The bridge to possible

Real-Time Media in a Cloud-Native World

Giles Heron, Principal Engineer
@gilesheron

cisco Live!

BRKETI-2006

Cisco Webex App

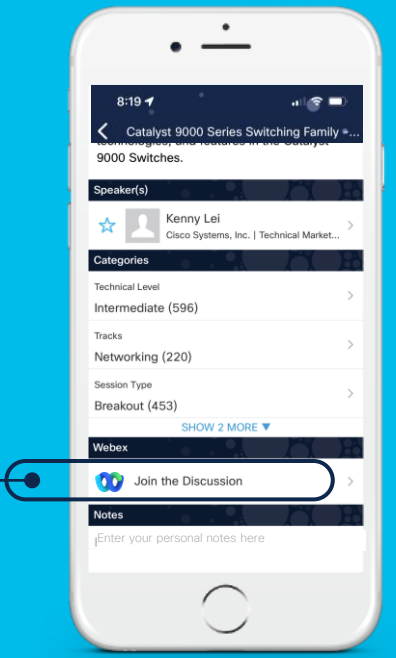
Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

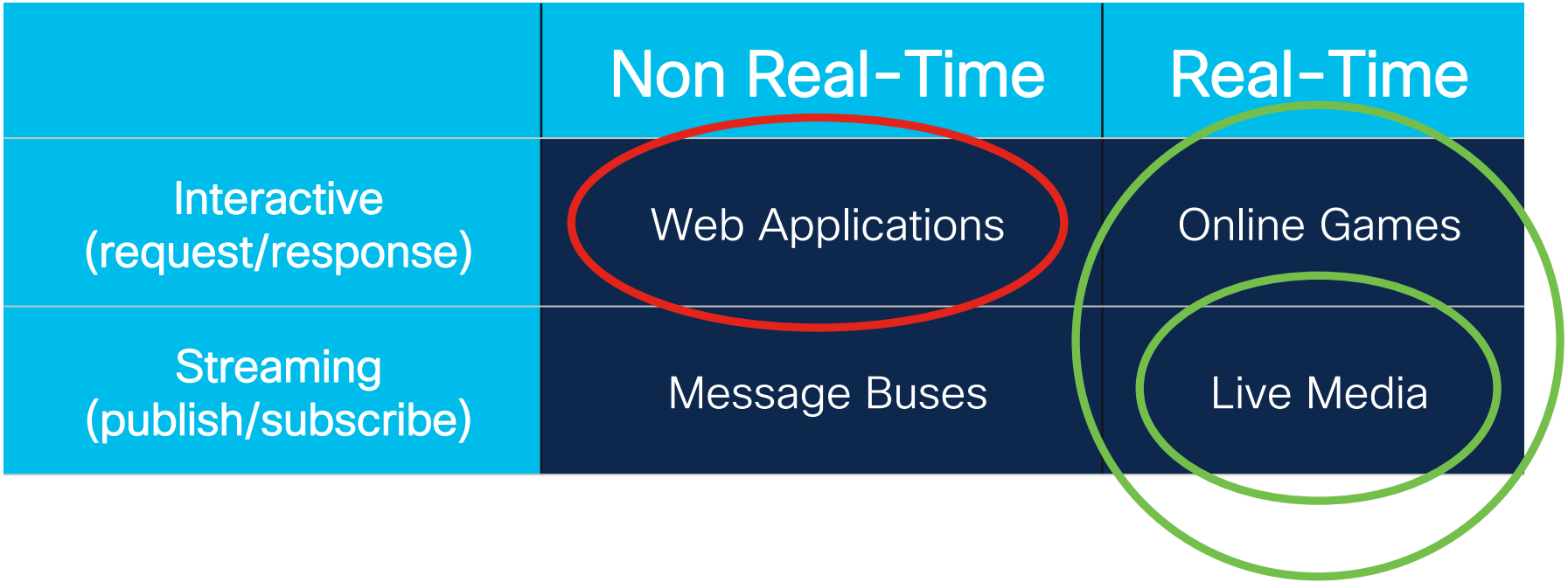
Webex spaces will be moderated until February 24, 2023.



Agenda

- Motivation
- Benefits
- Use-Cases
- Architecture
- Demo
- Call to Action

A (fuzzy) Application Taxonomy



Kubernetes Media Connectivity Options

	TCP	HTTP	UDP	RTP-based media (RTSP, SIP, etc.)
Service Mesh & Load Balancers	★	★★	⊘	⊘
Kube Proxy & NodePort	★	★	★	⊘
Host Networking	★	★	★	★
Media Streaming Mesh	★	★	★	★★

One Pod Per Node!!!

Benefits of Media Streaming Mesh

Observability

Media Streaming Mesh monitors jitter and packet loss across the mesh, enabling DevOps teams to quickly locate and resolve connectivity issues.

Low-Latency

The Media Streaming Mesh RTP data plane proxy adds minimal latency, in contrast to web proxies that terminate
TCP connections at each hop.

Security

Media Streaming Mesh authenticates traffic senders using SPIFFE/SPIRE and can encrypt traffic using SRTP. Proxies reduce attack surface and ensure protocol conformance.

Deployability

Lightweight per-node data plane proxy, and per-cluster control plane proxy ensures a much lower footprint than per-pod web proxies, making it suitable for deployment at the edge.

Use-Cases for Media Streaming Mesh

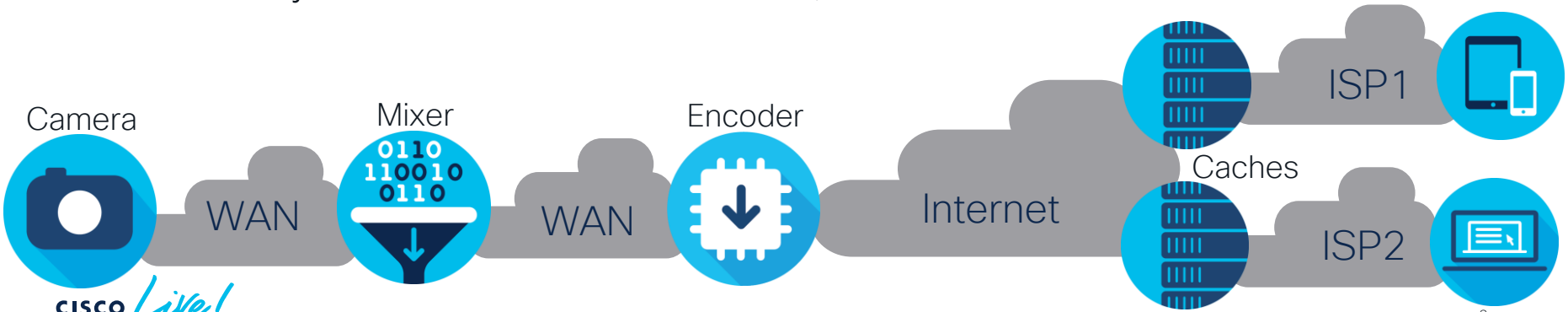
- Contribution video (high bandwidth, RTP-based, no tolerance for loss/jitter)
- Live video distribution (RTP-based, low tolerance for loss/jitter)
- Retail/Industrial Edge (need low footprint, lots of RTSP video streaming/analytics)
- Real-Time Collaboration (WebRTC etc.)

Previously – no longer in focus (though potential scope to leverage RTP here):

- Gaming (latency is key so “action” games use UDP rather than TCP)
- Finance (high frequency trading etc. - latency critical)
- Mobile (GTP is tunnelled over UDP – and is latency sensitive)

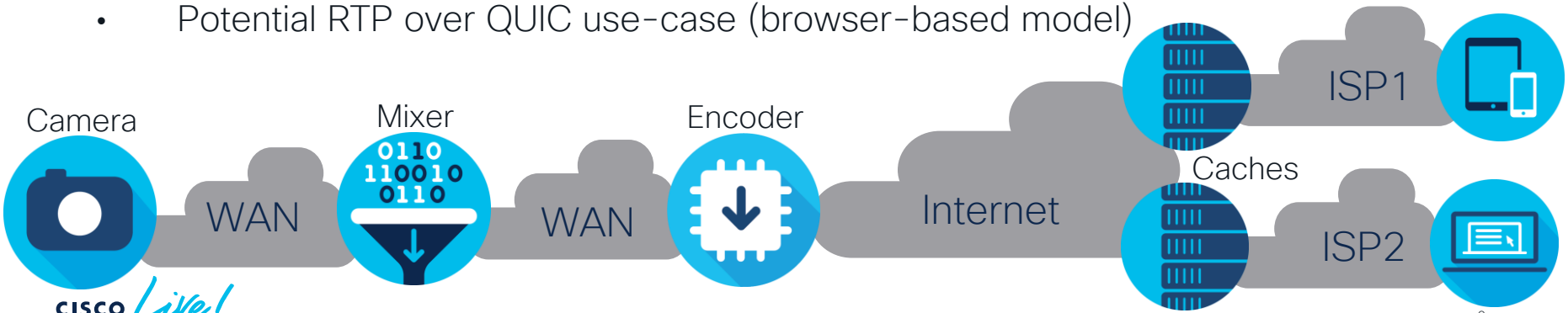
The Live Video Chain

1. Camera feeds (SDI or IP)
2. Mixing (generating a broadcast quality feed)
3. Encoding into multiple formats (resolutions, bitrates, protection…)
4. Distribution to CDN caches (diverse unicast feeds?)
5. Delivery to End users over DASH, HLS etc.



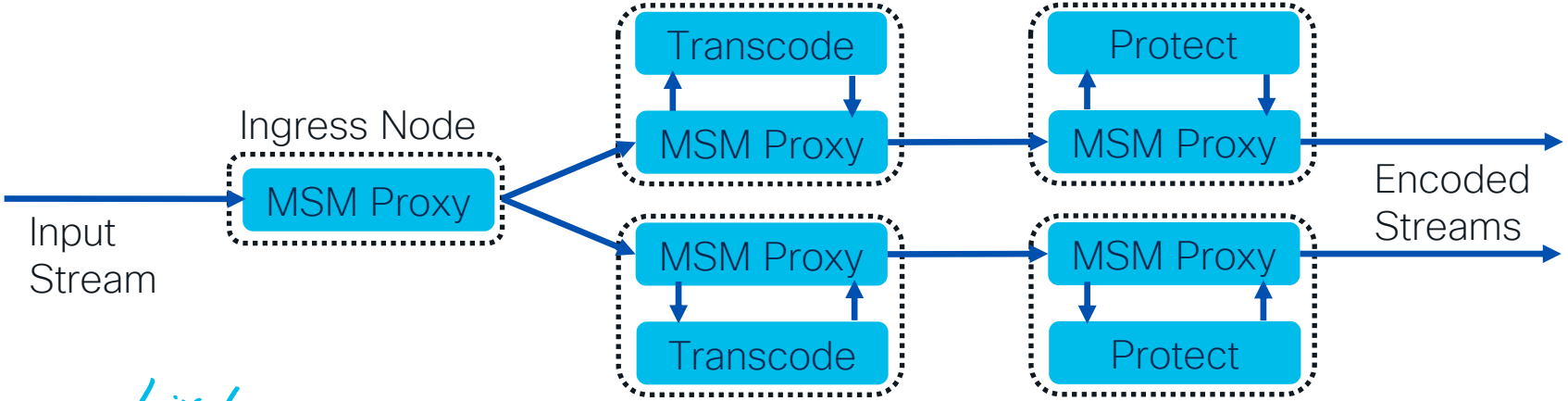
Live Video Use-Cases for Media Streaming Mesh

1. Contribution video (camera to studio and in-studio mixing)
   - Longer-term goal perhaps as cameras/mixers are dedicated hardware platforms
2. Interconnection of cloud-based encoders
   - Most likely an intra-cluster Kubernetes use-case
3. Distributing live streams from encoders to caches
   - Proxies handle fan out, and can add FEC, send dual streams over dual paths etc.
4. Streaming RTP to clients
   - Potential RTP over QUIC use-case (browser-based model)



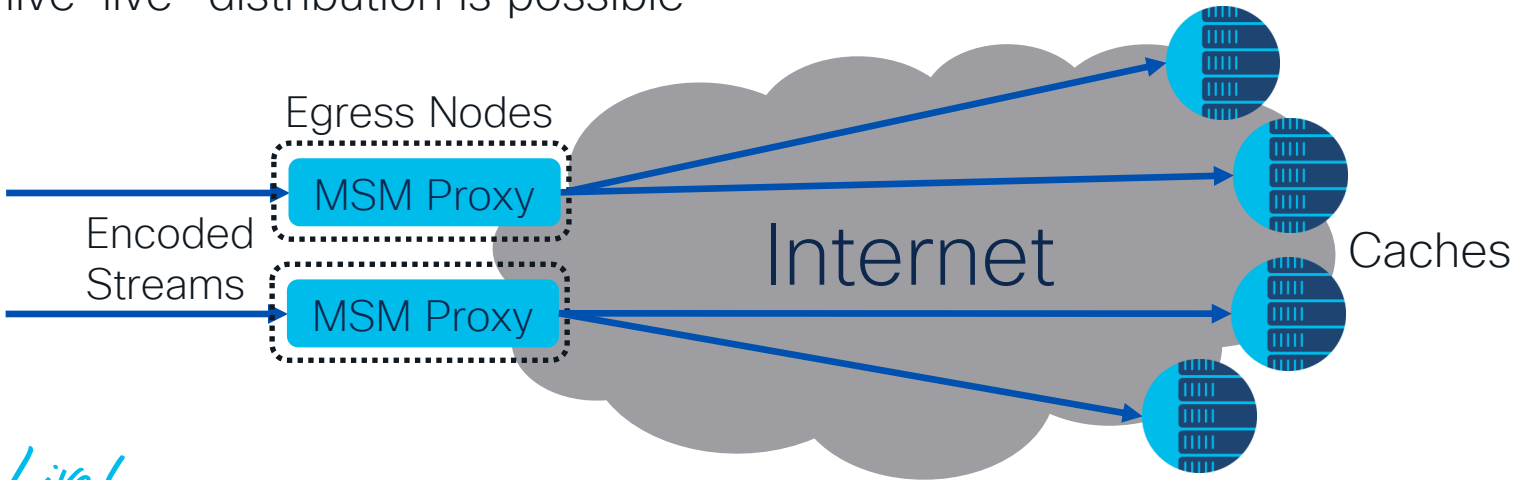
Interconnection of Cloud-Based Encoders

- We assume that for one input stream we may wish to:
  - Create multiple lower resolution / bitrate streams
  - Add content protection
- Deployment model is a single K8S cluster for multiple input streams
  - The same cluster can be used for distribution towards caches



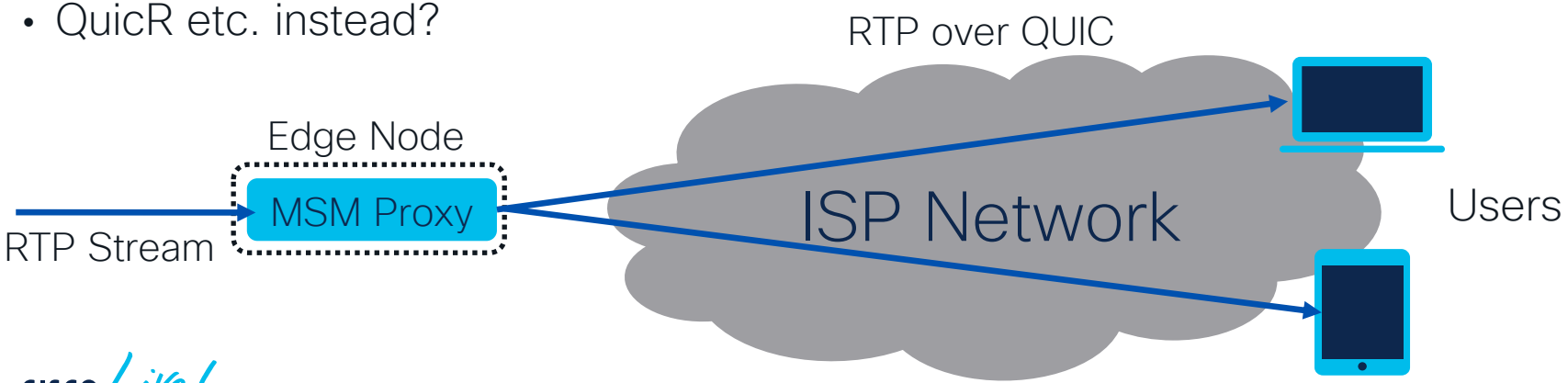
Distributing Live Streams from Encoders to Caches

- Egress node has one or more MSM proxies and can ”pull” any stream
- MSM proxy can replicate towards multiple caches
  - Can add FEC to streams from egress nodes towards caches
  - Caches could also use MSM at ingress to remove FEC etc.
  - ”live-live” distribution is possible



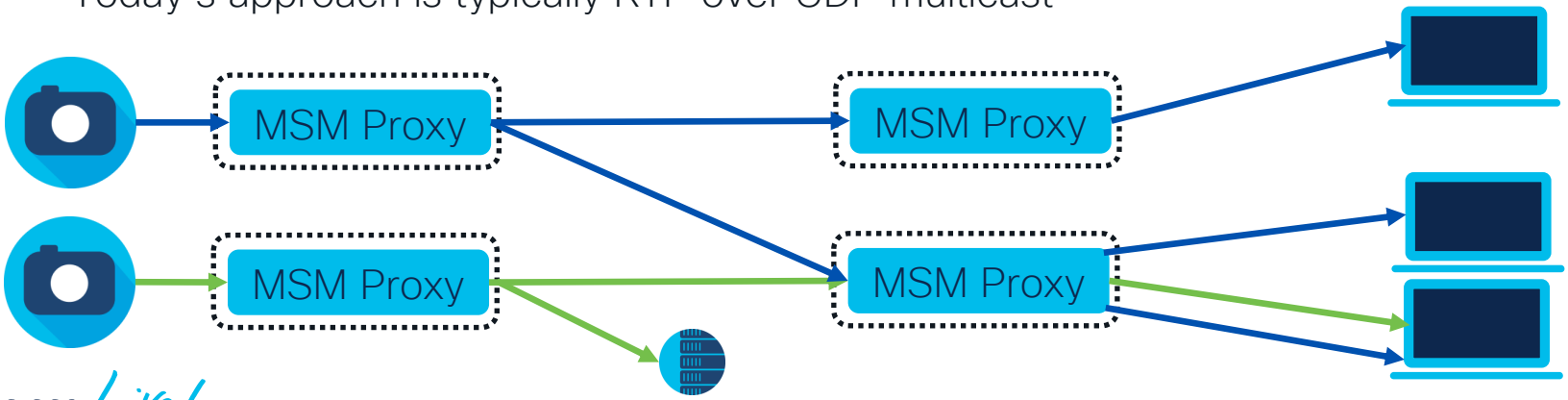
Streaming RTP over QUIC to Clients

- Edge proxy translates from RTP (over UDP) to RTP over QUIC
- Can use FEC, Live-Live etc. to optimise delivery to edge node
- Fan-out from proxy to multiple users
- Filter-based architecture enables plugging in congestion control algorithms
- Modified control plane required (negotiate flow IDs, not pairs of UDP ports)
- QuicR etc. instead?

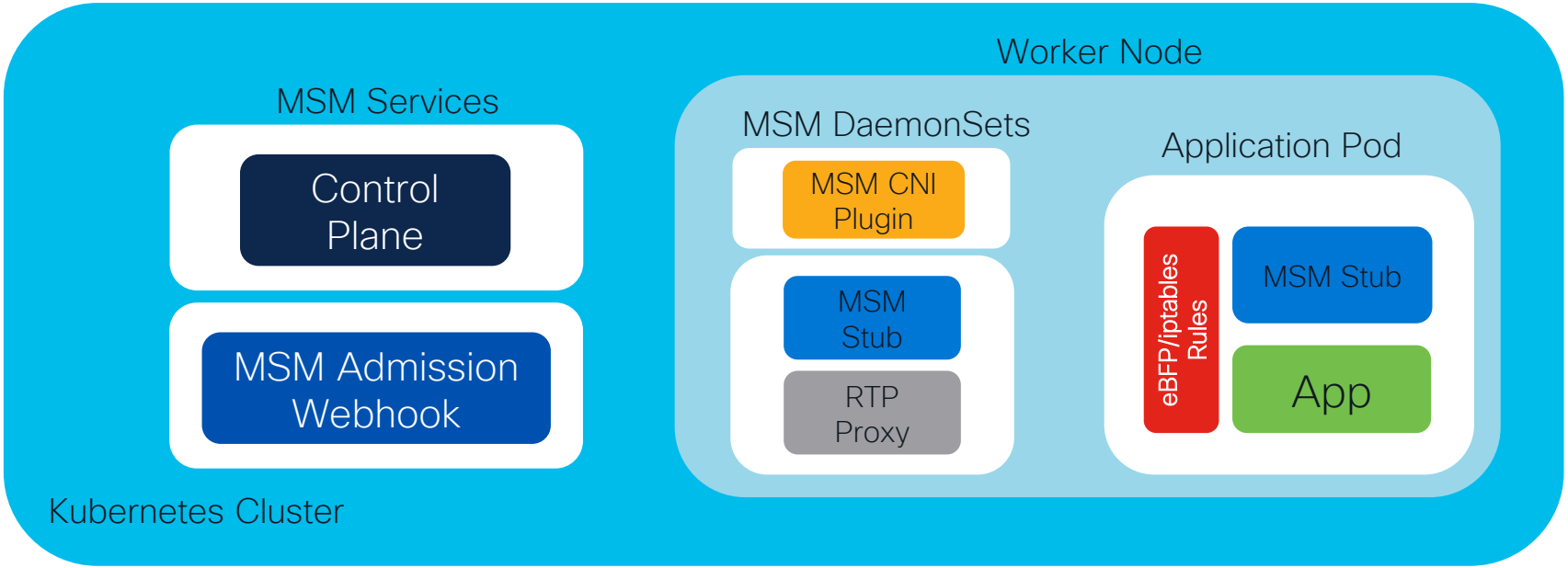


Video Monitoring Use-Cases for MSM

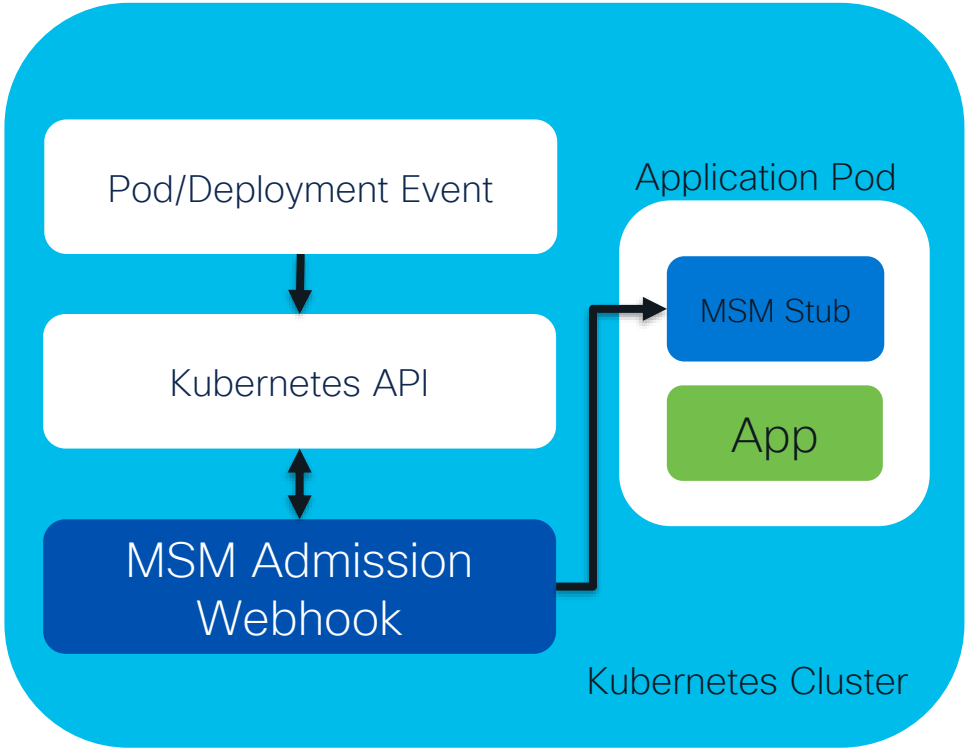
- Large number of cameras
  - few per site in many small sites (e.g. retail)
  - large numbers in a few big sites (airports, factories etc.)
- Multiple viewers – probably remote from the camera locations
- One or more proxies per camera site and a proxy at each viewer site
  - Today's approach is typically RTP over UDP multicast



MSM Software Architecture

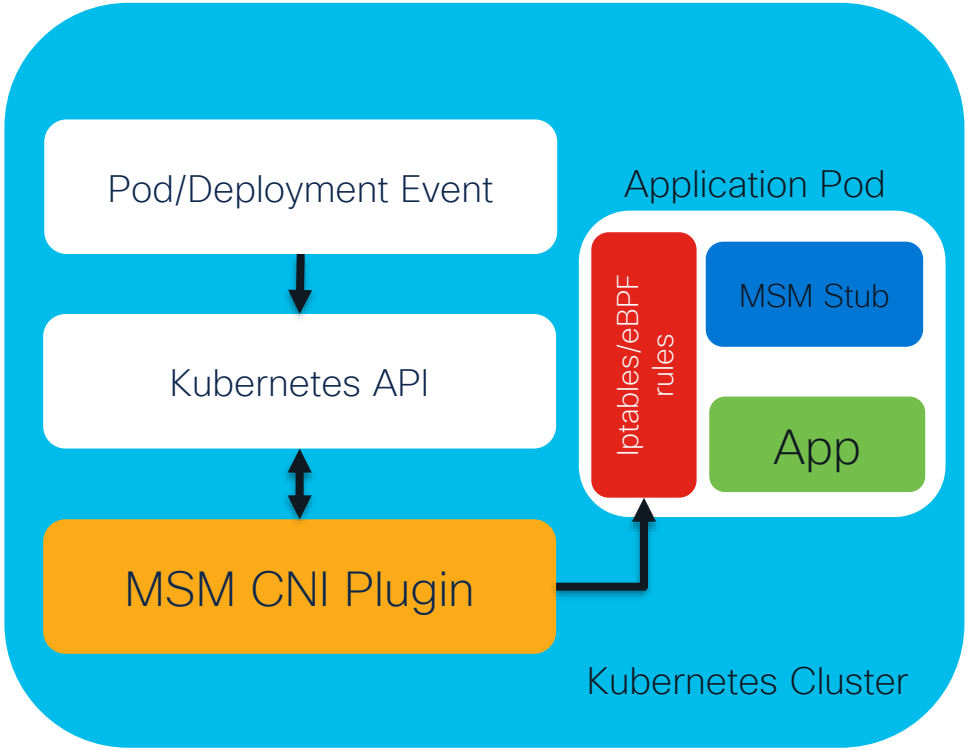


MSM Admission Webhook



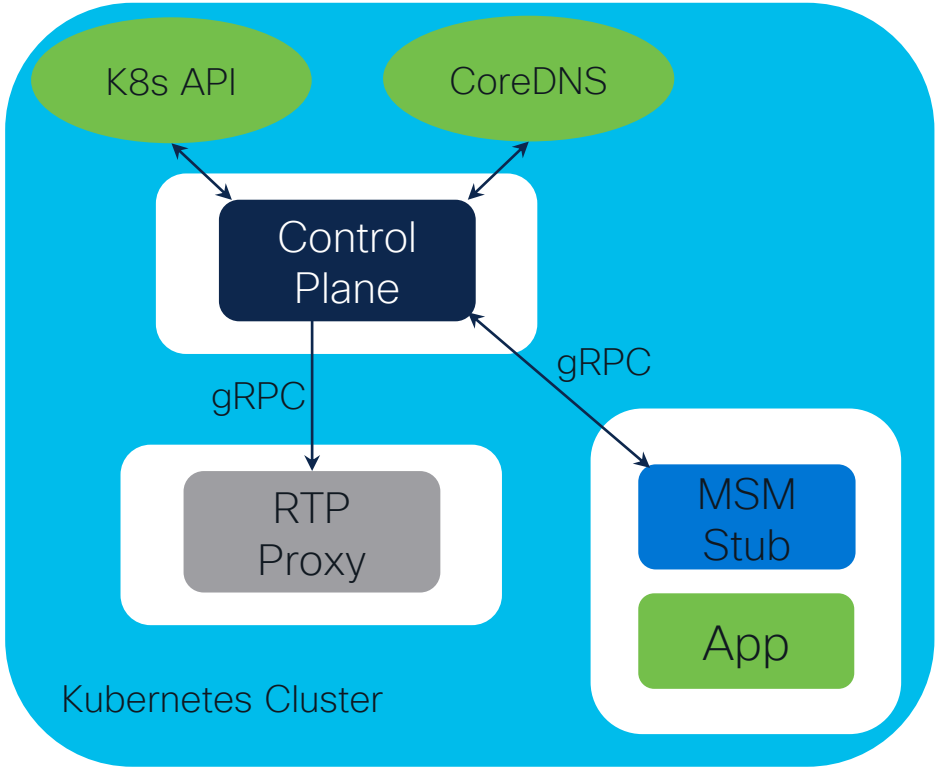
- Admission Webhooks are HTTP callbacks that receive admission requests and do something with them
- MSM implementation uses mutating admission webhooks to automatically inject the MSM stub (sidecar proxy) into an application pod
- Implemented an Admission Controller that listens for Pod/Deployment events and modifies the pod spec on runtime
- Controller is triggered if the pod or deployment specification is annotated with a custom key:
  `"sidecar.mediastreamingmesh.io/inject"="true"`

MSM CNI Plugin



- Any MSM enabled pod will need to redirect incoming traffic to the injected MSM stub
- Works as a chained plugin, meaning that it appends to the existing CNI configuration of a cluster, and its task is to install the appropriate iptables or eBPF rules
- As with the MSM Admission Webhook, it only acts on pods annotated with our custom key: `"sidecar.mediastreamingmesh.io/inject"="true"`
- Uses netns commands on an MSM enabled pod and runs on every node in the cluster

Per-Cluster Control Plane



Deploys as a Kubernetes Service
- Likely 3 replicas with RAFT etc.

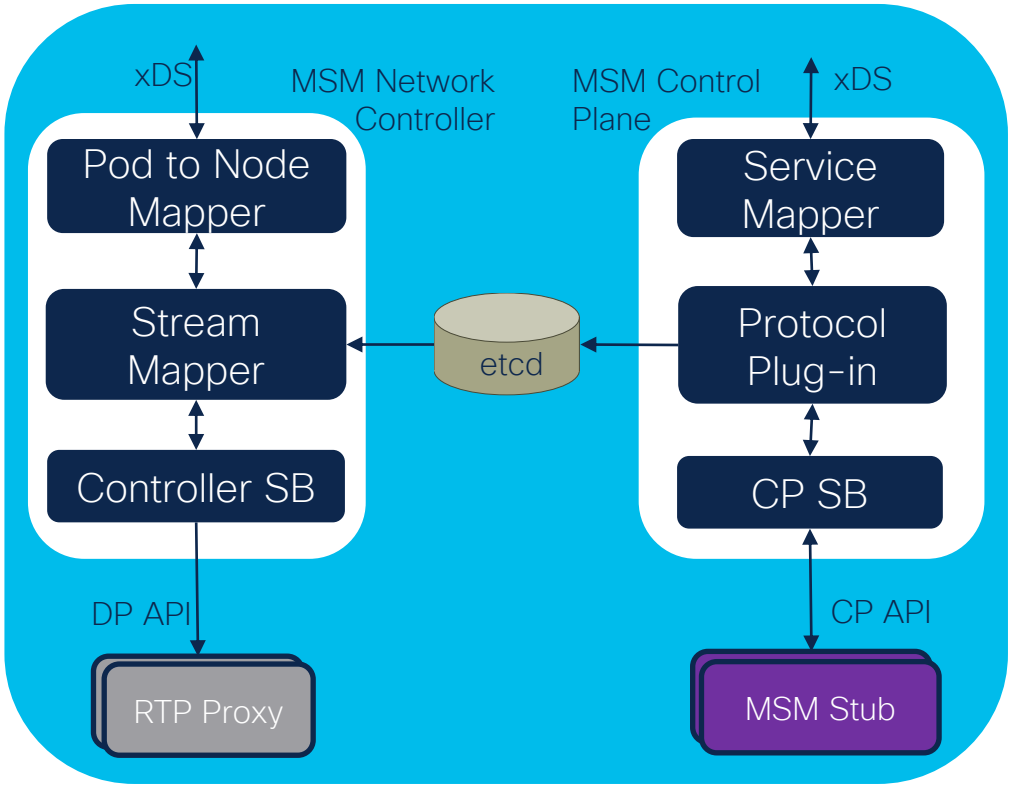
Uses gRPC Southbound:
- send/receive commands to/from MSM Stubs
- program the RTP Proxies

Uses K8s API and DNS Northbound
- In future use xDS interface here?

Written in Golang
- Leverage existing media libraries

L7 protocols implemented as plug-ins:
- RTSP
- WebRTC
- RIST
- SIP
- Others?

MSM Control Plane – Future Architecture



MSM Control Plane is K8s service

- Only required for call-setup and keepalives
- Runs a goroutine for each stream segment
- Segment is from CP to endpoint (via stub)
- Maintains svc mapping from URL/URI to pod
- Writes logical stream graphs to etcd cluster

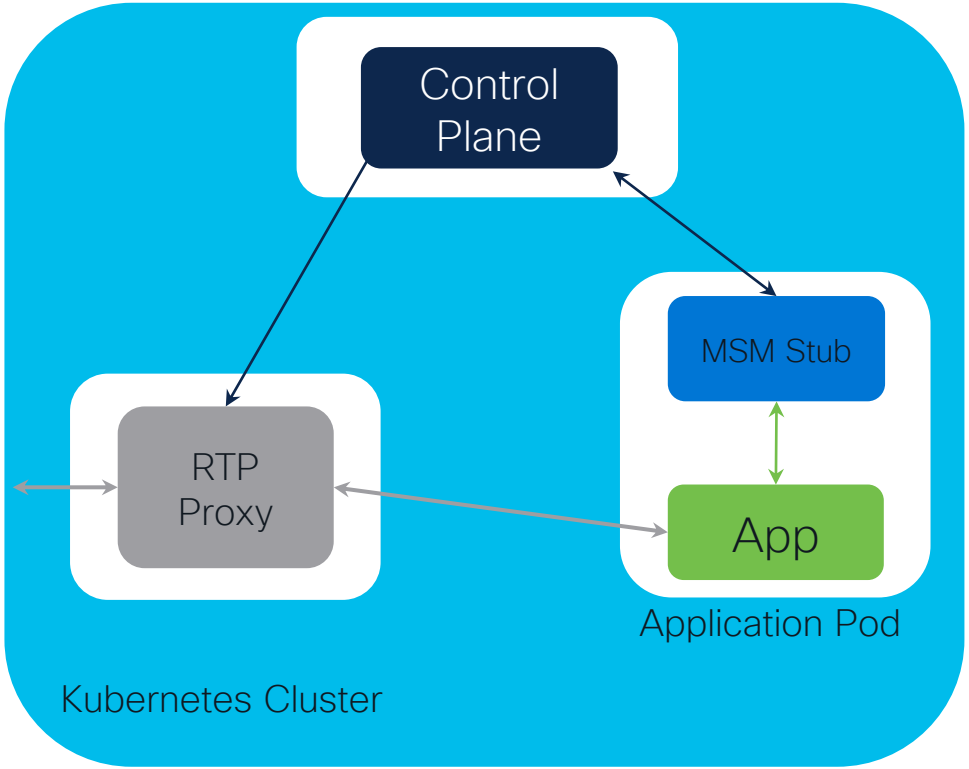
Network Controller is K8s service

- Maintains Pod to Node mapping (from xDS)
- Reads logical stream graphs from etcd cluster
- Creates physical stream graphs
- Pushes stream segments to RTP proxies
- Supports multiple control planes

Use xDS to externalise K8s dependencies

- URL to pod IP mapping
- Pod CIDR to node mapping

The MSM Stub



Deployed:

1. In each application pod that uses MSM
2. With RTP Proxy as "gateway"

Terminates App Control Plane

- Punts to per-cluster CP over gRPC

May intercept the Data Plane

- RTSP interleaved data case
- Monitoring at pod
- "Live-Live" replication/de-duplication?

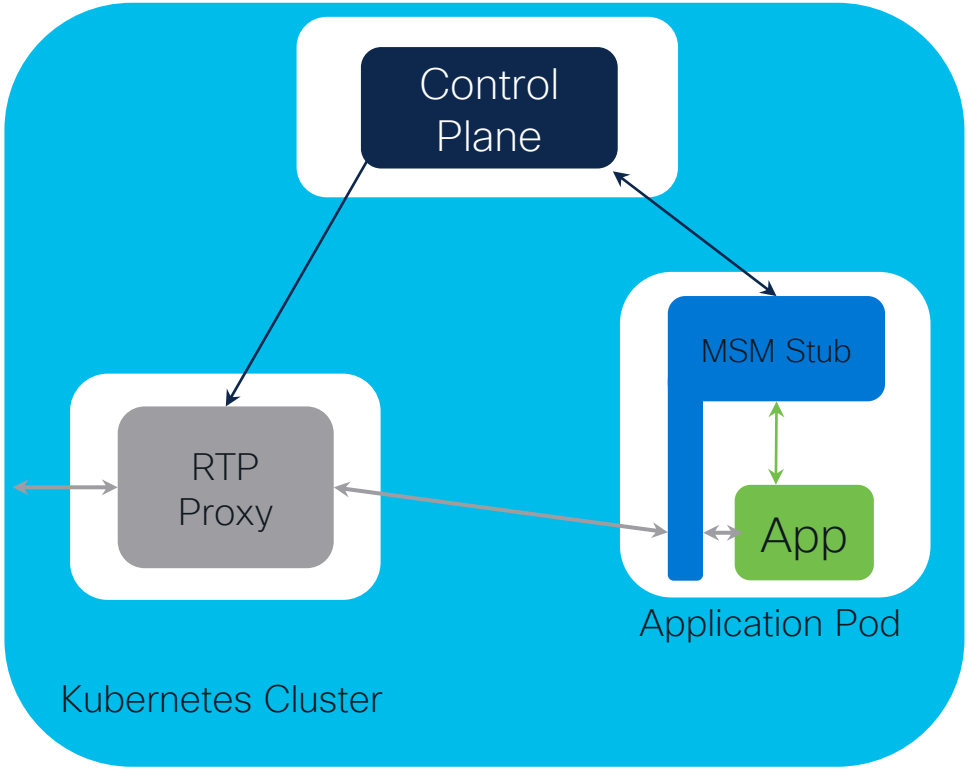
Potentially uses SPIFFE/SPIRE

- RTP Proxy does RTP/SRTP, will need end-to-end authentication

"Stub" because footprint is minimal

- Complexity is in the CP and the RTP Proxy
- Written in Rust to reduce footprint/latency

The MSM Stub



Deployed:

1. In each application pod that uses MSM
2. With RTP Proxy as "gateway"

Terminates App Control Plane

- Punts to per-cluster CP over gRPC

May intercept the Data Plane

- RTSP interleaved data case
- Monitoring at pod
- "Live-Live" replication/de-duplication?

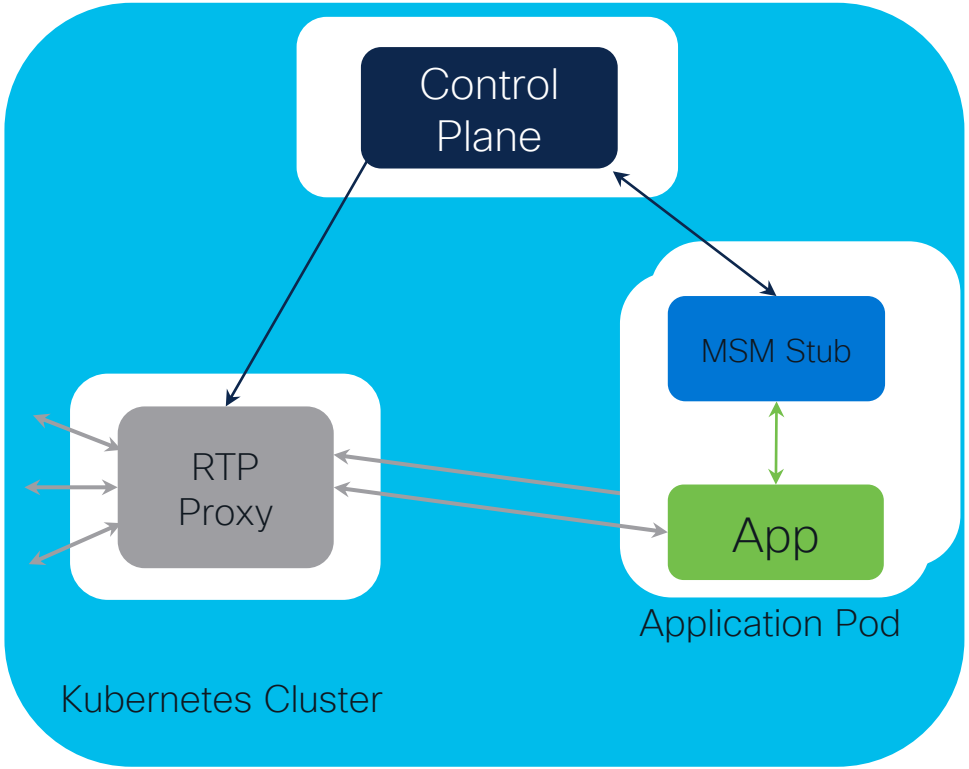
Potentially uses SPIFFE/SPIRE

- RTP Proxy does RTP/SRTP, will need end-to-end authentication

"Stub" because footprint is minimal

- Complexity is in the CP and the RTP Proxy
- Written in Rust to reduce footprint/latency

The RTP Proxy



Deployed as a per-node DaemonSet

- In host network namespace
- supports North/South and East/West flows

RTP Translator (RFC3550)

- Unicast to multicast, IPv4 to IPv6, RFC1918 to public IP, tunnelling, MTU conversion etc.
- RTP/UDP, RTP/TCP and RTP/QUIC support
- Also acts as a UDP, TCP and QUIC proxy
- Minimises attack surface

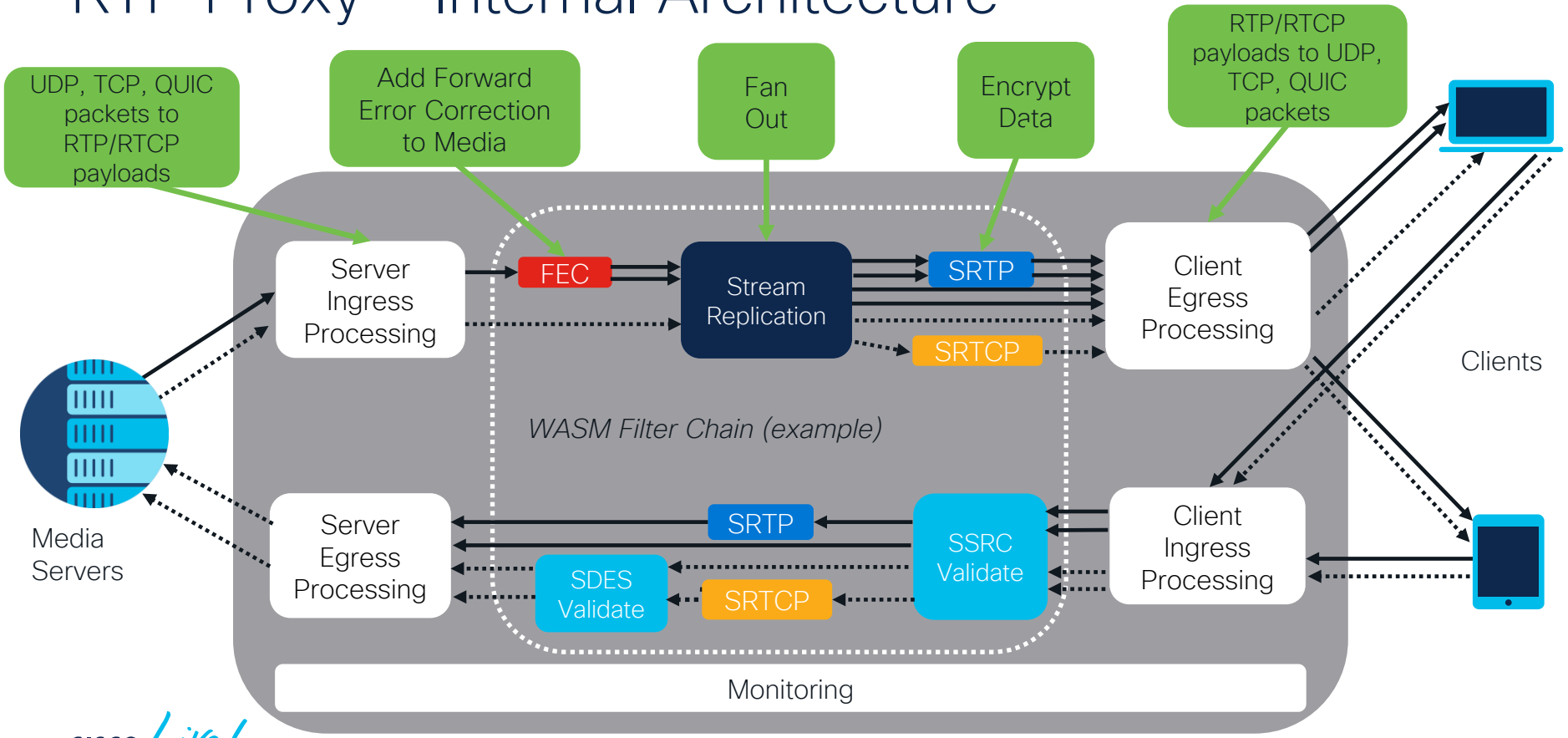
Prototype implementation in Golang

- Performance limited by kernel sockets and GC

Future is Async Rust with WASM filters

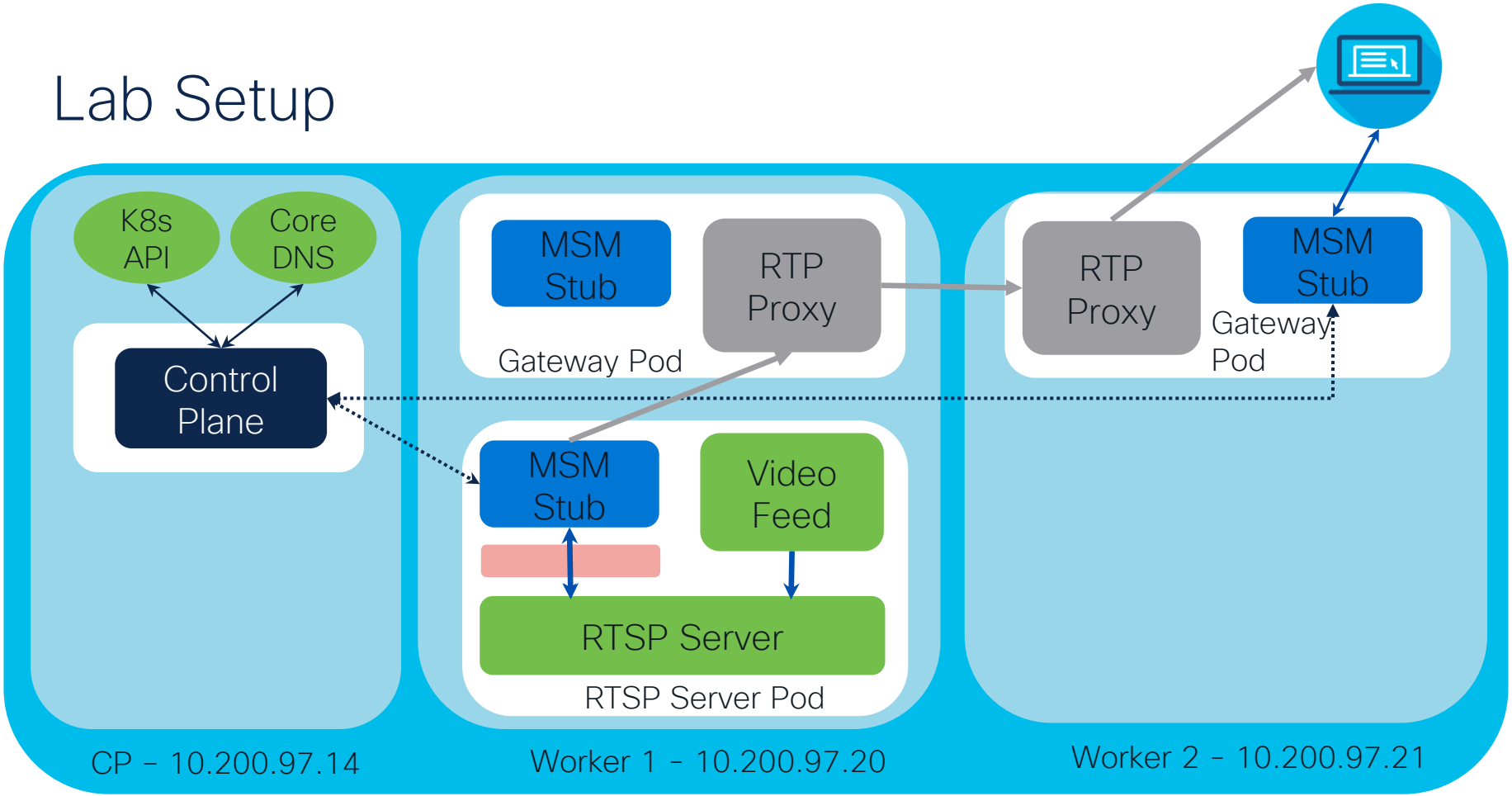
- Validation, replication, encryption, protection, congestion control etc.
- Key is to drive a filter ecosystem

RTP Proxy – Internal Architecture



Demo

Lab Setup



In conclusion

- Media Streaming Mesh enables real-time media applications to be first-class citizens in today's cloud native world
- MSM is a work in progress and is in open-source
  - https://www.mediastreamingmesh.io
  - https://www.github.com/media-streaming-mesh
- Please collaborate with us to make it a success!

Complete your Session Survey

- Please complete your session survey after each session. Your feedback is important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (open from Thursday) to receive your Cisco Live t-shirt.

- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the "Attendee Dashboard" at https://www.ciscolive.com/emea/learn/sessions/session-catalog.html

Continue Your Education

Visit the Cisco Showcase for related demos.

Book your one-on-one Meet the Engineer meeting.

Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.

Visit the On-Demand Library for more sessions at ciscolive.com/on-demand.


cisco
The bridge to possible
Thank you
cisco Live!


CISCO Live!
ALL IN