Paradrop: Enabling Lightweight Multi-tenancy at the Network's Extreme Edge

Peng Liu, Dale Willis, Suman Banerjee
University of Wisconsin-Madison
2016 IEEE/ACM Symposium on Edge Computing

Presented by Allen Leis and Patrick Jennings

Agenda

1. Problem Definition
2. Novel Approach
3. Background
4. Proposed / Developed Solution
5. Evaluation
6. Critique

Disclaimer

Unlike previous papers, this paper is light on certain technical details.

Much of the detail actually included compares topics like Docker vs LXC.

There are no proofs, formulas, etc.

There is no novelty per se other than the management and deployment of containers to WiFi access points.

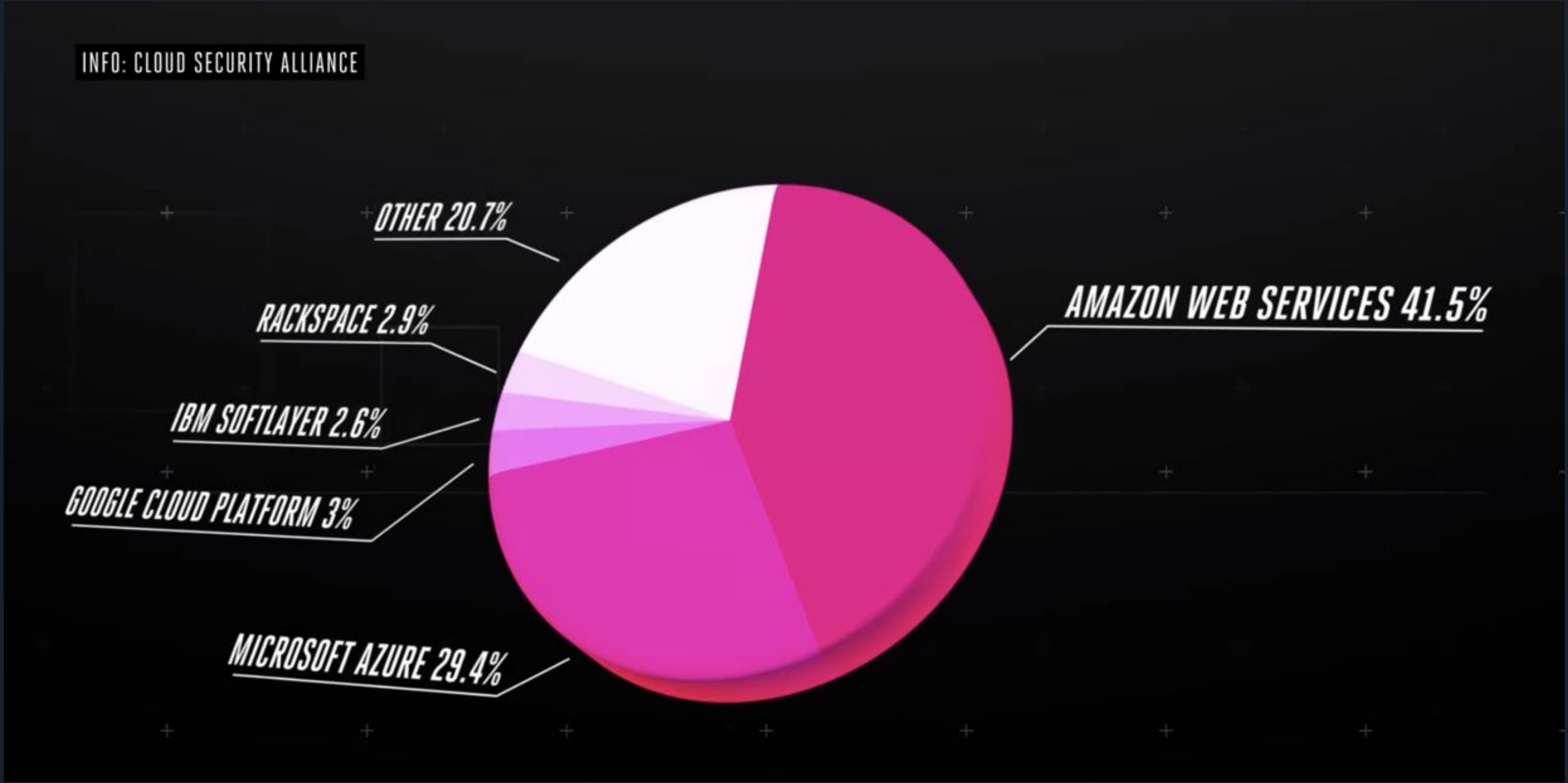
Problem Definition

Problem Definition

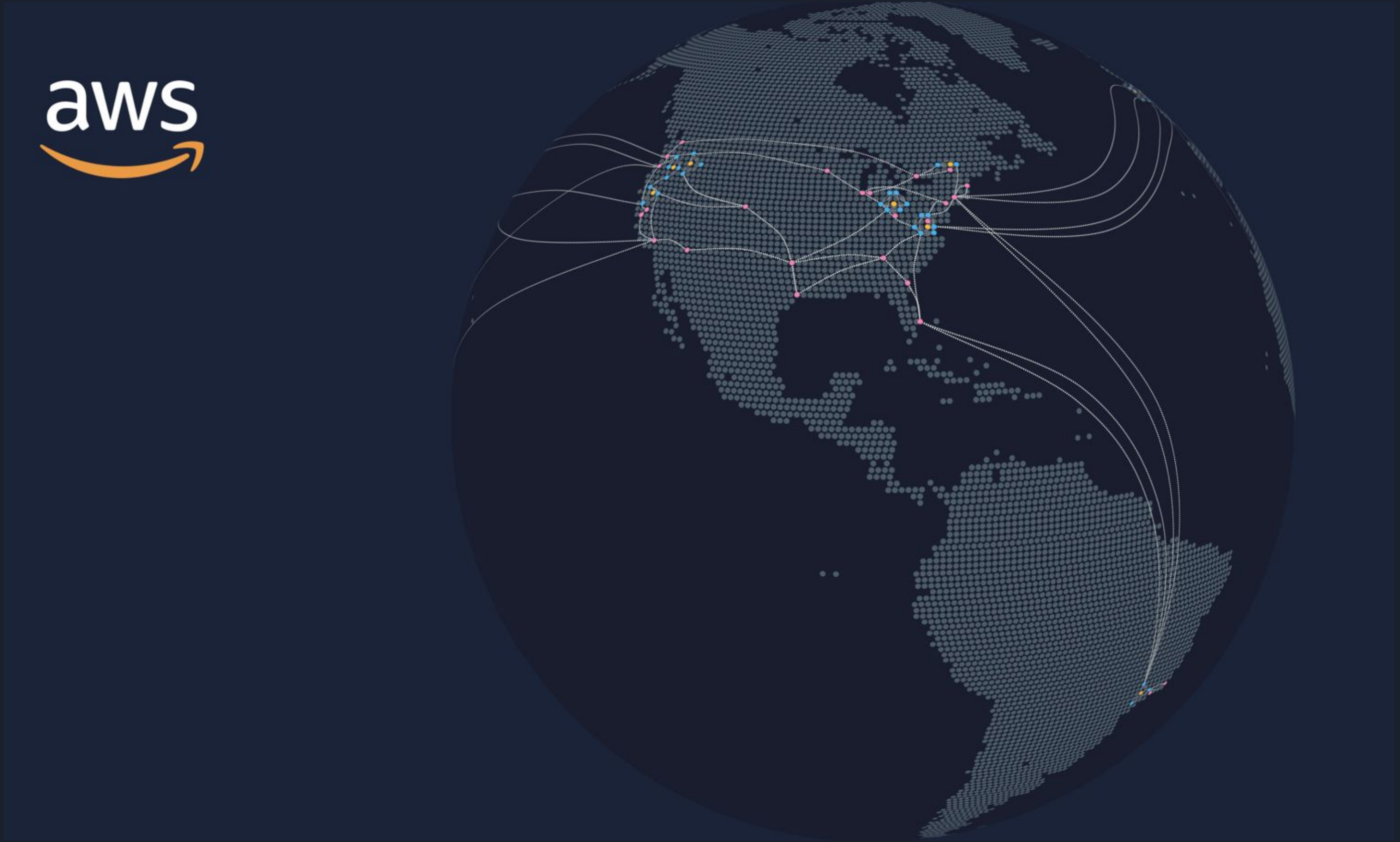
There is a growing need for edge computing however offloaded computation often occurs in the cloud far away from us.

Think about how many applications are located just in AWS which in the US has data centers in Northern Virginia, Ohio, Oregon, and Northern California.

There are limited options to move computation closer to the end user.



"Web programs running in the public cloud" - The Verge



AWS data center locations in gold
https://www.infrastructure.aws/

Novel Approach

Novel Approach

Let's use the Wireless AP/Gateway as a local resource for edge computing.

We will allow developers (service providers) to build application images that can be pushed to a user's AP.

We will provide a management layer and developer tools.

We will leverage standard tools like Ubuntu (Snappy) and Docker.

Why the Wireless AP / Gateway?

Modern APs/gateways are already quite powerful (and growing stronger).

They sit dormant most of the day.

Everyone has them (basically).

Within reach of all connected devices at the home.

They are always on.

Benefits for Developers (And Users)

Privacy
Sensitive data never leaves the home

Low Latency
Faster response times compared to cloud processing

Proprietary friendly
Virtual environment under developer's control

Local networking
No round trips to the cloud. Store data locally.

Additional wireless context
Can sense information about end-devices

Internet disconnectivity
Provide some mission critical service even during internet outage

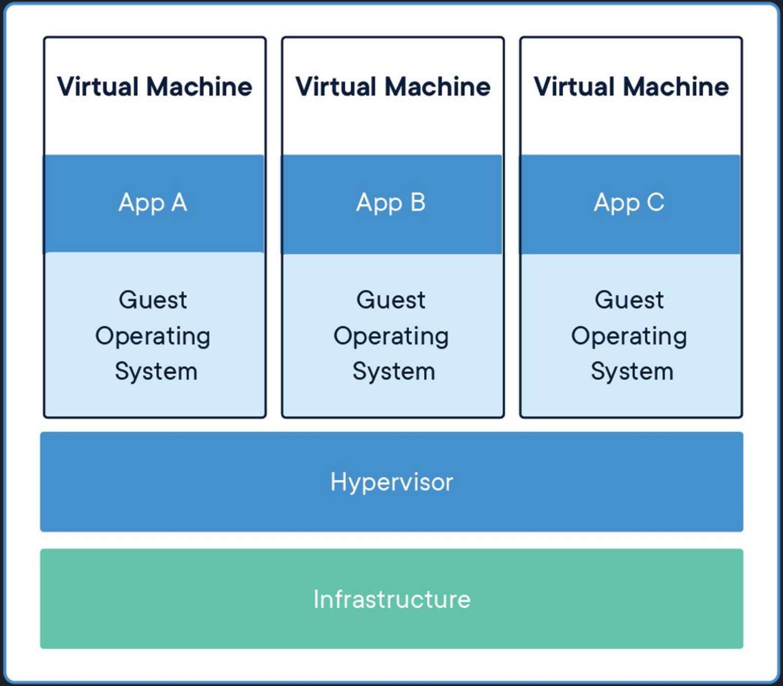
Background

What is a Virtual Machine?

A VM is focused on providing virtual hardware.

You need to install a full OS in the machine.

You run all the overhead of the virtual hardware and the OS.

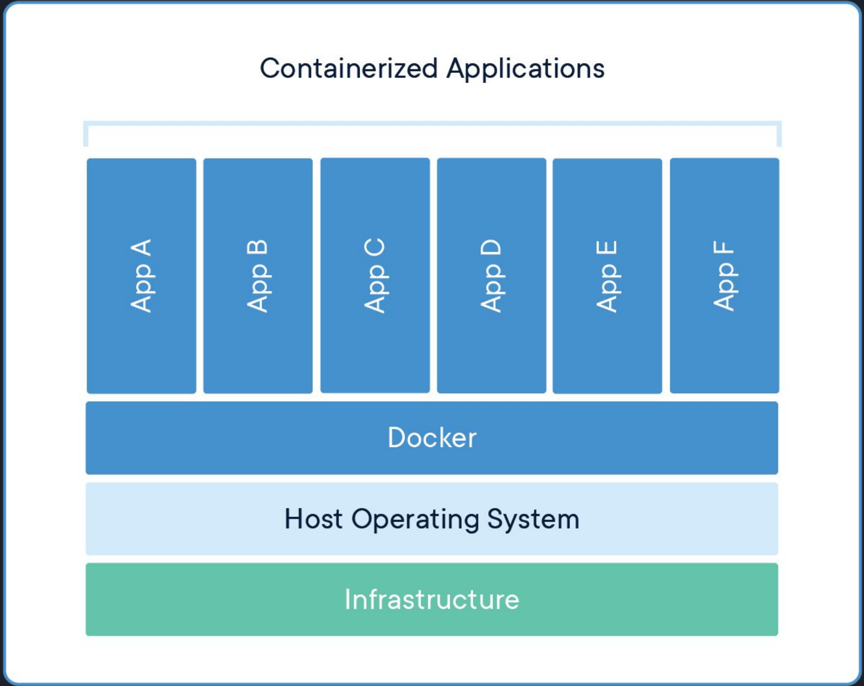


What is a Container?

A container is focused on providing a virtual operating system.

The OS kernel is shared (with Linux namespaces).

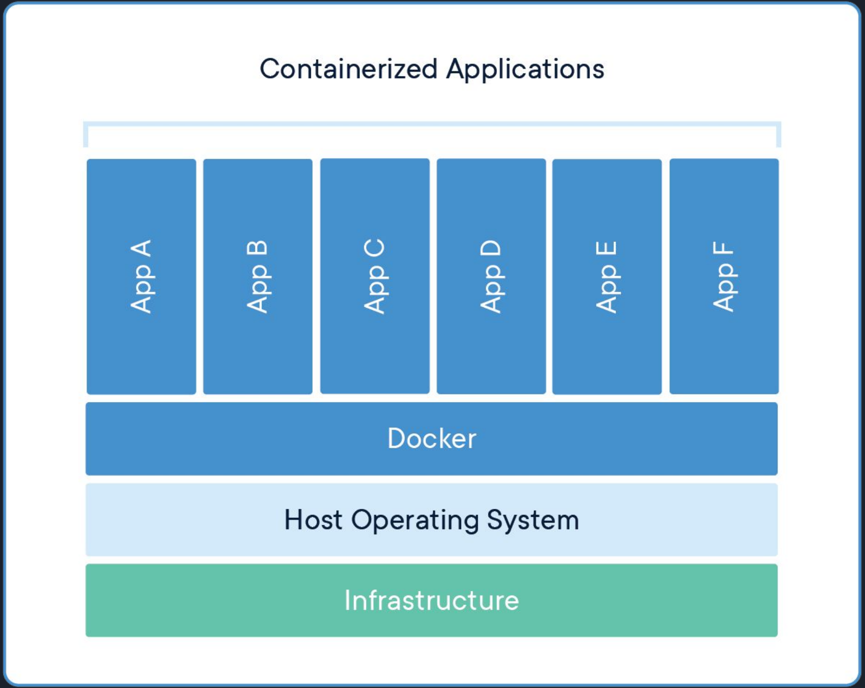
Less overhead and better performance. Less security, etc.



What is a Container (cont)?

Typically, a container runs one application process (application server, web server, database server, etc.)

Multiple containers make for a great microservices architecture.

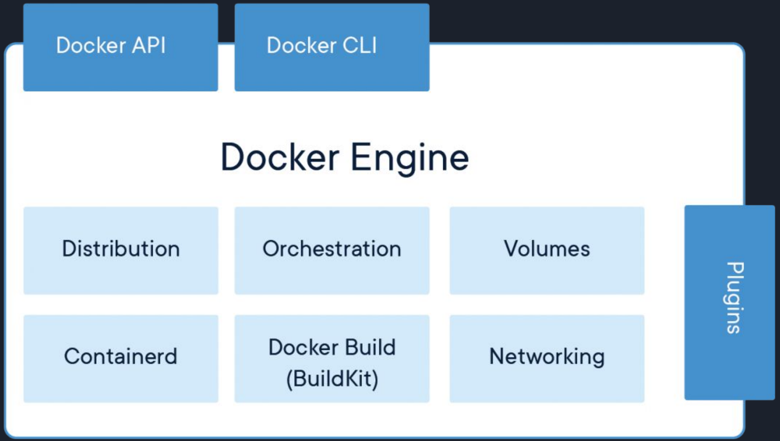


What does the Container Runtime do?

The runtime creates containers based on images.

It also manages the images and can usually retrieve them from elsewhere.

It handles networking, volumes, etc.



What about Container Orchestration?

You typically want a system for automating the management, placement, scaling and routing of containers.

Kubernetes is one of the most well known tools for this.

Proposed / Developed Solution

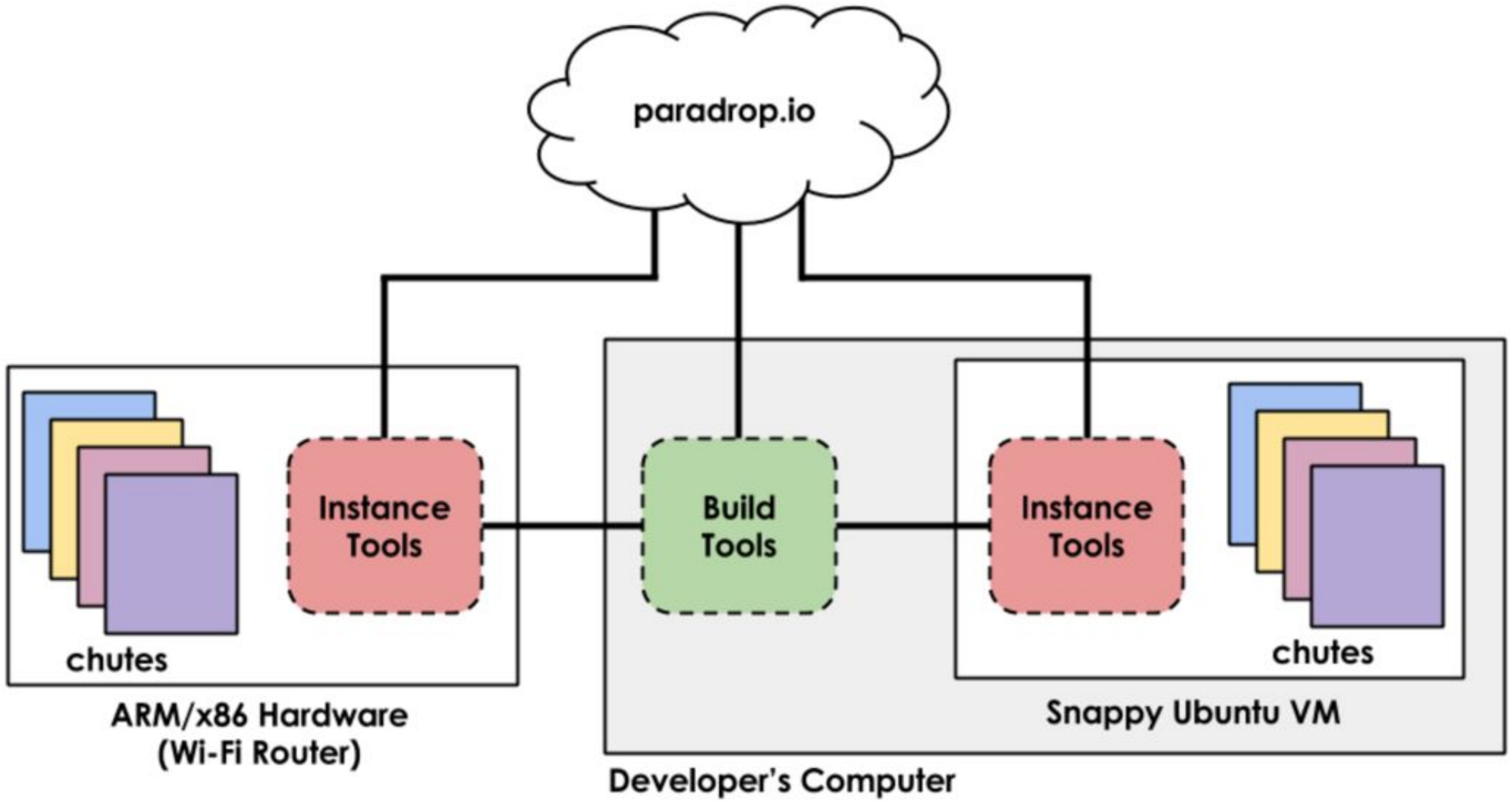
High Level Architecture

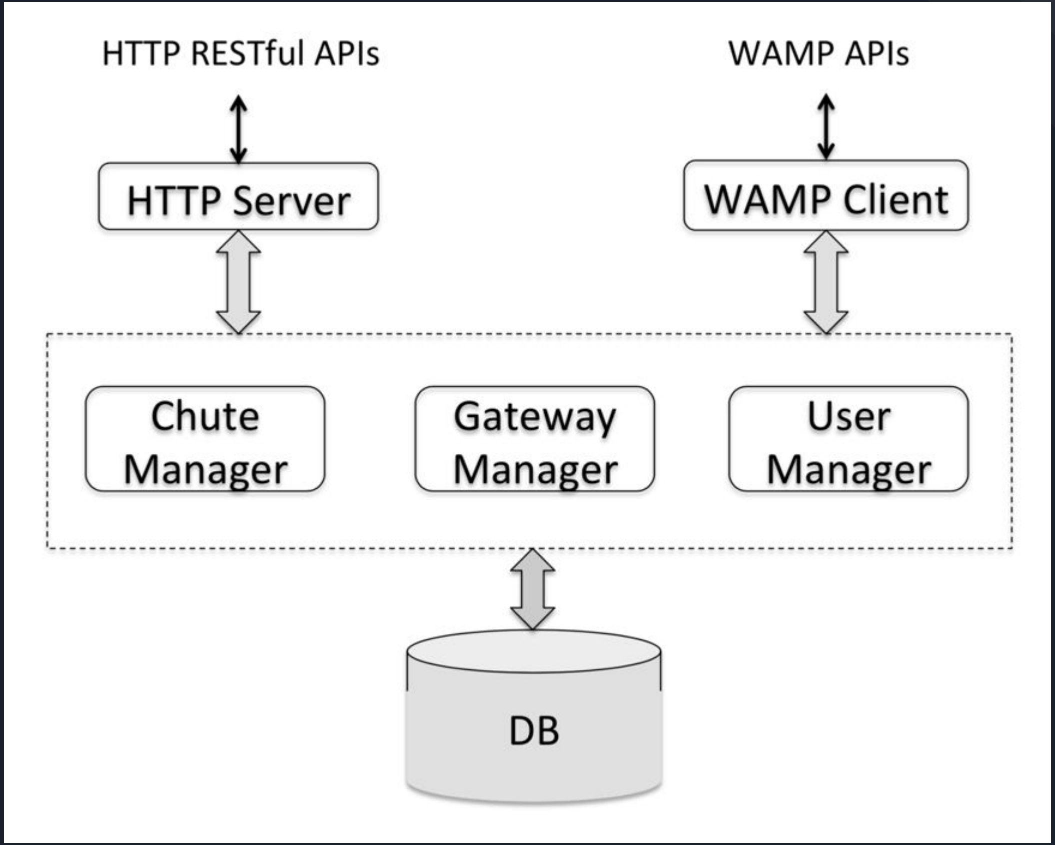
Access Points
The hardware/software solution in the home running containerized applications.

Cloud Management System
Provides centralized management and communication between developers and APs.

Developer Tools
Allow developers to build images and notify CMS of the end user to receive the “chute”.

Communication
Web Application Messaging Protocol (WAMP) used to communicate from CMS to AP. All other communications over HTTP.


paradrop.io
Instance Tools
Build Tools
Instance Tools
chutes
chutes
ARM/x86 Hardware (Wi-Fi Router)
Snappy Ubuntu VM
Developer's Computer



An alternate view of the architecture

Platform considerations

Installation should be as simple as possible; a user can add services to their gateway by simply registering a new account with an application developer

Developers should be able to provide services to their users easily, so a RESTful API is provided to control and configure services.

Resource management is done through the API; currently CPU, memory, and networking can all be managed dynamically by the developer.

Design Challenges

Virtualization
Containers were chosen over VMs due to superior performance and less overhead.

Application Management within the AP
WAMP message routing is used between the consoles and gateways. A local Paradrop daemon manages the AP.

AP Software Security and Maintenance
Used Ubuntu Snappy - a minimalist version of Ubuntu, it is a lightweight, transactionally updated OS designed for embedded and IoT devices.

Deployment Workflow

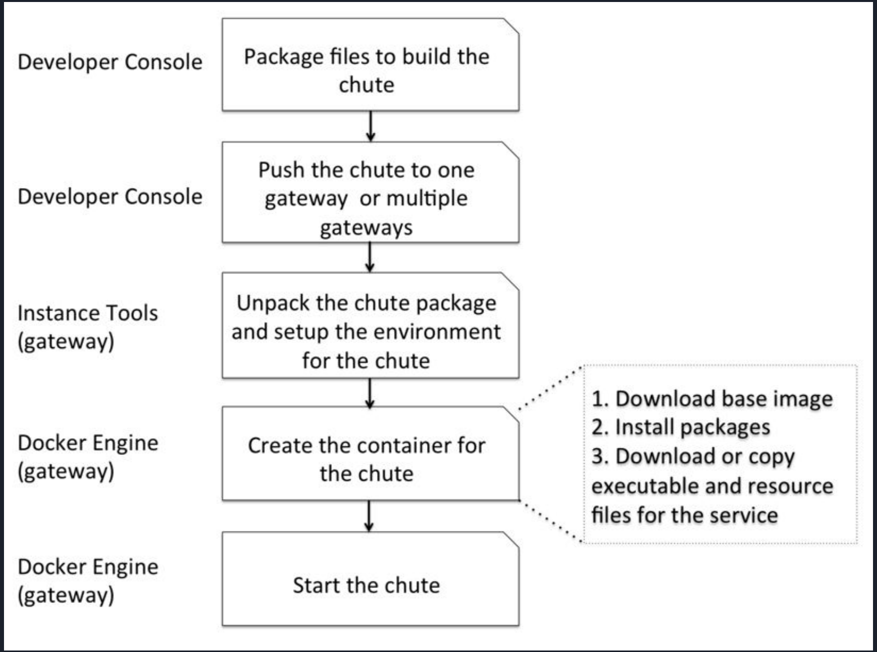
Developer creates application (chute)

Developer pushes to AP via the cloud manager.

Paradrop daemon receives deployment command and performs setup.

Paradrop daemon issues commands to Docker which provisions resources.

Docker starts up new container application (chute).


Developer Console
Package files to build the chute
Developer Console
Push the chute to one gateway or multiple gateways
Instance Tools (gateway)
Unpack the chute package and setup the environment for the chute
Docker Engine (gateway)
Create the container for the chute
1. Download base image
2. Install packages
3. Download or copy executable and resource files for the service
Docker Engine (gateway)
Start the chute

Access Points

The component local to the user containing a functional wireless gateway/AP as well as the ability to instantiate local applications for edge computing.

The AP is entirely under command of Paradrop cloud manager.

A Paradrop daemon runs locally to manage the OS, deploy applications (chutes), and all resources (routing,

Typical small board computing (SBC) hardware is envisioned.

Docker service installed through OS

Speaks to cloud manager through Web Application Messaging Protocol (WAMP)

Gateway Paradrop Daemon

A daemon that runs on the local gateway and manages Docker, controls AP services, and handles communication with the cloud manager.

Local network communication is usually HTTP while communication with the cloud manager is WAMP.

Paradrop daemon also controls the firewall, DHCP, WiFi, etc.

It also controls resource usage by the chutes.

Registers the gateway to the Paradrop backend.

Monitors gateway's status and reports to the Paradrop backend.

Receives RPCs and messages from the Paradrop backend and manage containers on the gateway accordingly, e.g. install, launch, stop, uninstall, etc.

Cloud Manager

Centralized management and middle man between the developers and APs. It communicates with all the gateways to dispatch commands and receive responses and status reports

Aggregates the information from all the gateways

Will eventually include a web frontend for visualization, user registration, chute installation, etc.

Stores information about the users, gateways and chutes in a MongoDB database

Still under development (at time of writing). A chute package must be available locally for the Paradrop developer console but future work would have the manager house the chute images.

Developer Tools

Allows the developer (service provider) to build and deploy chutes to end users.

Allows a developer to create chutes locally, upload them to the Paradrop backend.

Allows ability to install chutes to gateways that they have direct access to (local).

The created applications are basically Docker image definitions and support files with a Paradrop configuration file in YAML.

Resource Management

Resource policies are used to control the amount of CPU, network bandwidth, and RAM used by chutes.

CPU allocation is handled by Docker though Paradrop can provide direction.

CPU shares are in a chute's config and are given as abstract values with a default of 1024.

Network sharing is handled through the `tc` (traffic control) Linux utility as it provides for traffic shaping to limit bandwidth, etc.

Memory maximum is standard for all chutes.

A 1 GB limit on disk space is standard for all chutes.

Evaluation

Hardware

The evaluation hardware consisted of an off the shelf SBC was procured from PCEngines https://www.pcengines.ch/apu.htm.

Aside from network interfaces, WiFi, etc., it comes with an AMD APU 1GHz processor and 2GB of RAM.

Snappy Ubuntu was used for the local OS.

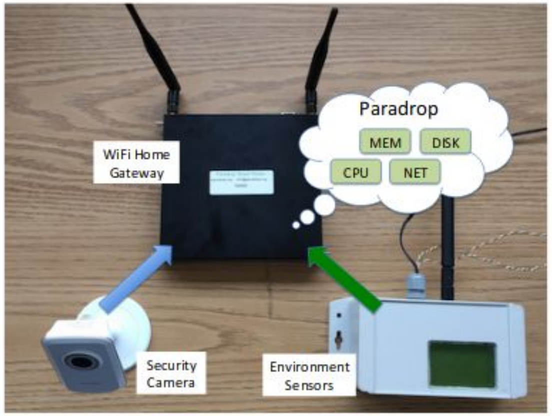
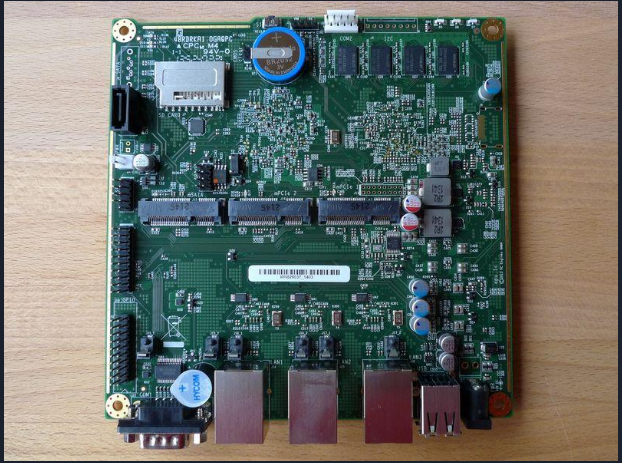


Fig. 2. The fully implemented Paradrop gateway, which shares its resources with two wireless devices including a Security Camera and an Environmental Sensor.



Sample Applications

SecCam

An application for introducing intelligent processing on video camera feeds.

Collects live video and analyzes for motion detection.

Implements user defined alerts.

EnvSense

Collect data from local environmental sensors.

After collection, it processes, stores, and visualized the data.

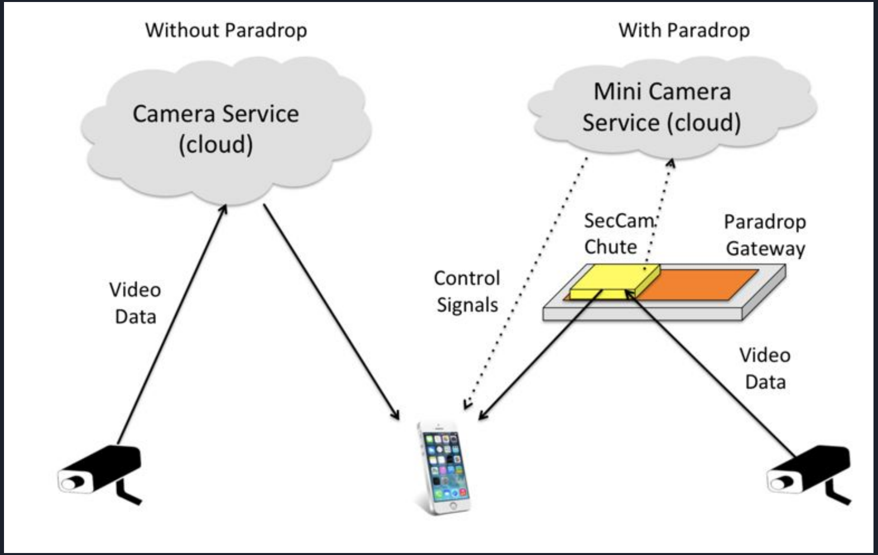
SecCam

SecCam

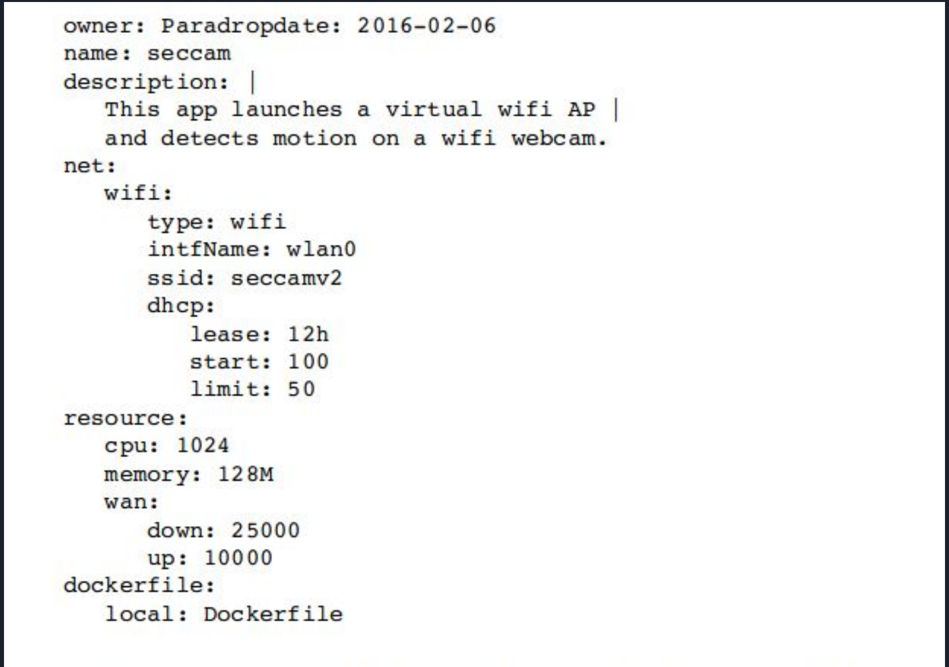
Motion detection using Python libraries

Visualization using PHP

Sensitive video never saved in the cloud

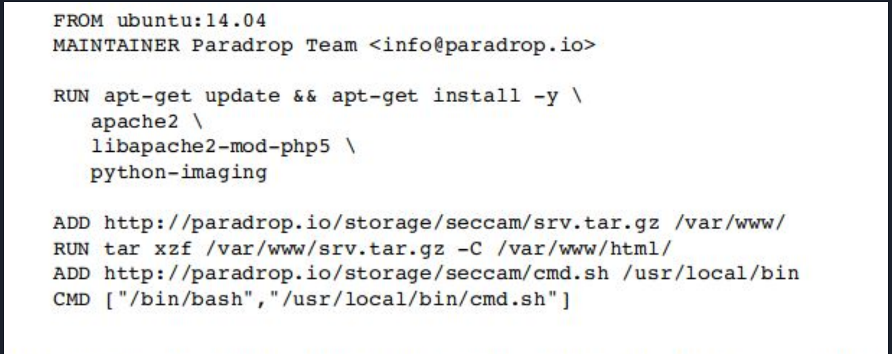


SecCam implementation



```
owner: Paradropdate: 2016-02-06
name: seccam
description: |
   This app launches a virtual wifi AP |
   and detects motion on a wifi webcam.
net:
   wifi:
      type: wifi
      intfName: wlan0
      ssid: seccamv2
      dhcp:
         lease: 12h
         start: 100
         limit: 50
resource:
   cpu: 1024
   memory: 128M
   wan:
      down: 25000
      up: 10000
dockerfile:
   local: Dockerfile
```

Fig. 9. The configuration file of the SecCam chute. It defines the environment of the container to run the chute.



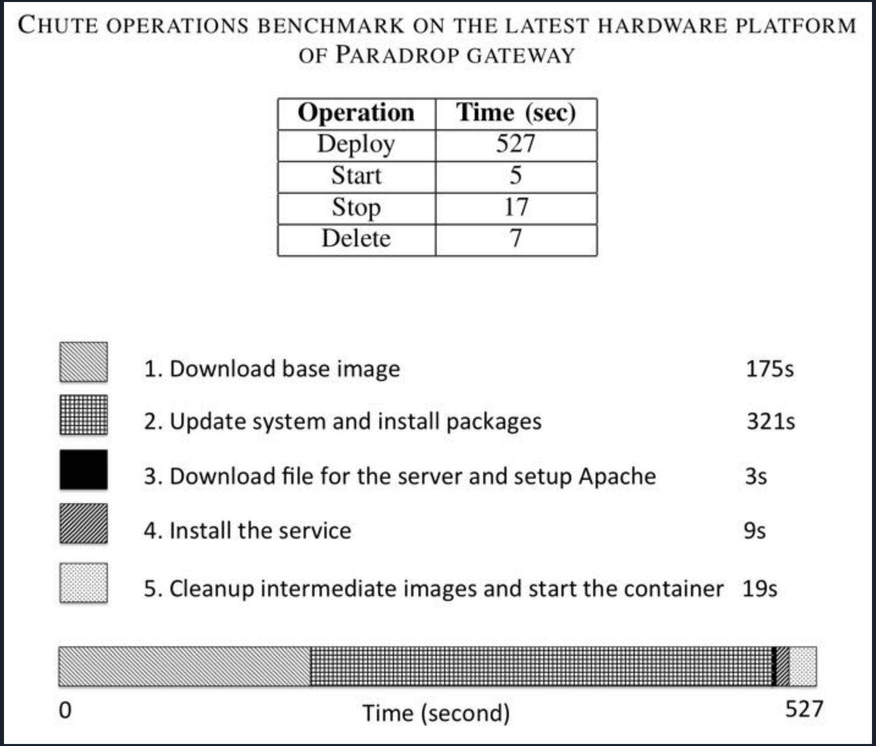
```
FROM ubuntu:14.04
MAINTAINER Paradrop Team <info@paradrop.io>

RUN apt-get update && apt-get install -y \
   apache2 \
   libapache2-mod-php5 \
   python-imaging

ADD http://paradrop.io/storage/seccam/srv.tar.gz /var/www/
RUN tar xzf /var/www/srv.tar.gz -C /var/www/html/
ADD http://paradrop.io/storage/seccam/cmd.sh /usr/local/bin
CMD ["/bin/bash","/usr/local/bin/cmd.sh"]
```

Fig. 10. The Dockerfile of the SecCam chute. This Dockerfile is a simplified version. We need to download some files for the chute from a web server. In deployment, these files can be included in the chute package.

Benchmarks for Chute Deployment


CHUTE OPERATIONS BENCHMARK ON THE LATEST HARDWARE PLATFORM OF PARADROP GATEWAY

Operation	Time (sec)
Deploy	527
Start	5
Stop	17
Delete	7

▨	1. Download base image	175s
▦	2. Update system and install packages	321s
■	3. Download file for the server and setup Apache	3s
▨	4. Install the service	9s
░	5. Cleanup intermediate images and start the container	19s

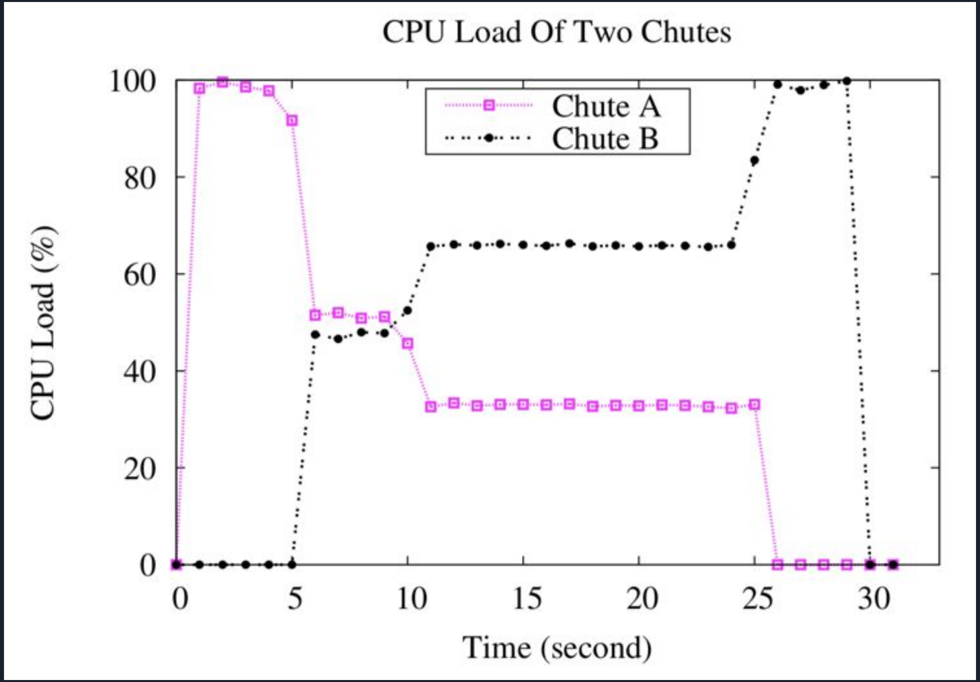
Deployment is broken down in lower half of graphic.

Test results highly dependent on network bandwidth.

The image is built "just in time" and then used to create the container.

You could alternatively pre-build the image and store in in a private repository to skip this phase.

Evaluation of CPU Resource Management



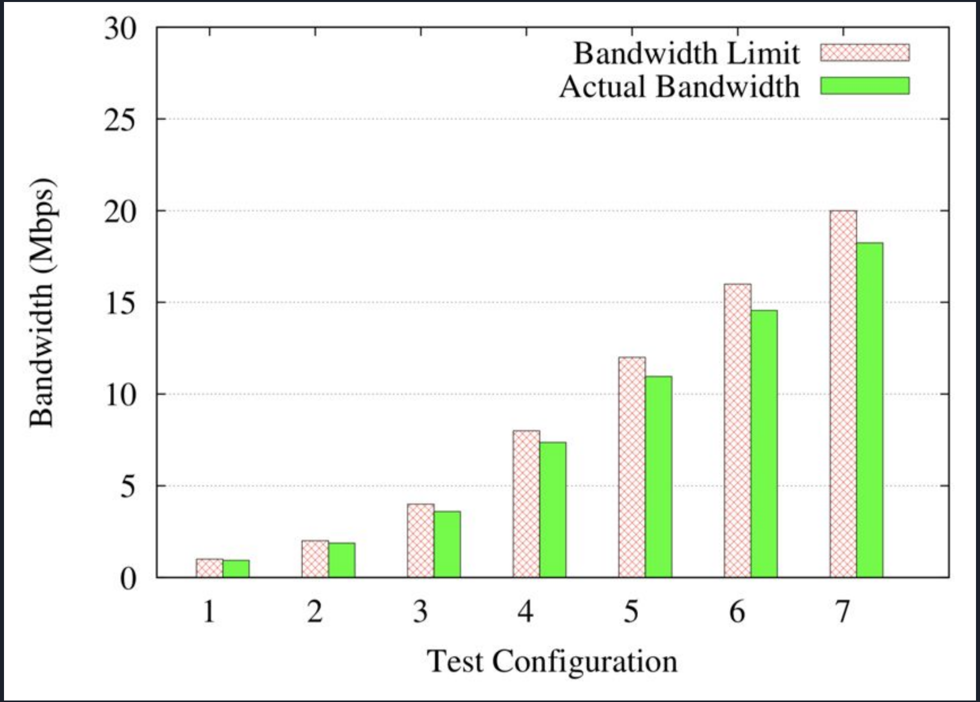
Chute A and B will attempt to use all available CPU when activated.

Chute A is a share of 512 and B a share of 1024. The values are abstract and only relative to each other.

As designed, once both chutes are online they content for resources.

Chute B correctly ends up with ⅔ of the CPU.

Evaluation of Network Bandwidth Management



The linux utility tc is used for traffic shaping to limit bandwidth use per chute.

Tests were conducted by transferring (HTTP) a 100MB file from chutes over ethernet.

Seven tests were performed each with a different limit.

Critique

Critiques

They are literally describing Kubernetes (initial release June 2014) and Swarm much of the time. This is probably due to the need for orchestration of Docker containers however it often feels like they are reinventing the wheel.

Previous versions of ParaDrop ran on OpenWRT but the switched to Ubuntu because "the operating system disparity on the gateway and the cloud platform could be an obstacle to develop or port applications" however this is literally the point of using containers.

The idea of containers at the edge are not new and companies have been rushing to fill this market for a while though none have gotten into the home. See k3s, KubeEdge, etc.

The demo system consists of a SBC by PCEngines that uses a 1Ghz AMD cpu with 2GB of RAM (https://www.pcengines.ch/apu.htm) on Ubuntu Snappy however they do not offer comparisons with popular home APs.

Critiques

The authors describe their system as "proprietary friendly" because it is running in a virtualized environment under their "complete control". However, this is hardware colocated with the user and is definitely not under their complete control and does not employ untrusted computing techniques.

Developers are overly smug about renaming containers as "chutes". This rebranding seems to imply that their contribution is more than orchestration of containers on APs. Many of their "challenges" they rely on Docker (or Ubuntu, etc.) to solve for them.

Authors seem a little hand-wavey and overly optimistic with some of their claims.

The authors built a REST API that passes JSON as if this is some ground breaking idea.