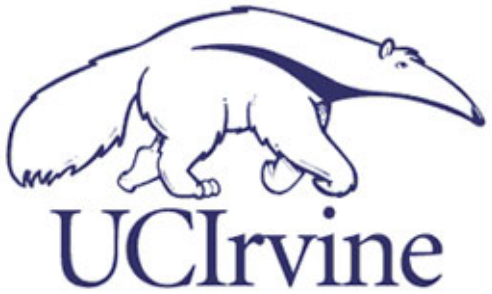
Deploying IoT devices to blind users

Austin Marron

About Me


UCIrvine

BLIZZARD
ENTERTAINMENT

intuit
TurboTax

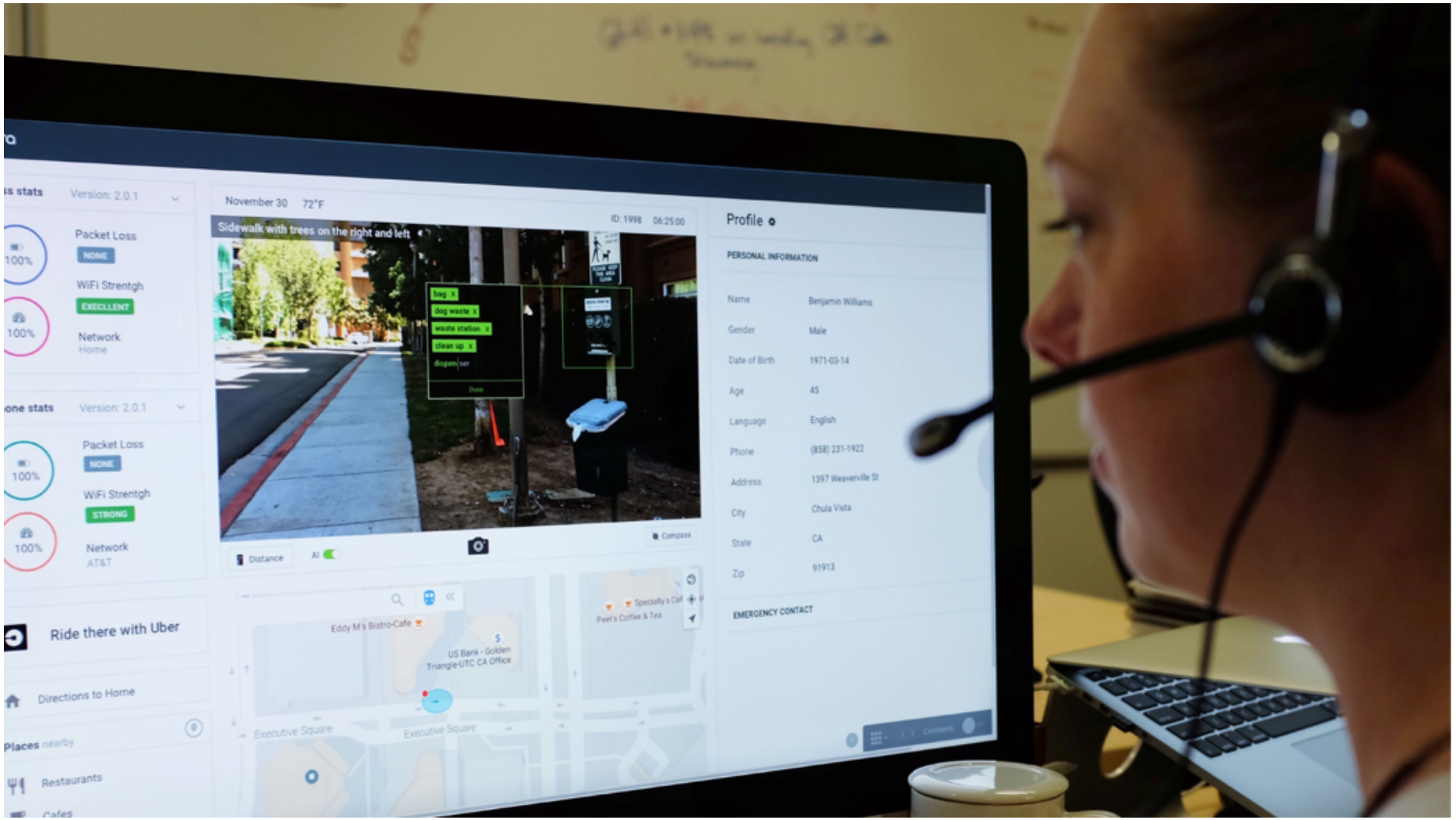
techstars

common form

aira

About Aira

[Intro Video]



Agent Dashboard

Our Team

Investors

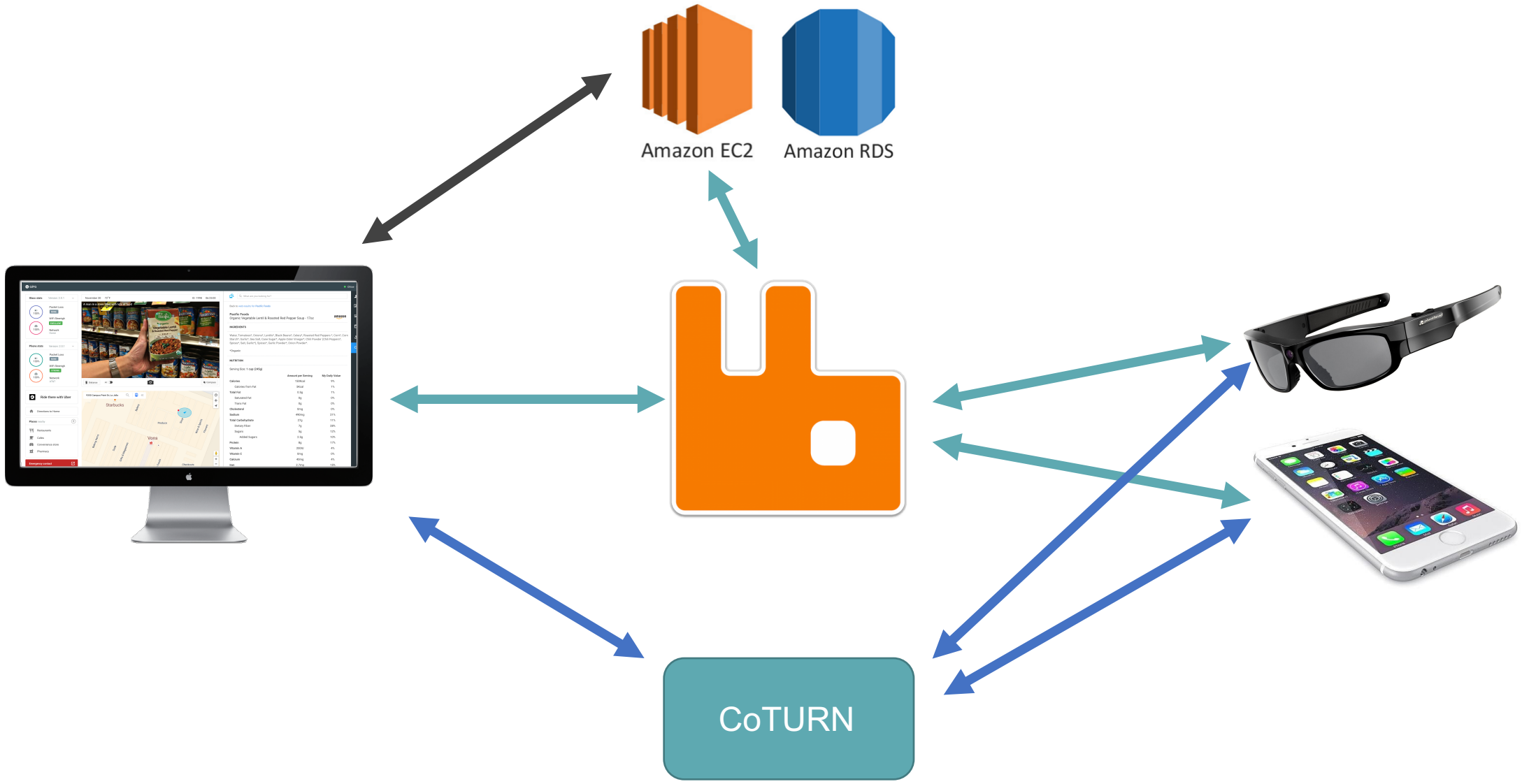
ARCH Venture Partners

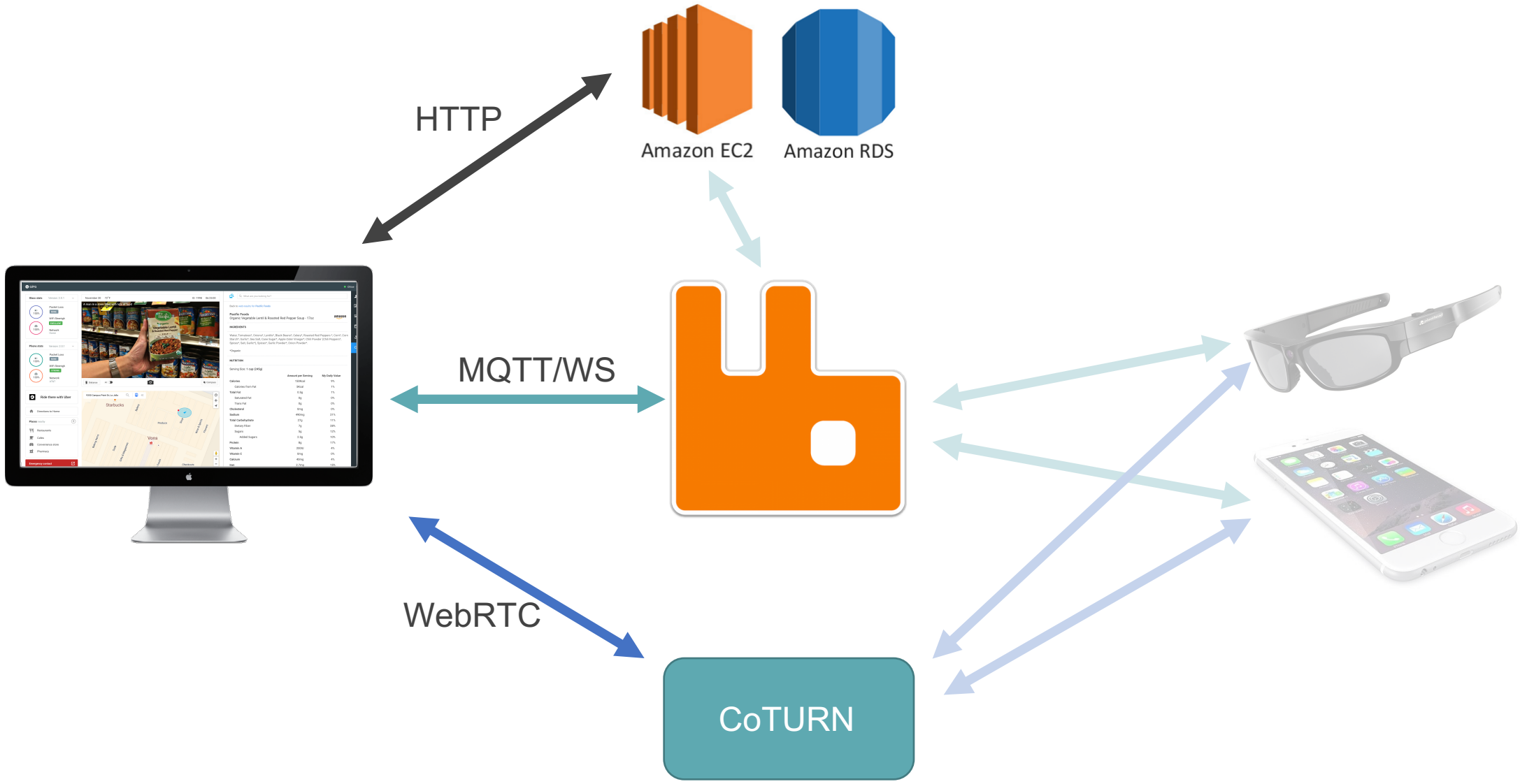
LUT

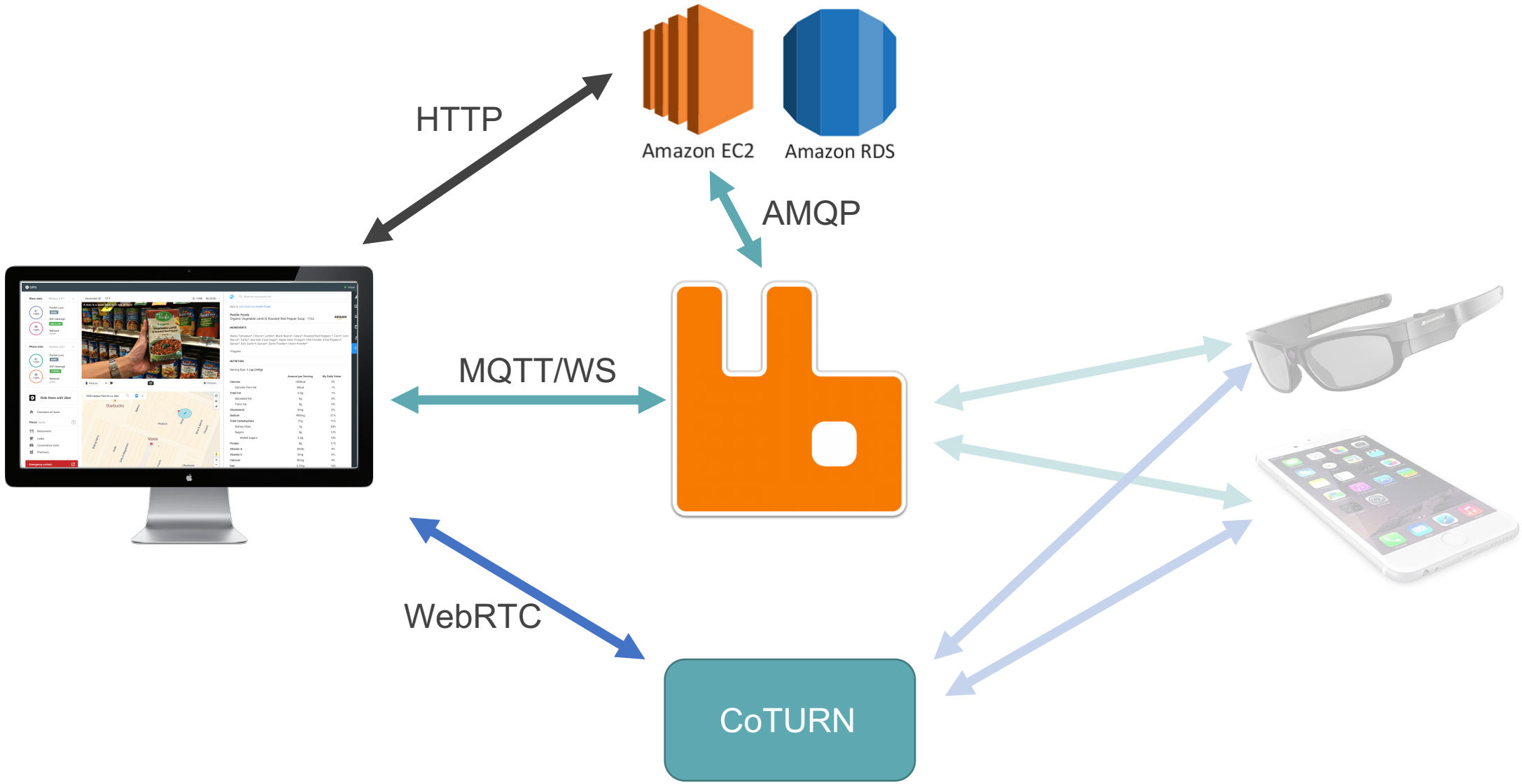
Felicis Ventures

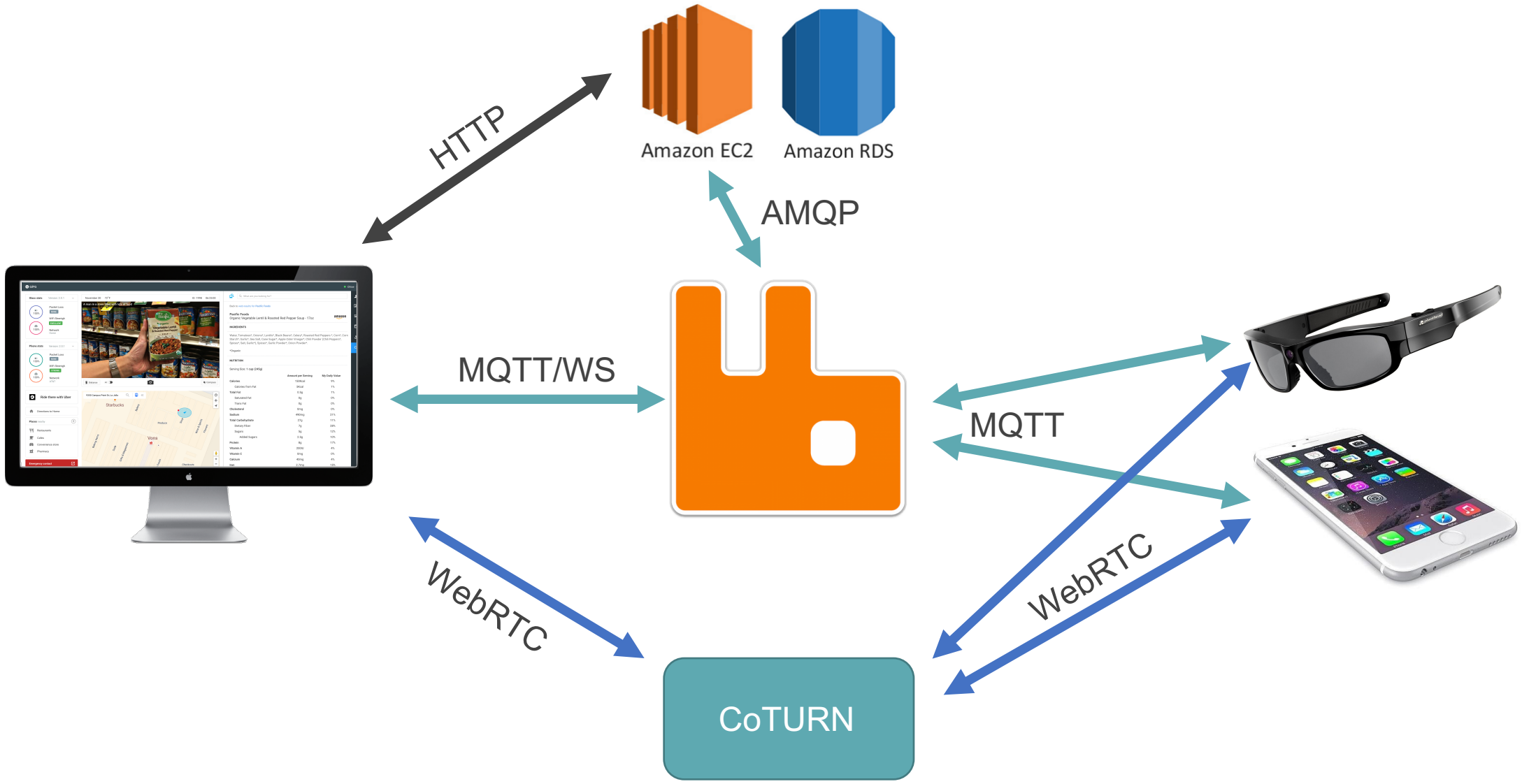
Team

What is the one thing, that if removed from your product, would make the whole thing fall apart?


Amazon EC2
Amazon RDS
CoTURN


HTTP
Amazon EC2
Amazon RDS
MQTT/WS
WebRTC
CoTURN


HTTP
Amazon EC2
Amazon RDS
AMQP
MQTT/WS
WebRTC
CoTURN


HTTP
Amazon EC2
Amazon RDS
AMQP
MQTT/WS
MQTT
WebRTC
WebRTC
CoTURN

Brokers, MQTT, AMQP, ?!

MQTT: Message Queueing Telemetry Transport

- Basic pub/sub architecture where clients interact based on topics.
- First used in the late 90s for oil rigs. Has been deployed on satellites and embedded systems that demand light-weight footprints.

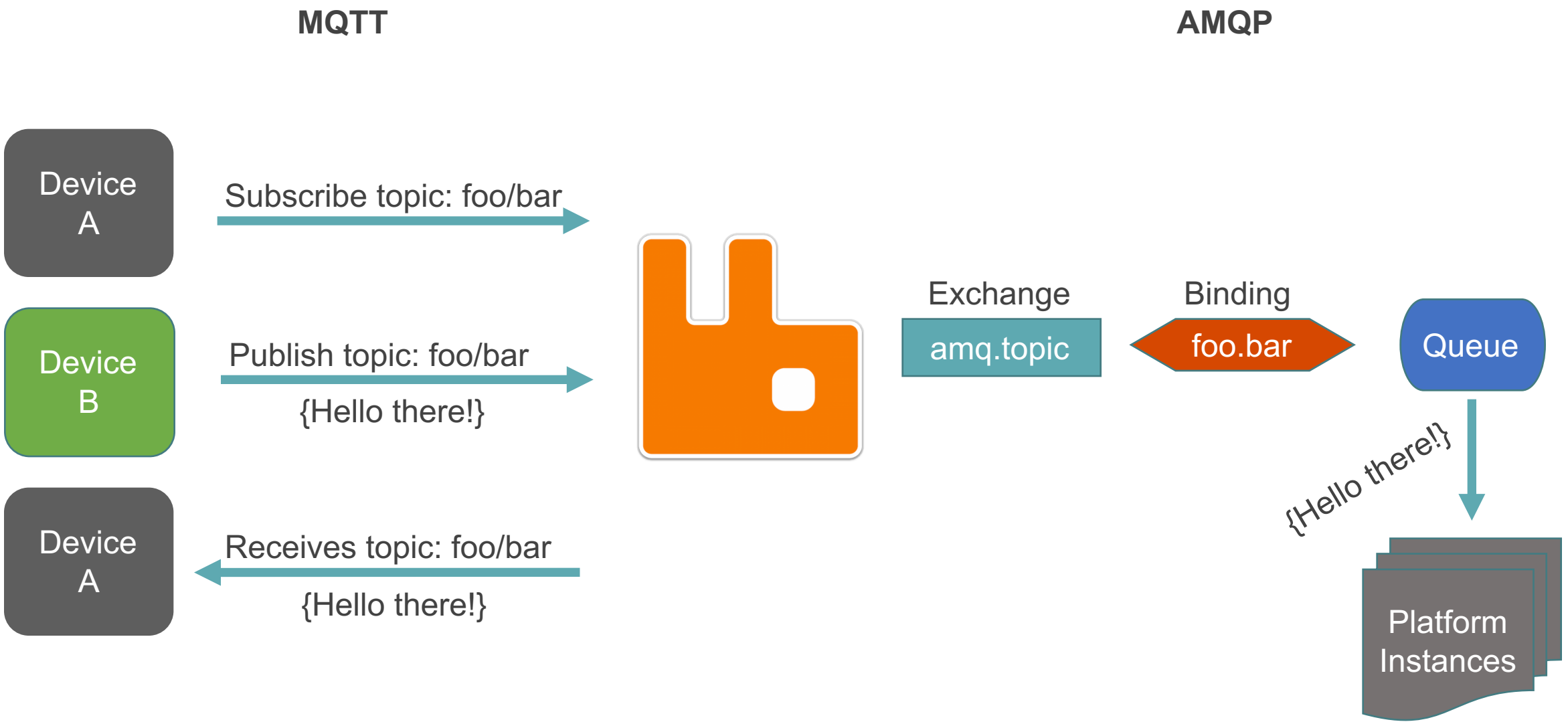
AMQP: Advanced Message Queueing Protocol

- Robust message delivery protocol that can support pub/sub, rpc, fanout, queues, the list goes on.
- Based on the concept of sending messages to exchanges and delivering them to queues.

Our Device Communication

Requirements:

- Each client needs its own persistent connection
- Some clients are single instance, others multi-instance: shared subscriptions
- Free to use (read: Not HiveMQ)
- On-site Deployment option (read: Not cloud-locked)
- Had MQTT already deployed to client devices


MQTT
AMQP
Device A
Subscribe topic: foo/bar
Device B
Publish topic: foo/bar
{Hello there!}
Device A
Receives topic: foo/bar
{Hello there!}
Exchange
amq.topic
Binding
foo.bar
Queue
{Hello there!}
Platform Instances

Our IoT Device: Glasses + LTE ”Puck”

- Worked with numerous glasses:
  - Google Glass
  - Pivothead
  - Vuzix
  - SeeU
  - ODG
  - And more…

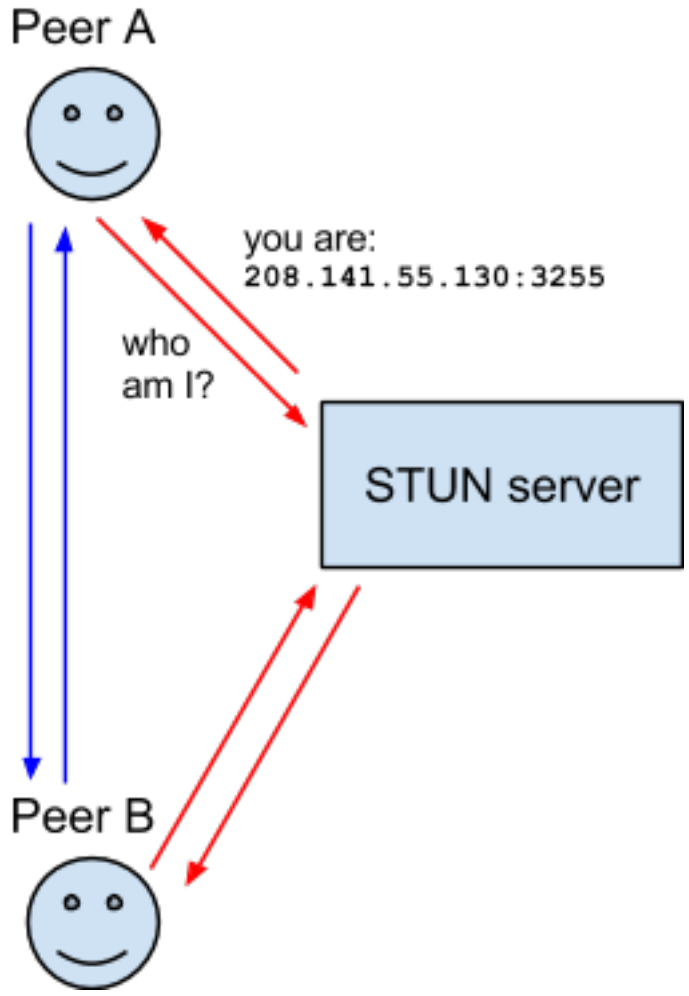
Our Video/Audio Streams: WebRTC

Video Requirements:

- Low latency (< 200 ms)
- Low bandwidth (< 100 kbps)
- Works on low-end devices
- Inexpensive infrastructure
- Works in browser, glasses, and phones.

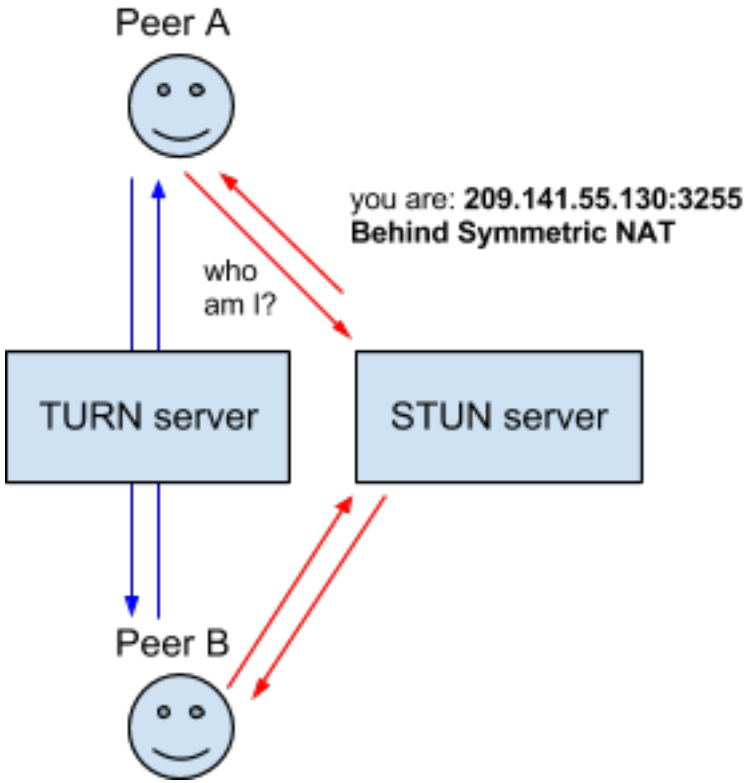
STUN – Session Traversal Utilities for NAT

- Find your public IP
- Determine restrictions in your router that would prevent a direct peer connection.
- Client makes request to STUN server, which replies with public and IP and status of accessibility.
- This is used to build a variety of candidates.



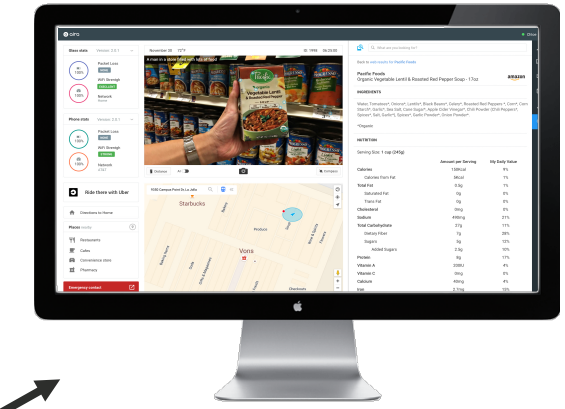
TURN – Traversal Using Relays around NAT

- Meant to bypass symmetric NAT.
- All traffic between peers is “relayed” through this server.
- Variable overhead.
- Relay endpoints are a type of candidate.
- This is an easy place to drop in a media server, such as Kurento.



WebRTC Overview

ICE Framework



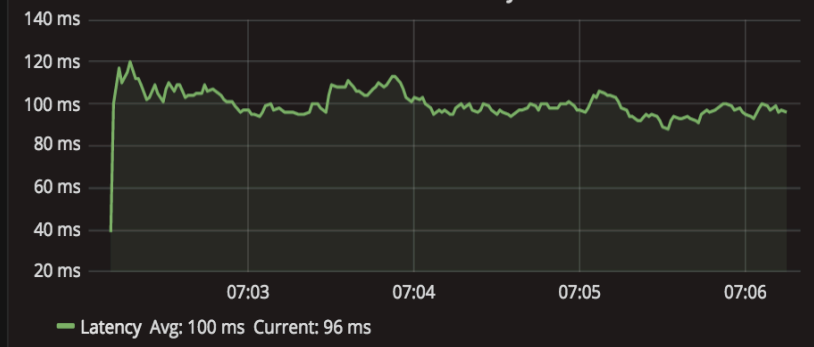
1. Publish Offer
4. Receive Answer
2. Receive Offer
3. Publish Answer
5. Candidates
6a. Peer-To-Peer Connection
Video, Audio, Data
STUN (for candidates)
STUN (for candidates)
6b. TURN Connection
CoTURN
6b. TURN Connection
Video, Audio, Data

aira | visual interpreter for the blind

Time To Pick

N/A to Pick

Video/VOIP Latency



WebRTC Connection

UDP

WebRTC Candidate Type

Candidate is TURN

Time to First Video

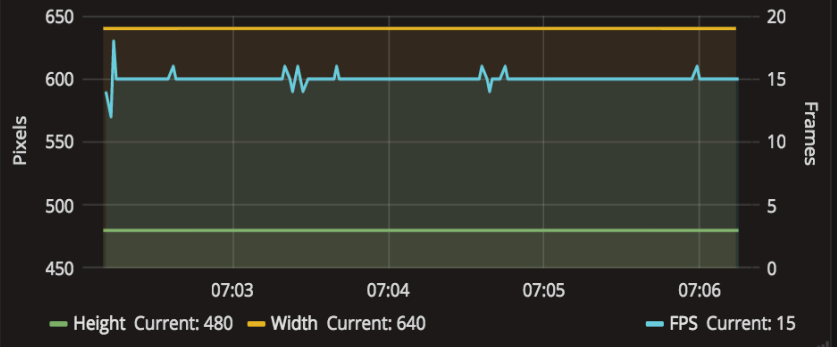
6.98 s to Video Start

Call Duration

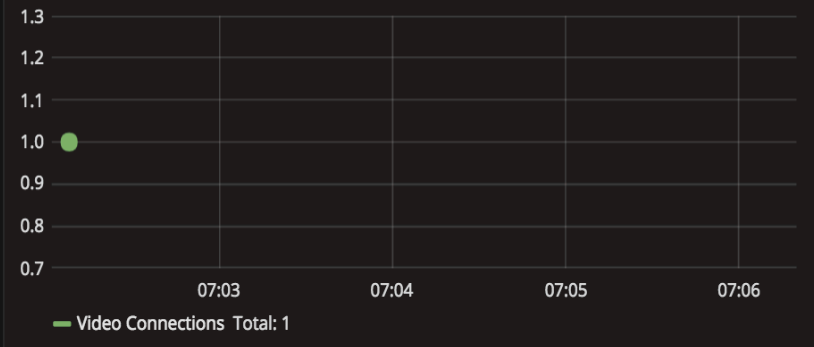
3.62 min

Video/Audio Metrics

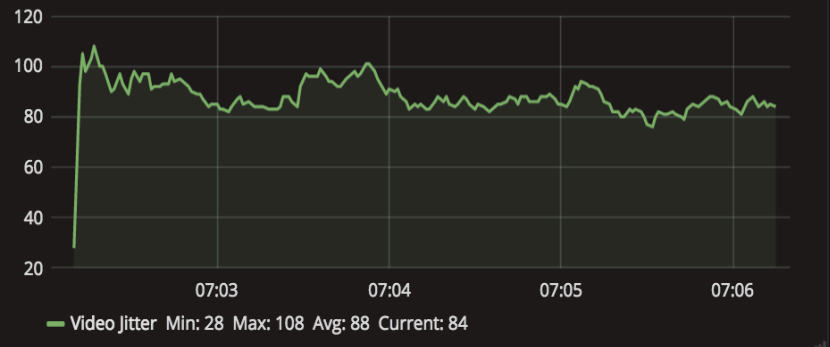
Video Dimensions Captured on Dashboard



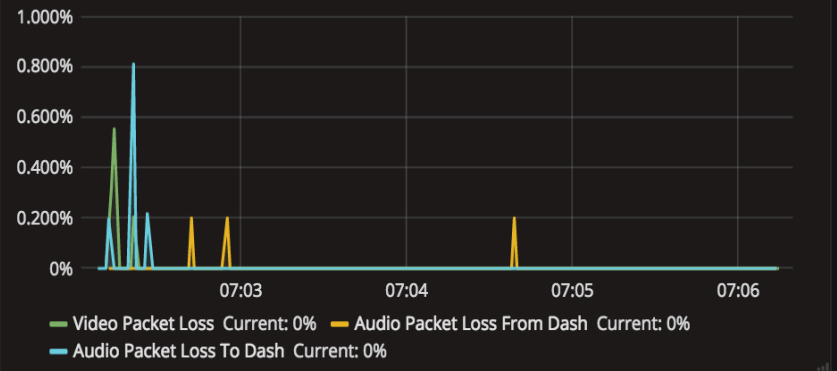
Video Connections



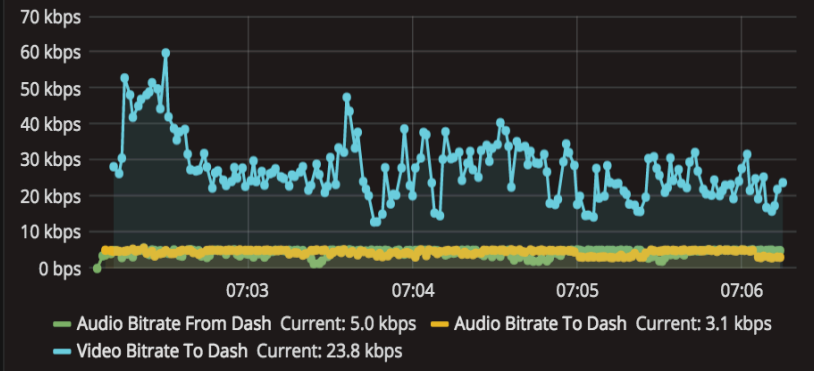
Jitter



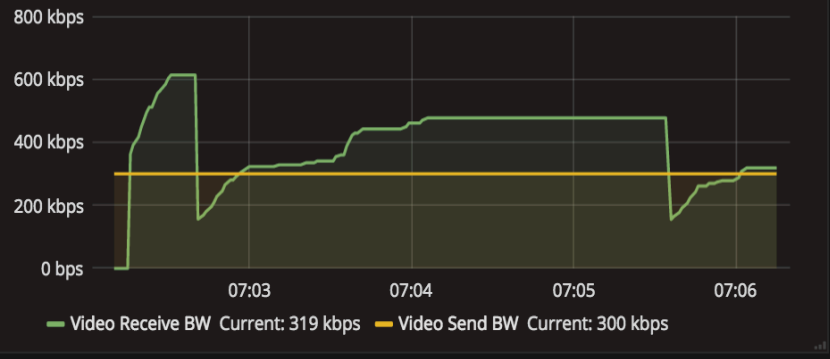
Packet Loss



Audio and Video Bitrate



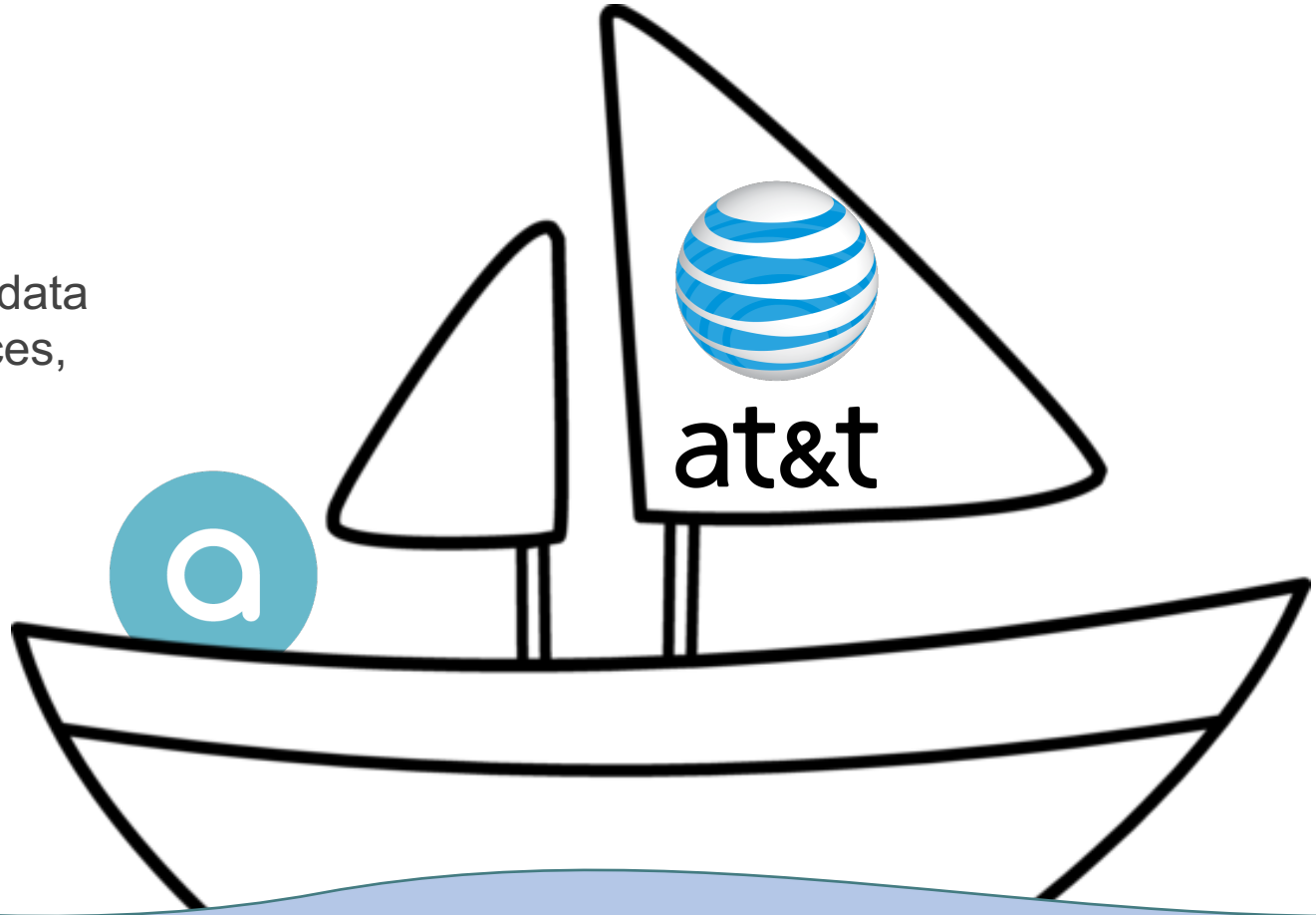
Available Bandwidth



What about connectivity?

AT&T Dynamic Traffic Management

- Provides network prioritization for data from critical services, health services, first responders.
- More info at AT&T CES Keynote!



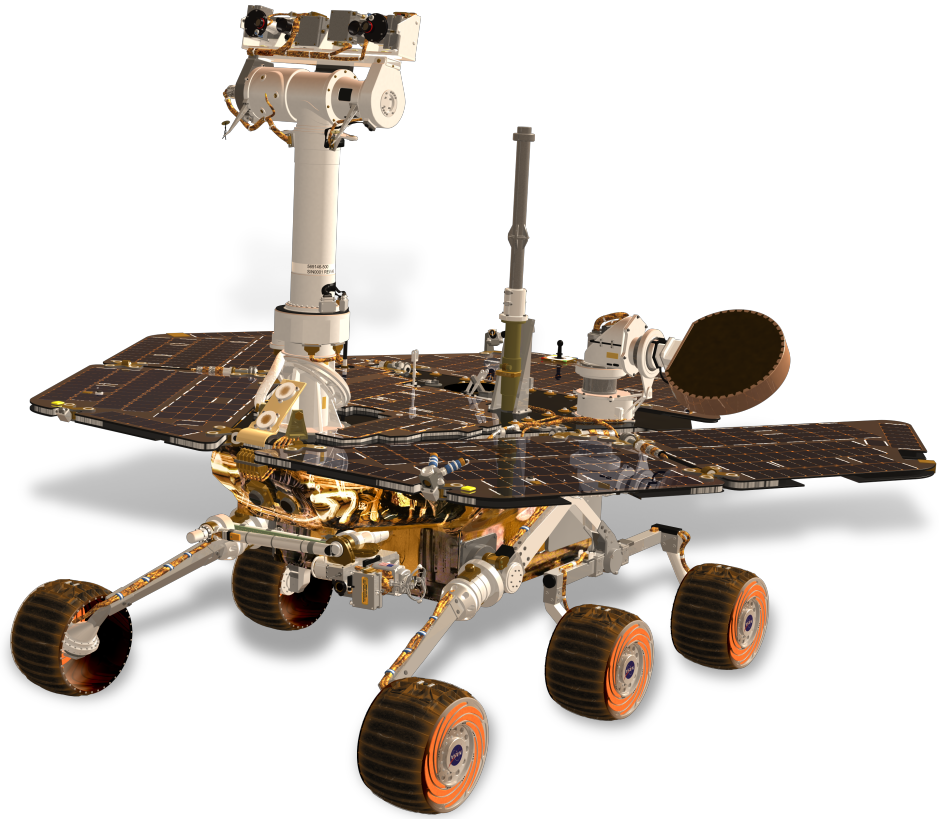
NETFLIX YouTube facebook amazon instant video

Managing an IoT Device

We treat each device like a rover. Once it's shipped, we will never get physical access to it again.

Issues Faced

- Intermittent connectivity
- Recovery from crashes
- Remote Control (Telemetry/Telecommand)
- Software Updates and Version Skew
- Securing devices



Handling Intermittent Connectivity

- Have a “sleep mode”, enter it intelligently.
  - Folding the glasses
- Wake the device to check connectivity.
  - Plugging in the glasses
- Create a device shadow.
  - AWS IoT does this well.
- Robust reconnection logic.
  - This code will execute all the time.

Crash Recovery

- Crashes happen, you need to know about them
  - Splunk Mint helps with this
- Restart in a good state automatically
  - Don’t assume user will do this for you
- Create some redundancy
  - Multiple processes to accept commands

Remote Control

- Do everything remotely
  - Trigger all user actions.
  - Build recovery commands.
- Report full state
  - Always have your device’s telemetry
- Automate commands to run later
  - Run on the device next time it’s online

Software Updates and Version Skew

- Have an automatic update plan
  - Know when it's safe to update
  - Check automatically, or push
- Minimize breaking changes
  - Users don't like waiting for updates
  - Latest versions will take time to fully roll out
- Keep logic off the device
  - Let the device ask the server what to do.

Secure the Devices

- Don’t contribute to the next Mirai Botnet
- Provision devices with secure credentials
  - AWS IoT does this well
- Be able to block a device, even brick it

Questions?

Thank You!