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

PENG LIU, DALE WILLIS, SUMAN BANERJEE
UNIVERSITY OF WISCONSIN - MADISON
Smart home apps
Should the cloud manage your home thermostats?
Should the cloud manage your home thermostats?
Solution:
Home gateways

The gateway which could be a home WiFi Access Point (AP) or a cable set-top box provided by a network operator is a platform that is continuously on, and due to its pervasiveness is a primary entry point into the end-user premises for such third-party services.

Advantages:

- Have decent processing power
- Gives local foot print
- Pervasive hardware
What is Paradrop??

Edge computing in the extreme

Providing a cloud-like abstraction in your home WiFi router
Paradrop Capabilities

- Privacy
- Low latency
- Proprietary friendly
- Local networking
- Additional wireless context
- Can work with Internet disconnectivity
How it works?
How it works?
Design challenges

1. Dynamic Installation – possible with help of chutes. If the service utilizes a wireless device, the gateway can fully integrate with the device without any interference from the end-user.

2. Paradrop API – In order for third party developers to provide high quality services RESTful paradigm allows developers to configure their chutes.

3. Network setup - The networking topology of a dynamic, virtualized environment controlled by several entities is very complex.

4. Resource Policies - The multitenancy aspects of Paradrop require tight policy control over the gateway and its limited resources.
Paradrop Components

- Virtualization substrate in the WiFi APs that host third-party computations in isolated containers (which we call, chutes)
- Cloud backend through which all of the Paradrop APs and the third-party containers are dynamically installed, instantiated, and revoked
- Developer API through which such developers can manage the resources of the platform and monitor the running status of APs and chutes
Gateway Implementation

- Should be capable of running multiple chutes on the gateway
- We want to provide an isolated and virtualized environment for the chutes deployed on the gateway.
- We want to keep the virtualization overhead minimum because the hardware platform of the gateway is not as powerful as the server in the cloud computing platform.
VM vs Containers

- Hypervisor-based VMs virtualize at the hardware layer, while containers virtualize at operating system layer.
- Hypervisor-based virtualization provides access to hardware only, we still need to install an operating system in the virtual environment, which quickly gobbles up resources on the gateway, such as RAM, CPU and bandwidth
- Docker adds features such as layered image and NAT, which ease the development, management and deployment of chutes.
Paradrop Gateway

[Diagram showing Paradrop Gateway components including APIs (WAMP and HTTP RESTful), Paradrop Daemon (autobahn and docker-py), Docker Engine, Chute A, Chute B, dnsmasq, and hostapd.]
Resource management

- Every chute needs to have the parameters to define the resource requirement in a config file.
- Paradrop supports policies for:
  1. CPU – specified by share of CPU resource that chute can use that one chute can use while competing with others
  2. Memory - The maximum memory size can be used by a chute is restricted by a value defined in its config file.
  3. Disk Size – Maximum disk size for the chute
  4. Network - The Paradrop daemon tracks all the network interfaces used by chutes, and restrict their speed by traffic shaping
Communication

- **WAMP** (Web Application Messaging Protocol) is an open standard Web Socket subprotocol that provides two application messaging patterns in one unified protocol: **Remote Procedure Calls** + **Publish & Subscribe**.

- Crossbar.io is a networking platform for distributed and microservice applications, implementing the open WAMP.
Two Interfaces

1. **WAMP API** – for Paradrop gateways.
2. **HTTP RESTful API** – for users, developers

Also stores persistent information about location of gateways, configuration, users etc. in a MongoDB
Developers view
Workflow

1. Download base image
2. Install packages
3. Download or copy executable and resource files for the service

- 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
- Docker Engine (gateway)
  - Start the chute
Resource management

1. CPU - every chute can use all the memory resource in the gateway.
2. Network - Docker does not provide the support to throttle bandwidth. For the chutes that need network interfaces, we use “tc” command to limit the bandwidth of these network
3. Memory - Paradrop gateway has an SD card with 16GB capacity. By default every container in the Docker can get 10GB of space.
Application: SecCam

- WiFi-based video camera
- DLink provides cloud service “mydlink” to which user can register their camera
- The applications for SecCam allow for motion detection from the webcam, user defined alerts, as well as visualization of the detected images
- We need to create a chute for the “SecCam” service, which has the following responsibilities:
  1. Create the SecCam SSID.
  2. Image capture service.
  3. Web server
Application: EnvSense

- This service is a wireless environmental sensor designed as part of the Emonix research platform.
- The original service runs in a server to collect data from the sensors, process and store the data, and visualize the data.
Chute operations benchmarking

TABLE II
CHUTE OPERATIONS BENCHMARK ON THE LATEST HARDWARE PLATFORM OF PARADROP GATEWAY

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy</td>
<td>527</td>
</tr>
<tr>
<td>Start</td>
<td>5</td>
</tr>
<tr>
<td>Stop</td>
<td>7</td>
</tr>
<tr>
<td>Delete</td>
<td>7</td>
</tr>
</tbody>
</table>

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

0 527 Time (second)
Shortcomings in paper

- Fault tolerance is not specified in the paper i.e. if the gateway fails or if the backend is not available to user.
- With docker being app container, can handle single process chutes. In case of major service with multiple micro services cannot be handled.
- Disk size is limited
- WAMP message router(crossbar.io) is the bottleneck for communications.
Questions?
References

- https://www.youtube.com/watch?v=yyuM7W-HjQo
- https://developer.ubuntu.com/core
- http://wamp-proto.org/