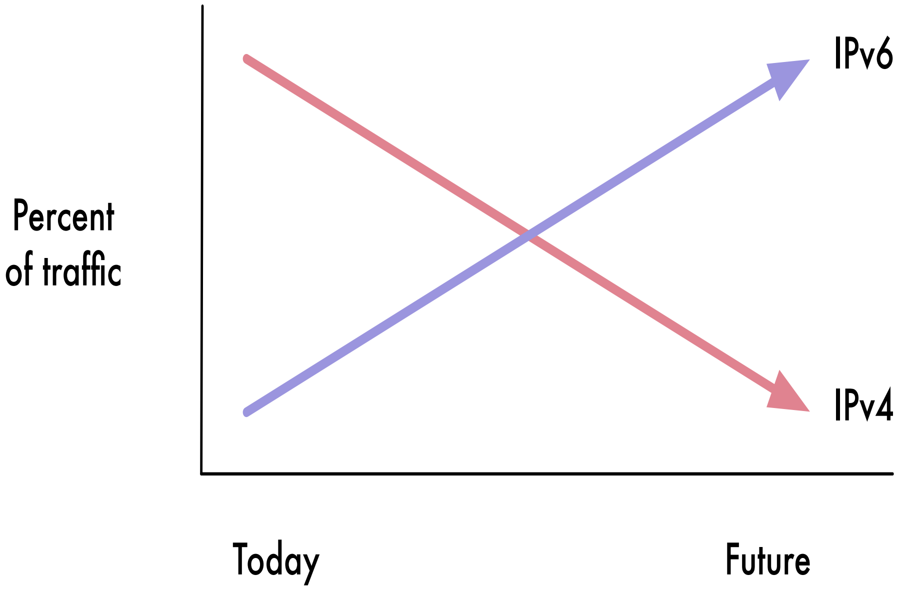
Motivation, Analysis, and Architecture for IPv4aaS
[implemented using cloud infrastructure]

Brian Field
Comcast

Motivation: Eventually IPv4 will taper off



- Much like X.25, FR, ISDN, ATM, etc.
- Hardware and software removed from router platforms
- Does the same apply for IPv4 (eventually?)

An IPv6 optimized network

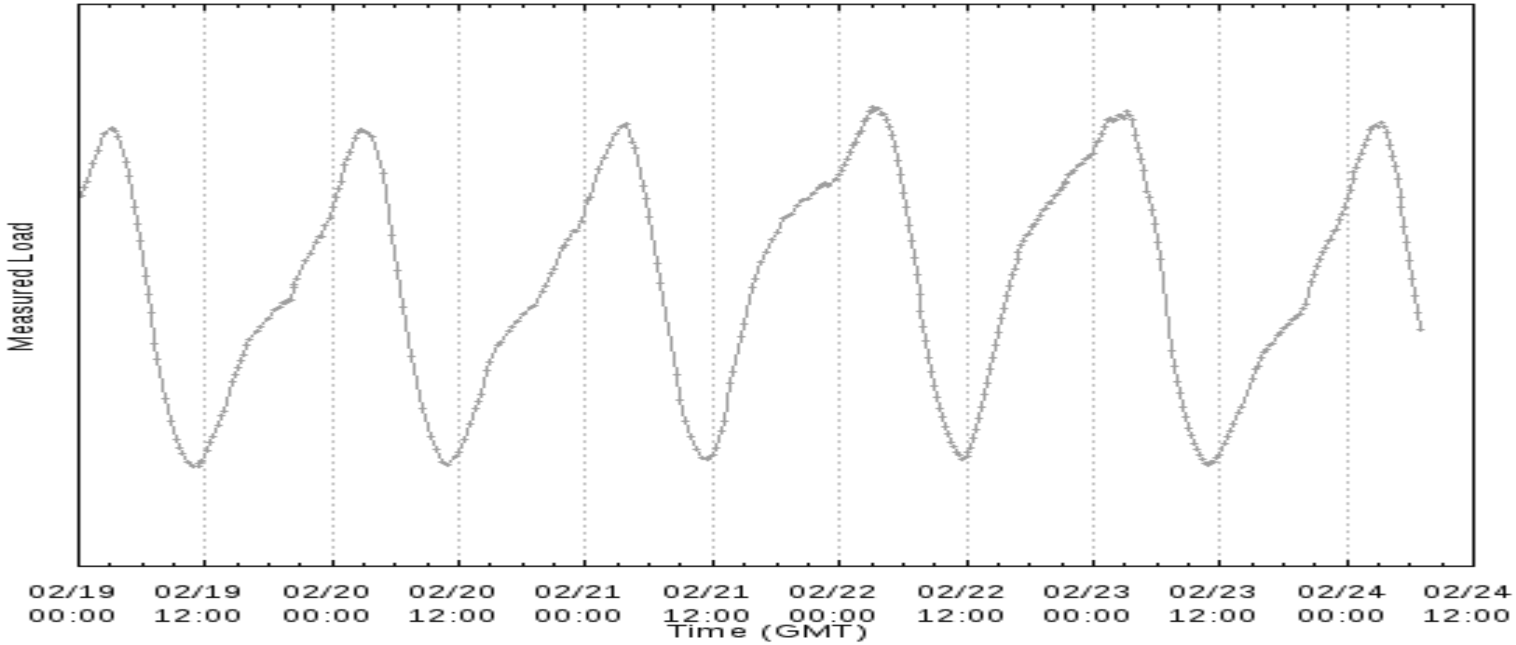
- Does it make sense to think about a next-generation network infrastructure to be IPv6 “optimized”?

- Concepts:
  - Lean Core
  - IPv6 only core
  - Lean IPv6 core

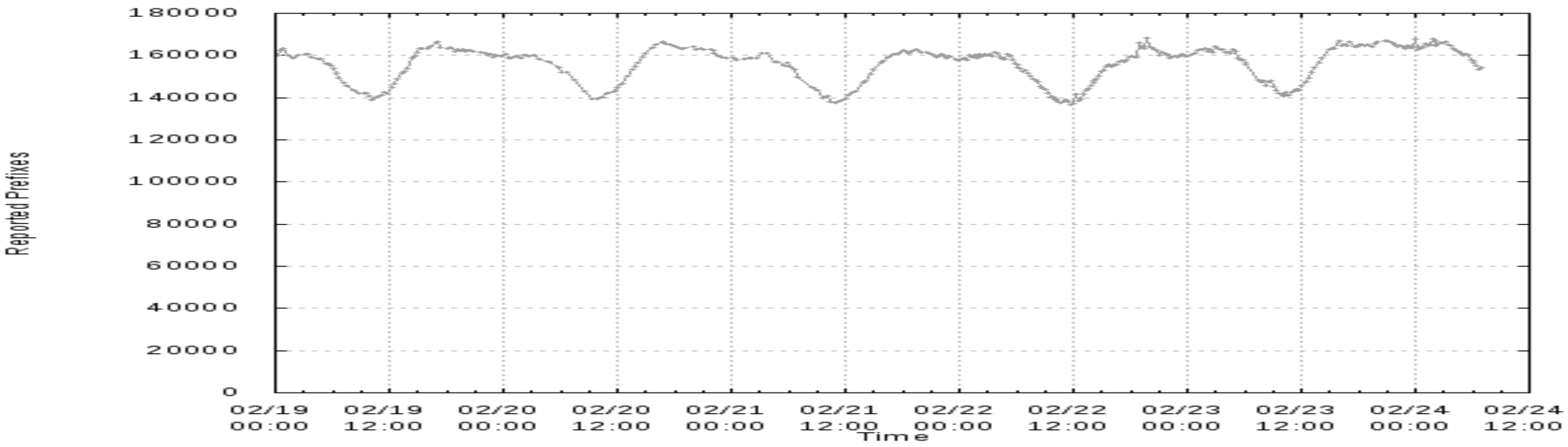
- Benefits to Lean Core (thin FIB)

- How might we handle IPv4 traffic if we start thinking about optimizing our physical infrastructure for IPv6.

Analysis: IPv4 traffic profile today

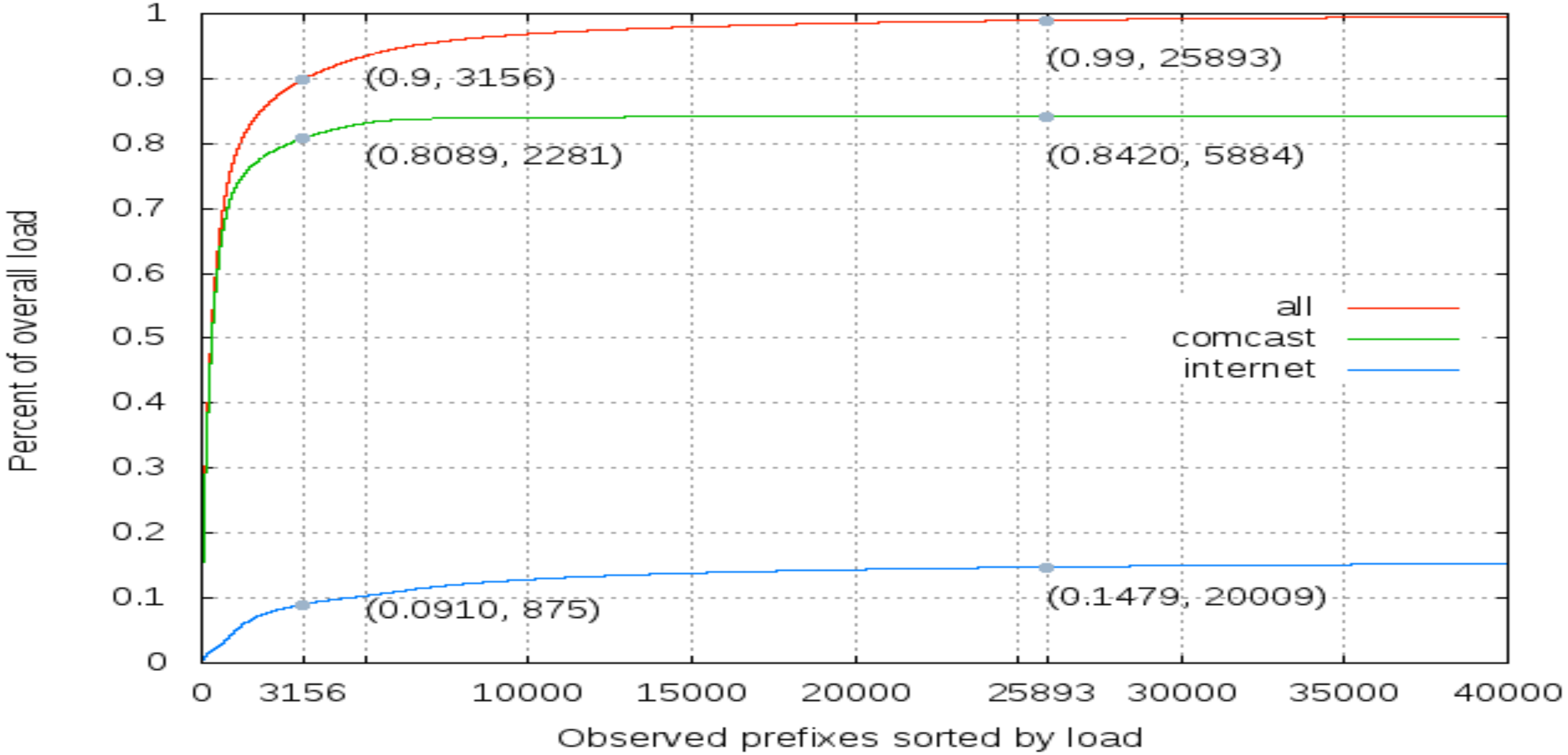


IPv4 prefixes with traffic



- 575k prefixes in the FIB
- 160k prefixes have traffic
- 415k prefixes (72%) with no measureable traffic.

Details on those 160k prefixes



What does this mean?

- Out of 575k IPv4 FIB entries:
  - No traffic to 415k prefixes (72%)
  - Observe traffic for 160k prefixes (28%)

- Out of the 160k prefixes with traffic:
  - 90% of traffic carried by 3156 prefixes (0.005%)
  - 99% of traffic carried by 25893 prefixes (4.5%)

  - 549k prefixes carry 1% of traffic

COMCAST

IPv4aaS is practical today

- 99% of traffic carried by 4.5% of the FIB entries (26k prefixes)

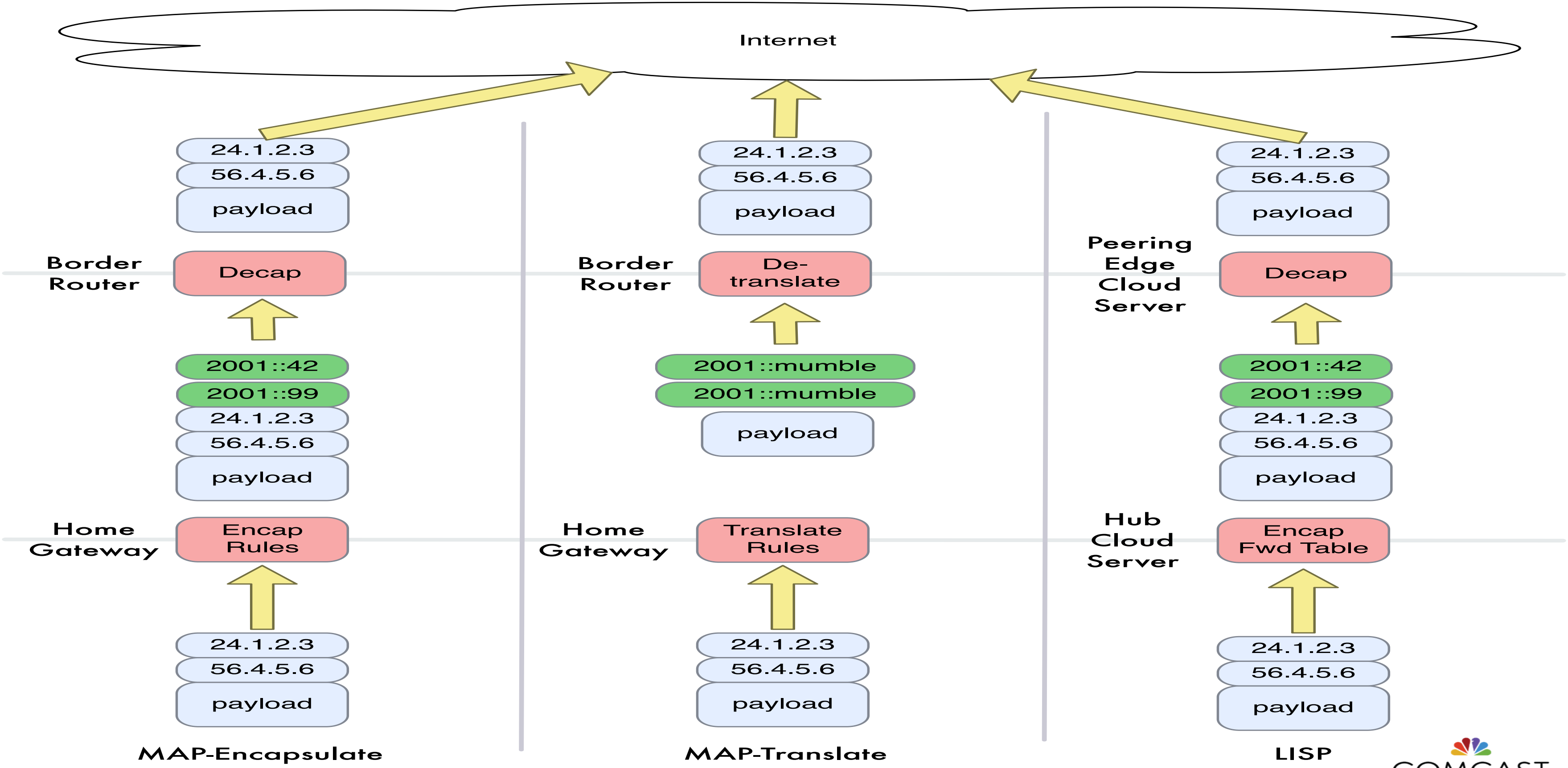
- Which means:
  - 1% of the traffic carried by 95.5% of the FIB entries (549k)

- An IPv4aaS overlay could:
  - Reduce the FIB size drastically.
  - Only need to carry a small amount of traffic.

- Doesn’t help today but allows us to prove out these ideas in advance of requirements for next-generation routing platforms.

- How might we build this IPv4aaS overlay network?

MAP and LISP

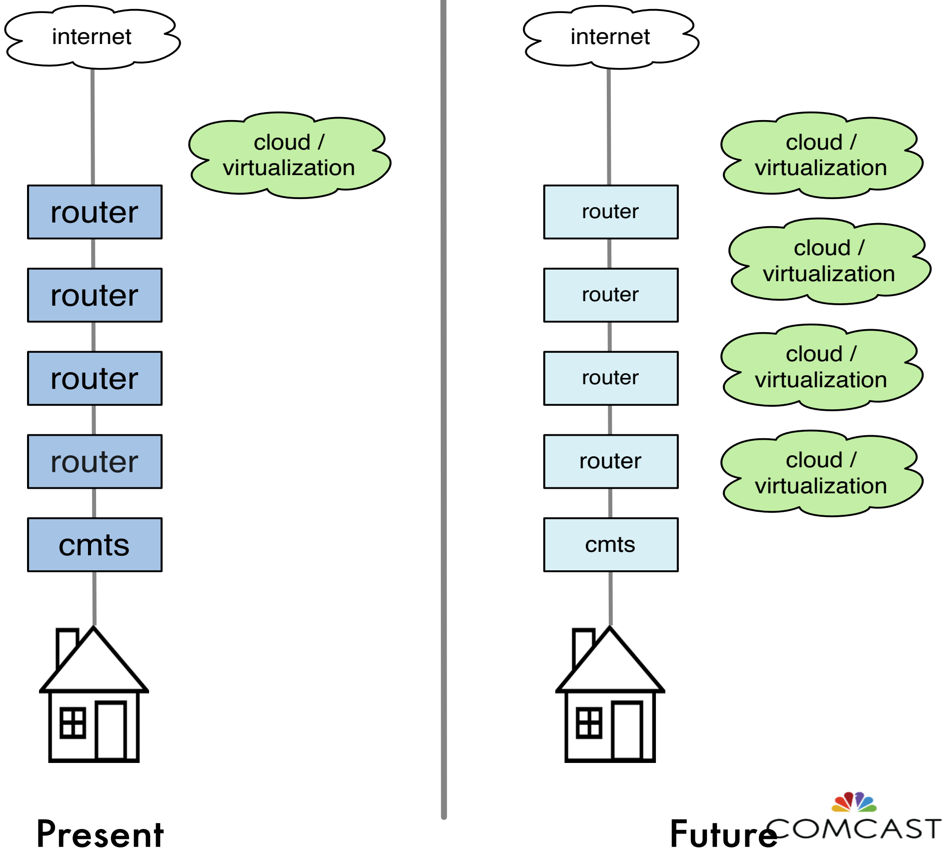


COMCAST

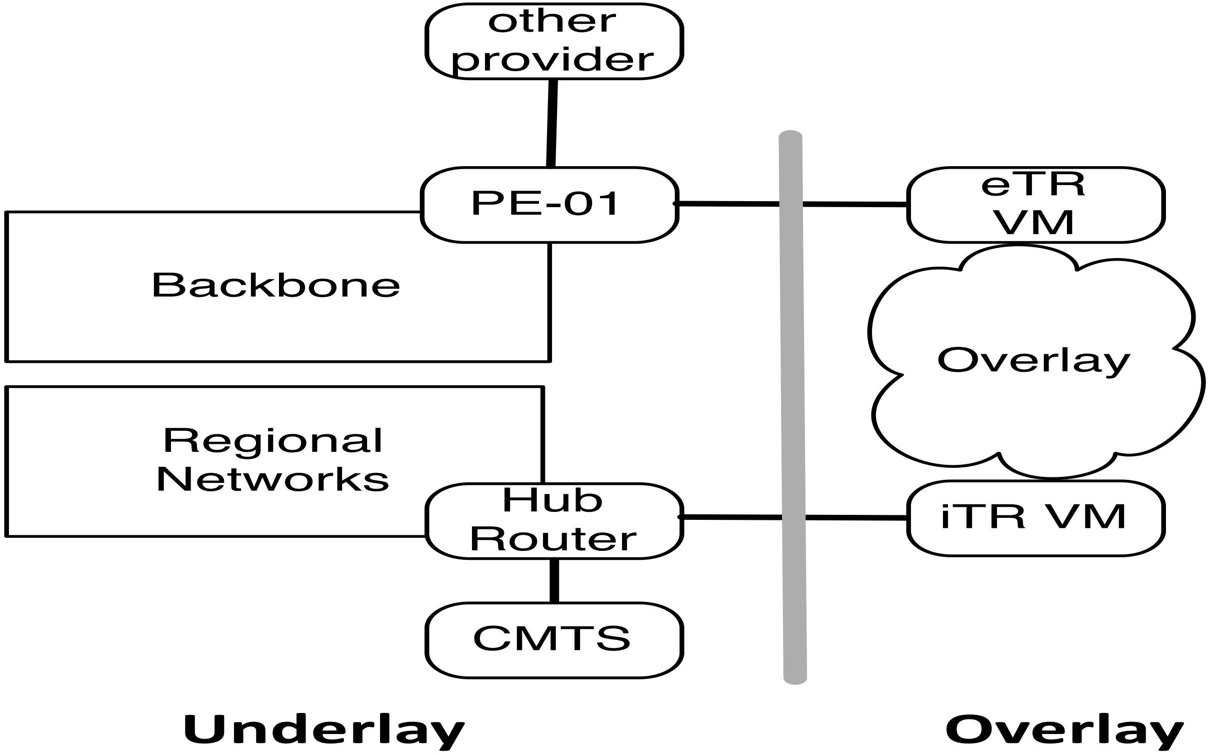
Our Thinking

- LISP encapsulation
- Routing based control plane

- Cloud / virtualization is enabler
- Open source preferred, home grown as needed

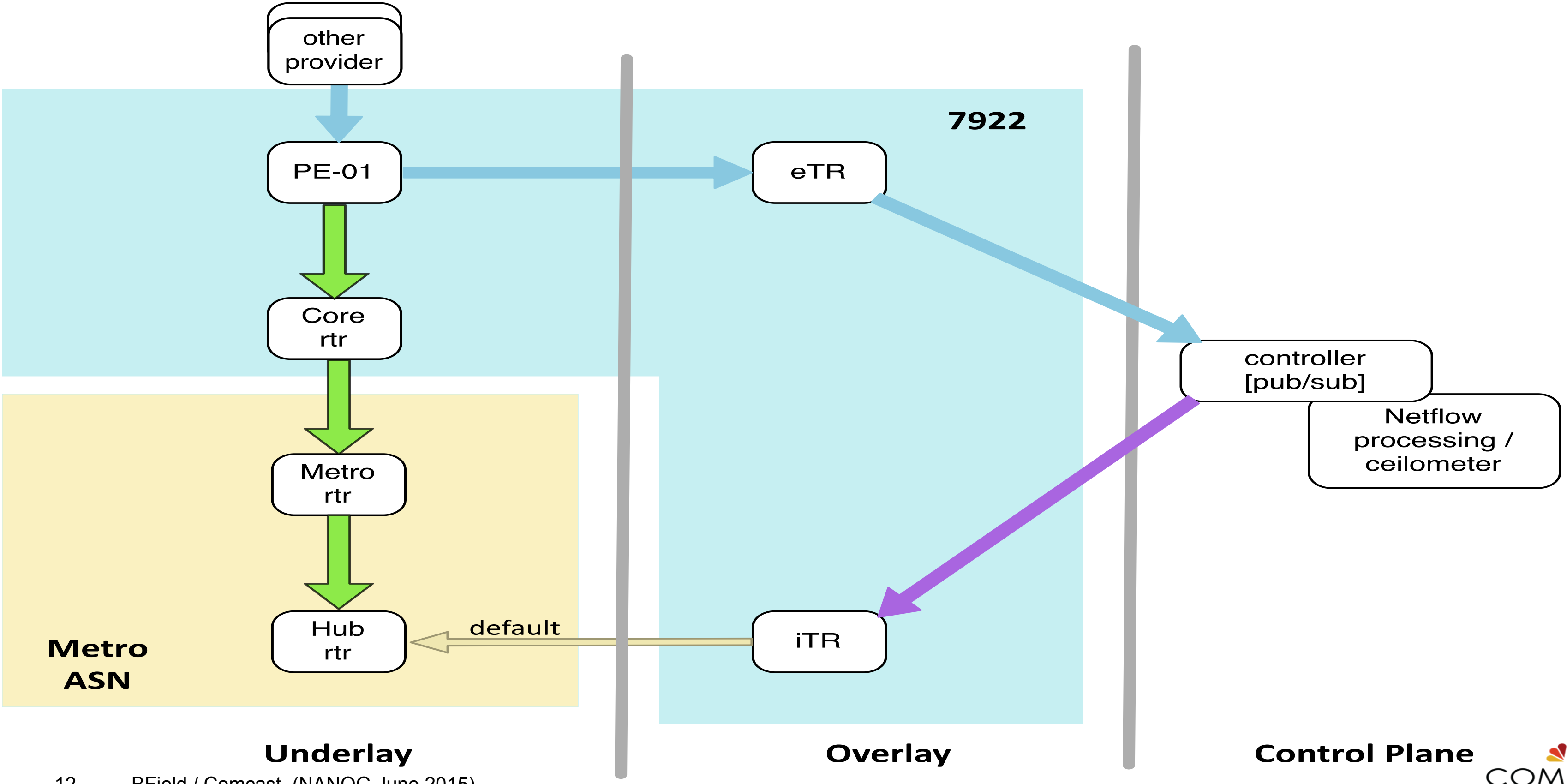


High-level IPv4aaS Architecture



COMCAST

IPv4aaS Routing Architecture

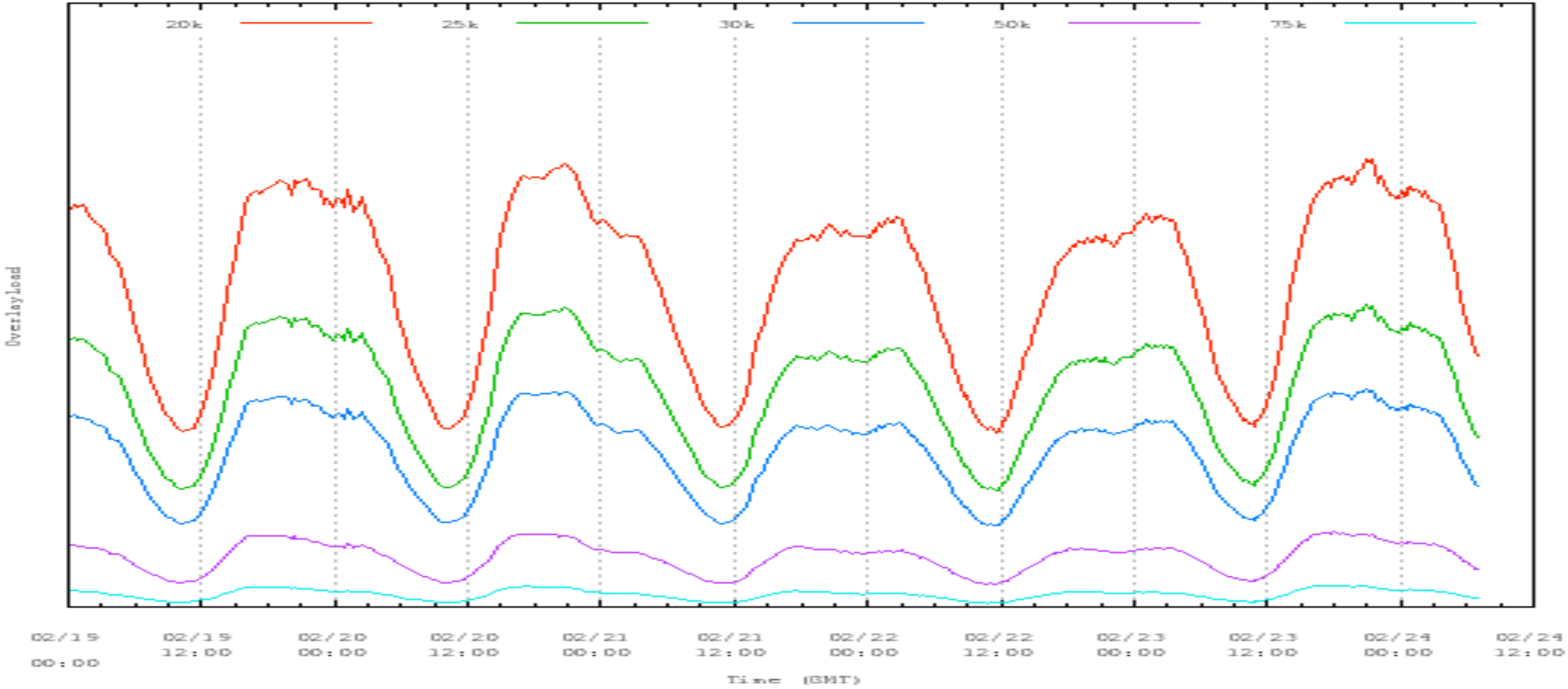


COMCAST

BGP as JSON (with LISP wrapper)

```
{
    "LISP_header" : {
        "RLOC" : "2001:558:fc05:0:f816:3eff:febb:6f65"
    },
    "BGP_msg" : {
        "time" : 1423603076,
        "pid" : "6359",
        "host" : "etr-01",
        "exabgp" : "3.4.8",
        "type" : "update",
        "ppid" : "6358",
        "neighbor" : {
            "asn" : {
                "local" : "65000",
                "peer" : "7922"
            },
            "ip" : "96.119.41.74",
            "message" : {
                "eor" : {
                    "safi" : "unicast",
                    "afi" : "ipv4"
                }
            },
            "address" : {
                "local" : "96.119.41.7",
                "peer" : "96.119.41.74"
            }
        }
    },
    "CP_header" : {
        "Type" : "bgp-message"
    }
}
```

Underlay FIB size (# IPv4 prefixes) and resulting [theoretical] Overlay load



What are we doing?

- Constructing an overlay “IPv4aaS” based on open source plus additions, on cloud infrastructure.

- Beginning the process to transition “services” that used to be implemented in the “underlay” (proprietary closed vendor lock eco-system) into an open source cloud environment.
  - Program the network or make it “lean”?

- Observation:
  - Mainline Openstack does not yet appear ready for all network services

Ongoing eco-system evolution

NANOG 2013 (New Orleans). Applying web principles to the Network	For new BGP "features" or extensions use JSON and HTTP	**Control plane evolution**
NANOG 2014 (Bellevue). Hybrid Routing Platform ("Hopen")	We want router platforms where we can incrementally add ourselves new features to the routing platform	**Legacy platform evolution**
NANOG 2014 (San Francisco). IPv4aaS.	Incrementally tease features from underlay network into open source overlay infrastructure built on cloud / OpenStack. Simplify underlay "incremental white box"	**Simplify Underlay, incrementally transition services to Overlay**

COMCAST

brian_field@cable.comcast.com