OneOS: IoT Platform based on POSIX and Actors

Kumseok Jung, UBC
Julien Gascon-Samson, ÉTS Montréal
Karthik Pattabiraman, UBC
Motivation

Near-future IoT Project

City of Vancouver
Motivation

Near-future IoT Project

City of Vancouver

Public Infrastructure
Motivation

Near-future IoT Project

City of Vancouver

Public Infrastructure
Motivation

Near-future IoT Project

City of Vancouver

Public Infrastructure
Motivation

Near-future IoT Project

City of Vancouver

Public Infrastructure

3rd Party Service
Motivation

City of Vancouver

Alice
Engineer
Motivation

City of Vancouver

Alice
Engineer
Motivation

City of Vancouver

Magic

Alice
Engineer
Motivation

City of Vancouver

Magic

Alice
Engineer
Motivation

City of Vancouver

Magic

Alice Engineer

UBC
Motivation

City of Vancouver

Magic

Alice
Engineer
Challenges

Example Problem:
Collect GPS data from the 2 subsystems (Bus & Bike), compute an optimal scheduling policy for another subsystem (Subway)
Challenges: Heterogeneity

Company A

Company B
Challenges: Heterogeneity

Company A
- Windows 64bit

Company B
- Linux 32bit

Heterogeneity in Hardware and Operating Systems
Challenges: Heterogeneity

Company A - Windows 64bit, Azure IoT Edge Runtime

Company B - Linux 32bit, AWS IoT Greengrass

Heterogeneity in Languages and Frameworks
Challenges: Heterogeneity

Software within a **specific framework** is **not portable** across different frameworks.
Challenges: Heterogeneity

Company A
- Windows 64bit
- Azure IoT Edge Runtime
- Solution

Company B
- Linux 32bit
- AWS IoT Greengrass
- Solution

Software within a **specific framework** is not portable across different frameworks
Challenges: Heterogeneity

Company A
- Windows 64bit
- Azure IoT Edge Runtime

Company B
- Linux 32bit
- AWS IoT Greengrass

Need to rewrite the same application logic for different frameworks
Challenges: Heterogeneity

Company A
- Windows 64bit
- Azure IoT Edge Runtime

Company B
- Linux 32bit
- AWS IoT Greengrass

Difference in application semantics resolved by more glue software
Challenges: Heterogeneity

Company A
- Windows 64bit
- Azure IoT Edge Runtime
- Solution

Company B
- Linux 32bit
- AWS IoT Greengrass
- Solution_1

We end up with: **Heterogeneity** in Application Software
Challenges: Scale & Dynamicity

Company A

Company B
Challenges: Scale & Dynamicity

Company A

Company B
Challenges: Scale & Dynamicity

Company A

Company B
Challenges: Scale & Dynamicity

Company A

Company B
Challenges: Scale & Dynamicity

Company A

Company B
Challenges: Scale & Dynamicity

Company A

Company B
Related Work

Distributed Computing Platforms

Application Portability

Device Independence

IoT Platform
- AWS Greengrass
- Azure IoT Edge
- Google Cloud IoT
Related Work

Distributed Computing Platforms

Device Independence
Interface Transparency

IoT Platform
AWS Greengrass
Azure IoT Edge
Google Cloud IoT

Application Portability
Related Work

Distributed Computing Platforms

Device Independence

Application Portability

Cluster Platform
- Mesosphere DC/OS
- Kubernetes
- Docker Swarm
- OpenHPC

IoT Platform
- AWS Greengrass
- Azure IoT Edge
- Google Cloud IoT
Related Work

Distributed Computing Platforms

- Interface Transparency
- Topology Independence
- Failure Tolerance

Cluster Platform
- Mesosphere DC/OS
- Kubernetes
- Docker Swarm
- OpenHPC

IoT Platform
- AWS Greengrass
- Azure IoT Edge
- Google Cloud IoT
Related Work

Distributed Computing Platforms

**Distributed OS**
- Amoeba
- Sprite
- Plan 9
- Inferno
- Barrelfish

**Cluster Platform**
- Mesosphere DC/OS
- Kubernetes
- Docker Swarm
- OpenHPC

**IoT Platform**
- AWS Greengrass
- Azure IoT Edge
- Google Cloud IoT

Device Independence vs. Application Portability
Related Work

Distributed Computing Platforms

- **Distributed OS**
  - Amoeba
  - Sprite
  - Plan 9
  - Inferno
  - Barrellfish

- **Cluster Platform**
  - Mesosphere DC/OS
  - Kubernetes
  - Docker Swarm
  - OpenHPC

- **IoT Platform**
  - AWS Greengrass
  - Azure IoT Edge
  - Google Cloud IoT

- Interface Transparency
- Application Portability
- Topology Independence
- Failure Independence

Application Portability vs Device Independence
Our Goal: OneOS

Distributed Computing Platforms

Distributed OS
- Amoeba
- Sprite
- Plan 9
- Inferno
- Barrelfish

Cluster Platform
- Mesosphere DC/OS
- Kubernetes
- Docker Swarm
- OpenHPC

IoT Platform
- AWS Greengrass
- Azure IoT Edge
- Google Cloud IoT

Device Independence

Application Portability
OneOS: Approach

Chain of Programming Interfaces

- User App
- API
- Framework
- Language
- Runtime
- API/ABI (system call)
- OS
- Hardware
OneOS: Approach

Chain of Programming Interfaces

High-Level
- Framework
- Language

Low-Level
- Runtime
- API/ABI (system call)
- OS
- Hardware

User App

Application Programming

Systems Programming

Low-Level Language High-Level System Interfaces
OneOS: Approach

High-Level

- User App
- Framework
- Language

Low-Level

- Runtime
- API/ABI (system call)
- OS
- Hardware
OneOS: Approach

High-Level

- User App
- Framework
- Language

Low-Level

- Runtime
  - Linux
    - armv7
  - OS
    - Hardware
  - Windows
    - x64
- API/ABI (system call)
OneOS: Approach

Platform-Independence by using High-level Language
OneOS: Approach

Platform-Independence by using High-level Language, not by using Framework API
OneOS: Approach

**Platform-Independence**
by using High-level Language, 
*not by using Framework API*
OneOS: Approach
OneOS: Approach

Heterogeneity in Software unresolved by frameworks
OneOS: Approach

Heterogeneity in Software unresolved by frameworks

Application Portability worsens with more frameworks
OneOS: Approach

Adding Abstraction Layers on top
OneOS: Approach

Adding Abstraction Layers on top leads to “API hell”
OneOS: Approach

Adding Abstraction Layers on top leads to “API hell”
OneOS: Approach

Our approach:
Not a high-level framework
OneOS: Approach

Our approach:
Not a high-level framework
OneOS: Approach

Embrace heterogeneity in software
allow existing technology to work together
OneOS: Approach

High-level Language VMs share a common interface to the underlying abstract machine

- JavaScript: Node.js
- Python: CPython
OneOS: Approach

High-level Applications are agnostic about the underlying abstract machine
OneOS: Approach

High-level Applications are agnostic about the underlying abstract machine. System call modeled as message between Actors.
OneOS: Approach

Applications make system calls to interact with other agents
OneOS: Approach

Hijack low-level Abstraction Layer
alter the operational semantics of high-level software
OneOS: Approach

**Hijack low-level Abstraction Layer**
alter the operational semantics of high-level software

**Intercept system calls**
redirect to various **distributed services**
OneOS: Approach

- App: yourApp, herApp
- Runtime: Node.js, CPython
- Language: JavaScript, Python
- OneOS Middleware
- Services: Scheduler, File System, IPC
- Interfaces: File System, Network, I/O Streams
OneOS: Design
OneOS: Design

App 1
- fs.writeFile
- OneOS Middleware
- Windows
  - x86

App 2
- OneOS Middleware
- MacOS
  - x64

App 3
- OneOS Middleware
- Linux
  - armv7

Network

IPC
FileSystem
Scheduler
Storage
Session

OneOS

Component Details:
- fs.writeFile
- Windows
- MacOS
- Linux
- x86
- x64
- armv7
OneOS: Design

- App 1
- App 2
- App 3

OneOS Middleware

- Windows (x86)
- MacOS (x64)
- Linux (armv7)

IPC
FileSystem
Scheduler
Storage
Session

Network

fs.writeFile
OneOS: Design

- App 1
- App 2
- App 3

Middleware components:
- IPC
- FileSystem
- Scheduler
- Storage
- Session

Process stdout.write

Operating systems:
- Windows (x86)
- MacOS (x64)
- Linux (armv7)

Network
OneOS: Design
OneOS: Proof-of-Concept Demo
OneOS: Discussion

Feedback Wanted:
- Evaluation strategies
- Practicality of Actor-based micro-kernel
- Suitability of high-level language for systems programming

Controversial Points:
- *Single system image* appropriate for a *geographically distributed grid*?
- Mapping *POSIX interface* over an *inherently distributed and concurrent architecture*?
- *Limiting application space to high-level languages*?

Open Issues & Future Work:
- Security and Privacy model
- Failure handling
- Semantics of cyber-physical resources

Potential Drawbacks:
- Fundamental tension between cyber-physical resources and their abstract representations
- Reasoning about security concerns within high-level programming space
- Inability to make low-level optimizations

✉ kumseok@ece.ubc.ca  📞 DependableSystemsLab/OneOS