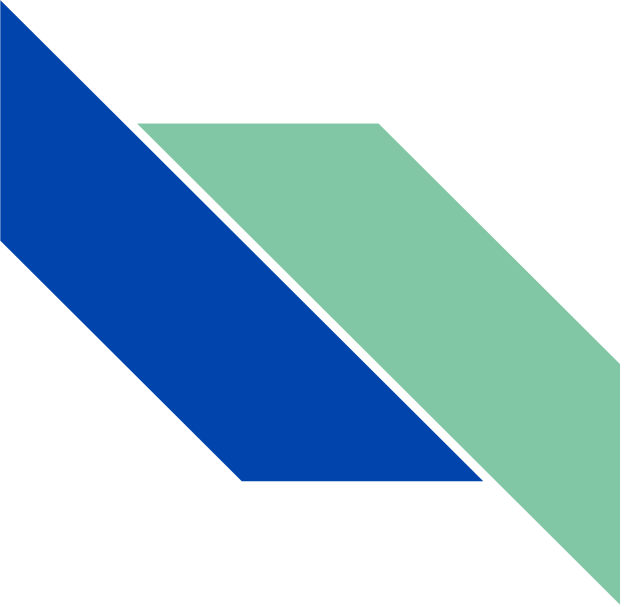


PacketStream

(a platform for analyzing
network traffic in near real-time)

Leonid Isaev

Insight Data Eng. Fellowship
NY 2019

Overview

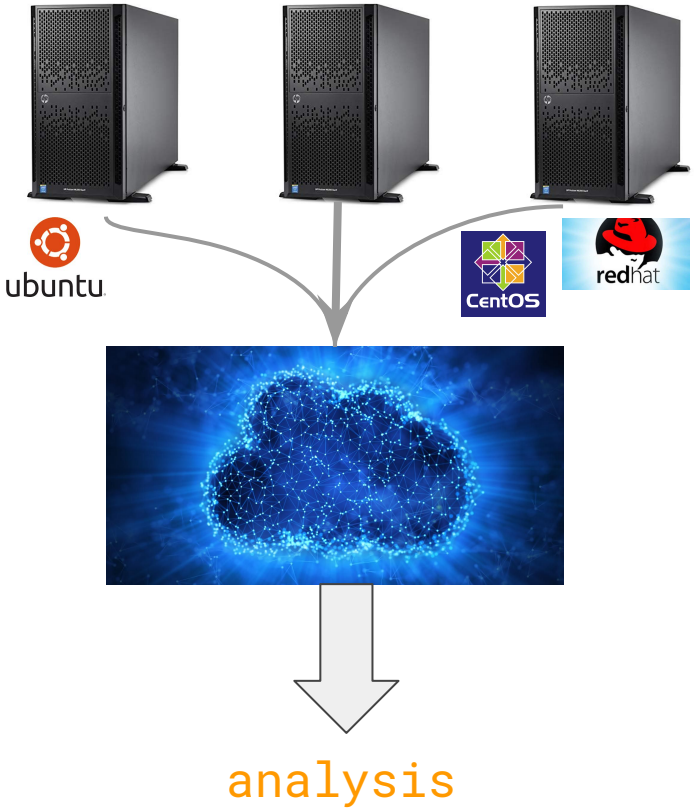
Goals:

- Build a distributed platform for troubleshooting network failures;
- Make it understand Wireshark-like rules.

Constraints:

- Must be field-ready
  - A data producer is a CentOS/Ubuntu server;
  - No “pip install <200 packages>” on clients.

PacketStream architecture



Examples of broken networks
(early connection resets)

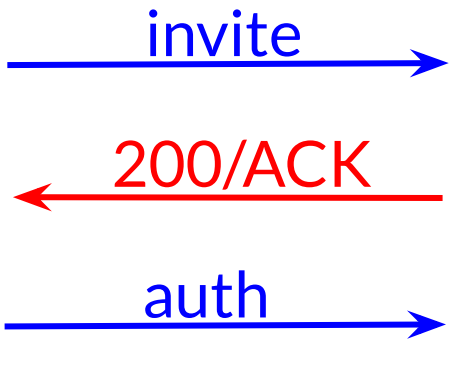
invite

Examples of broken networks
(early connection resets)

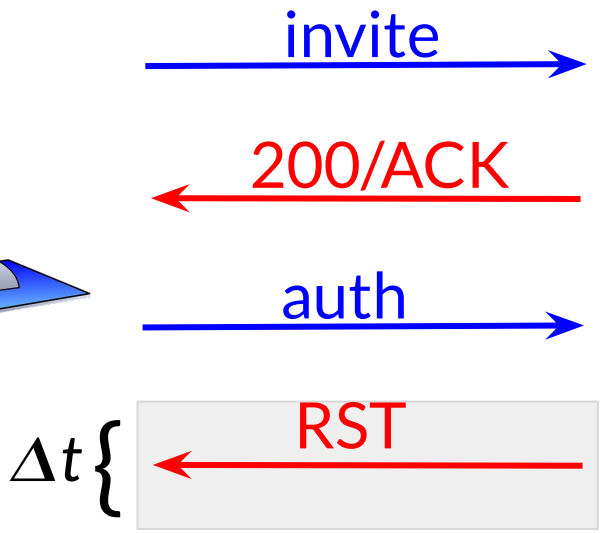
invite

200/ACK

Examples of broken networks
(early connection resets)



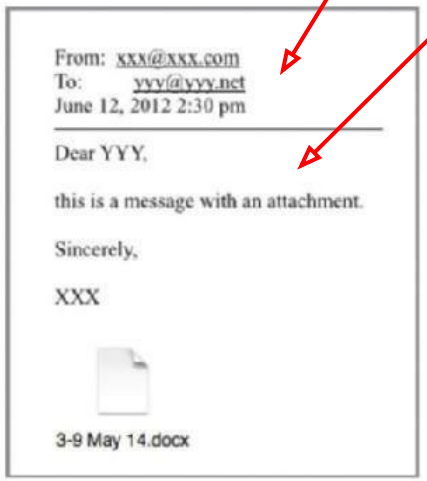
Examples of broken networks
(early connection resets)



can’t RST connection during wait window

Sample packet structure
(T.38 fax protocol)

- Packets have headers and payload, like email messages
- Headers are key-value pairs



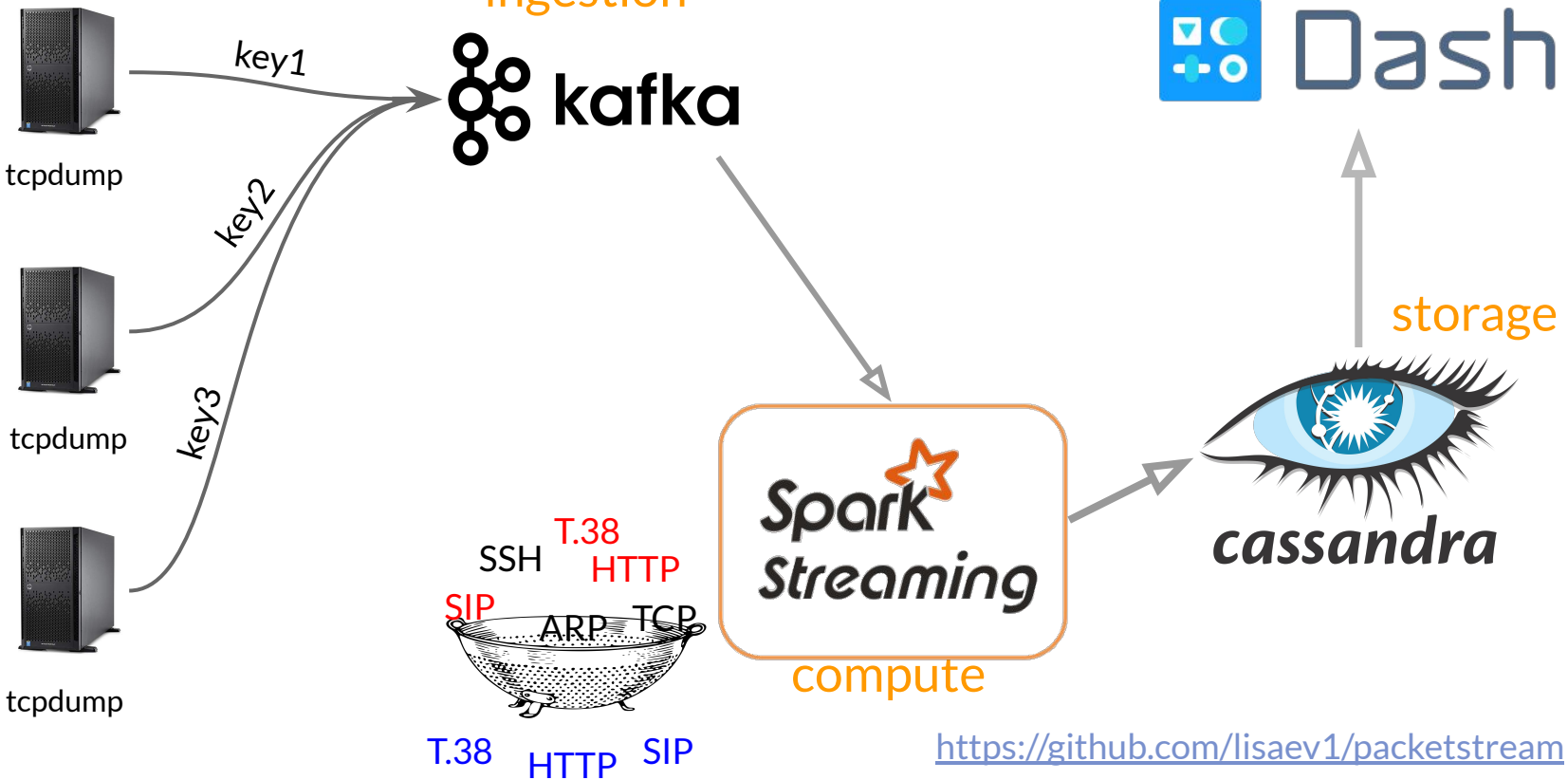
Header

Body

Attachment

```
Session Initiation Protocol (INVITE)
  Request-Line: INVITE sip:8553922666@199.199.10.4:5060 SIP/2.0
  Message Header
  Message Body
    Session Description Protocol
      Session Description Protocol Version (v): 0
      Owner/Creator, Session Id (o): CiscoSystemsSIP-GW-UserAgent 217 1007 IN IP4 172.20.4.10
      Session Name (s): SIP Call
      Connection Information (c): IN IP4 50.198.1.182
      Time Description, active time (t): 0 0
      Media Description, name and address (m): image 18824 udptl t38
      Connection Information (c): IN IP4 50.198.1.182
      Media Attribute (a): T38FaxVersion:3
      Media Attribute (a): T38MaxBitRate:33600
      Media Attribute (a): T38FaxFillBitRemoval:0
      Media Attribute (a): T38FaxTranscodingMMR:0
      Media Attribute (a): T38FaxTranscodingJBIG:0
      Media Attribute (a): T38FaxRateManagement:transferredTCF
      Media Attribute (a): T38FaxMaxBuffer:200
      Media Attribute (a): T38FaxMaxDatagram:320
      Media Attribute (a): T38FaxUdpEC:t38UDPRedundancy
```

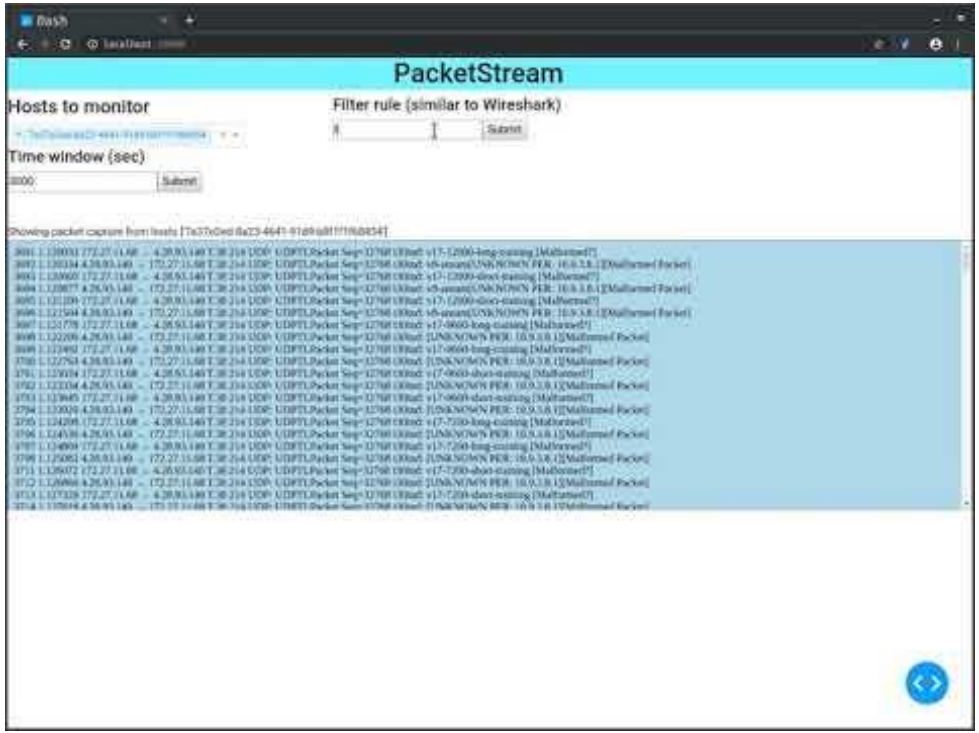
Pipeline



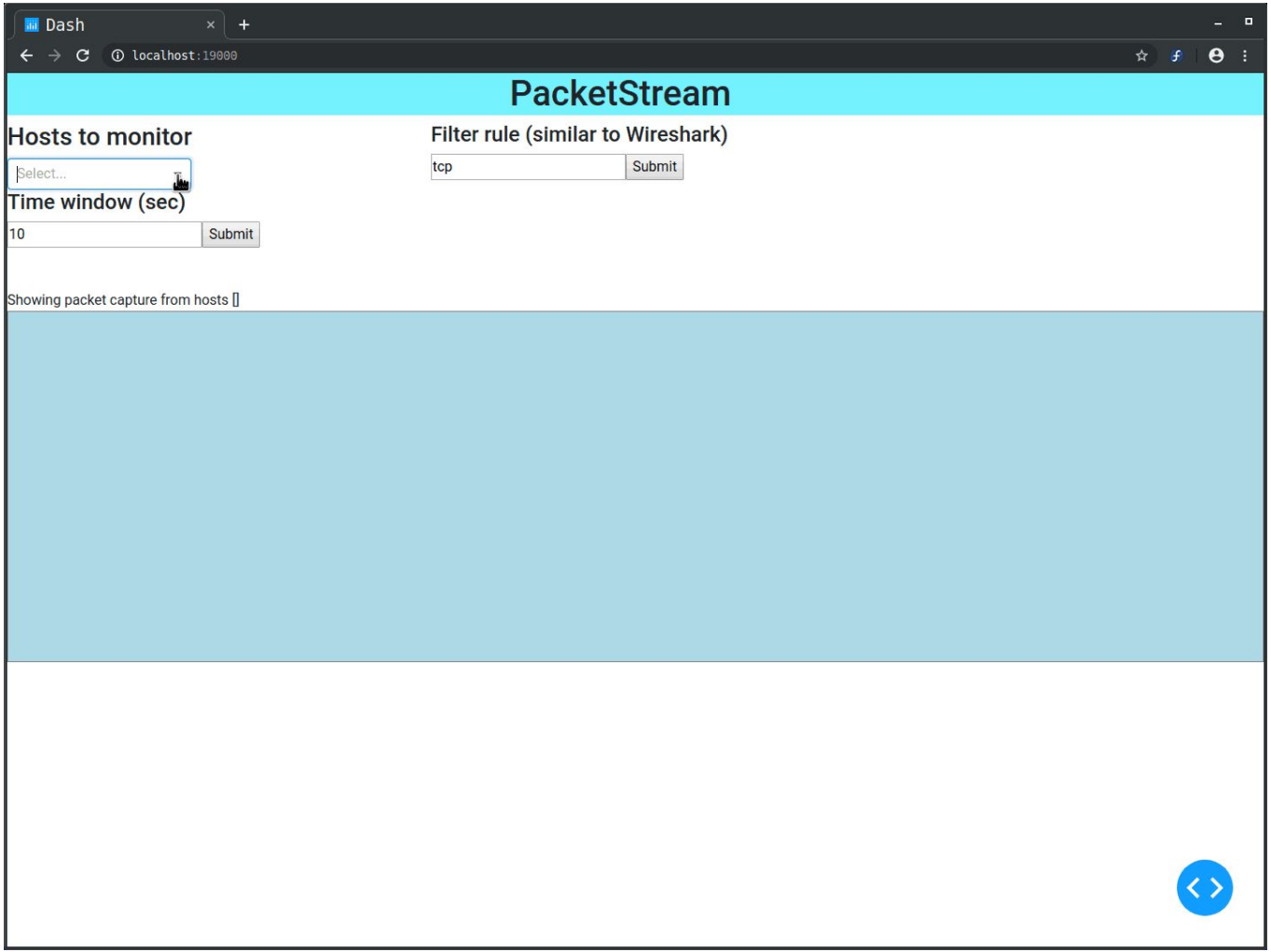
https://github.com/lisaev1/packetstream

Demo

(PacketStream)

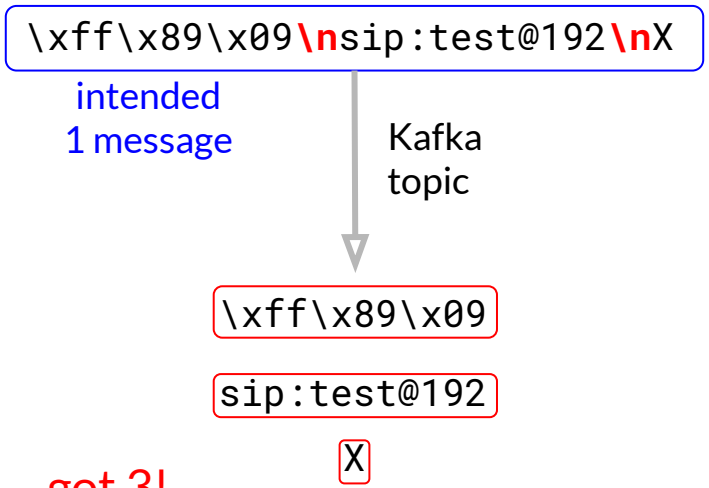


Demo
(fallback)



Challenges
(messages with $'\n' in Kafka)

Problem: Kafka treats '\n' as message separator

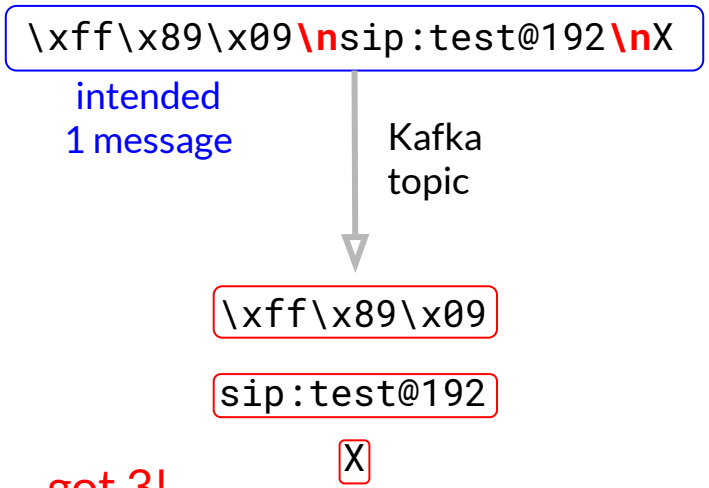


Solution: Use hexadecimal representation of the Python bytes objects

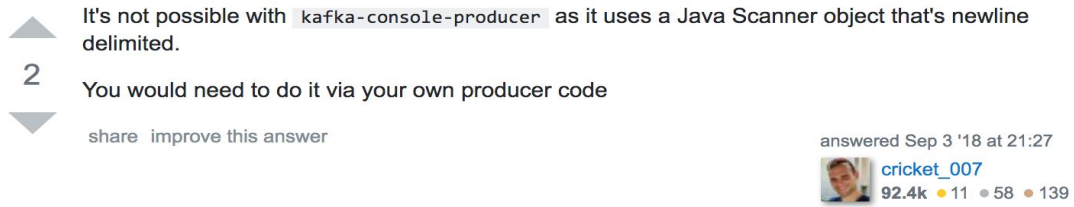
```
C02RJ1JPG8WM:packetstream insight$ cat misc/ser.py
#!python
"""
Kafka Consumer/Producer Java and Python API has problems handling messages with
$'\n' characters [1]. This program consumes a multiline message from stdin and
writes its hex-encoding to stdout. The latter can be pushed to Kafka as a singl
e
message.

[1] https://stackoverflow.com/questions/52151816/push-multiple-line-text-as-one
-message-in-a-kafka-topic
"""

import sys, binascii

for l in sys.stdin.buffer:
    sys.stdout.buffer.write(binascii.hexlify(l))

#sys.stdout.buffer.write(b'\n')
```

Challenges
(messages with $'\n' in Kafka)

Problem: Kafka treats '\n' as message separator



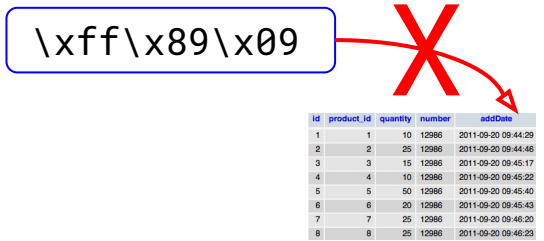
got 3!

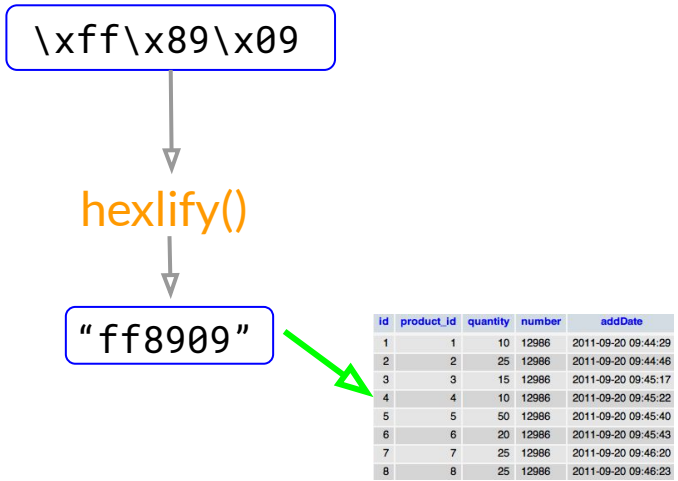



```
C02RJ1JPG8WM:packetstream insight$ cat misc/ser.py
#!python
"""
Kafka Consumer/Producer Java and Python API has problems handling messages with
$'\n' characters [1]. This program consumes a multiline message from stdin and
writes its hex-encoding to stdout. The latter can be pushed to Kafka as a singl
e
message.

[1] https://stackoverflow.com/questions/52151816/push-multiple-line-text-as-one
-message-in-a-kafka-topic
"""

import sys, binascii

for l in sys.stdin.buffer:
    sys.stdout.buffer.write(binascii.hexlify(l))

#sys.stdout.buffer.write(b'\n')
```

Challenges
(writing network packets to Cassandra from PySpark)

- Spark DataFrame doesn't support Python bytes object
  - Network packets look like b"\xff\x89\x09...";
  - Need to convert RDD -> DF for writing to Cassandra.



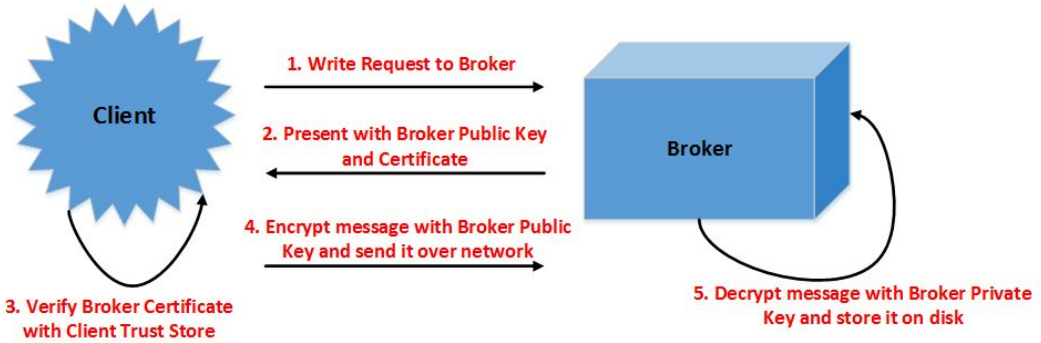


- **Solution:** store hex-encoded bytes
  - Base64 encoding can be wasteful;
  - Use binascii.hexlify(): b"\xff\x89\x09" -> "ff8909"

Challenges
(encrypted traffic from producers)

- Initial approach: wrap Kafka traffic in SSH tunnel
  - Works, but Kafka broker readvertisement doesn't happen
  - Slow -- SSH limits packet length (compared to http/https)

- **Solution:** use native Kafka TLS protocol
  - Certs are a PAIN
  - CA vs self-signed



Short bio

Background

- Indiana Univ Bloomington (computational physics)
- Postdoc at LSU Baton Rouge, Los Alamos, JILA/NIST/CU Boulder
- **Linux support/software engineer @ iFax Solutions, Norristown PA**

Fun facts/hobbies:

- Real-life Sheldon
- Science-fiction, hacking on Arch Linux

Appendix

Kafka topic structure

9f1faa88-fa44-4ce9-b2ab-eb267630cebf

packets: bytestream


```
message = {
    “key”: 9f1faa88-fa44-4ce9-b2ab-eb267630cebf,
    “value”: b”...”
}
```

- Assign a UUID to each machine
  - Can be license number;
  - Python `uuid.uuid4()`;
- UUID -> key in Kafka topic;
- Choose # of partitions = # of nodes in the cluster
  - Packets from each host end up in the same partition!

Cassandra table structure

hostid (uuid)	pktid (timeuuid)	payload (text)
7e37e2ed-8a23-4641-91d9-b8f1f1f68454	6cde8df0-f019-11e9-b24e-52a33b5da58a	0050569e7115b...
89eab76b-2969-4031-9f27-7e53614edc55	6c6cd4ee-f019-11e9-b24e-52a33b5da58a	0050716e294b6...
a704ae05-032f-426c-a211-e6249e885287	6c6d31a0-f019-11e9-b24e-52a33b5da58a	0050af98223c6...

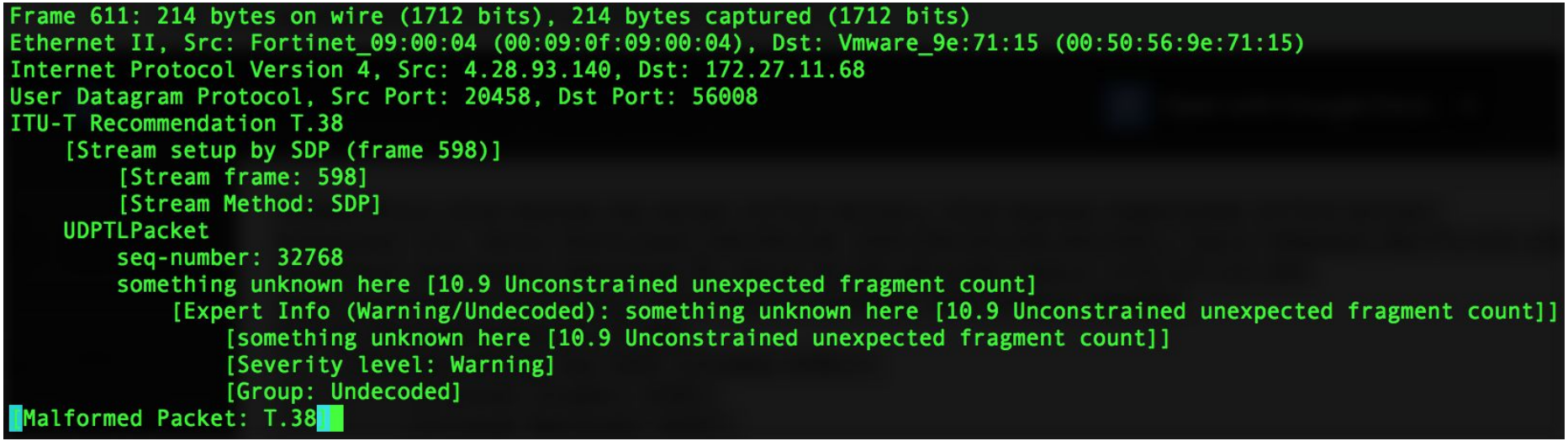
```
CREATE TABLE t8 (
    hostid uuid,                          # analogous to uuid.uuid4()
    pktid timeuuid,                       # same as uuid.uuid1()
    payload text,
    PRIMARY KEY ((hostid), pktid)
) WITH CLUSTERING ORDER BY (pktid DESC);
```

Examples of broken networks

No.	Time	Source	Destination	Protocol	Info
1	0.000000	ipphone	Broadcast	ARP	Who has 192.168.1.10? Tell 192.168.1.151
2	0.052480	pbxhost	ipphone	ARP	192.168.1.10 is at 00:10:5a:e1:90:6e
3	0.000146	ipphone	pbxhost	SIP/SDP	Request: INVITE sip:sip%3a613@fwd.pulver.com, with session description
4	0.001480	pbxhost	ipphone	SIP	Status: 407 Proxy Authentication Required
5	0.052920	ipphone	pbxhost	SIP	Request: ACK sip:sip%3a613@fwd.pulver.com
6	0.062692	ipphone	pbxhost	SIP/SDP	Request: INVITE sip:sip%3a613@fwd.pulver.com, with session description
7	0.063859	pbxhost	ipphone	SIP	Status: 407 Proxy Authentication Required
8	0.123275	ipphone	pbxhost	SIP	Request: ACK sip:sip%3a613@fwd.pulver.com
9	0.132364	ipphone	pbxhost	SIP/SDP	Request: INVITE sip:sip%3a613@fwd.pulver.com, with session description
10	0.135772	pbxhost	ipphone	SIP	Status: 407 Proxy Authentication Required
11	0.193816	ipphone	pbxhost	SIP	Request: ACK sip:sip%3a613@fwd.pulver.com
12	0.202913	ipphone	pbxhost	SIP/SDP	Request: INVITE sip:sip%3a613@fwd.pulver.com, with session description
13	0.208640	pbxhost	ipphone	SIP	Status: 407 Proxy Authentication Required
14	0.258099	ipphone	pbxhost	SIP	Request: ACK sip:sip%3a613@fwd.pulver.com

- Broken firmware -> timing issues, random connection resets;
- Wrong credentials, encoding, etc.

Packet structure
(broken packets)



Repository

https://github.com/lisaev1/packetstream

```
usage: scapy_prod.py [-h] [-p FILE] [-t TOPIC] [-s SERVER:PORT]

optional arguments:
  -h, --help            show this help message and exit
  -p FILE, --pcap FILE  Packet capture (pcap0 file to parse (default: none)
  -t TOPIC, --topic TOPIC
                        Kafka topic (default: sharks)
  -s SERVER:PORT, --servers SERVER:PORT
                        Kafka broker (default: none). Can be specified
                        multiple times.
```

lisaev1 / packetstream

Watch 0 | Star 0

Code | Issues 0 | Pull requests 1 | Projects 0 | Security | Insights

Distributed platform for analyzing streaming packer captures

9 commits | 2 branches | 0 releases | 1 contributor

Branch: exp | View #2 | Find File | Clone or download

This branch is 3 commits ahead of master. #2 Compar

lisaev1 Miscellaneous utilities: a simple script to send a file as a single K... Latest commit 89baa77 5 hours ag

archlinux	Add the "python-kafka" Arch Linux package	16 hours ag
examples/small-cluster	Add Apache Spark buildscripts and example configuration.	18 hours ag
misc	Miscellaneous utilities: a simple script to send a file as a single K...	5 hours ag
producers_consumers	* Rename producers/ -> producers_consumers/	5 hours ag
security	New dir security/ that will contain crypto-related stuff. So far, the...	4 days ag
streaming	Start work on the Spark Streaming code.	18 hours ag
README.md	Initial import	14 days ag

```python
#!python
"""
Kafka producer for packets stored in a pcap file.
Sends 1 scapy packet per message.
"""

import os, sys, kafka, common
import scapy.all as scapy

#-- parse cmdline options
pcap_f, topic, srv_list = common._parseopts()

#-- sanity checks
if (not os.path.isfile(pcap_f)):
    print(f"File \"{pcap_f}\" does not exist! Aborting.")
    sys.exit(1)

if (len(srv_list) == 0):
    print("Please specify at least one Kafka broker! Aborting.")
    sys.exit(1)

#-- define a producer
producer = kafka.KafkaProducer(bootstrap_servers = srv_list)

#-- read the pcap file
capture = scapy.rdpcap(pcap_f)
for p in capture:
    producer.send(topic, bytes(p))

producer.close()
```

Test run -- a look inside Spark Stream

```
C02RJ1JPG8WM:packetstream insight$ python3 producers_consumers/python/scapy_prod.py -p ~/Downloads/_l
ifax.pcap -t "test1" $(for i in {1..3}; do echo -nE "-s 192.168.122.7${i}:9092 "; done)
```

```
lisaev@spark-node2 ~$ spark-submit --master spark://spark-node1:7077 --jars ks.jar ./echo.py 192.1
68.122.71:9092 test1

<bound method Packet.summary of <Ether  dst=00:09:0f:09:00:04 src=00:50:56:9e:71:15 type=IPv4 |<IP
version=4 ihl=5 tos=0x0 len=931 id=28856 flags=DF frag=0 ttl=64 proto=udp chksum=0xad8a src=172.27.1
1.68 dst=4.28.93.140 |<UDP  sport=sip dport=sip len=911 chksum=0x1ca8 |<Raw  load='INVITE sip:0ae9fa
eaad38d22c779775e03bb9d755@4.28.93.140:5060 SIP/2.0\r\nFrom: <sip:4012830857@sbc01-stl.sip.univergeb
lue.com>;tag=dcfef6b0-0-13c4-65014-a5c-75d5ed11-a5c\r\nTo: "PHILADELPHIA PA"<sip:2153830199@sbc01-st
l.sip.univergeblue.com>;tag=4.28.93.140+1+3218a8f7+270ff97\r\nCall-ID: 0gQAAC8WAAACBAAALxYAANYw/3hJv
01vNq45pvOsNQbJaGaS1oYg5/3tEuM5AjEX@4.28.93.140\r\nCSeq: 1 INVITE\r\nVia: SIP/2.0/UDP 172.27.11.68:5
060;branch=z9hG4bK-a60-28871c-296f3a47-dd2ff5e8\r\nSupported: 100rel\r\nMax-Forwards: 70\r\nUser-Age
nt: Brktsip/6.9.6B10 (Dialogic)\r\nContact: <sip:172.27.11.68>\r\nContent-Type: application/sdp\r\nC
ontent-Length: 286\r\n\r\nv=0\r\no=- 2208997233 0083442001 IN IP4 172.27.11.68\r\ns=no_session_name\
r\nt=0 0\r\nm=image 56008 udptl t38\r\nc=IN IP4 172.27.11.68\r\na=T38FaxVersion:0\r\na=T38MaxBitRate
:14400\r\na=T38FaxRateManagement:transferredTCF\r\na=T38FaxMaxBuffer:200\r\na=T38FaxMaxDatagram:72\r
\na=T38FaxUdpEC:t38UDPRedundancy\r\n' |>>>>>

<bound method Packet.summary of <Ether  dst=00:50:56:9e:71:15 src=00:09:0f:09:00:04 type=IPv4 |<IP
version=4 ihl=5 tos=0x0 len=491 id=49541 flags= frag=0 ttl=64 proto=udp chksum=0x9e75 src=4.28.93.14
0 dst=172.27.11.68 |<UDP  sport=sip dport=sip len=471 chksum=0x9451 |<Raw  load='SIP/2.0 100 Trying\
r\nCall-ID: 0gQAAC8WAAACBAAALxYAANYw/3hJv01vNq45pvOsNQbJaGaS1oYg5/3tEuM5AjEX@4.28.93.140\r\nCSeq: 1
INVITE\r\nFrom: <sip:4012830857@sbc01-stl.sip.univergeblue.com>;tag=dcfef6b0-0-13c4-65014-a5c-75d5ed
11-a5c\r\nTo: "PHILADELPHIA PA"<sip:2153830199@sbc01-stl.sip.univergeblue.com>;tag=4.28.93.140+1+321
8a8f7+270ff97\r\nVia: SIP/2.0/UDP 172.27.11.68:5060;branch=z9hG4bK-a60-28871c-296f3a47-dd2ff5e8\r\nS
erver: SIP/2.0\r\nContent-Length: 0\r\nUser-Agent: SoTel\r\n\r\n' |>>>>>

...
```

Roadmap: putting all this into production

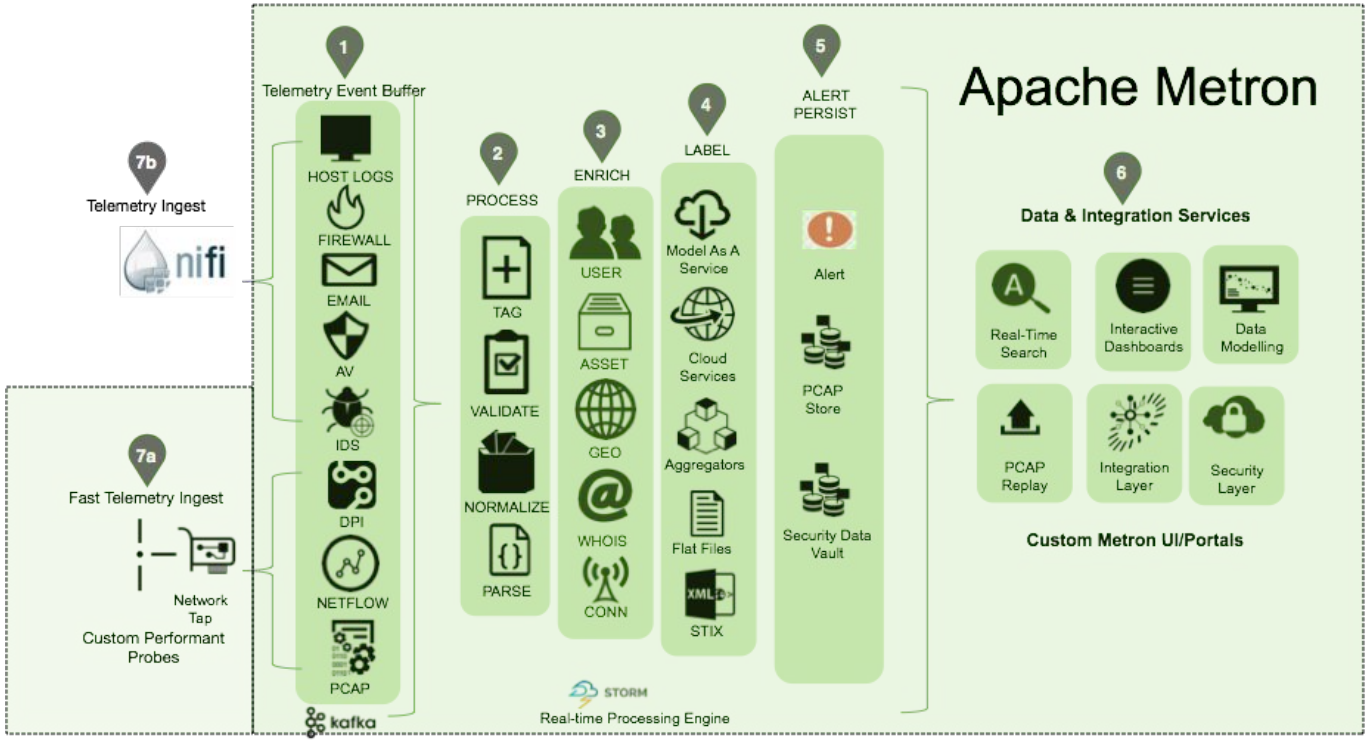
Nobody should start to undertake a large project. You start with a small *trivial* project, and you should never expect it to get large. If you do, you'll just overdesign and generally think it is more important than it likely is at that stage. So start small, and think about the details. Don't think about some big picture and fancy design. If it doesn't solve some fairly immediate need, it's almost certainly over-designed.

-- Linus Torvalds, 2004

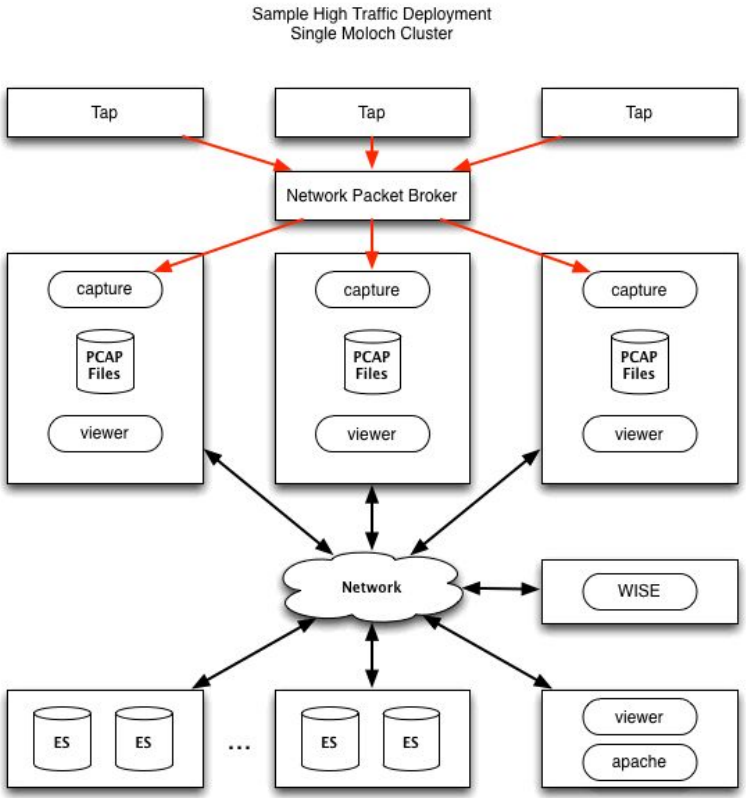
- CLI interface (Dash is great, but non-functional):
  - Need to look at raw packets;
  - Wireshark rules/filters;
- Richer processing rules in Spark, incl wireshark rule files;
- Dynamic provisioning of Kafka brokers and Spark nodes;
- Dedicated distributed storage (perhaps outside Cassandra).

complexity

Related work: Apache Metron



Related Work: AOL Moloch



Why not use Kafka streaming?

- Streaming apps have to be started manually on each broker with # of instances = # of partitions in the consumed topic;
- No automatic re-spawning in case of node failure;
- Complex code (lots of async stuff);
- Processing and message handling are co-located.

We did test Faust streaming for Kafka

Faust

A library for building streaming applications in Python.

Star 3,691

build failing

Navigation

Faust - Python Stream Processing

```
# Python Streams ʕ◕ᴥ◕ʔ
# Forever scalable event processing & in-memory durable K/V store;
# w/ asyncio & static typing.
import faust
```

Faust is a stream processing library, porting the ideas from Kafka Streams to Python.

It is used at Robinhood to build high performance distributed systems and real-time data pipelines that process billions of events every day.

Faust provides both *stream processing* and *event processing*, sharing similarity with tools such as Kafka Streams, Apache Spark/Storm/Samza/Flink,

It does not use a DSL, it's just Python! This means you can use all your favorite Python libraries when stream processing: NumPy, PyTorch, Pandas, NLTK, Django, Flask, SQLAlchemy, ++

Faust requires Python 3.6 or later for the new async/await syntax, and variable type annotations.

Here's an example processing a stream of incoming orders:

```
app = faust.App('myapp', broker='kafka://localhost')

# Models describe how messages are serialized:
# {"account_id": "3fae-...", amount": 3}
class Order(faust.Record):
    account_id: str
    amount: int

@app.agent(value_type=Order)
async def order(orders):
    async for order in orders:
```