Compute-First Networking
Summary of Jan 24th - 25th 2019 Workshop
IETF 104, Prague
Berkeley Workshop on 1/24-1/25:

• A discussion of Compute-First Networking, trying to answer questions:
  • How CFN will enable new business models and new service scenarios (Who will sell what new services to whom?)
  • Gap Analysis/problem statement (Why the incremental extension of current practice is not enough?)
  • Technical proposals (what architecture/solution is required?)
  • Other questions that you would like to discuss?
CFN: the big picture
Edge Computing location in the eye of China Mob

Facility: Enough central offices fully distributed at the edge locations

<table>
<thead>
<tr>
<th>Distance From UE</th>
<th>1km</th>
<th>10km</th>
<th>50km</th>
<th>200~300km</th>
<th>500~1500km</th>
<th>2000~3000km</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (us)</td>
<td>10us</td>
<td>100us</td>
<td>500us</td>
<td>1500us</td>
<td>15000us</td>
<td>40000us</td>
</tr>
</tbody>
</table>

- **100K*10=1M Server**
- **3K*60=180K Server**
- **300*197=60K Server**

Source: ONS/Europe September 2018
CFN architectural elements:

Includes both control plane and forwarding plane.

- APP
- Data1
- Data2
- CFN
- IP Underlay

Smart phone

- Compute Resource
- func1
- func2
- Data 3

Smart Home Gateway/Edge

- Lambda Store as unikernels/VM
- Unikernel container/VM as a Service

- Data1 read
- write

- CFN
- IP Underlay

- Smart Central Office/Cloud DC

- Compute Resource
- func3
- func4
- Data 4

- Lambda Store as unikernels/VM
- Unikernel container/VM as a Service

Request Func1 (input:Data1, Output:Data2)
CFN: an in-network function deployment layer?

- Goal of CFN is to run function in a way that maximizes the end user’s QoE
  - Select the better location so that the response time is optimized
  - Taking into account: processing time, network delay, availability of the resources, availability of the function
  - Need to support a wide range of functions and deployment scenarios

- Key question: how to instantiate CFN in a practical, efficient, economical way?
What are the requirements for CFN?

| - Expressivity of the service request | - Dynamic provisioning |
| - Ability to instantiate a wide range of services/programmability | - Agnostic to network technology and to computing platform technology |
| - Security (for user, for network) | - Incremental deployment/migration mechanism |
| - Optimization of the resource allocation (optimization objective set by operator) | - Mobility support |
| - Support for a wide range of applications | - Meets delay requirements |
| - Scale (applications/users) | - Distributed deployment |

* Can we design a solution that meets all these requirements?
* Can we build upon a solution that partially meets these requirements and expand it to meet the missing ones?
  - if we build upon an existing solutions, do we need to add other requirements (existing customer base, open source, ease of implementation)
CFN: top down or bottom up?

Is CFN

- a network solution that is enhanced with the ability to deploy functions
  Or
- a service layer that is enhanced with network awareness
  ?
  (or something else altogether)

Is it build by adding function execution environment to network router, or by adding network APIs to a managed function deployment environment
CFN architectural elements:

CFN-Net: monitors network delay and congestion; updates network controller; route to service instance; CFN-SL: service layer monitors server/CPU allocations, function performance, server load

Request Func1 (input: Data1, Output: data2)
Top-down approach: Serverless architectures?

- Current widely deployed architecture to run services on demand in the cloud
Lambda Use-cases: Mobile Backend

1. Mobile users retrieve an identity from Amazon Cognito, which offers mobile identity management and data synchronization across mobile devices. Once a mobile user has retrieved an identity, the user is granted access to other AWS services.

2. User-generated media files are stored in Amazon Simple Storage Service (Amazon S3), a highly available and durable storage service.

3. Mobile users can access their uploaded digital assets stored in Amazon S3 through Amazon CloudFront, a low latency, content delivery network.

4. Mobile users send requests to Amazon API Gateway to access application logic and dynamic data. API Gateway acts as an entry point for mobile applications to access functionality from code running on AWS Lambda.

5. Mobile applications require a highly scalable backend infrastructure to support the variable usage created by mobile users. AWS Lambda runs code in response to requests and automatically manages and scales the underlying resources. Lambda Function 1 provides a synchronous endpoint for users to store and retrieve unstructured data from Amazon DynamoDB.

6. Lambda Function 2 uses Amazon DynamoDB streams to retrieve changes made by users, creates a searchable document, and inserts it into Amazon CloudSearch.

7. Lambda Function 3 provides a synchronous interface for users to search for data from CloudSearch.

8. Lambda Function 4 provides an asynchronous endpoint for mobile users to communicate with each other within a mobile application. The function formats each communication request and sends a push notification to specific users with Amazon SNS.

Source: https://aws.amazon.com/lambda/resources/reference-architectures/

Is this CFN-SL?
Bottom-Up Approach: Named-Function Networking

- Leverages the abstraction of Information-Centric Networking, where the networking layer is able to route content by name.
- Named Data Networking exchanges data using the data name as the network handle.
- NFN generalizes this to include function names as well, so that a network request carries multiple names:
  - `compute(/name/of/fct, /name/of/arg)`
- Up to the network to forward request (for `fct` and `arg`) to the node that will execute the function.
Our preliminary attempt at implementing some form of CFN

- Built on top of Sapphire, a distributed programming platform built for mobile/cloud applications
- Focuses on application developers need to specify application logic and deployment logic in separate but structured way
- Deployment scenarios range from replication and client-side caching to dynamic code offloading
- Any node in the cloud or edge, along with user devices can provide execution environment – provided they run sapphire components
- Research prototype level code publically available
- Implementation by Aziz Albalawi (UCSC) and Asit Chakraborti
CFN controller architecture

- **Cloud**
  - Controller
  - Resource
- **Edge**
  - Sapphire
  - Object
  - User
  - Device
- **Network performance**
  - BW utilization, RTT

**Resource**

<table>
<thead>
<tr>
<th>ID</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host1 CPU</td>
<td>50%</td>
</tr>
<tr>
<td>Host2 Mem</td>
<td>100GB</td>
</tr>
</tbody>
</table>

**Object**

<table>
<thead>
<tr>
<th>ID</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>IP1</td>
</tr>
<tr>
<td>O2</td>
<td>IP2</td>
</tr>
</tbody>
</table>

**Cluster Resource**

- CPU 10%, Mem 20%, …
CFN controller design choices…

- How to allocate requests based upon service requirements and availability.
- For instance, for a latency sensitive application CFN can place the workload based upon network latency
- CFN: latency-aware, vs other workload placement policy:

CFN vs non-CFN

RTT

0 10 20 30 40

Requests

Clie

non-CFN Edge non-CFN Cloud non-CFN RR CFN

cloud

Edge1 Edge2
CFN controller design choices…

- Multihoming scenario:
- CFN can decide which server to allocate based upon performance

**Multihoming**

![Graph showing RTT for CFN and non-CFN (RR)]
Questions to consider

- How to deploy services at the edge (heterogeneous devices within different administrative domains): is there a simple and robust way to deploy applications?
- What compatibility with the current network layer?
- How to abstract the network so that application developers do not worry about it?
- Security models, distributed trust, delegation of security?
- Mobility – of user and of function?
- What are the next steps: towards standardization?
# Agenda:

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>9am-10am</td>
<td>Breakfast, welcome, intro, motivation – Cedric Westphal (Huawei) &amp; Dirk Kutscher (Emden University)</td>
</tr>
<tr>
<td>10-am-12pm</td>
<td>Use-case session – What are the business models? What are the applications for CFN? Marie-Jose Montpetit - COIN Ignacio Solis (LinkedIn) – Stream processing Alexey Tumanov (UC, Berkeley) - <a href="https://ray.io">Ray</a> software project and prediction serving work and latency constraints Khachik Sahakyan (Grovf) – Accelerated computing using FPGA on premise, at edge or in the cloud</td>
</tr>
<tr>
<td>12pm-1:30pm</td>
<td>Lunch</td>
</tr>
<tr>
<td>1:30pm-3:30pm</td>
<td>Gaps - What is missing today? Where are we now? Quinton Hoole (Huawei) – Amino open source project Zack Butcher (Tetrate/CNCF) - Overview of Envoy and application to operators Igor Tarasenko (Bayware) – Network Microservices</td>
</tr>
<tr>
<td>3:30pm-4pm</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>4pm-5pm</td>
<td>Discussion</td>
</tr>
<tr>
<td>5pm-6pm</td>
<td>Reception</td>
</tr>
</tbody>
</table>
### Agenda:

**Friday:**

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>9am-11am</td>
<td>Technical solutions – Going forward – What is the research community looking at?</td>
</tr>
<tr>
<td></td>
<td>Stratis Ioannidis (Northeastern University) – In-Network Caching</td>
</tr>
<tr>
<td></td>
<td>Alex Afanasayev (Florida International University) – NDN as Compute-First Networking</td>
</tr>
<tr>
<td></td>
<td>KK Ramakrishnan (UC, Riverside) – Scheduling in NFV - OpenNetVM</td>
</tr>
<tr>
<td></td>
<td>Dirk Kutscher (Emden University) – Remote Method Invocation in ICN</td>
</tr>
<tr>
<td>11am-12pm</td>
<td>Discussion moderated by Dave Oran</td>
</tr>
<tr>
<td>12pm-1pm</td>
<td>Lunch</td>
</tr>
</tbody>
</table>
Talks at the workshop: Use case session

■ COIN and its motivations (M-J. Montpetit)

Go to the BOF!

■ Stream Processing at LinkedIn (Ignacio Solis)

how LinkedIn relies on stream processing, and on Kafka in particular. The numbers are staggering: 4.5B messages. Stream processing touches pretty much everything, including the end user’s activities, security, monitoring, infrastructure, etc.

■ Ray from the RISE Lab (Alexey Tumanov)

Ray: a library of tools for distributed machine learning. It is one system that makes it possible to implement functionality as libraries over a distributed eager dynamic task graph.

■ Accelerated computing using FPGA (Khachik Sahakian, Grovf)

use cases for FPGA acceleration at the edge significantly overlap with CFN use cases: 1- autonomous driving; 2- edge AI; 3- industrial IoT; 4- M2M communication; 5- RF/DSP; 6- in memory caching; 7- network edge firewalls; 8- High Frequency Trading; 9- real-time network monitoring/filtering; 10- real-time control systems
Session 2:

- **Amino Open Source Project (Quinton Hoole, Huawei)**
  Open source product-grade version of Sapphire, distributed programmable platform

- **Network Micro Services (Igor Tarasenko, Bayware)**
  an API to program network flows that express the intent of application services across an overlay network. This is done by micro-service packaging into IPv6 extension header. Permissions to run microprogram are granted via 100-byte certificate; a graph description supports 12 and 20-bit segment identifiers and 8-bit program counters
Session 3: Research

- Stratis Ioannidis (Northeastern University) – In-Network Caching
  routing and caching are jointly optimized; mathematical framework to formulate the
  optimization problem and can solve the optimization within \((1-1/e)\) of the optimal in a
  distributed manner

- KK Ramakrishnan (UC, Riverside) – Network scheduling of workloads
  OpenNetVM: an open source project to implement NFV; NFVnice: an NF scheduler that
  combines hardware packet schedule, and software process scheduling. While this was in the
  context of NFV, service chaining (and scheduling of functions) would be required in CFN

- Alex Afanasayev (Florida International University) – NDN support for CFN
  NDN as a substrate for CFN. Application naming at the network layer can re-use the content
  naming semantics. The recursivity of the content recovery gives NDN an advantage.
Thank You