OPEN NETWORK EDGE SERVICES SOFTWARE (OPENNESS): APIS

Edge Computing and Ecosystem Enabling
Network Platforms Group (NPG)
Topics

• Quick Overview of OpenNESS
  • High performance, secure, modular edge platform
  • Microservices-based
  • Customizable platform by selective use of services

• Classes of interfaces
  • RESTful interfaces
  • Protobuf interfaces
  • “Kubernetes” interfaces
Edge Definitions

Edge Computing: Placement Of Data Center-Grade Network, Compute and Storage Closer to the Endpoint Devices

- Comply with data locality
- Reduce application latency
- Meet reliability requirements
- Deliver rich user experiences
- Optimize TCO
- Improve service capabilities
OPENNESS: WHAT DO YOU GET

An Optimized and Cohesive Framework for Managing Apps and Services

OPENNESS EXPERIENCE KIT:
Available at openness.org

- Microservices
- Sample applications (with relevant SDKs integrated – eg. Smart city)
- SDKs (OpenVINO, IPP, MKL, DPDK...)
- OS Kernel and bios configurations
- Reference hardware specs
- Performance benchmarks (in the future)
- Ansible Playbooks
## Intel’s Edge Platform Offering

<table>
<thead>
<tr>
<th>VIRTUAL NETWORK FUNCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPENNESS (Open Source), Intel® Smart Edge (Commercial distribution of OpenNESS)</td>
</tr>
<tr>
<td>OTHER DATA-CENTRIC APPS (analytics, cloud)</td>
</tr>
<tr>
<td>OPENVINO Open Visual Cloud</td>
</tr>
<tr>
<td>Vertical Recipes (Retail, Industrial, Healthcare...) &amp; Solutions (CERA, EIS)</td>
</tr>
<tr>
<td>Services (IOT, Comms, Cloud, Enterprise)</td>
</tr>
<tr>
<td>RESOURCE ORCHESTRATION AND VIRTUALIZATION</td>
</tr>
<tr>
<td>NETWORK PLATFORM INFRASTRUCTURE SOFTWARE (DPDK)</td>
</tr>
<tr>
<td>EDGE HARDWARE PLATFORM</td>
</tr>
</tbody>
</table>

**Multi-Radio LTE/5G/Wi-Fi**

**Cloud**
OpenNESS Edge Node* can be deployed on Network Edge or On-Premise Edge
OpenNESS Functionality: A Closer Look (1/2)

<table>
<thead>
<tr>
<th>Data Plane</th>
<th>Multi-Access Networking</th>
<th>Enhanced Platform Awareness (EPA)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OVN/OVS-DPDK</strong></td>
<td><strong>5G – AF, NEF</strong></td>
<td><strong>NFD (Node Feature Discovery)</strong></td>
</tr>
<tr>
<td>High-performance Data Plane</td>
<td>3GPP Network function microservices enabling deployment of</td>
<td>Kubernetes add-on that detects and advertises hardware and software capabilities of a platform</td>
</tr>
<tr>
<td>microservices supporting</td>
<td>edge cloud in a 5G network</td>
<td>that can, in turn, be used to facilitate intelligent scheduling of a workload.</td>
</tr>
<tr>
<td>a Container Network</td>
<td><strong>4G-LTE OAM</strong></td>
<td><strong>TAS (Telemetry Aware Scheduler)</strong></td>
</tr>
<tr>
<td>Interface (CNI) and can</td>
<td>Microservice extending the capability of 3GPP 4G core network</td>
<td>Making available hardware and software Telemetry data for scheduling and de-scheduling decisions</td>
</tr>
<tr>
<td>be managed by a</td>
<td>enabling deployment of edge cloud</td>
<td>Kubernetes.</td>
</tr>
<tr>
<td>standard SDN controller</td>
<td><strong>CNCA (Core Network Configuration API)</strong></td>
<td><strong>CMK (CPU Manager for Kubernetes)</strong></td>
</tr>
<tr>
<td>via Open Flow</td>
<td>REST Based interface for configuring 4G and 5G Network</td>
<td>Kubernetes plugin that provides core affinity for applications deployed as Kubernetes pods</td>
</tr>
<tr>
<td>Functions in edge cloud</td>
<td><strong>Multus</strong></td>
<td><strong>Topology Manager - NUMA</strong></td>
</tr>
<tr>
<td>deployments</td>
<td>Container network interface (CNI) plugin for Kubernetes that</td>
<td>Solution permitting CPU Manager and Device Manager, to coordinate the resources allocated to a</td>
</tr>
<tr>
<td></td>
<td>enables the attachment of multiple network interfaces to</td>
<td>workload.</td>
</tr>
<tr>
<td></td>
<td>pods.</td>
<td><strong>RMD, SST-BF (QoS)</strong>*</td>
</tr>
<tr>
<td>*Coming soon as part of</td>
<td></td>
<td>Provide a central uniform interface portal for hardware resource management tasks on x86</td>
</tr>
<tr>
<td>future enhancements</td>
<td></td>
<td>platforms.</td>
</tr>
<tr>
<td><strong>OpenNESS Functionality: A Closer Look (2/2)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Accelerator</strong></td>
<td><strong>Application</strong></td>
<td></td>
</tr>
<tr>
<td><strong>FPGA Orchestration</strong></td>
<td><strong>Service Mesh</strong></td>
<td></td>
</tr>
<tr>
<td>Enable Programming, Configuration and resource allocation of FPGAs</td>
<td>Provide a common message bus for applications and services on the platform to publish and subscribe.</td>
<td></td>
</tr>
<tr>
<td><strong>HDDL Orchestration</strong></td>
<td><strong>DNS</strong></td>
<td></td>
</tr>
<tr>
<td>Enable allocation of Intel® Movidius™ Myriad™ X High Density Deep Learning (HDDL) cards to cloud native applications</td>
<td>DNS microservices with forwarding capability for edge applications and network functions</td>
<td></td>
</tr>
<tr>
<td><strong>VCA Orchestration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable allocation of Intel® Visual Compute Acceleration cards to Cloud native applications</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Multi Cloud</strong></td>
<td><strong>Security</strong></td>
<td></td>
</tr>
<tr>
<td>Enable deployment of Public Cloud IOT gateways on Edge platform</td>
<td><strong>Securely Store Data</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enable Applications to securely store and retrieve data</td>
<td></td>
</tr>
<tr>
<td><strong>Isolate Secure Data</strong></td>
<td><strong>Isolate Secure Data</strong></td>
<td></td>
</tr>
<tr>
<td>Enable Isolation of data from application instances on the edge node</td>
<td><strong>SGX (Software Guard Extensions)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enable allocation of Key storage and secure enclave resource to applications</td>
<td></td>
</tr>
<tr>
<td><strong>SGX (Software Guard Extensions)</strong></td>
<td><strong>RBAC (Role-Based Access Control, PKI (Public Key Infrastructure))</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configure fine-grained and specific sets of permissions that define how a user, or group of users, can interact with the system objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ISEC-L</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Intel® Security Libraries extend the concept of the &quot;chain of trust&quot; using a remote Verification Service to verify the measured server components against previously-stored known-good measurements</td>
<td></td>
</tr>
</tbody>
</table>

*Coming soon as part of future enhancements*
OpenNESS follows a microservice architectural style:

- gRPC used as underlying API framework
- RESTful APIs provided for integration with external subsystems and apps
- Microservices implemented in a split Service/Agent (in Controller/Platform) style for resilience
- External interfaces are also supported
  - Open source APIs (e.g., OpenStack, Kubernetes)
  - Network Interfaces (e.g., sockets, DPDK)
  - LTE Control and User Plane interfaces

OpenNESS Microservices Architecture

- Edge Cloud Application (Container)
- Edge Cloud Application (Container)
- EAA
- DNS Server
- Data plane
- Data plane NIS
- Traffic rules
- Socket
- EPC User Plane
- EPC Control Plane
- PDN
- Local Breakout
- HTTP/REST
- gRPC

**CentOS\* + Docker\* Container Runtime**

**Hardware:** CPU, Memory, NIC, Accelerator, and FPGA

**OpenNESS Controller Community Edition**

**Microservices**

- GUI
- EVA Controller
- ELA Controller
- Core Network Configuration Agent (CNCA)
- Authentication
- Telemetry
- Policy Database

**OpenNESS Edge Platform**

- MEC App using cloud connector (AWS\* Greengrass (Docker\* container)
- App and Device Authentication
- HTTPS/REST
- gRPC

**Controller Gateway**

- Authentication

**Cloud/Host Deployable**

- Kubernetes\* Master

**K8s based pod deployment**

**Unmodified Standard APIs**

*Other names and brands may be claimed as the property of others*
Platform/Controller Gateway

- API gateway for Edge Platform and controller communication.
- Controller and agent microservices communicate through these gateways.

EDGE PLATFORM ARCHITECTURE

**OpenNESS Edge Platform**

- Edge Application (Container)
- Edge Cloud Application (Container)
- DNS Server
- Data plane
- Traffic rules
- EAA
- EPC User Plane
- EPC Control Plane
- PDN
- Local Breakout
- HTTPS/REST
- Unmodified Standard APIs

**OpenNESS Controller Community Edition**

- EAA
- DNS Server
- EVA and ELA
- EDA
- Appliance Gateway
- Controller Gateway
- Authentication
- GUI
- EVA Controller
- ELA Controller
- Configuration Agent (CNCA)
- Authentication
- Telemetry
- Policy Database
- Kubernetes* Master
- PDN
- Local Breakout
- HTTPS/REST
- Unmodified Standard APIs

**Microservices**

- GUI
- EVA Controller
- ELA Controller
- Core Network Configuration Agent (CNCA)
- Authentication
- Telemetry
- Policy Database
- Kubernetes* Master
- HTTPS/REST
- Unmodified Standard APIs

**Hardware**

- CentOS* + Docker* Container Runtime
- Hardware: CPU, Memory, NIC, Accelerator, and FPGA

*Other names and brands may be claimed as the property of others
Edge Application Service/Agent

Allows applications to act as Edge Services
Acts as a message bus for inter-application communication
• Service registration
• Service discovery
• Communication support for services
• Application availability

• Edge Application API: EAA

*Other names and brands may be claimed as the property of others
Edge Lifecycle Service/Agent

Manage configuration of the Edge Platform deployment

• Enroll Edge Platform with controller and create certificate

• Configure edge platform network ports for use in traffic steering

Configure Application rules

Support application authentication for application services

Carry out application lifecycle support procedures

• Edge Lifecycle Management API: ELA

*Other names and brands may be claimed as the property of others
Edge Virtualization Service/Agent

Manage virtualized resources on edge platform

Ensure proper operation both in presence and absence of Virtualization Manager (i.e., OpenStack, Kubernetes)

- Instantiate, start, stop containers/VMs via platform virtualization manager if present in the deployment environment
- Instantiate, start, stop containers/VMs directly via libraries if virtualization manager is not present

**OpenNESS**

**Microservices**

Dataplane traffic from LTE/IP Network

REST API calls

gRPC calls

Reference 4G EPC stack

Producer/Consumer Edge cloud application

Open source stack, modules and libraries

*Other names and brands may be claimed as the property of others*
Controller APIs

APIs implemented by the Controller

Used by Orchestrator or (web-based) UI application to carry out activities on the system

Objects controlled by APIs are:

- **Node**: refers to an edge platform; API performs an operation on a platform
- **Application**: refers to an application image managed by controller.
- **Policy**: refers to a traffic policy (to be applied to an application or to a node)
- **Interface**: refers to a network interface on a node, to be configured
- **DNS**: refers to a DNS configuration (combination of "A" records and forwarding rules) to be applied to a node

---

**On Premises Controller API: EAA**
Configuring the Edge App in the 5G Network

**Core Network Configuration API**

**gNB identifiers and data elements**
- User Location Information
- UPF IP Address
- GTP-U Tunnel ID for UPF

**UPF identifiers and data elements**
- **App info:**
  - App ID
  - Traffic Filters
- **UE info:**
  - IP Address (PDU layer)
  - MAC address
  - GPSI (Generic Public Subscriber Identifier)
  - Any UE or group of UEs (selected)

**DN identifiers and data elements**
- DNN (Data Network Name)
- Route (IP address)
- GTP-U Tunnel ID

**System Operator**
- Tenant
- System Operator must agree on
  - DN Information
  - App ID information
  - Population of UEs authorized to use app

**Edge App Information**
- Location of app in DN
- Application/Service ID

**Tenant and System Operator must agree on**
- DN Information
- App ID information
- Population of UEs authorized to use app
Topology Manager Microservice

Node without Topology Manager

- Network Card
- Memory 1
- App
- Memory 2
- Socket 1
- Socket 2

Not NUMA Aware
Application may be deployed to the wrong socket leading to performance degradation

Node with Topology Manager

- Network Card
- Memory 1
- App
- Memory 2
- Socket 1
- Socket 2

NUMA Aware
Application always deployed on the socket with the desired resources for deterministic & reliable performance

What does it do?

For un-balanced NUMA nodes with network card attached to only one socket:
- TM microservice exposes which socket is attached to the network card
- Enables the orchestrator to properly deploy performance sensitive edge applications (throughput, latency)

Alternative Solutions:
Suboptimal allocation of resources resulting in degraded performance

Makes the orchestrator NUMA Aware, enabling optimum deployment of performance sensitive edge applications
Telemetry Aware Scheduler (TAS) Microservice

What does it do?
TAS collects and exposes platform telemetry from collected and other sources to the Master. The Master, able to monitor performance of respective nodes and dynamically place/migrate workloads for optimum performance.

Example:
Content Delivery Network in Loc A has a service provider with a high revenue generating streaming app along with voice app. Loc A experiences voice app overloads due to local emergency. Using telemetry data from TAS the K8s Master observes bottleneck at Loc A and identifies Loc B that is capable of added workload. Service provider quickly and seamlessly moves the streaming app to Loc B with minimal impact to customer experience.

Alternative Solutions:
Absent this level of telemetry integration there is no way to get real time data needed to identify bottleneck and suitable node for dynamic offload.

Exposes edge node telemetry metrics enabling service providers to implement rules based workload placement for optimum performance and resilience.
OVN/OVS-DPDK Microservice

**What does it do?**
- Supports network overlay and dataplane using OVN/OVS-DPDK to accelerate data throughput
- OVN CNI (container network interface) helps
  - Configure network interfaces on OVS-DPDK
  - Configure data plane routing defined by SDN Controller (Openflow)
- Multus provides support for multiple network interfaces in the PODs deployed by Kubernetes

**Alternative Solutions:**
Absent this microservice, a more traditional approach around kernel stack packet routing would be needed which has significant performance implications

Enables implementation of high performance data plane (OVS-DPDK) on a cloud native edge environment with standard SDN controller
**Node Feature Discovery (NFD) Microservice**

Exposes node specific HW capabilities to the orchestrator to enable intelligent placement of workloads for optimized application performance and manageability.

**What does it do?**
Advertises edge node capabilities to the orchestrator
Enables the orchestrator to deploy the applications to the edge node with optimum capabilities that best meet edge KPIs

**Alternative Solutions:**
Without NFD, application is deployed with degraded performance
Implement a custom solution for the deployment

Unique HW capability examples: FPGA, Security, Performance, ...
What does it do?
Once NFD microservice checks for FPGA presence, version, status etc
FPGA RSU microservice programs and reboots the FPGA (as needed)
FPGA Config microservice configures it for deployment
FPGA Plugin allocates resources (eg. RAN-CNFR) to the FPGA

In-field configuration/programming capabilities are currently not available in off the shelf k8s

Alternative Solutions:
Physically program/config the FPGA (truck roll)
Implement custom/proprietary solutions
How to get started:

FREE SOFTWARE DOWNLOAD

COMMERCIAL RELEASES FOR FASTER TTM

TRAINING

Get access to Open Network Edge Services Software at OpenNESS.org

DOCUMENTATIONS:
- Architecture Overview
- User Guides
- OpenNESS White Paper
- OpenNESS Overview
- Webinar/Video
- Other Developer Resources here

Intel® Network Builder University Training