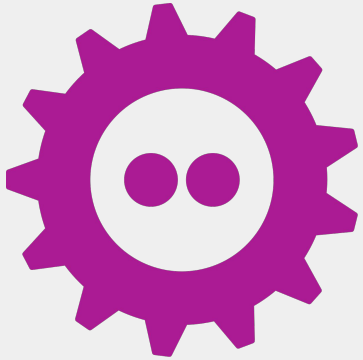
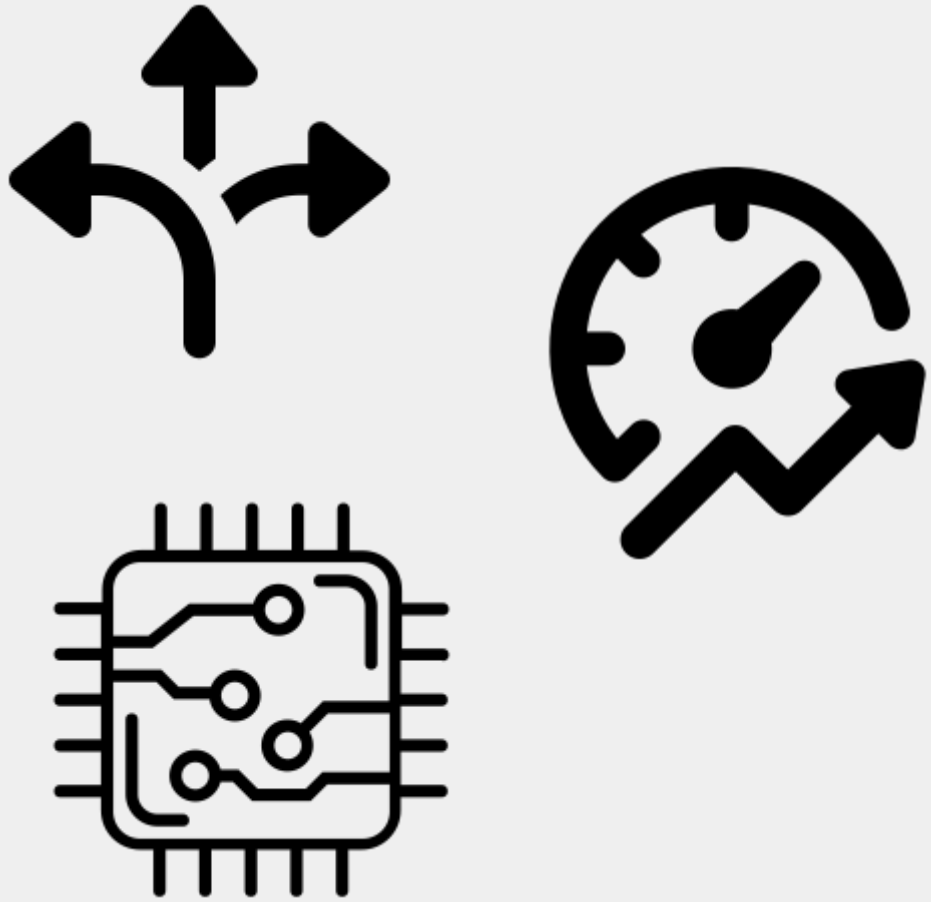
A Modular Approach to Effortless and Dependency-Aware LibOS Building



Charalampos Mainas, Anastassios Nanos

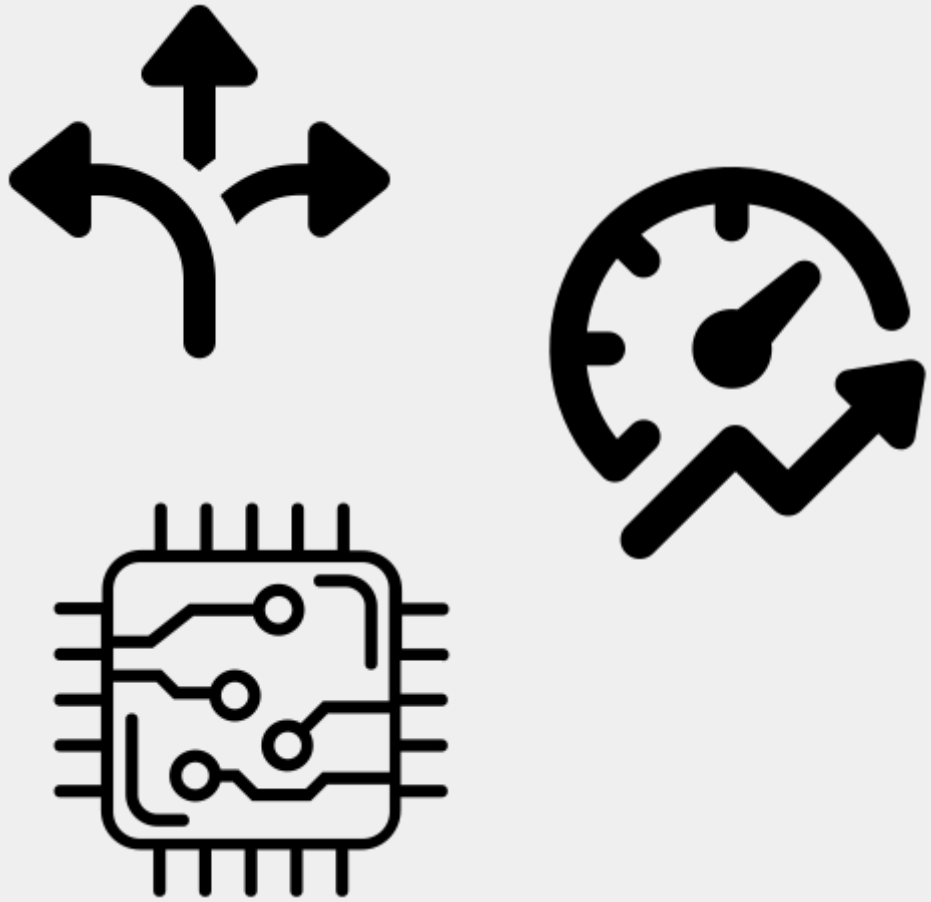
Specialization in OSes

- Normal in embedded world
  - Hardware constraints
  - Specialized use cases
- OS specialization in the cloud
  - Specialized LInux distributions for containers
  - Minimal LInux kernels



Specialization in OSes

- Normal in embedded world
  - Hardware constraints
  - Specialized devices
- OS specialization in the cloud
  - Specialized LInux distributions for containers
  - Minimal LInux kernels

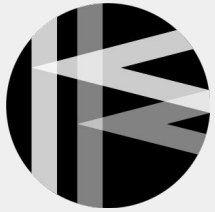


We can do better than a specialized Linux distro

Library Operating Systems

- Each OS component as a standalone independent library
- Modular, creation of tailored OS images for a purpose
- Use cases:
  - Cloud deployments
  - TEEs
  - Userspace drivers/OS
  - SoC and Embedded





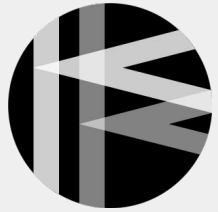

Occlum

GRAMINE

Building LibOSes

- Application compiles and links against libOS
  - Cross compilation of app with libOS tools
  - Static linking of app and libOS objects
- Two main building processes from libOSes
  - Install or build and use a cross-compiling suite
  - Integrate the building process of the app in LibOS building






Occlum

GRAMINE

Building LibOSes

- Developers are exposed on building an OS
  - Understand, configure and build the libOS framework
  - Handle dependencies
- Each libOS has its own
  - Building system / process
  - Library support
  - Dependency tracking

Building LibOSes

- Developers are exposed on building an OS
  - Understand, configure and build the libOS framework
  - Handle dependencies
- Each libOS has its own
  - Building system / process
  - Library support
  - Dependency tracking

Building libOSes can be challenging

How can we improve the building process of libOSes

Specialization in OSes

- ~~Library Operating Systems and their building process~~
- **Existing building frameworks**
- Bunny: A new approach for libOS building
- Bunny internals

Building tools and platforms

- A rich ecosystem in the embedded world
  - Multiple tools mostly for Embedded Linux
  - Each OS comes with each own framework
- A fast evolving ecosystem in the cloud world
  - Creation of distros based and focused on containers
  - Target only Linux

Building tools for unikernels

- Kraftkit: A suite of tools for building Unikraft unikernels
  - Great interface / Easy to use
  - Supports only Unikraft
  - Not just a building tool

- Unik: A platform for automating unikernel and MicroVM compilation and deployment
  - Support for various unikernels and Firecracker microVMs
  - Various outputs: VMs, OCI images etc.
  - Project seems inactive

A modern and unified building experience for library-based OSes

Specialization in OSes

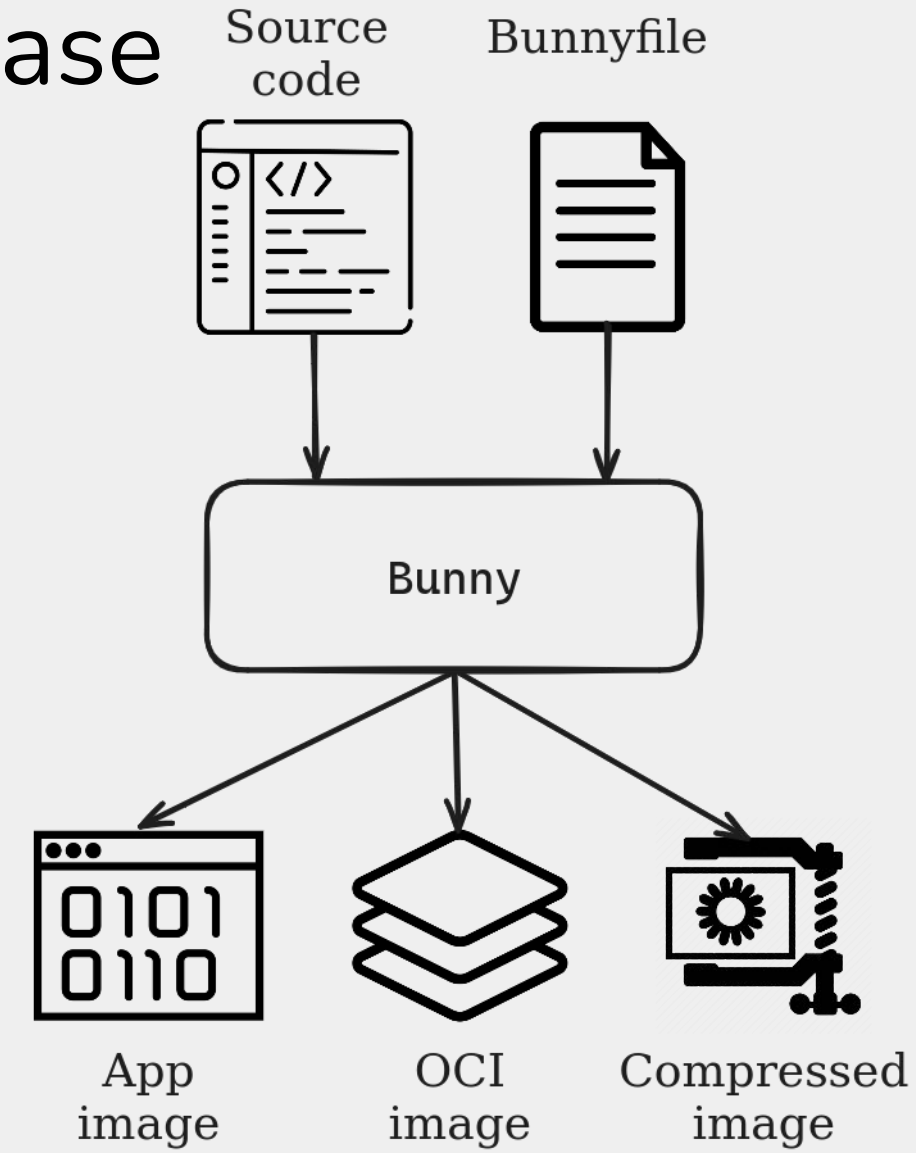
- ~~Library Operating Systems and their building process~~
- ~~Existing building frameworks~~
- **Bunny: A new approach on libOS building**
- Bunny internals

Bunny: Building libOSes with ease

- Configure once build against multiple libOSes
  - **Unified interface for libOSes**
- Simplify the process of building an app with a libOS
  - **Abstract away the diversity and complexity of each toolstack**
- No dependency hell
  - **Automatically identify and handle dependencies**

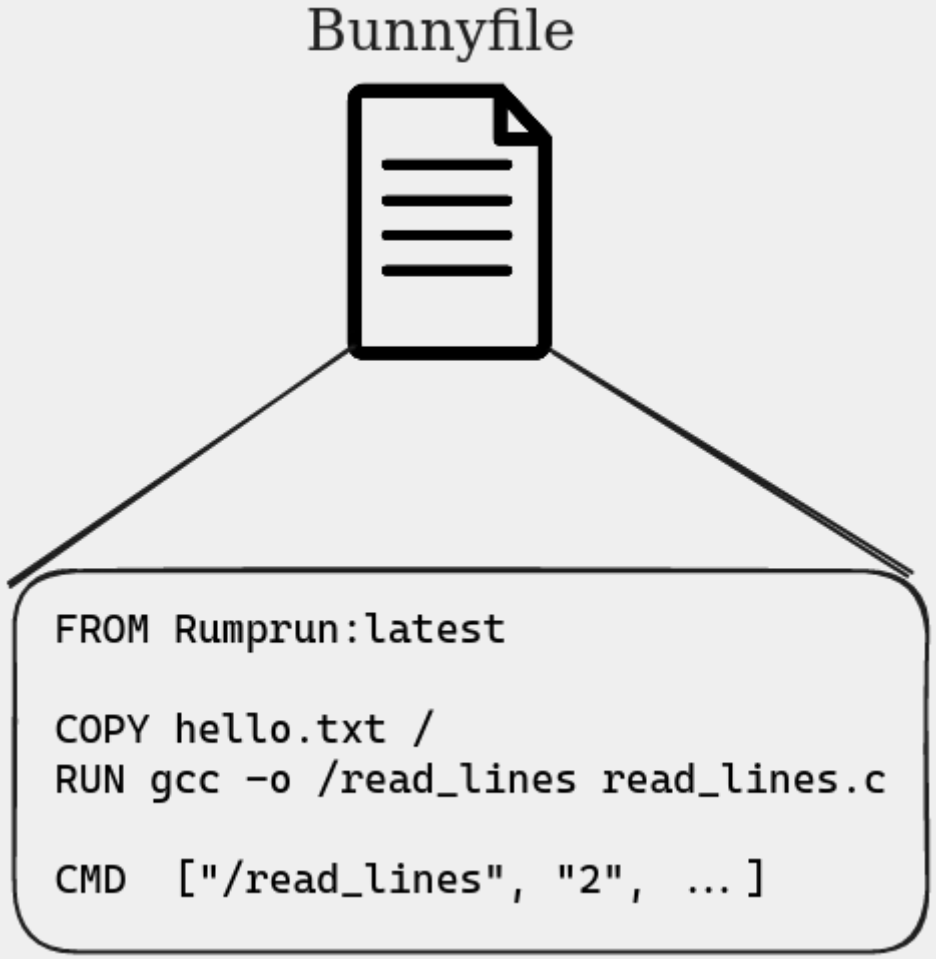
Bunny: Building libOSes with ease

- A container-like experience
  - Similar workflow with containers building
- Generate various outputs
  - VM images, OCI images, or other formats
- A layered building process
  - Reuse previously built components



Bunny: Interface

- User specifies in Bunnyfile:
  - The desired framework/OS
  - How to build their app
  - Application dependencies
  - Config files
- User specifies in Bunny cli parameters:
  - Target architecture
  - Target platform
  - Type of artifact



Specialization in OSes

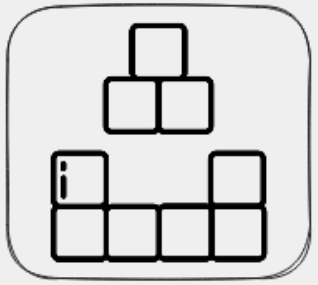
- ~~Library Operating Systems and their building process~~
- ~~Existing building frameworks~~
- ~~Bunny: A new approach on libOS building~~
- **Bunny internals**

Bunny: Building steps

- Analyze user input
  - Identify framework, dependencies, app language

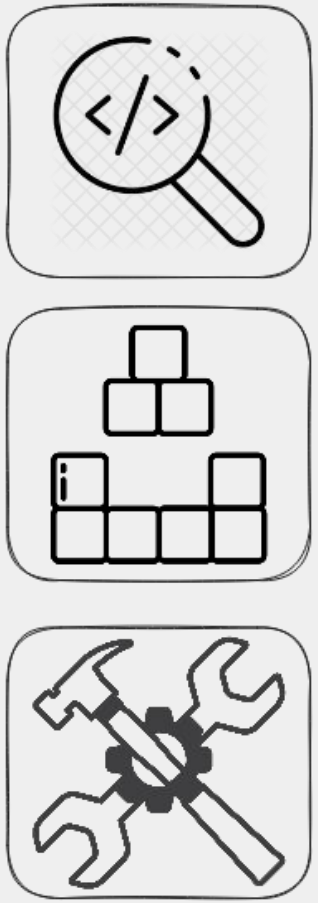
Bunny: Building steps

- Analyze user input
  - Identify framework, dependencies, app language
- Fetch building layers
  - Building tools and dependency layers (framework, libs)



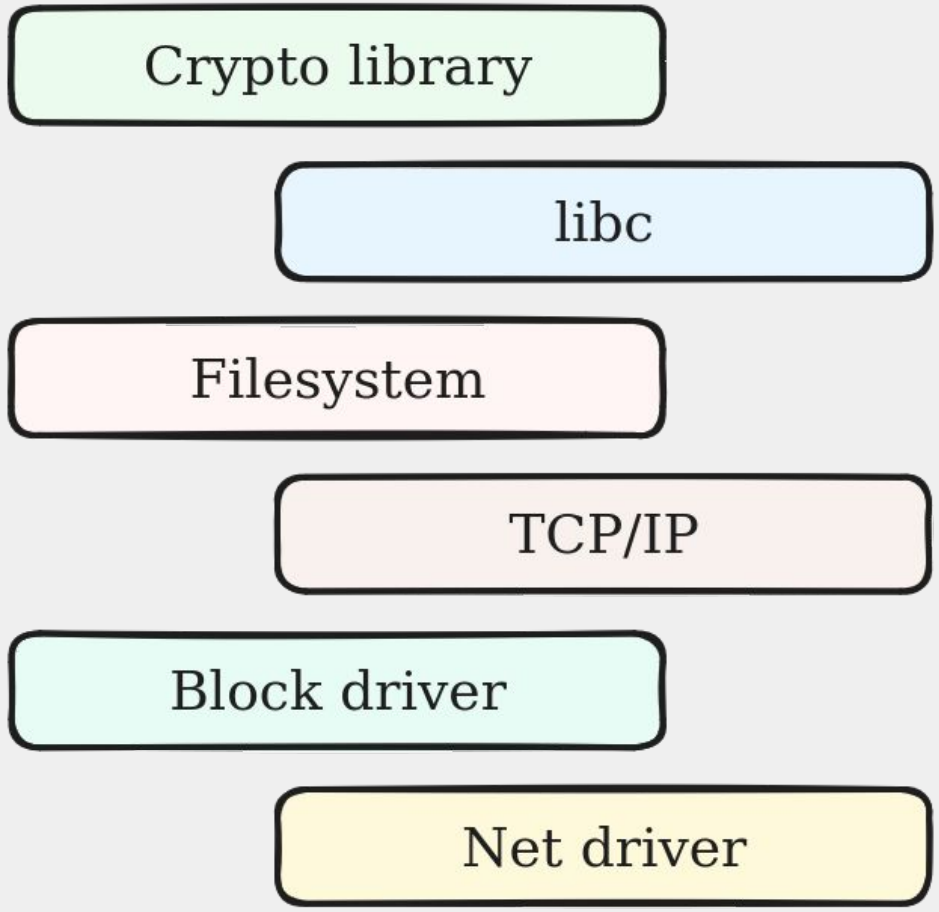
Bunny: Building steps

- Analyze user input
  - Identify framework, dependencies, app language
- Fetch building layers
  - Building tools and dependency layers (framework, libs)
- Build app
  - Build user code and link against all building blocks



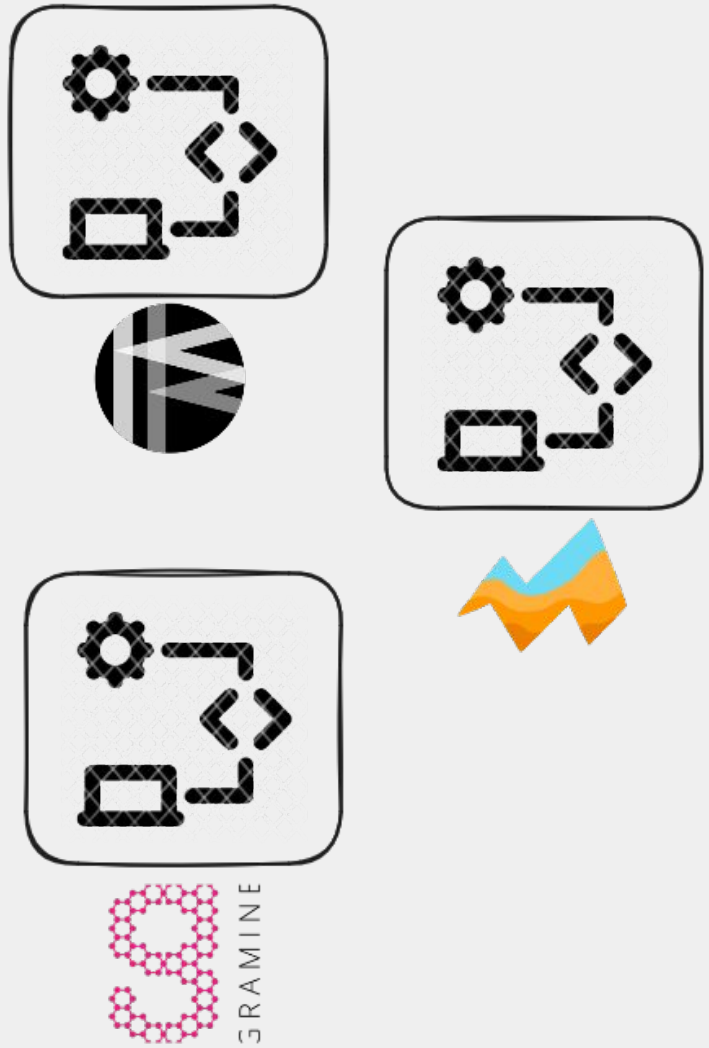
Bunny: Layers

- Leverage libOS's architecture
  - Every component is an independent library
  - Treat each component as a standalone layer
  - Reuse previously built layers
- Two types of layers
  - Tooling layers
  - Component layers



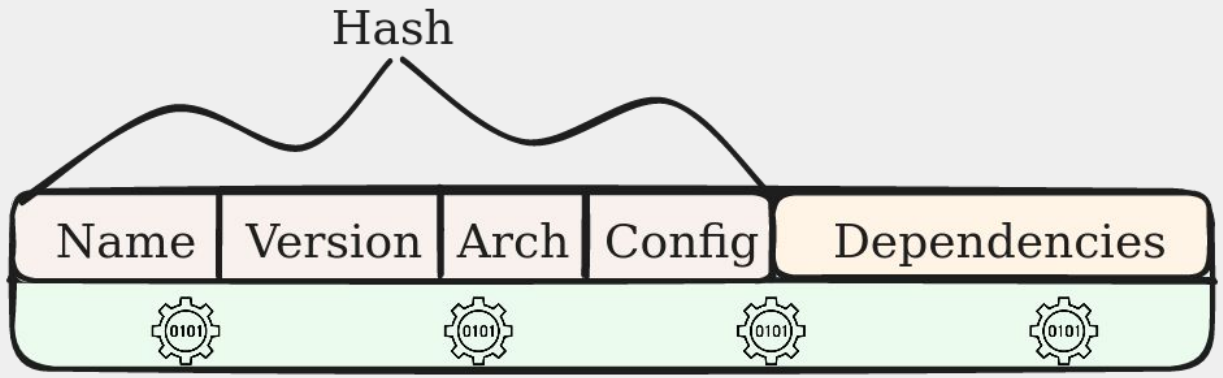
Bunny: Tooling layers

- Tooling layers contain the necessary tools to build the libOS
  - One tooling layer per libOS
  - Cross-compiler, Linker etc.
- Building process takes place inside a tooling layer
  - Cross-compilation of user's code
  - Creation of unikernel image



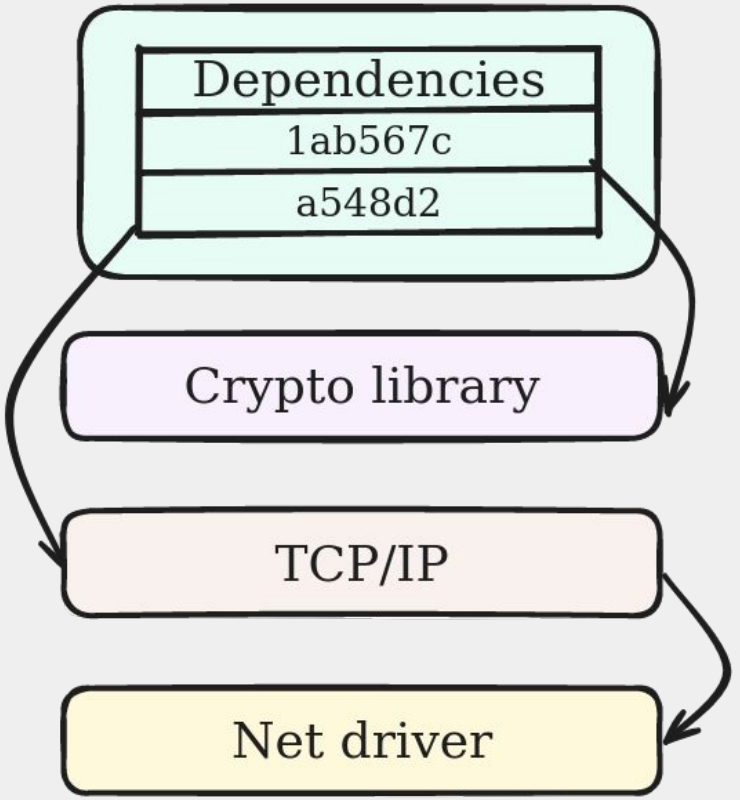
Bunny: Component layers

- Component layers contain
  - Metadata
  - Pre-built Object file
- One layer for each component-version-config
  - A unique hash identifies each layer
  - Hash generation by layer’s metadata



Bunny: Dependency handling

- Metadata contain all direct dependencies
  - **Declare dependencies during layer creation**
- Bunny creates a dependency tree during the analysis phase:
  - **Fetch all dependencies during fetch phase**



Summary

- Trend in OS specialization
- LibOSes are taking OS specialization seriously
- Diverse tools and building processes
- Bunny abstracts away the complexity of each framework
- A new layered approach on building applications with libOSes

This work is partially funded through Horizon Europe actions, MLSysOps (GA: 101092912) and DESIRE6G (GA: 101096466)

Summary

- Trend in OS specialization
- LibOSes are taking OS specialization seriously
- Diverse tools and building processes
- Bunny abstracts away the complexity of each framework
- A new layered approach on building applications with libOSes

Get in touch:

cmainas@nubificus.co.uk
ananos@nubificus.co.uk

https://github.com/nubificus/bunny
https://blog.cloudkernels.net