-vast

vast-io/vast

tenzir_company

tenzir.com