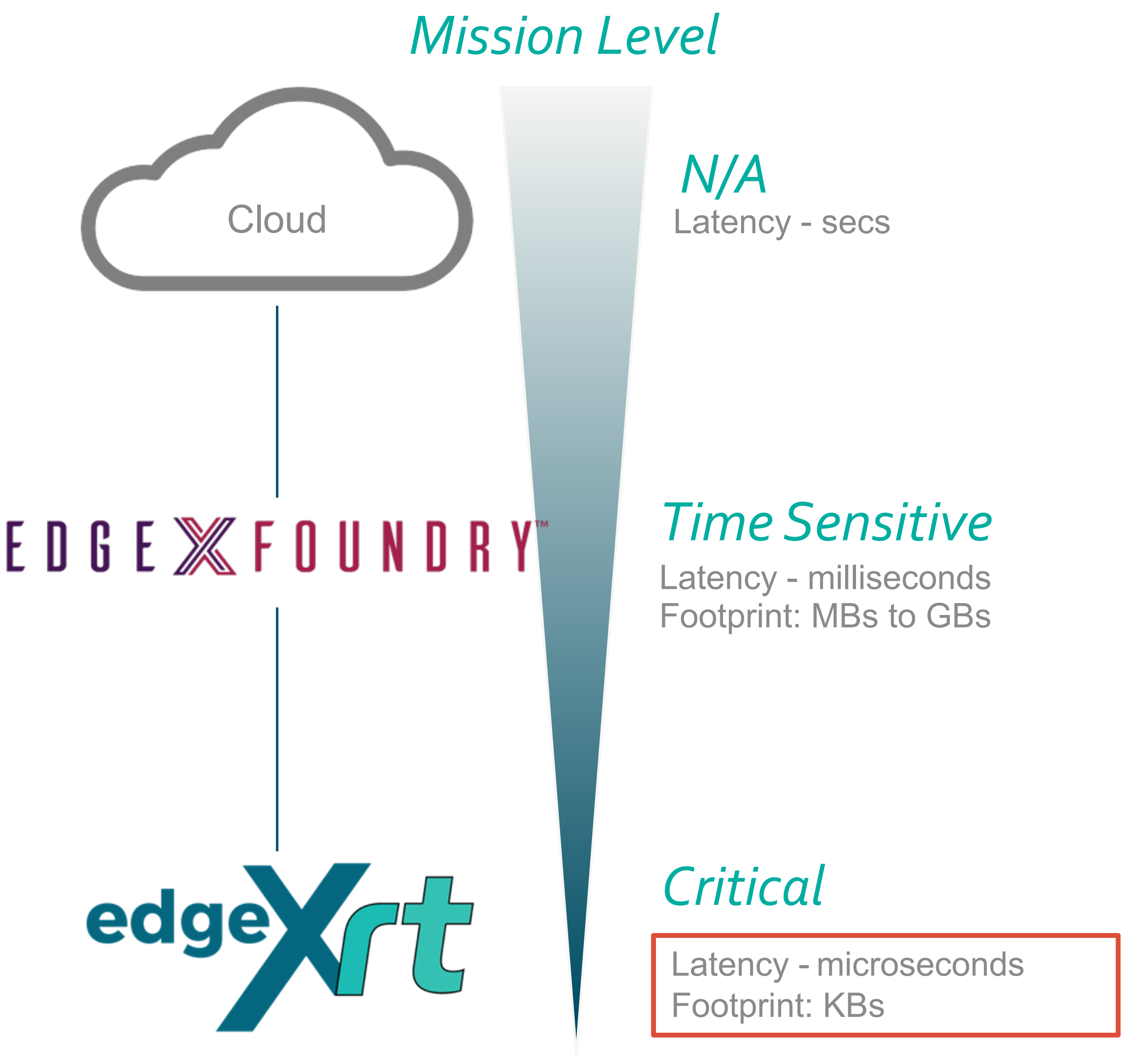
Time-Critical Edge Software Platform

Hans Chen

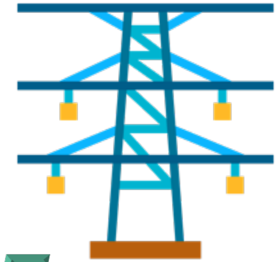
Internet of Thing

EdgeX Foundry is a perfect edge gateway to ingest /process /export data for IoT application



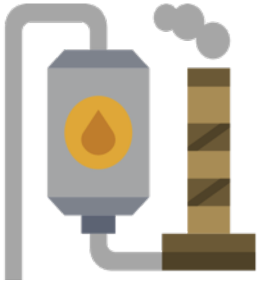
Time Critical Domain

Aerospace



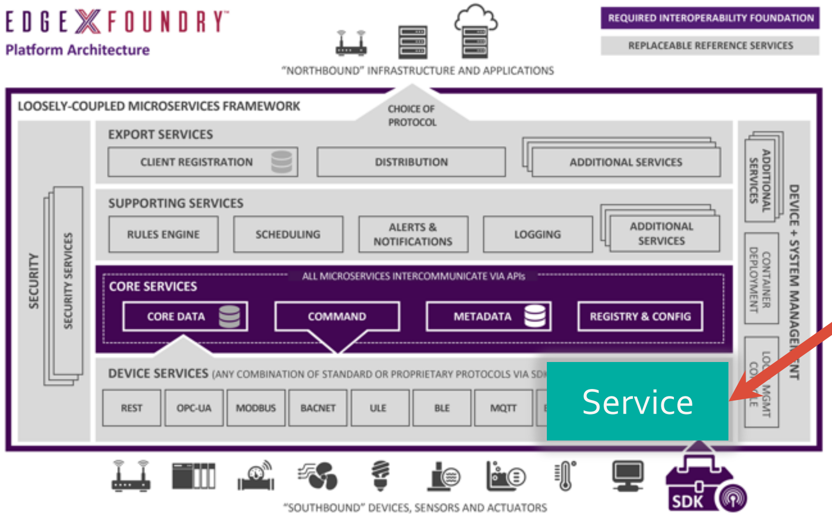
Energy

Defense



Gas & Oil

Time Critical Domain



edgeXrt

Aerospace

Defense

Energy

Gas & Oil

IOTech

Real Time? Real Fast?

= Determinism

Right Data

to

Right Place

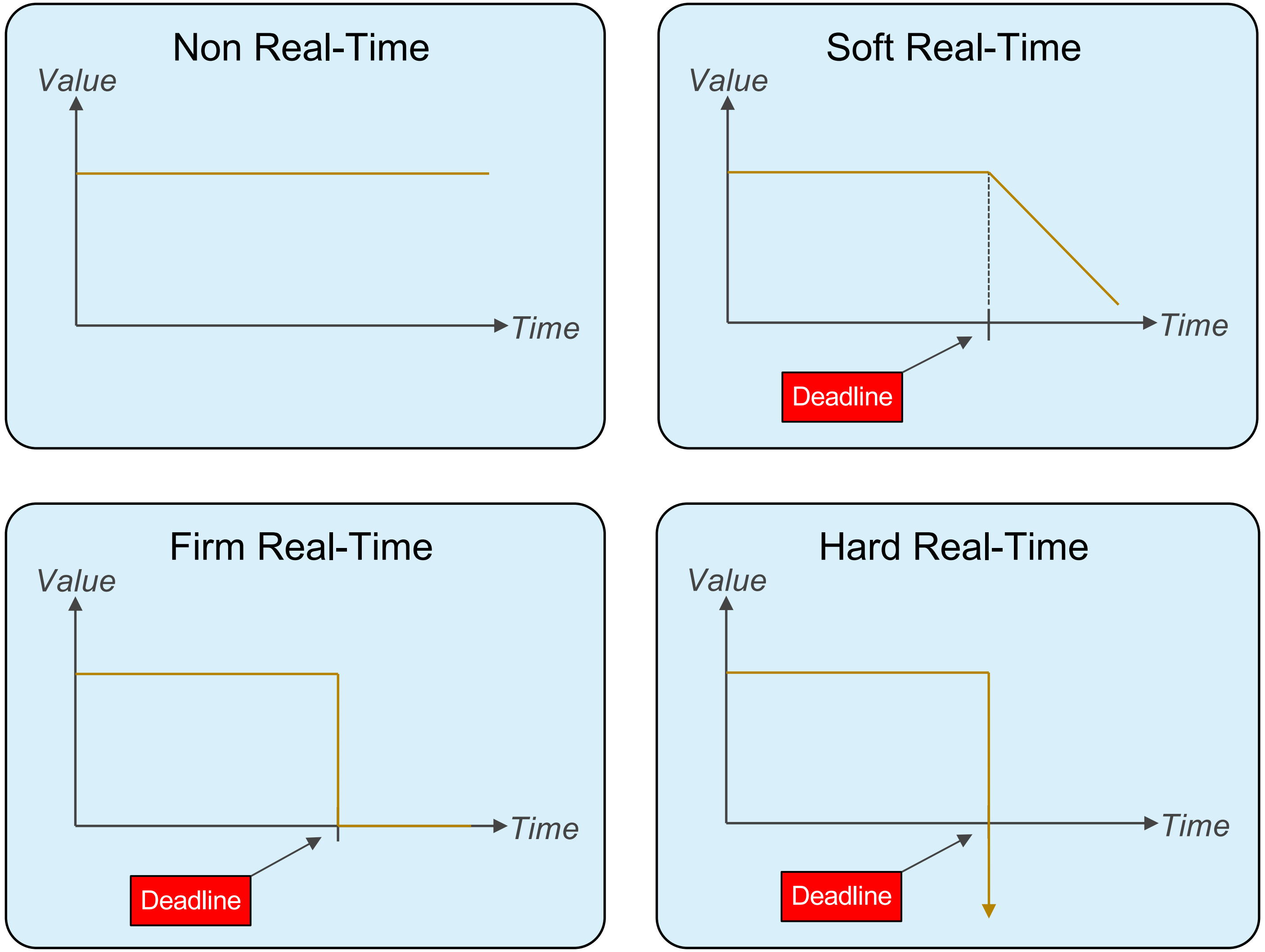
at

Right Timing

IOTech

‘Real-Time’ – Mutual Understanding

- ‘Real-time’ is not just about low latency, it is about ‘determinism’
- A real-time system is a time-bound system which has well defined fixed time constraints (deadlines or the maximum latencies that can be tolerated).
- Processing must be done within the defined constraints or the system will fail in a ‘soft’, ‘firm’ or ‘hard’ fashion.
- Examples of applications that have hard real-time constraints are the automotive industry (e.g. drive-by-wire and ABS), air traffic control, process controls, and medical devices (e.g. a pacemaker)



Edge XRT the Time-Critical Edge Software Platform

Product Overview

Introducing Edge XRT

- **XRT is the first Edge Software Platform designed specifically for Time-Critical and resource constrained IoT systems**

- **XRT supports IoT applications that have one or more of the following requirements:**
    - Ultra Low footprint (< 100 KBs)
    - Ultra low latency (< 100 μSec) cycle times
    - Real-time predictability (priority-aware guaranteed determinism)
    - ‘Brownfield’ Legacy portability

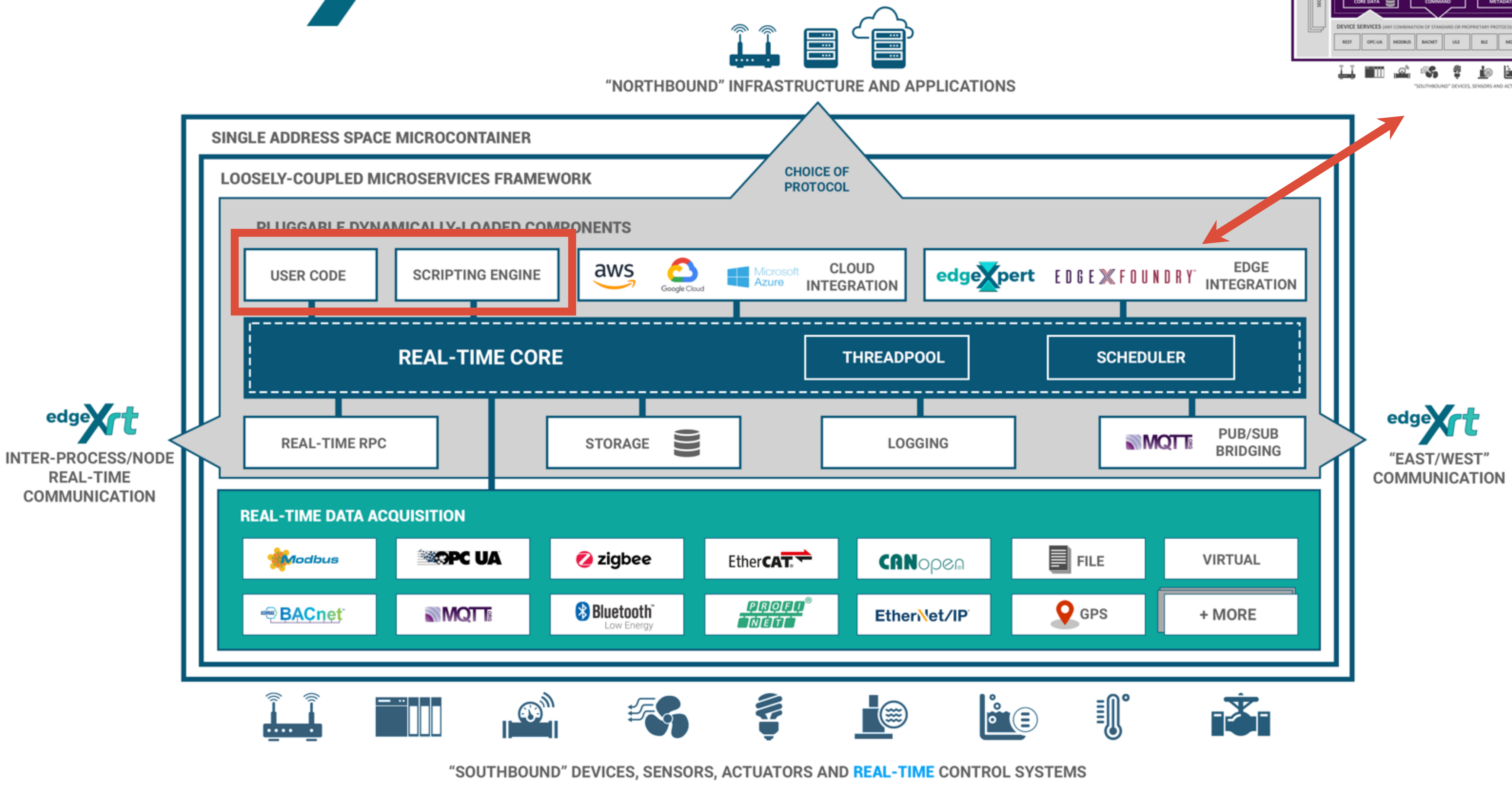
- **XRT reduces time-to-market for software defined Time-Critical IoT systems, by providing critical software infrastructure, allowing developers to focus on application value-add**

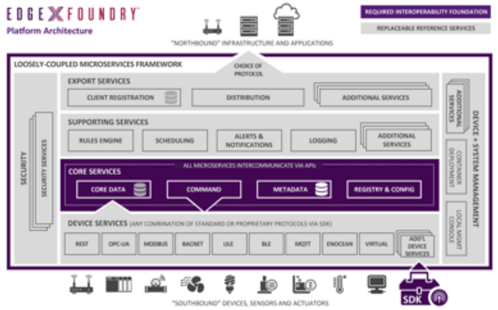
Why XRT?

- **Simplifies** real-time development, provides application portability, improved supportability, faster time-to-market and product evolution, allowing ease of hardware upgrade
- Provides productized **reusable real-time infrastructure** (e.g. real-time thread prioritization and scheduling) allowing users to focus on value-add applications
- **Optimized** – written in C for low footprint, portability, and real-time performance
- Complete deployment **flexibility** – containerized, non containerized and virtualized
- **Open** – runs on any silicon, operating system (with POSIX support) and hardware
- Increases application **re-use** across different use cases
- **Standard connectivity** "out-of-the-box" for southbound OT (e.g. Modbus, OPC UA etc.) and northbound IT (e.g. Multi-Cloud, MQTT, REST)

IOTech

EDGE XRT PLATFORM ARCHITECTURE





IOTech

Edge XRT
Product Availability

Processor and RTOS Support

XRT has a lightweight POSIX abstraction layer and can easily ported to any system that provides libc

Current OS Support

- Ubuntu Linux 16.04, 18.04, 20.04
- Debian Linux 9, 10
- Centos 7, 8
- Alpine Linux 3.9, 3.10, 3.11
- Clear Linux
- Photon Linux 3.0
- Fedora 30, 31
- OpenSuse 15 (Leap)
- Zephyr 2.0

Processor support: 32 bit and 64 bit ARM, 64 bit Intel

Initial release hardware that XRT has been tested on:

- Intel NUC (64 bit Intel)
- Dell 3000 and 5000 series Edge Gateways (64 bit Intel)
- Raspberry PI 2, 3, 4 (32 and 64 bit ARM)
- NXP SPC5567MVR132 (32 bit ARM)
- HP MP9

Language Support

Programming Language Support

- C and C++ language
- Lua scripting language
- Integrates with components written in other languages via Thrift RPC, MQTT or REST APIs

Edge XRT
Use Cases

Use Cases

1. **Programmable automation controllers** running closed loop control applications (e.g. gas flow valve control, pump control, motor control) with sample rates < 50ms and a requirement for predictable cycle times
2. **High frequency signal processing** applications (e.g. substation fault monitoring) where multiple voltage and current signals need to be analyzed and actions taken in real-time and with cycle times as low as 5ms
3. **Microcontroller applications** that require IoT platform capabilities but have SOC memory constraints as low as 126KB e.g. automobile engine management system, medical device, consumer electronic device
4. **Legacy embedded edge environments** that cannot support a modern operating system running docker or cannot support a modern software development tool chain e.g. legacy power meters in energy distribution

IOTech

XRT Use Case: Oil & Gas

Oil Well Control

Outcome: Provide real-time, closed-loop control of oil wells in order to optimise the oil production rate

- Prove the application of edge processing technologies to closed-loop control and to demonstrate feasibility

Key Problem:

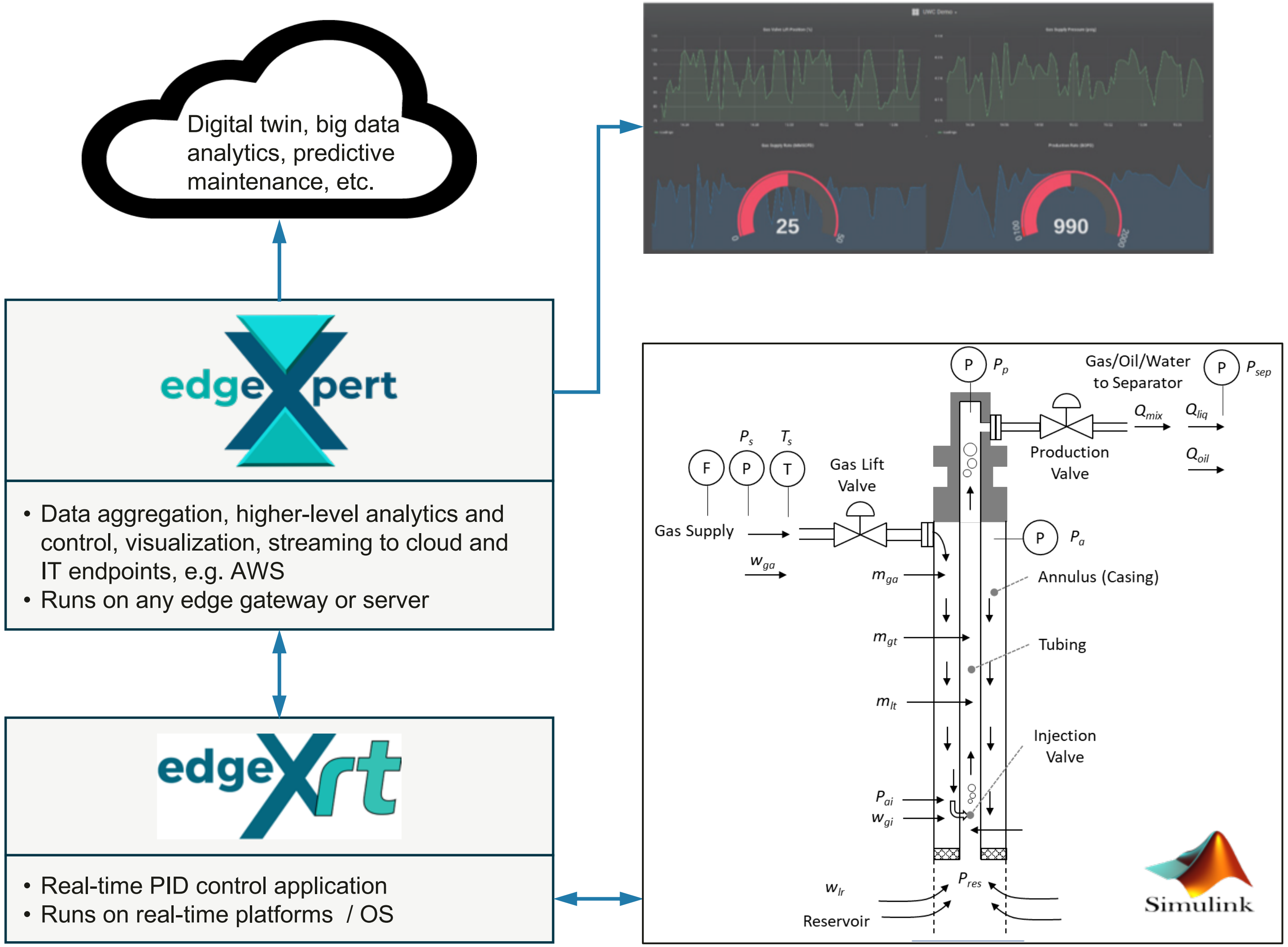
- The customer needs real-time, software defined control to run on-site, on a variety of devices, and to connect using a variety of data protocols

Solution:

- A full well simulation model was produced and interfaced with Edge XRT, demonstrating real-time PID control running on the platform

Other Benefits:

- Edge Xpert can be added to provide higher-level analytics, visualization, monitoring and well optimization (including AI / ML) capabilities
- Cloud-agnostic and on-premise solution



IOTech

XRT Use Case: Electrical Power Metering

Smart Meter Monitoring

<u>Outcome</u>: Better visibility on active energy consumption across power networks, via reliable data capture and aggregation from a variety and large number of smart meter concentrator devices

- Requires a solution for collecting smart meter data from all types of edge concentrator gateways and delivering it to the cloud

<u>Key Problem</u>:

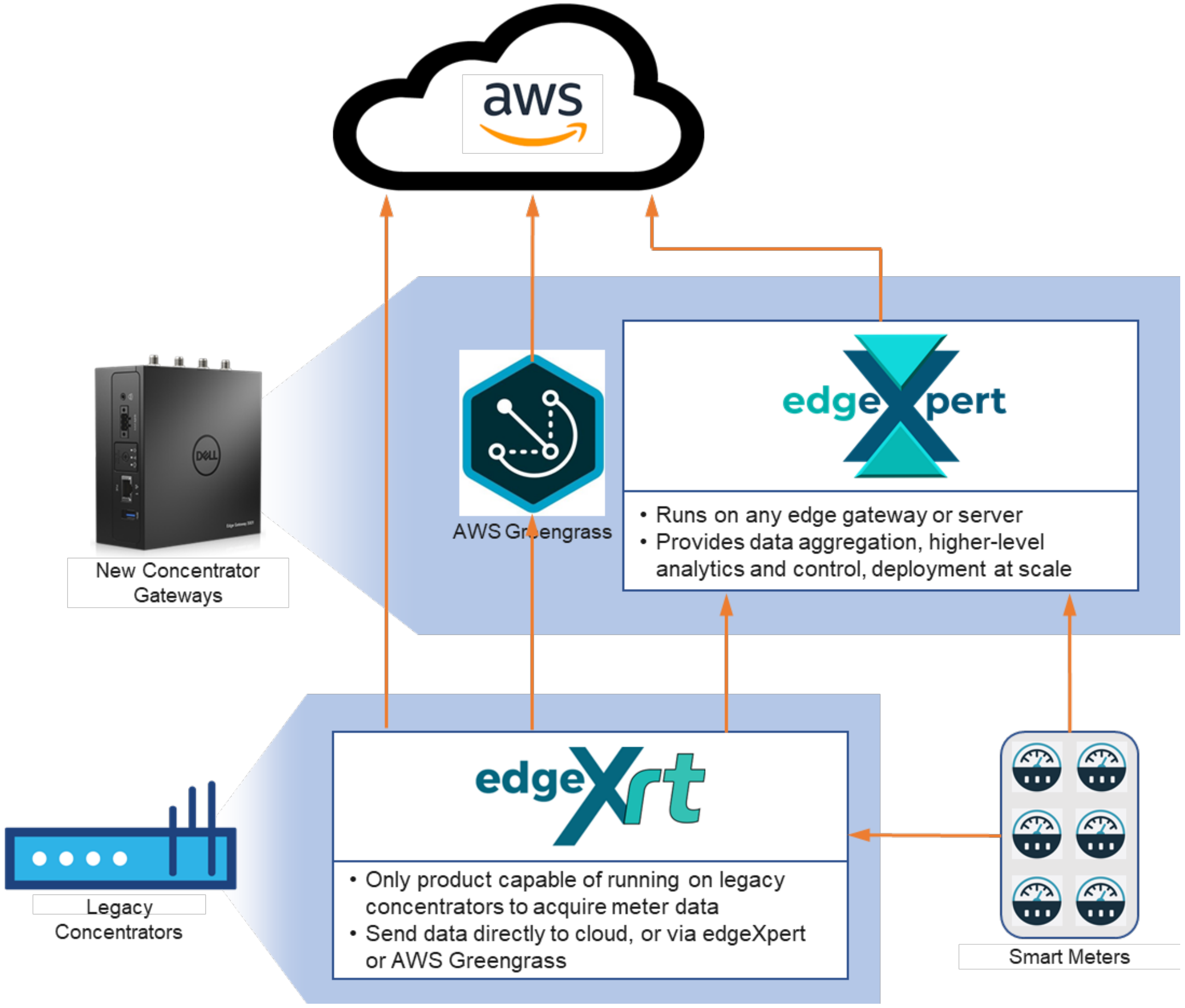
- The customer couldn't get any solution to run on the oldest (MSC) concentrators

<u>Solution</u>:

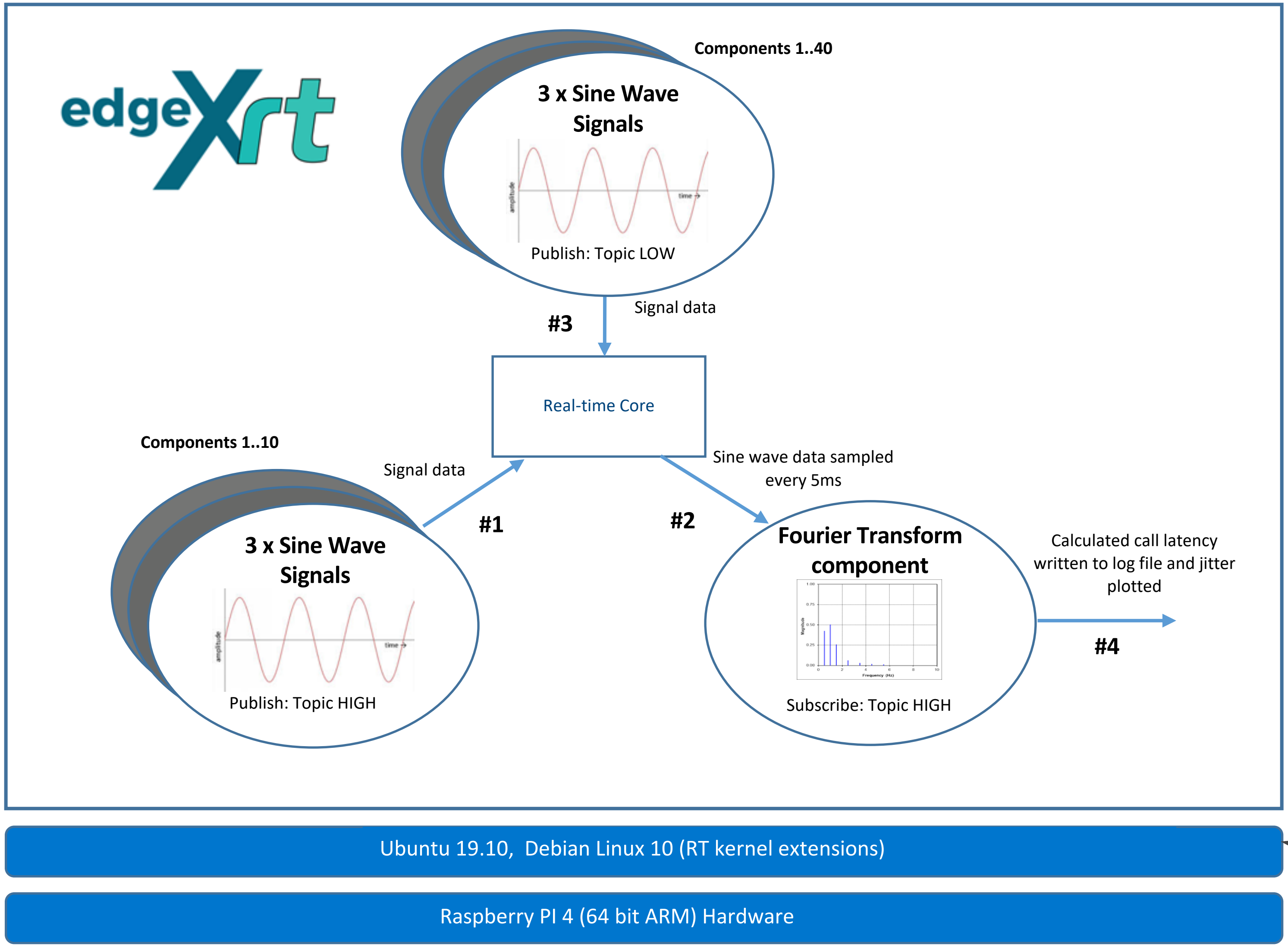
- Edge XRT was the only platform that could run on those concentrators

<u>Other Benefits</u>:

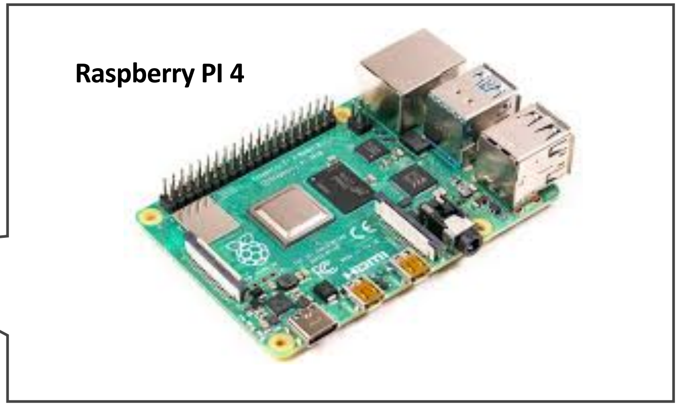
- Edge XRT and Xpert provide a clear upgrade path from the legacy devices
- Third-party products, e.g. AWS Greengrass, can be readily incorporated if required



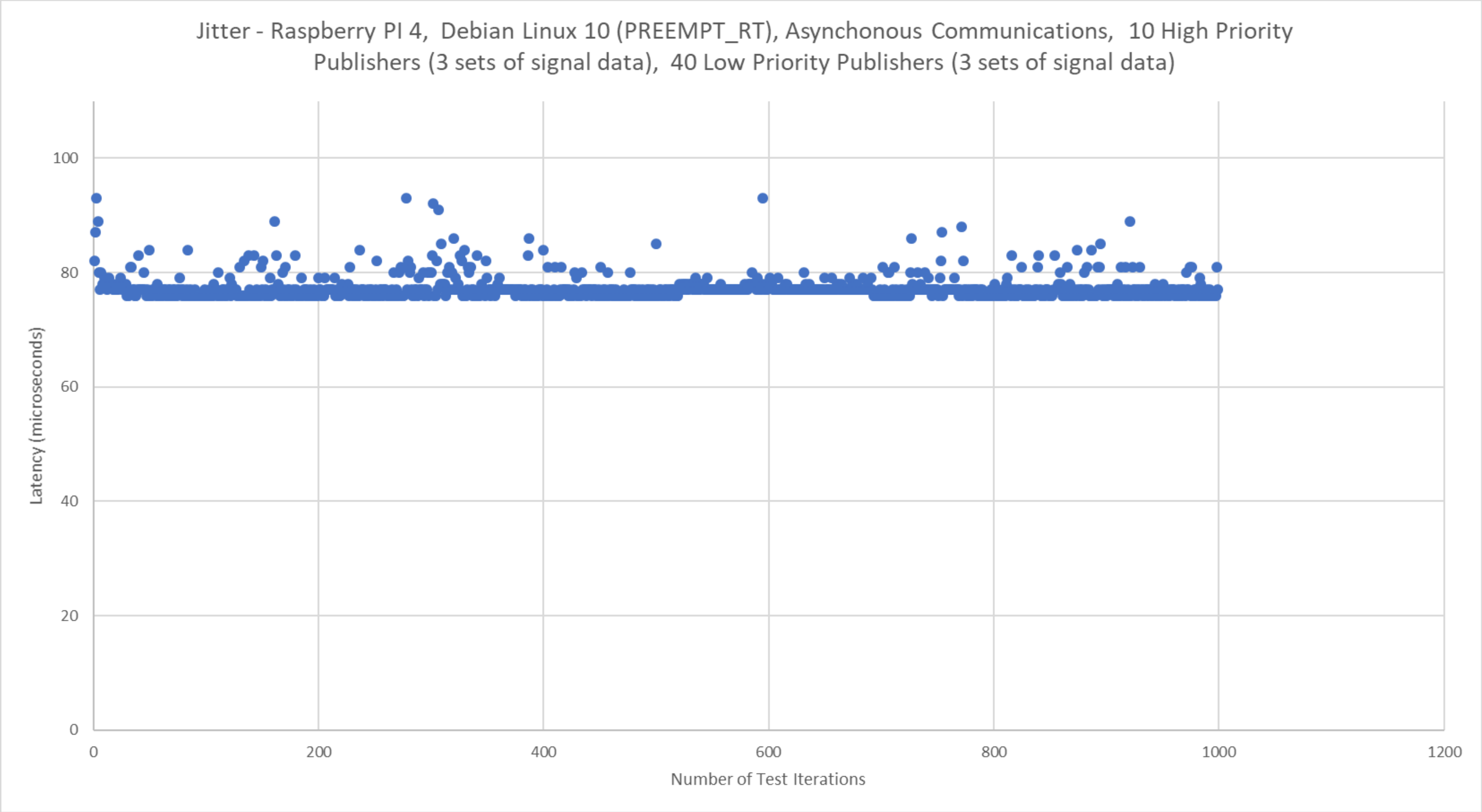
Fault Monitoring Example



1. **10 x C components** each **implementing 3 x Sine Wave signals**, publishes sine wave samples (3 x array of 1024 doubles) and origin time stamp on HIGH Priority Topic
2. 40 x C components implementing 3 x Sine Wave signals, publishes sine wave samples (3 x array of 1024 doubles)
3. C component implementing a Fourier Transform subscribes to Sine Wave data on HIGH Priority Topic, reads and processes **3 x array of 1024 doubles every 5ms** (3 signals) , calculates message latency and writes it to a log file
4. Jitter is then plotted using latency data written to log file



Fault Monitoring Test Results – Raspberry PI, Debian Linux 10 (PREMPT_RT), Asynchronous, 10 High Priority Publishers, 40 Low Priority Publishers



Results referred to HIGH priority publishers

30 Signals High Priority
120 Signals Low Priority

Avg Latency: 80 microsec
Max Latency: 95 microsec

Notes: The test process has two separate instances of bus/threadpool/scheduler, one set configured with no priority and with RT priority and processor affinity.

Thank You

www.iotechsys.com/

Great China Business Development
joey@iotechsys.com (Sales Manager)
hans@iotechsys.com (Solution Architect)

For additional information

Visit: http://www.iotechsys.com/

Thank You