PyTAPS
Implementation Report
Framers and Multicast

Max Franke
Jake Holland

TAPS
IETF 106, November 2019, Singapore

Implementation status

- What works so far
  - TCP, UDP, TLS (1.3)
  - Protocol & remote Endpoint racing
  - Framers
- What we want to do next:
  - Interface selection
  - QUIC

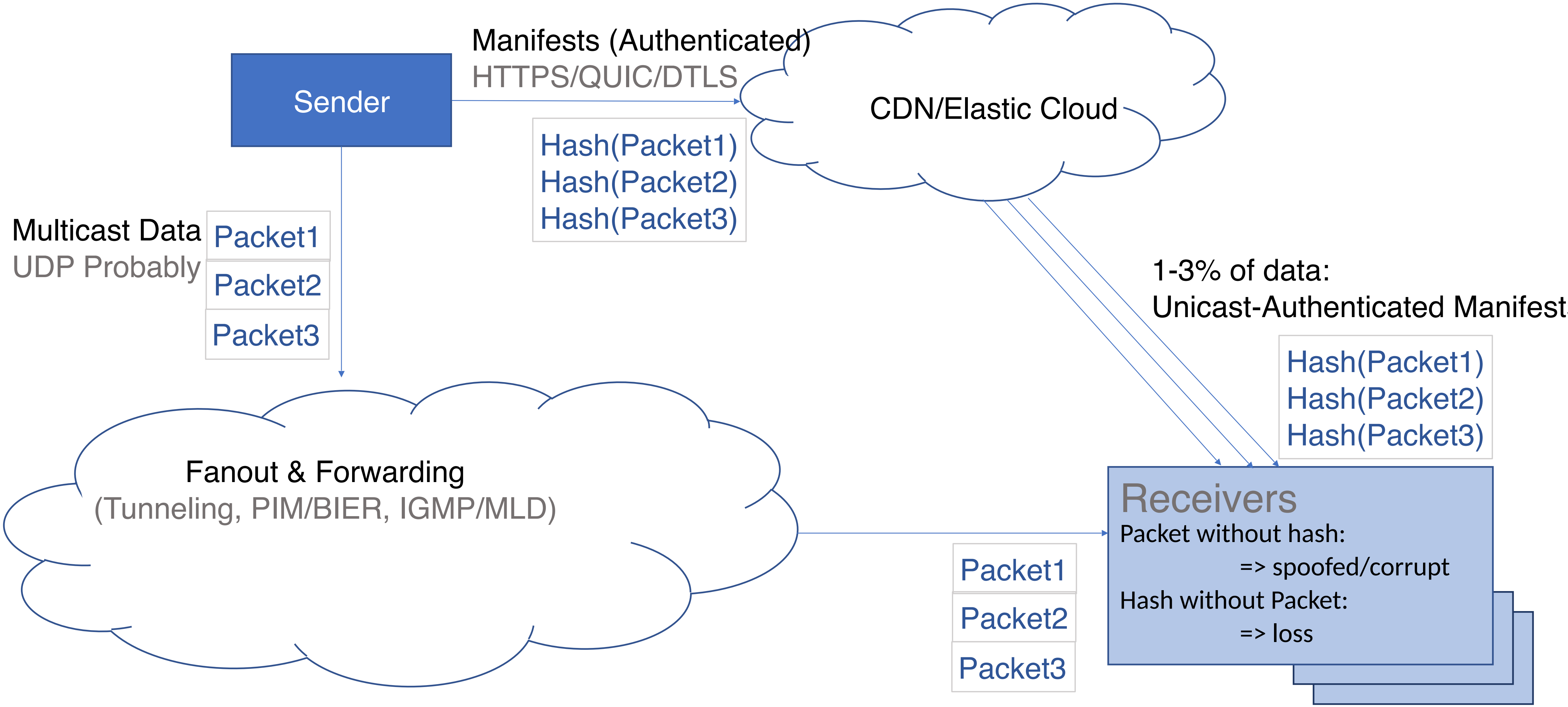
Framers in PyTAPS

- Less callback based than example in interface draft
- Deframes messages when they arrive not when the application calls receive()
- Much more than simple transformation of messages

Multicast in PyTAPS

- We don’t think there have to be any additions to API
- Pull request by Jake to add multicast receive to protocol specific considerations section in implementation
  - Local endpoint: group address
  - Remote endpoint: empty in ASM, source in SSM
  - Calling initiate results in InitiateError
  - Call listen to send join
  - Works like this in PyTAPS

AMBI (Asymmetric Manifest-Based Integrity)



Sender

Manifests (Authenticated)
HTTPS/QUIC/DTLS

Hash(Packet1)
Hash(Packet2)
Hash(Packet3)

CDN/Elastic Cloud

Multicast Data
UDP Probably

Packet1
Packet2
Packet3

1-3% of data:
Unicast-Authenticated Manifest

Hash(Packet1)
Hash(Packet2)
Hash(Packet3)

Fanout & Forwarding
(Tunneling, PIM/BIER, IGMP/MLD)

Packet1
Packet2
Packet3

Receivers

Packet without hash:
=> spoofed/corrupt

Hash without Packet:
=> loss

Multicast + Framers = AMBI

- AMBI is a way to verify integrity of multicast (draft-jholland-mboned-ambi)
- Uses secure connection to send manifests of multicast packets
- Implementation as a framer:
    - Create secure connection in framer on start
    - Receive manifests on that connection
    - When multicast packets arrive „deframe“ them by creating hash and checking against manifest
    - If its the same return unchanged message to application

Multicast + Framers = AMBI

- Is this a legitimate use of framers?
  - API text only talks about encapsulation and encoding, this is neither
  - Arch mentions only data translation
  - Impl also only mentions simple transformation
- Make text more consistent across docs
- If it is a legitimate framer also extend framer definition to include it more clearly

Framers on listen

- Framer object gets added on the preconnection
- Listen spawns multiple connections, only holds one instance of framer object
- In case of AMBI this is useful, but in general each connection might want its own framer instance
- Add section to impl that explains how to do this

Links

- https://github.com/fg-inet/python-asyncio-taps
- https://github.com/GrumpyOldTroll/ambi