HIVE STREAMING

Mikael Högqvist
Senior Research Engineer

ABOUT PEERIALISM

- Founded in 2007 by entrepreneurs and researchers from KTH/SICS
- Mix of business people, developers and researchers
- R&D driven and specialized in P2P products
- Focused on enterprise customers

ERICSSON

Time Warner Cable®

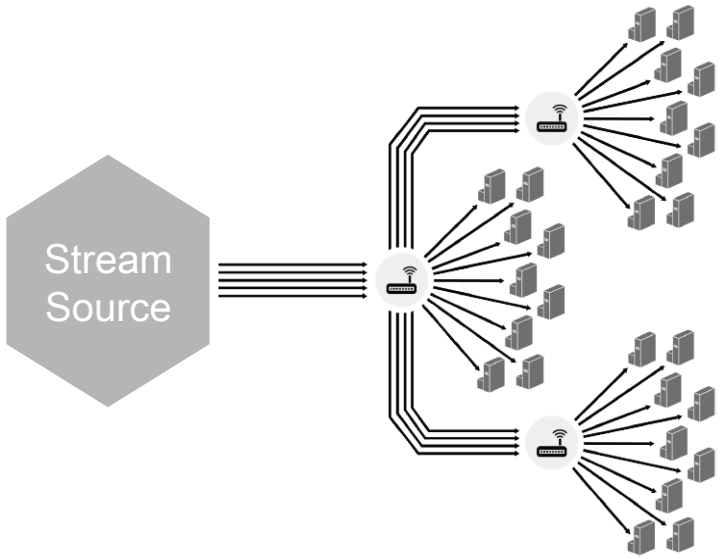
Microsoft

HIVE STREAMING

- **Peer based streaming technology** for public and private networks
- Offloads servers and networks with **97-99%**
- Standalone or as part of Microsoft Azure Media Services
- Live and Video-on-Demand (VoD) content via Adaptive HTTP Streaming

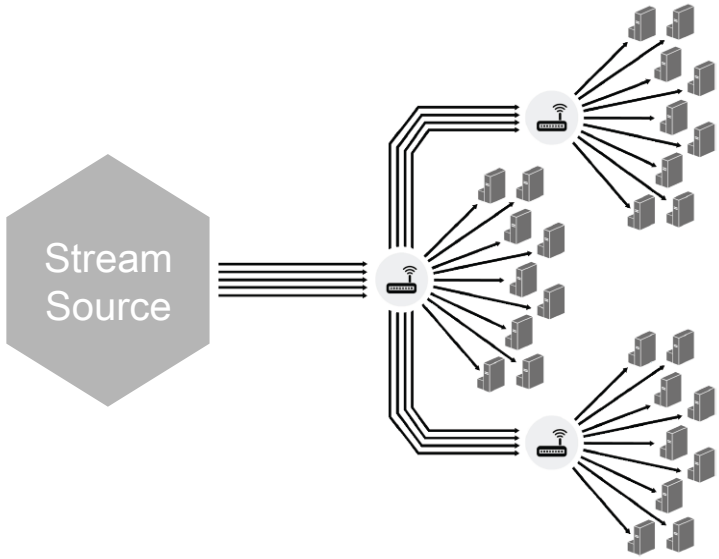
ADAPTIVE HTTP STREAMING

- Video (and audio) is divided into data fragments
- Each video stream can have several bitrates (video quality)
- A manifest describes video metadata, bitrates, fragment location, etc.
- The player decides which bitrate to retrieve based on available bandwidth, rendering capabilities, load on the host
- Fragments are retrieved with a HTTP GET to the source/CDN



THE ENTERPRISE CHALLENGE

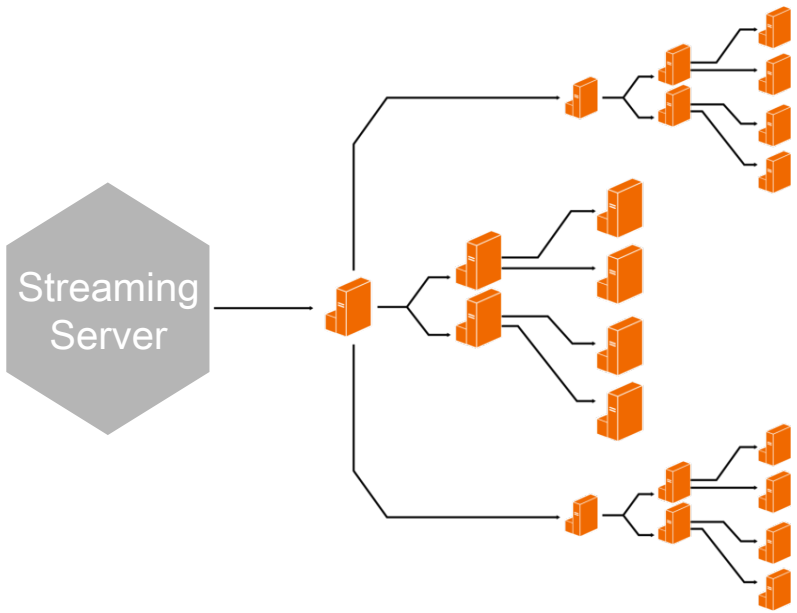
- Communication with **high quality video,** e.g. town-hall events
- Most corporate **networks can't handle the load**
- Existing solutions using hardware are not cost effective



One stream to each viewer causing overloaded servers and networks

HIVE – A SOFTWARE CDN

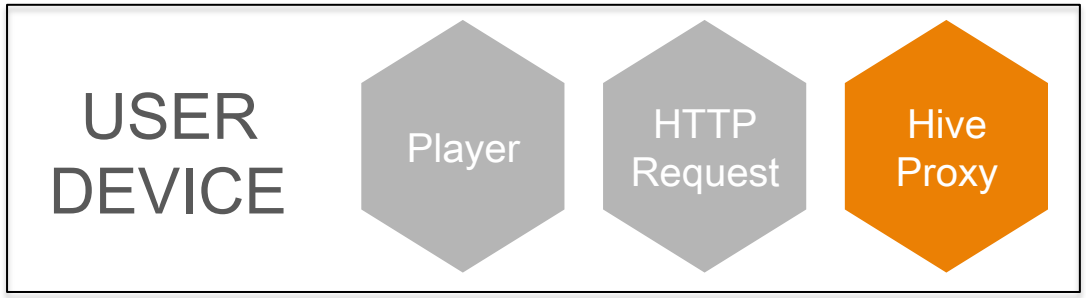
- High-level goals
  - Same Quality of Experience (QoE) as a hardware CDN
  - Efficiency through a single stream per network segment (locality awareness)
  - Support all Adaptive HTTP streaming protocols (Smooth Streaming, HDS, HLS, DASH)



Only one stream per link with HIVE

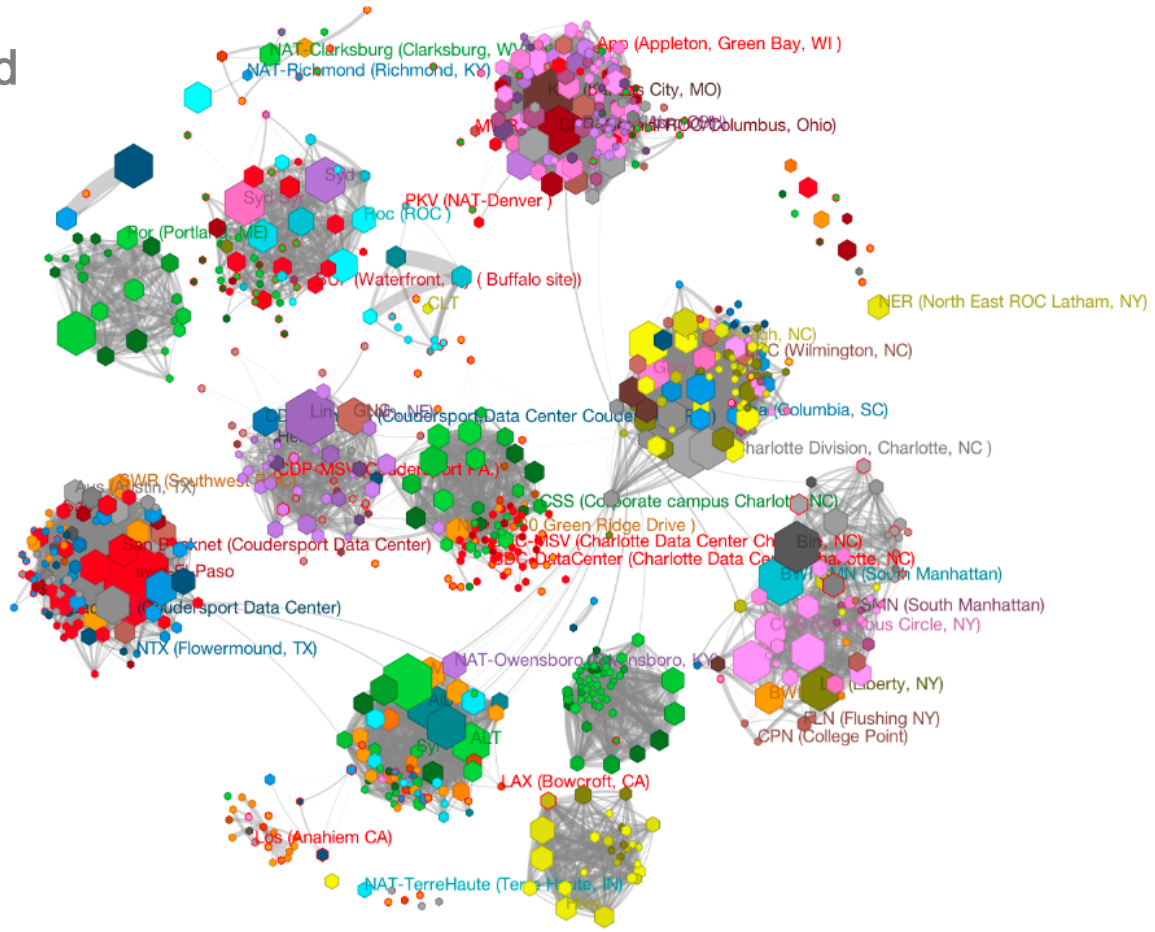
HIVE – ARCHITECTURE

- Each end-user device has a hive agent that intercepts all video player requests
- Hive acts as a distributed cache
  - Agents exchange Have messages indicating fragment availability
  - Cache miss: Retrieve from Source
  - Cache hit: Retrieve from another agent
- Same cache abstraction for both Live and VoD
  - Live data only available for a short time (~30 seconds)
  - VoD based on device storage capability
- Efficiency is determined by overlay construction



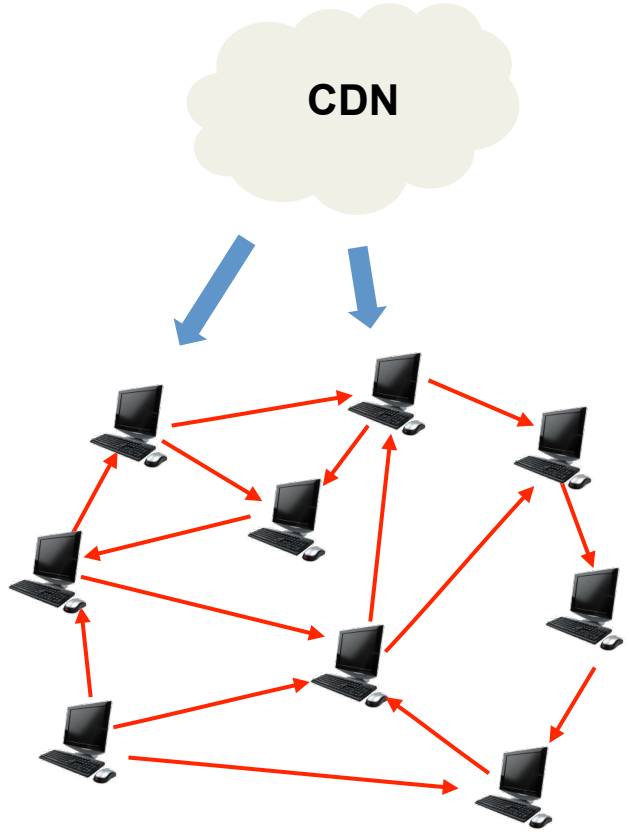
OVERLAY CONSTRUCTION – CHALLENGES

- Quality of user experience
  - Real-time requirements with hard deadlines, e.g. deliver fragment within 2 seconds
  - Quickly react to bitrate switches
- How to cope with
  - Limited bandwidth capacity
  - Network congestion
  - Churn
  - NAT/Firewalls
- Topology optimization
  - Content Locality
  - Bandwidth



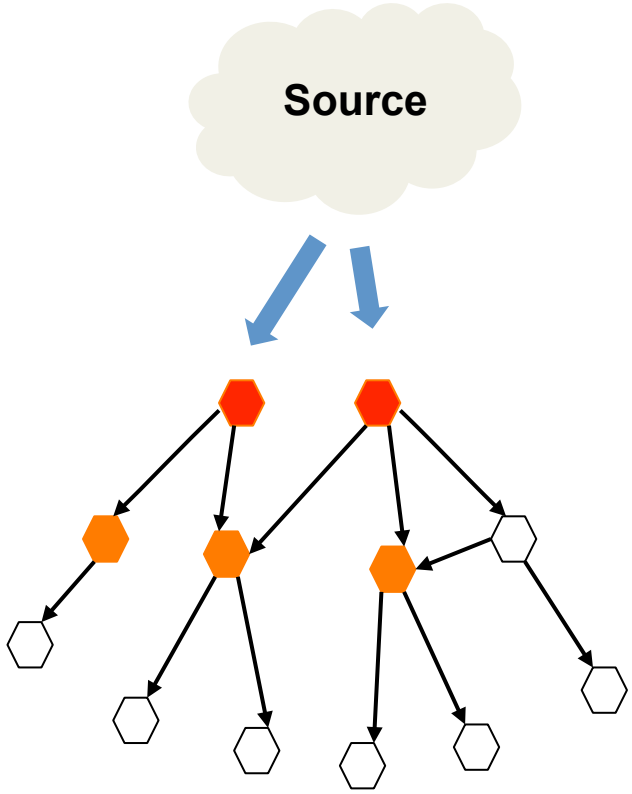
OVERLAY – PUBLIC NETWORKS

- Public networks
  - Several devices behind a home router (NAT)
  - Each device (network) is connected to the Internet via ISP
- Challenges
  - Heterogeneous device capacity and bandwidth
  - Detect the network structure, i.e. avoid traffic across ISPs/ countries/continents



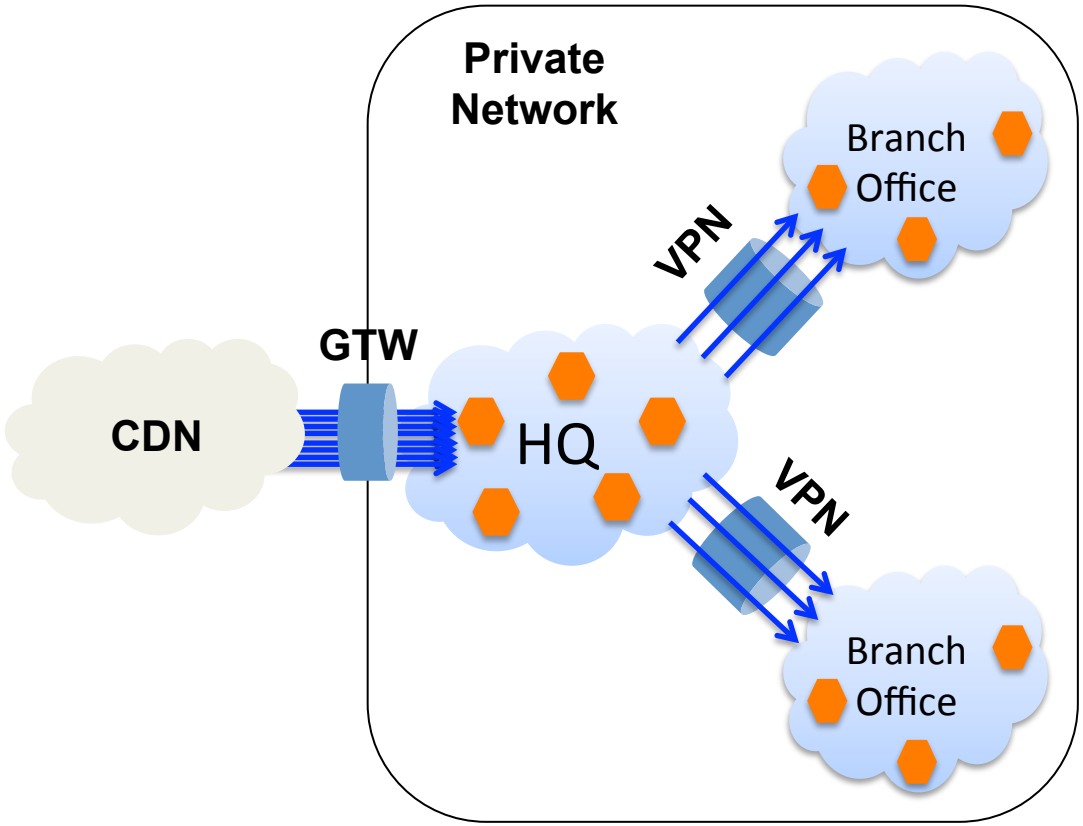
OVERLAY – PUBLIC NETWORKS

- Strive to minimize the load on the Source
- Construct a self-organizing mesh overlay network
  - Each node fully acts on local knowledge
- Emergent behavior
  - Nodes with higher capacity ends up closer to the source
  - Nodes within the same AS cluster together to achieve locality



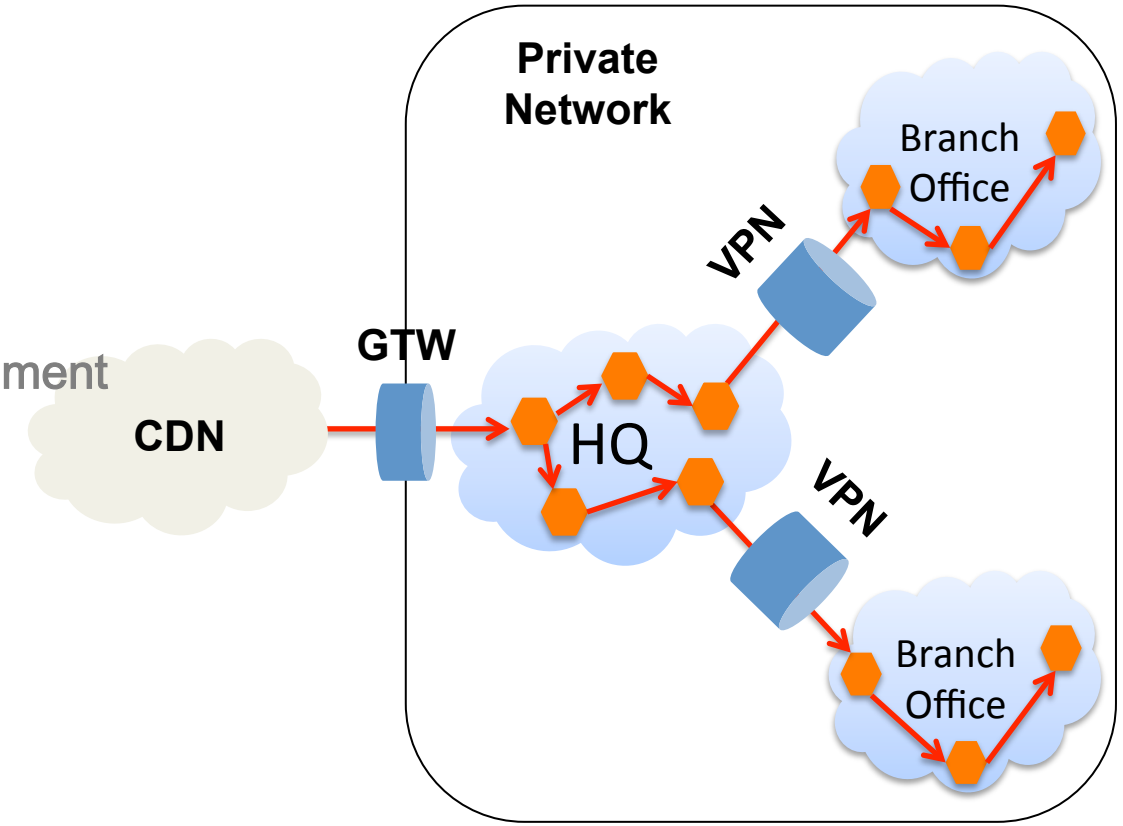
OVERLAY – PRIVATE NETWORKS

- Private networks
  - Network segments (offices)
  - VPN links between segments
  - Few gateway links towards the Internet
- Challenges
  - Gateway and VPN links are potential bottlenecks
  - Detect the network structure, i.e. avoid traffic cross branch offices or countries/continents

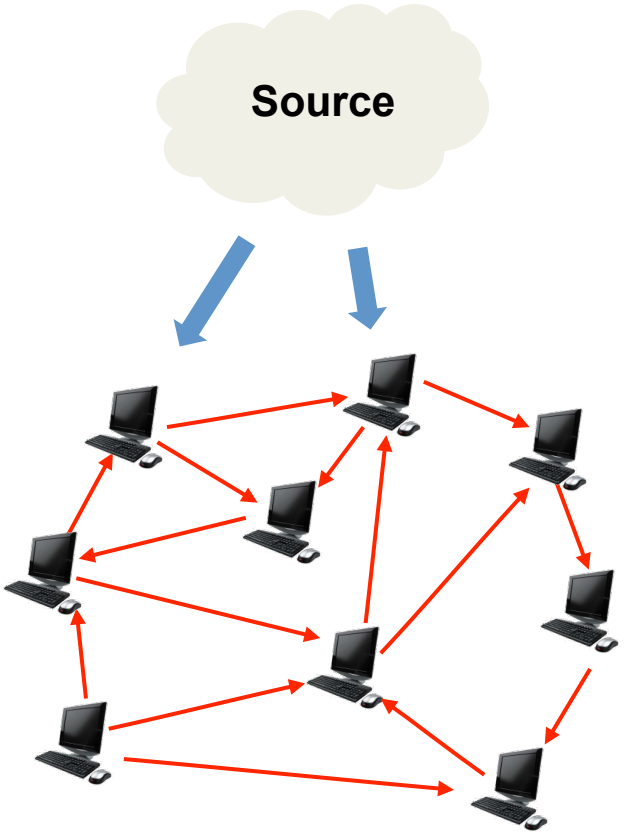


OVERLAY – PRIVATE NETWORKS

- Main goal to offload bottleneck links (VPN and Gateway)
  - A single incoming stream per bitrate
- Emergent behavior
  - Hierarchical overlay
  - Promoted supernodes per segment that fetch data before others



SYSTEM ARCHITECTURE



HIVE SERVICES

Registrar
Authentication server

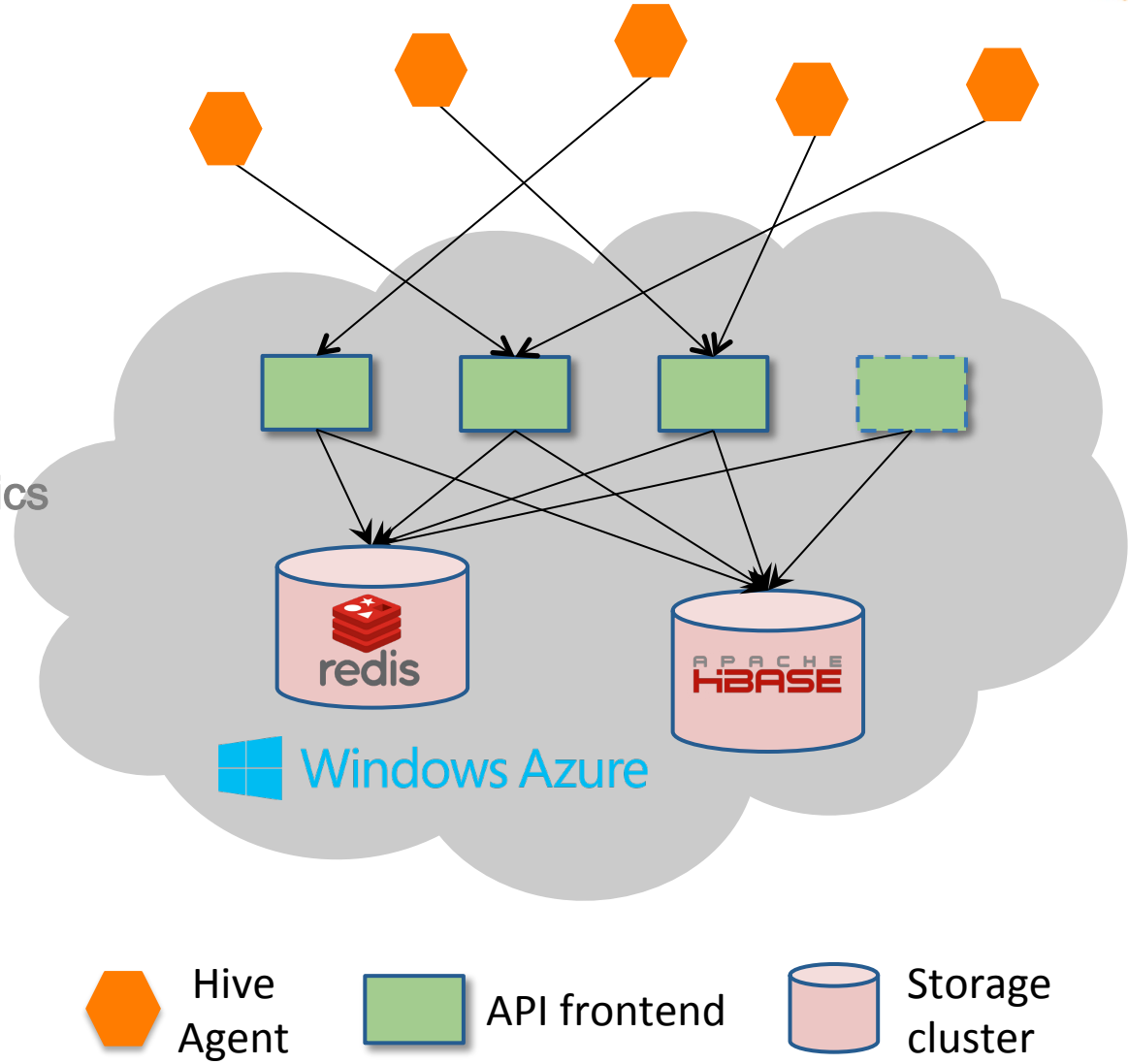
Connectivity
NAT Discovery and connection establishment helper

Tracker
Introduces nodes to each other

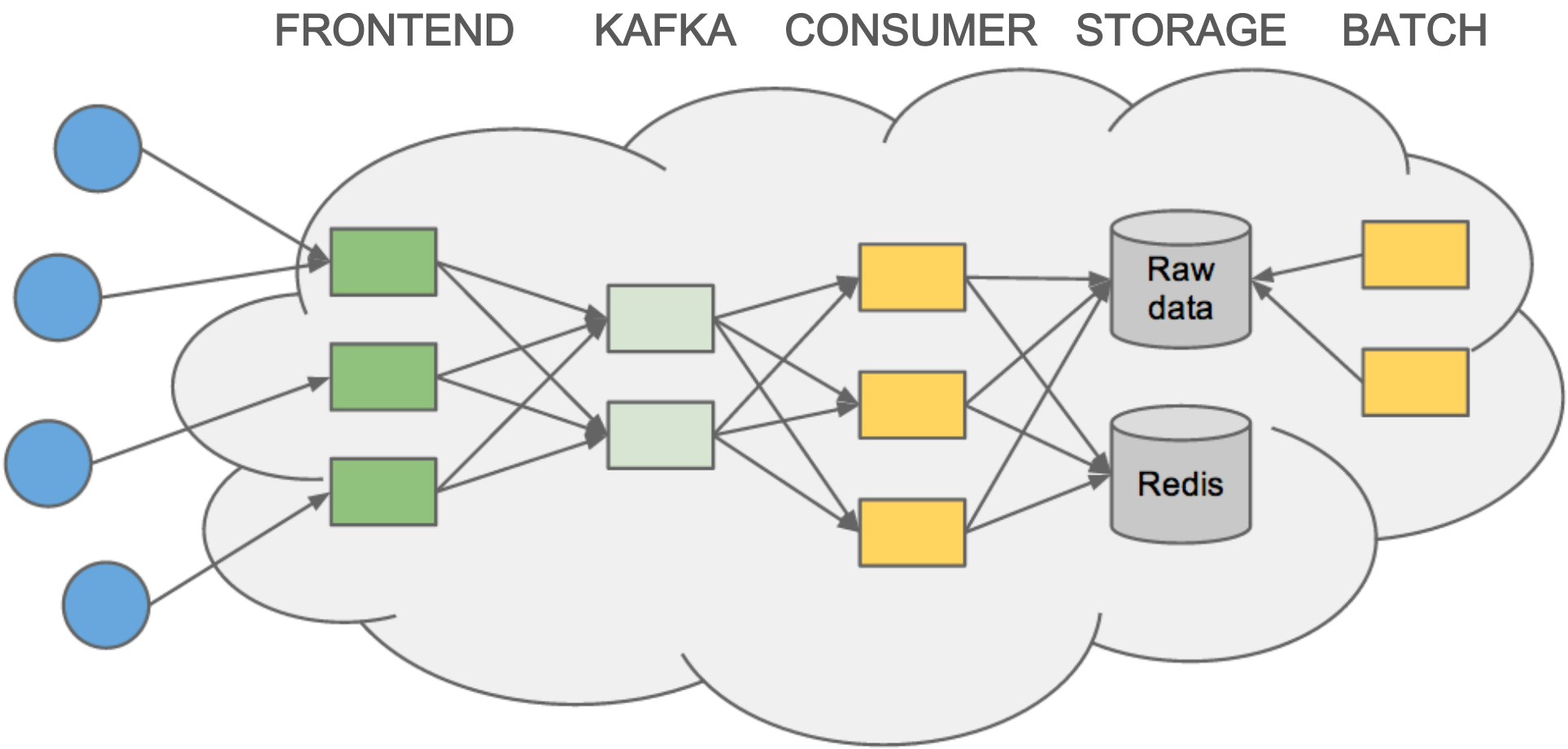
Statistics
Detailed stats for troubleshooting and billing

SERVICE ARCHITECTURE

- Stateless API Frontends
  - Scala + Finagle
- Storage backends
  - Redis for tracker/lookup
  - OpenTSDB/Hbase for statistics data
- Snapshots processing framework
  - Kafka
  - Custom built consumers



PROCESSING PIPELINE



CLIENT IMPLEMENTATION

- Based on Mesmerizer, a java component framework
- In-house network library
  - Fast SSL between all nodes
  - UDP-based transport with error detection and flow control
  - DTL: Dynamic priority congestion control
  - State of the art NAT traversal (NATCracker)

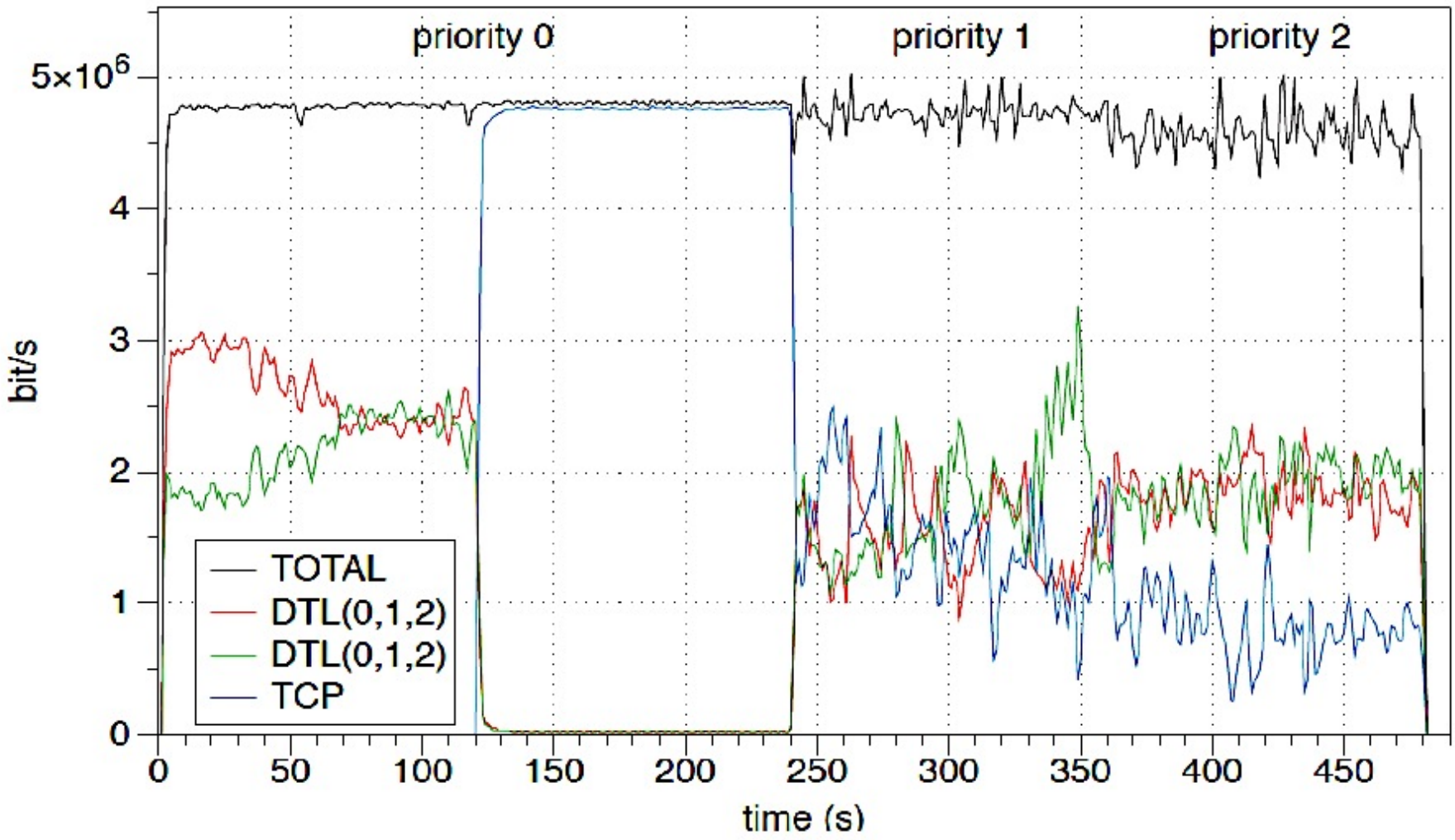
NAT Traversal Statistics

Sample of ~12000 peers
- Open Internet: 18.1%
- NATed: 78%

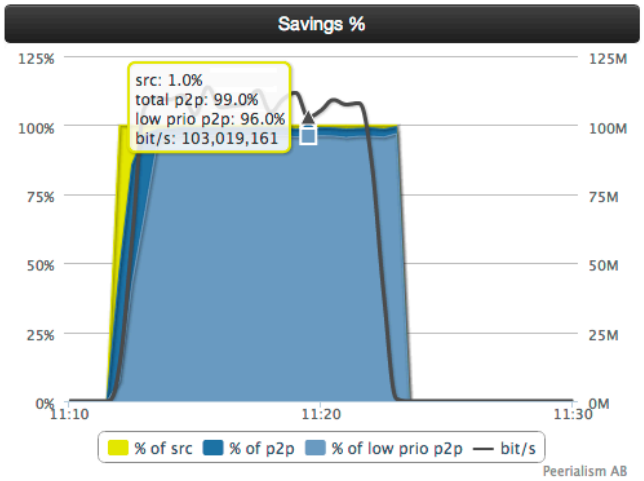
Average connectivity probability
- Direct both ways all-to-all: 85%
- One-way: 8%
- Direct excluded impossible: 92%

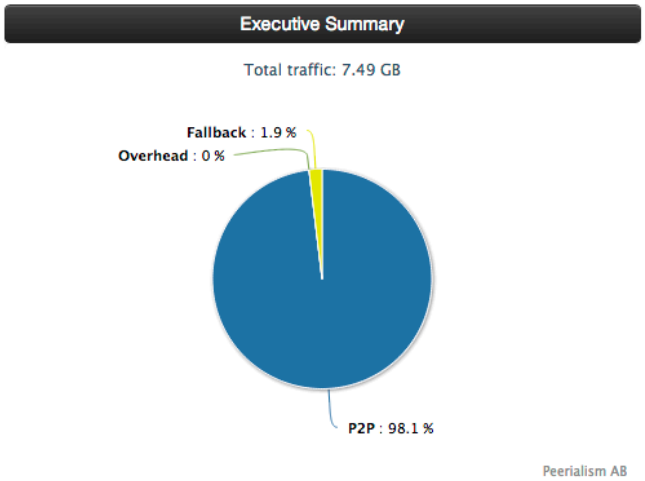
Avg. Connection establishment time: ~800ms

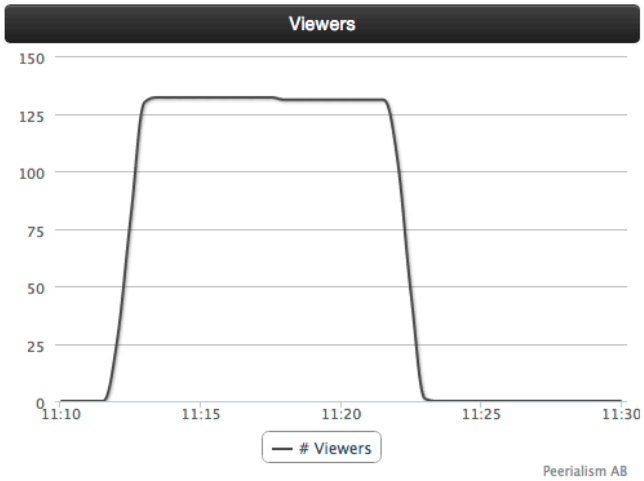
DTL CONGESTION CONTROL

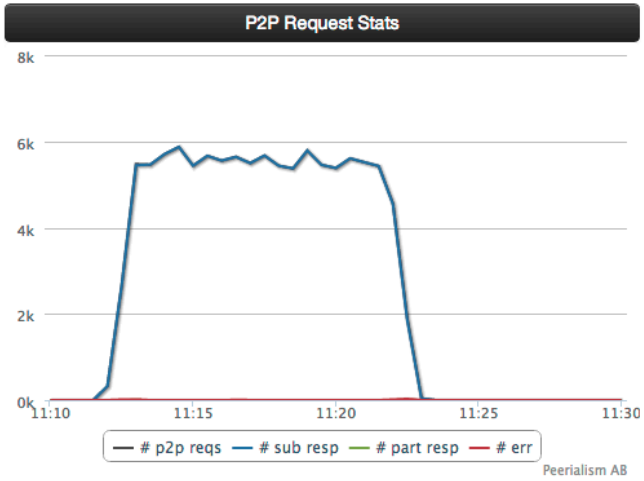


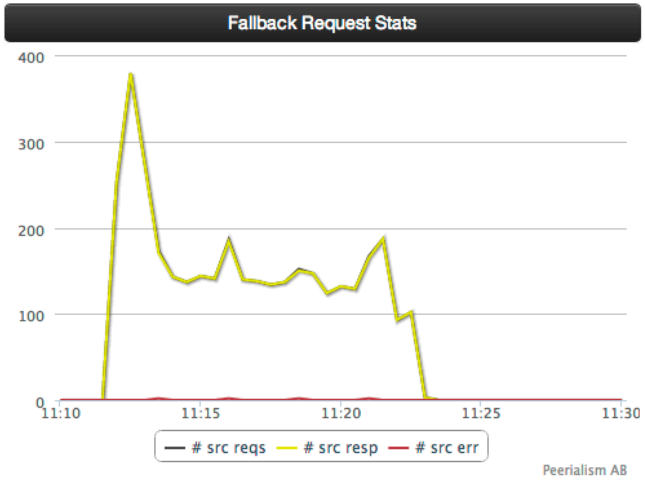
CORPORATE DEPLOYMENT – STATS

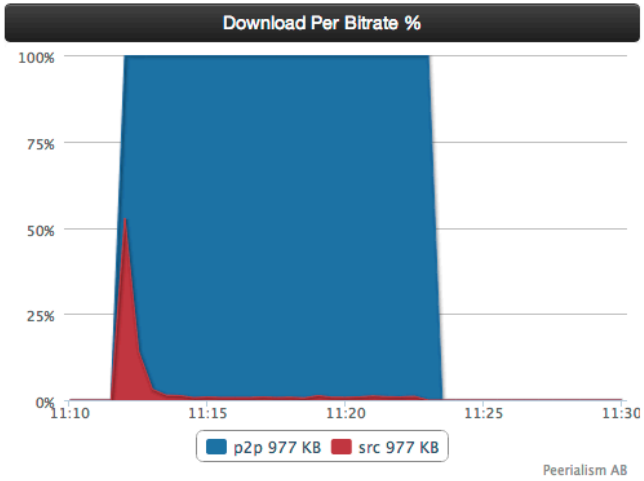






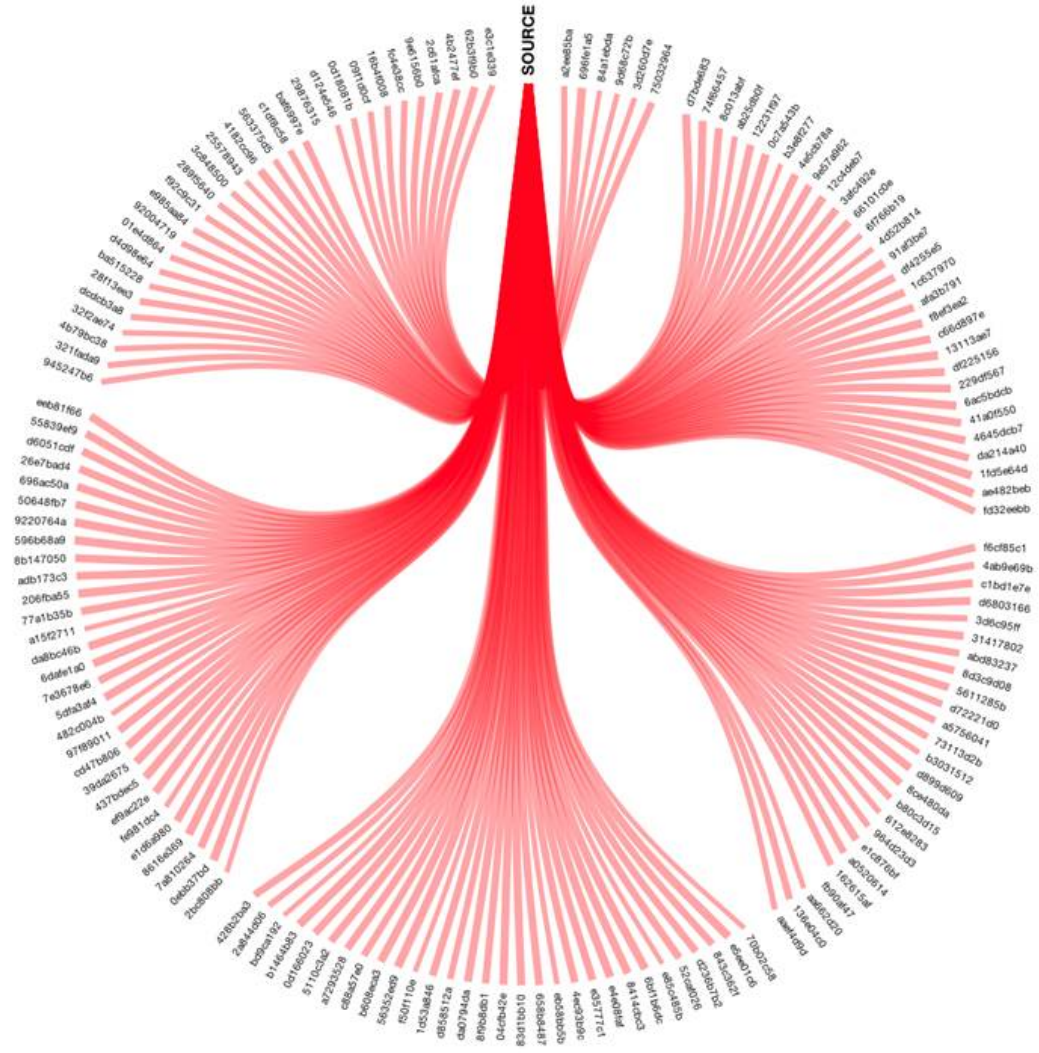




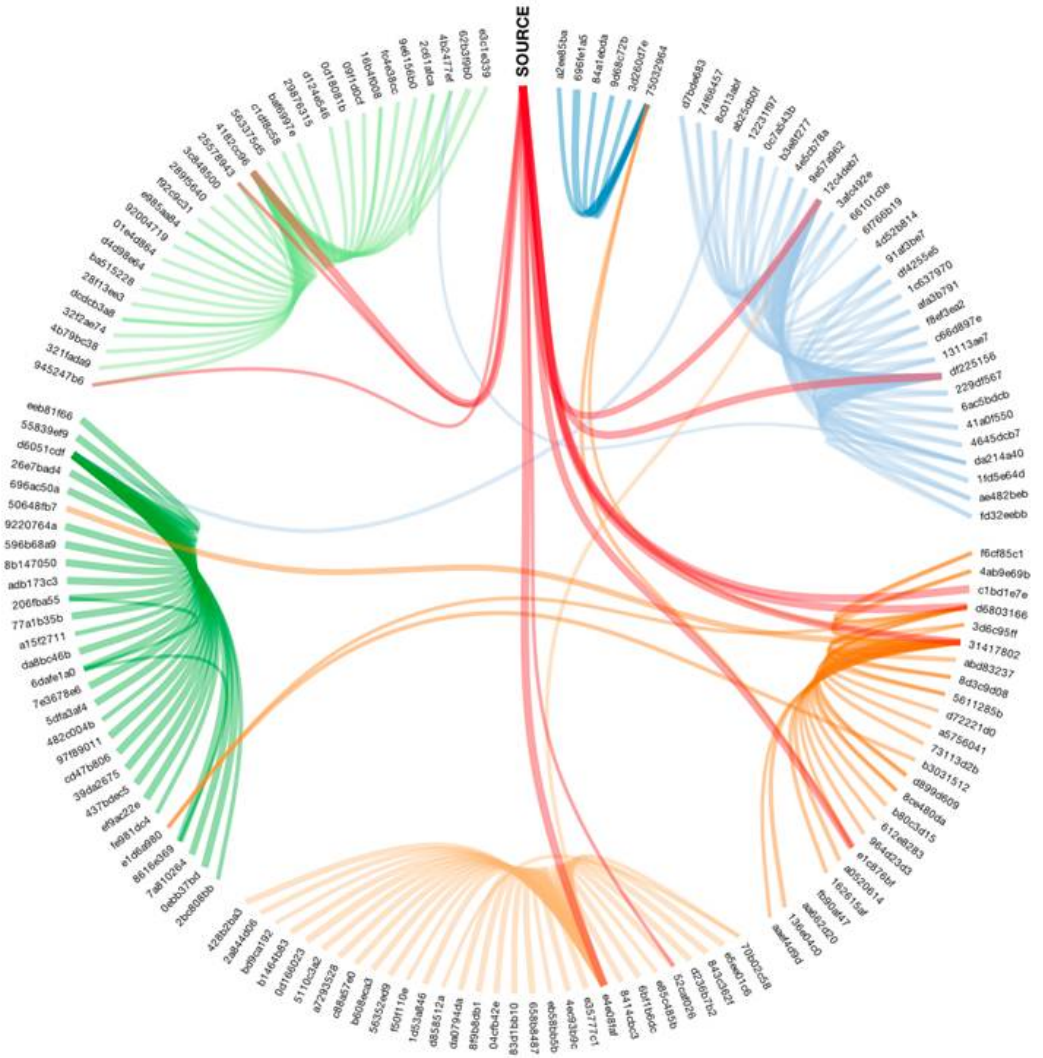


Test. Stream of 1Mbit/s, 130 concurrent clients

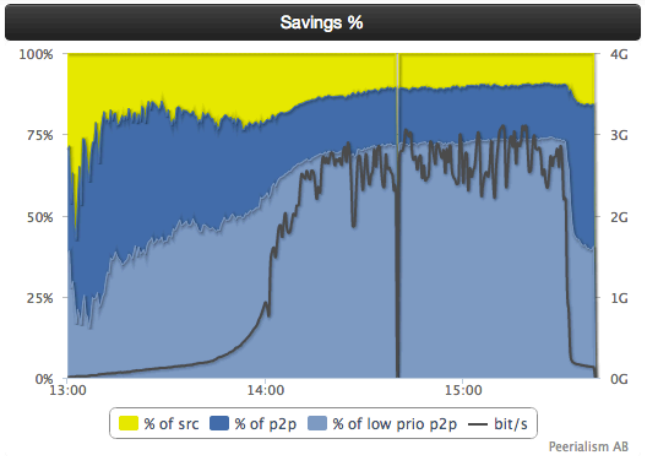
CORPORATE DEPLOYMENT – WITHOUT HIVE

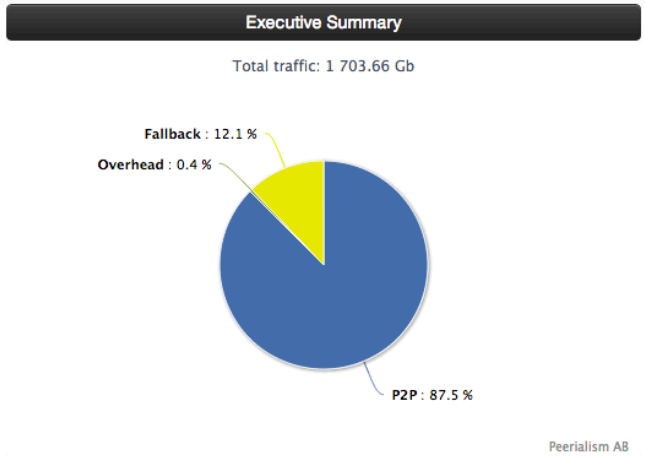


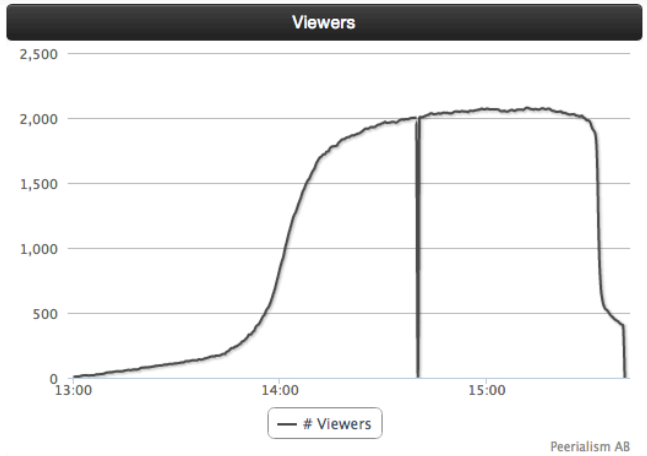
CORPORATE DEPLOYMENT – WITH HIVE

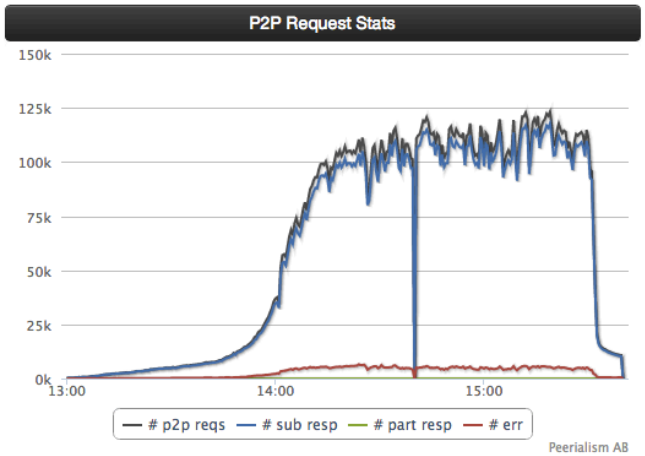


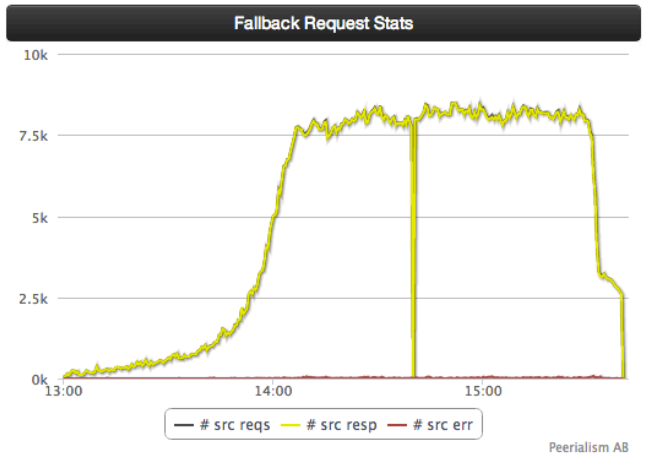
CORPORATE DEPLOYMENT – STATS

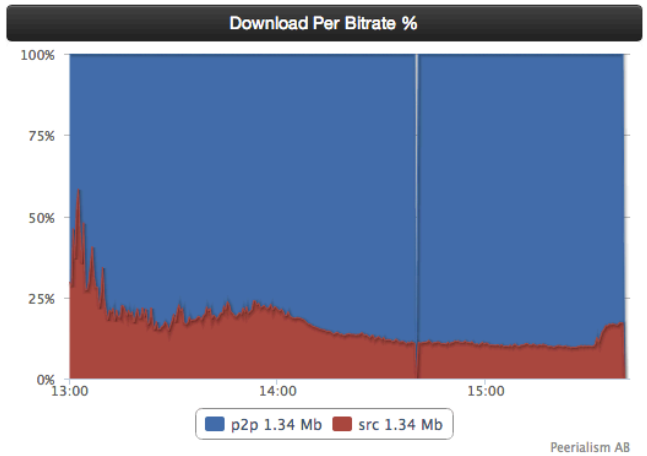






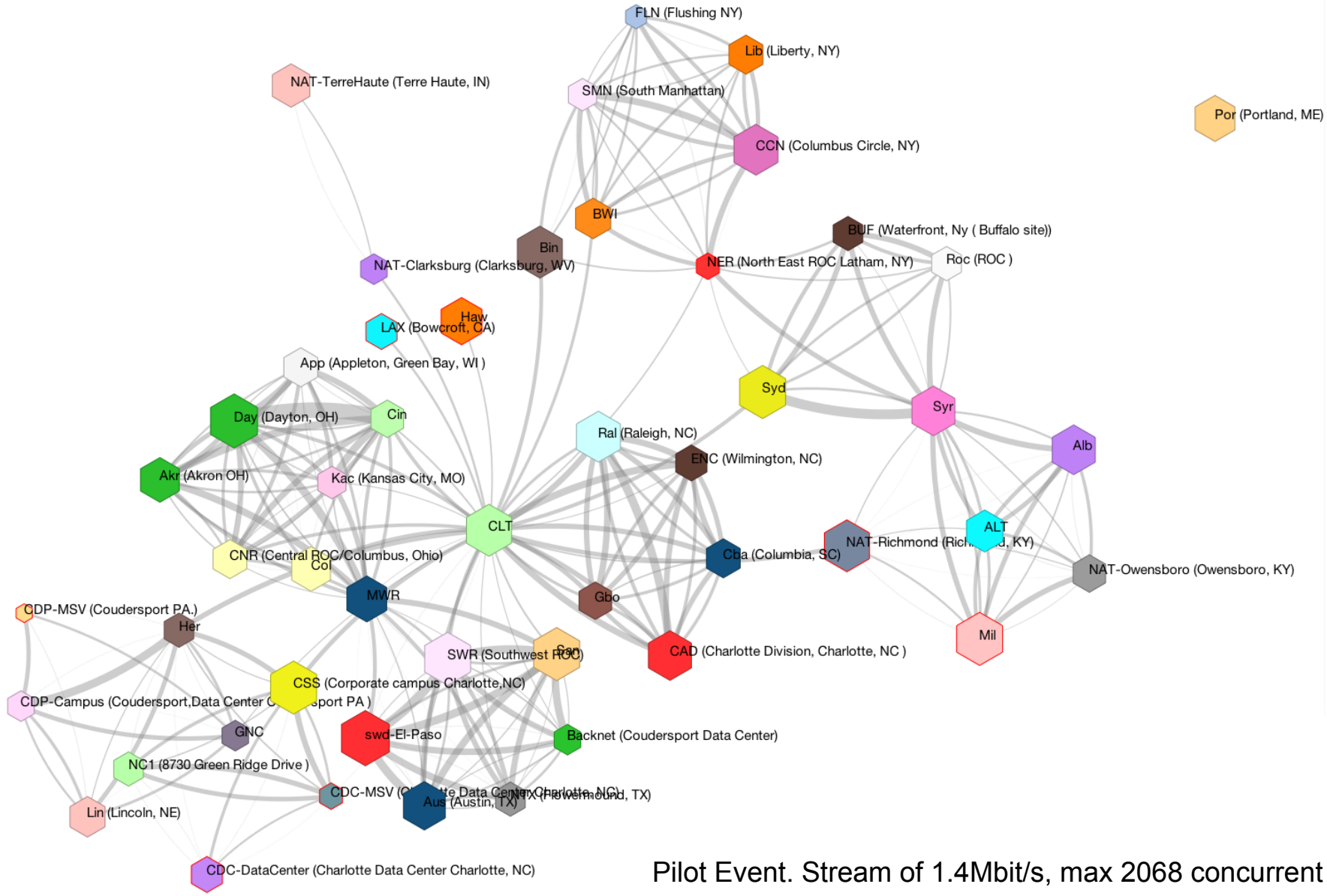






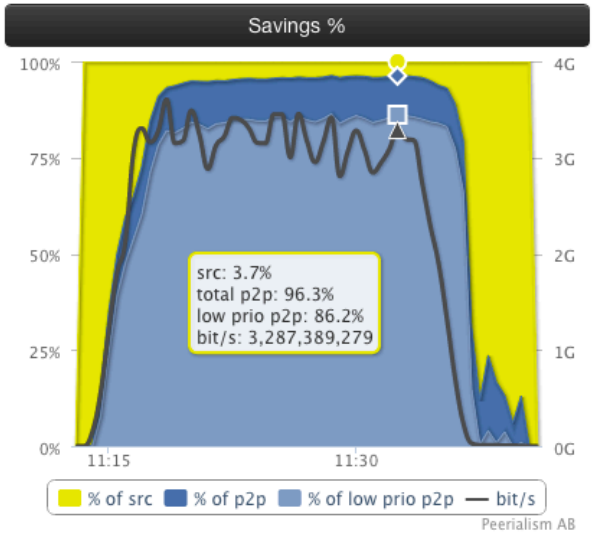
Pilot Event. Stream of 1.4Mbit/s, max 2068 concurrent viewers

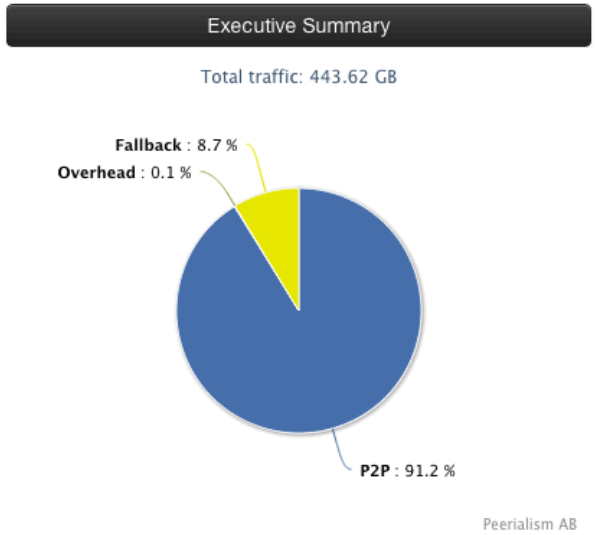
CORPORATE DEPLOYMENT – LARGE

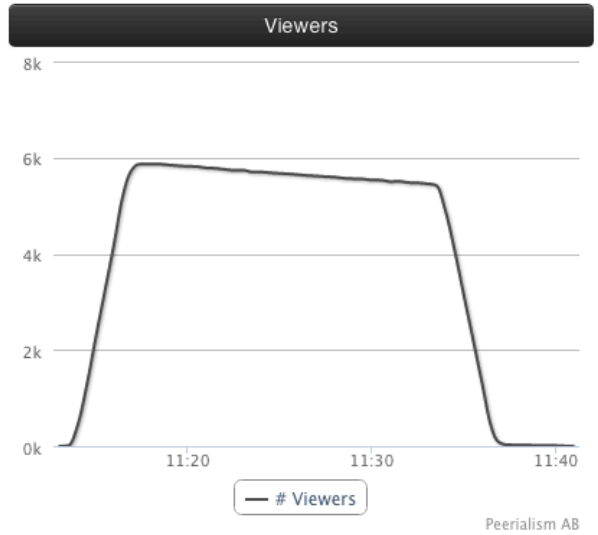


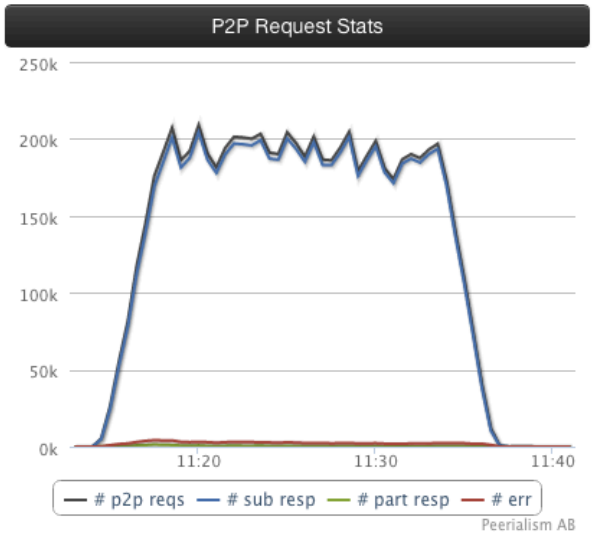
Pilot Event. Stream of 1.4Mbit/s, max 2068 concurrent viewers

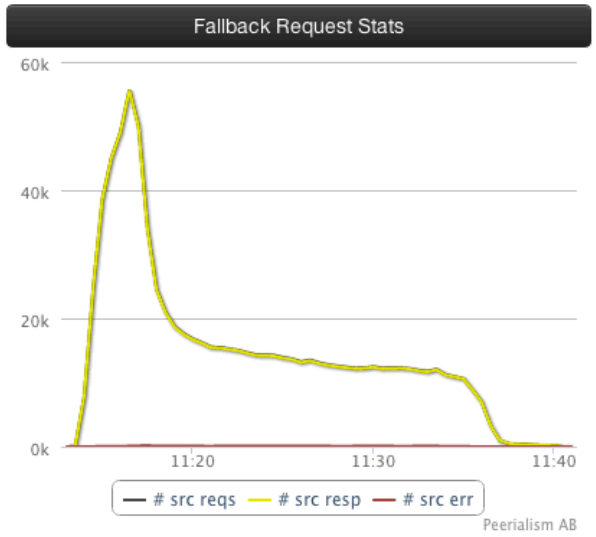
WAN DEPLOYMENT – STATS

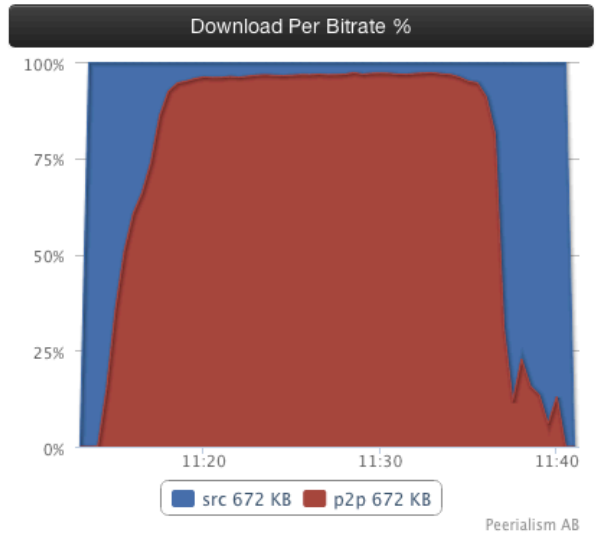








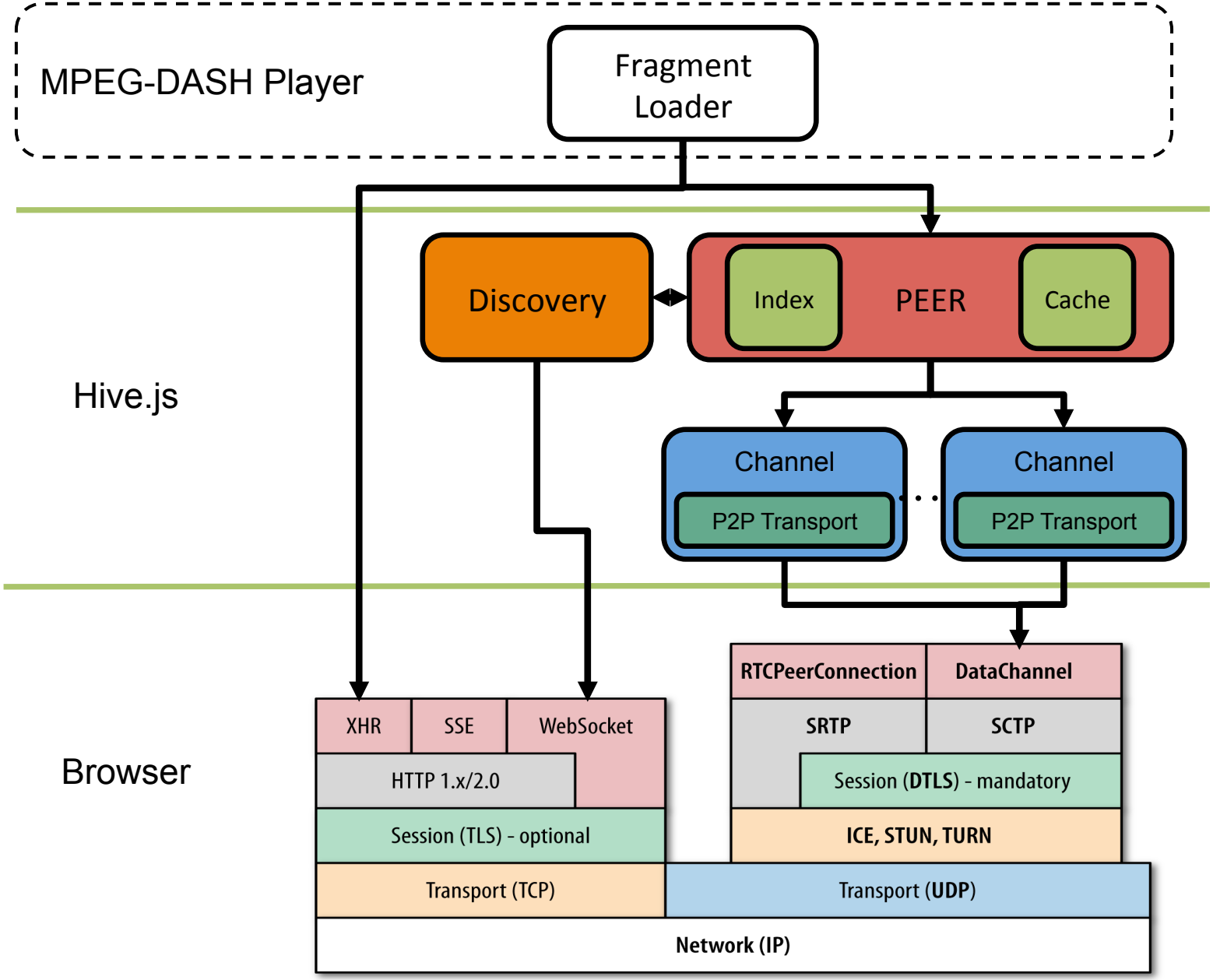




Test. Stream of 700Kbit/s, 6000 concurrent test

CURRENT RESEARCH

- Video on Demand (VoD)
  - Optimization based on real-world measurements
- Browser-based client (Hive.js)
  - No need to install plugins/native clients
  - Based on WebRTC (chrome + firefox)
  - Distributed caching algorithms
  - Proof of concept with base-line caching
- LEDBAT implementation as a contribution to the WebRTC stack


MPEG-DASH Player
Fragment Loader
Hive.js
Discovery
Index
PEER
Cache
Channel
P2P Transport
Channel
P2P Transport
Browser
XHR
SSE
WebSocket
HTTP 1.x/2.0
Session (TLS) - optional
Transport (TCP)
RTCPeerConnection
DataChannel
SRTP
SCTP
Session (DTLS) - mandatory
ICE, STUN, TURN
Transport (UDP)
Network (IP)

PUBLICATIONS

A System, Tools and Algorithms for Adaptive HTTP-Live Streaming on P2P Overlays, Roberto Roverso, PhD Thesis

On HTTP live streaming in large enterprises, Roverso et Al., SIGCOMM 2013, Hong Kong

Peer2View: a Peer-To-Peer HTTP-live streaming platform, Roverso et Al., P2P 2012, Sept, Tarragona (SP)

SmoothCache: HTTP-Live Streaming Goes Peer-To-Peer, Roverso et Al., IFIP Networking 2012, May, Prague (CZ)

DTL: Dynamic Transport Library for Peer-To-Peer Applications, Reale et Al., ICDCN 2012, January, Honk Kong (China)

NATCRACKER: NAT Combinations Matter, Roverso et Al., ICCCN 2009, July, San Francisco (CA)

Mesmerizer: a effective tool for a complete peer-to-peer software development life-cycle, Roverso et Al., Simutools 2011, Feb, Barcelona (SP)