The Internet of Things in a Laptop:

Rapid Prototyping for IoT Applications with Digibox

Silvery Fu, Hong Zhang, Sylvia Ratnasamy, Ion Stoica

UC Berkeley

IoT: devices and apps

~7.74 billion connected IoT **devices**

In homes, offices, retail locations, commercial buildings..

New **applications:** smart spaces, logistics, agriculture, urban sensing..

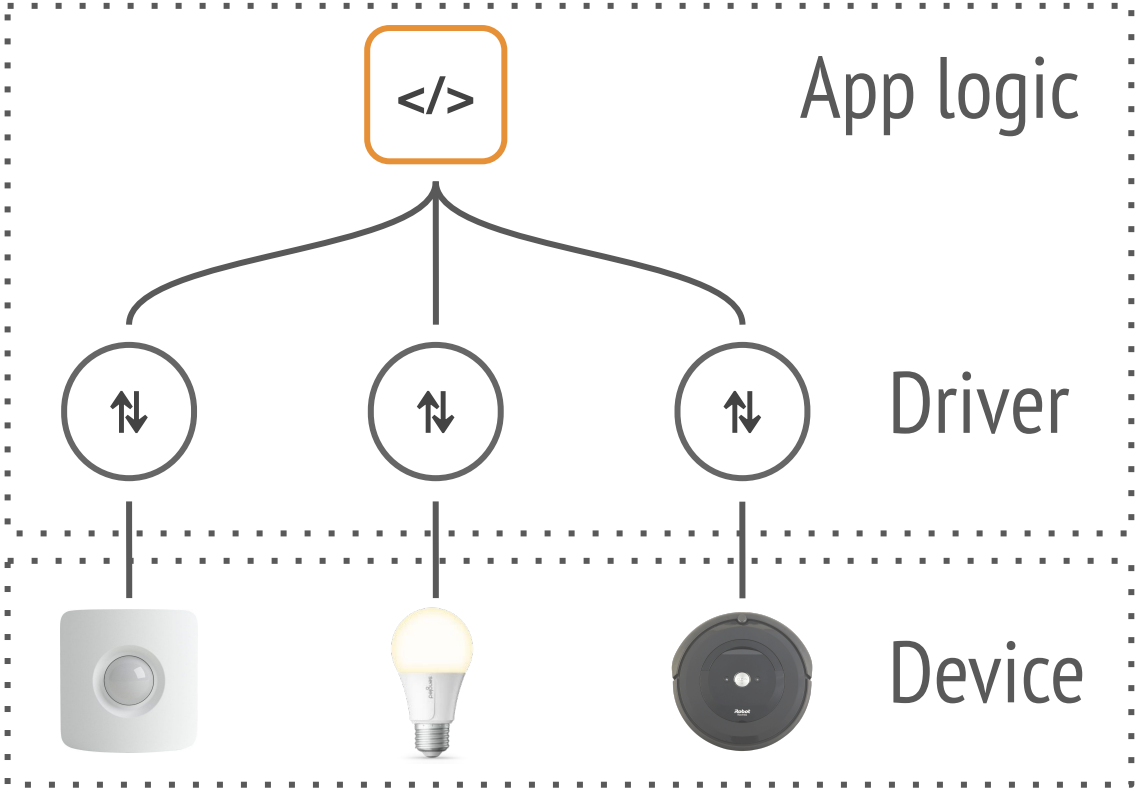
Picture courtesy of Comfy: https://comfyapp.com/

IoT: devices and apps

E.g. Smart building

Physical infrastructure

App framework



IoT: devices and apps

Setting up the testbed:
a bottleneck in prototyping
e.g., research, class, demo

Time, Cost

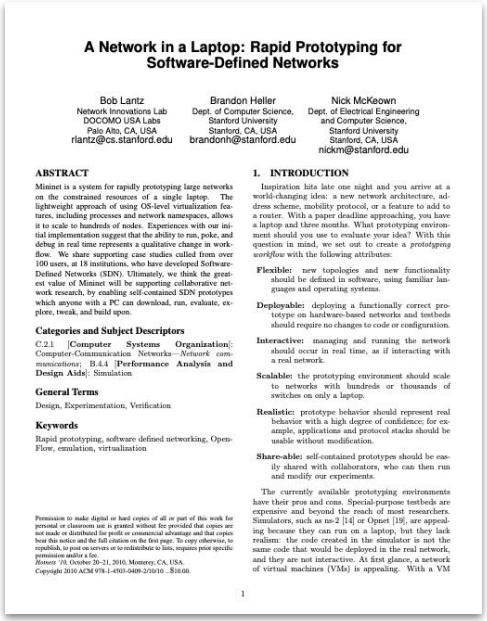
Physical
infrastructure



Device

Mininet

Bob Lantz, Brandon Heller, and Nick McKeown, HotNets 2010



A Network in a Laptop: Rapid Prototyping for Software-Defined Networks

Bob Lantz
Network Innovations Lab
DOCOMO USA Labs
Palo Alto, CA, USA
rlantz@cs.stanford.edu

Brandon Heller
Dept. of Computer Science,
Stanford University
Stanford, CA, USA
brandonh@stanford.edu

Nick McKeown
Dept. of Electrical Engineering
and Computer Science,
Stanford University
Stanford, CA, USA
nickm@stanford.edu

ABSTRACT

Mininet is a system for rapidly prototyping large networks on the constrained resources of a single laptop. The lightweight approach of using OS-level virtualization features, including processes and network namespaces, allows it to scale to hundreds of nodes. Experiences with our initial implementation suggest that the ability to run, poke, and debug in real time represents a qualitative change in workflow. We share supporting case studies culled from over 100 users, at 18 institutions, who have developed Software-Defined Networks (SDN). Ultimately, we think the greatest value of Mininet will be supporting collaborative network research, by enabling self-contained SDN prototypes which anyone with a PC can download, run, evaluate, explore, tweak, and build upon.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network communications*; B.4.4 [**Performance Analysis and Design Aids**]: Simulation

General Terms

Design, Experimentation, Verification

Keywords

Rapid prototyping, software defined networking, OpenFlow, emulation, virtualization

1. INTRODUCTION

Inspiration hits late one night and you arrive at a world-changing idea: a new network architecture, address scheme, mobility protocol, or a feature to add to a router. With a paper deadline approaching, you have a laptop and three months. What prototyping environment should you use to evaluate your idea? With this question in mind, we set out to create a *prototyping workflow* with the following attributes:

Flexible: new topologies and new functionality should be defined in software, using familiar languages and operating systems.

Deployable: deploying a functionally correct prototype on hardware-based networks and testbeds should require no changes to code or configuration.

Interactive: managing and running the network should occur in real time, as if interacting with a real network.

Scalable: the prototyping environment should scale to networks with hundreds or thousands of switches on only a laptop.

Realistic: prototype behavior should represent real behavior with a high degree of confidence; for example, applications and protocol stacks should be usable without modification.

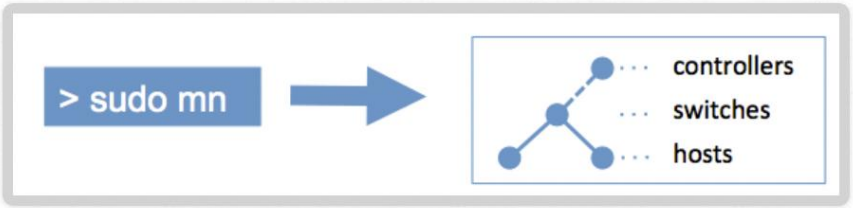
Share-able: self-contained prototypes should be easily shared with collaborators, who can then run and modify our experiments.

The currently available prototyping environments have their pros and cons. Special-purpose testbeds are expensive and beyond the reach of most researchers. Simulators, such as ns-2 [14] or Opnet [18], are appealing because they can run on a laptop, but they lack realism: the code created in the simulator is not the same code that would be deployed in the real network, and they are not interactive. At first glance, a network of virtual machines (VMs) is appealing. With a VM

Hotnets '10, October 20–21, 2010, Monterey, CA, USA.

SIGCOMM Test of Time Paper Award

Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine (VM, cloud or native), in seconds, with a single command:



Because you can easily interact with your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet is useful for development, teaching, and research.

Mininet is also a great way to develop, share, and experiment with Software-Defined Networking (SDN) systems using OpenFlow and P4.

Mininet is actively developed and supported, and is released under a permissive BSD Open Source license. We encourage you to contribute code, bug reports/fixes, documentation, and anything else that can improve the system!

Get Started

Download a Mininet VM, do the walkthrough and run the OpenFlow tutorial.

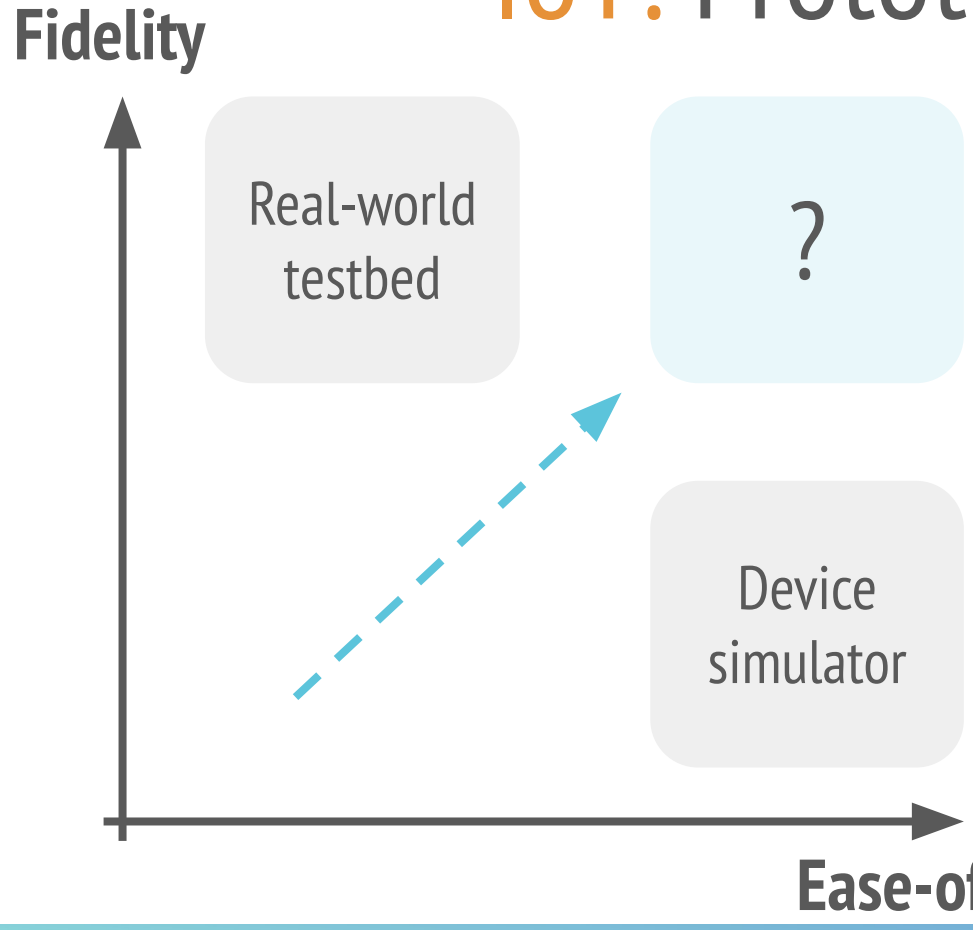
Support

Read the FAQ, read the documentation, and join our mailing list, mininet-discuss.

Contribute

File a bug, download the source, or submit a pull request - all on GitHub.

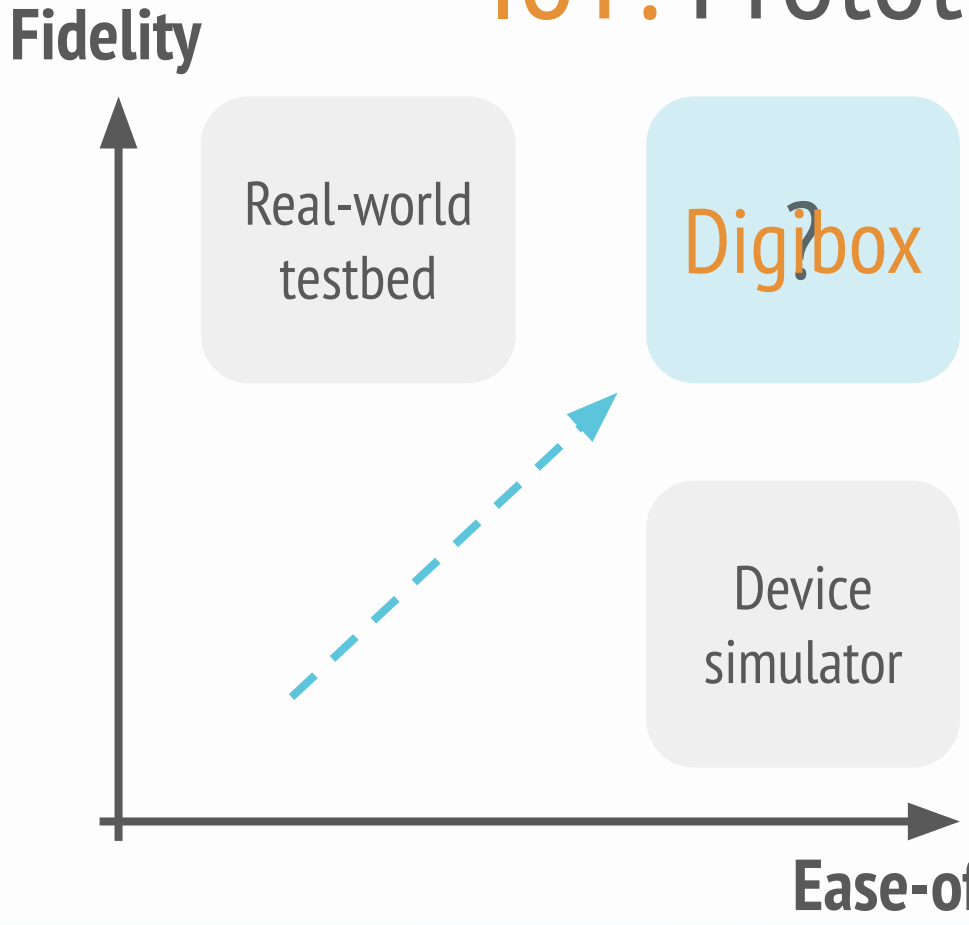
IoT: Prototyping Env.



Real-world testbed
- Human interaction
- Correlated behavior

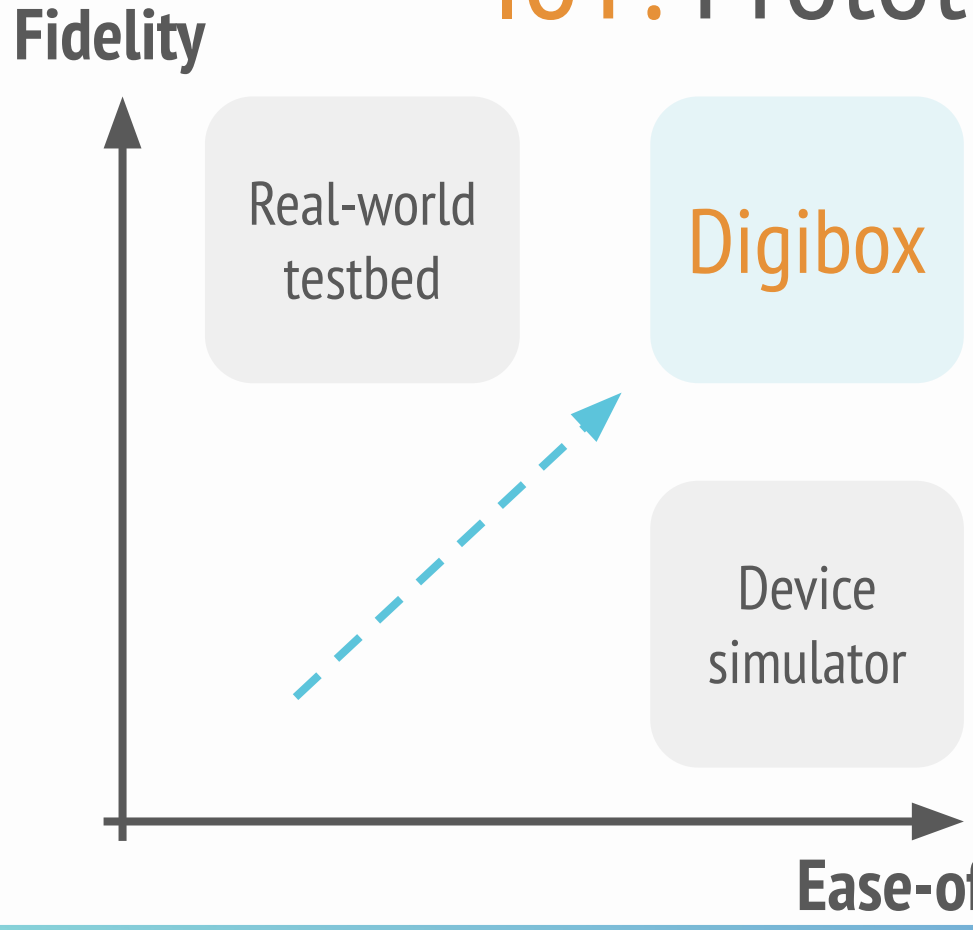
Today: device simulator
- H/W capability
- Individual device behavior
  - e.g. events, messages

IoT: Prototyping Env.



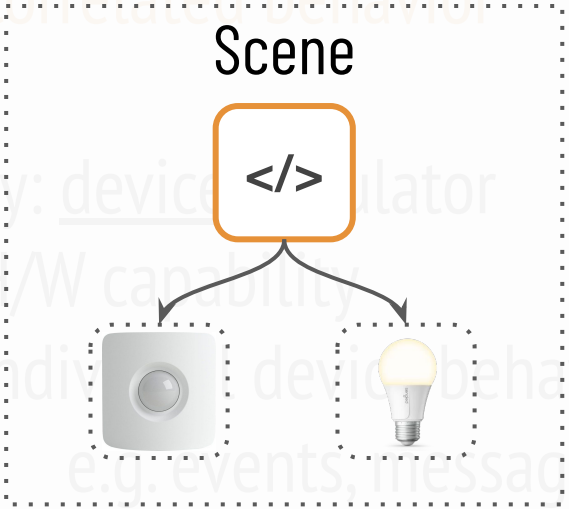
- Human interaction
- Correlated behavior
- Easy to reproduce
- Easy to (re)use
- Scalable

IoT: Prototyping Env.



Key insight:

Device-centric → **Scene**-centric



Digibox Walkthrough

w/Demo

Digibox

Design

Mock

Scene

Attach

Property

Interactive

Ensemble support

Reproducible

Customizable

Scalable

Digibox

Design

Mock

Scene

Attach

Property

Interactive

Ensemble support

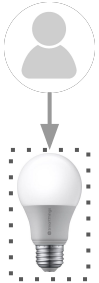
Reproducible

Customizable

Scalable

In the demo:

Mockup device



Lamp

CLI: digi
(alias: dbox)

Command:
pull/run,
check/edit/query

Goal: Interactivity

Digibox

Design

Mock

Scene

Attach

Property

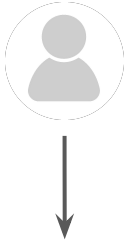
Interactive

Ensemble support

Reproducible

Customizable

Scalable



Model

lamp.power
.intent: **on**
.status: **off**

Simulator

get(lamp.intent)
.status = ...

Goal: Interactivity

Each mock has a **model** and a **simulator**

- Model: declarative interface, intent & status
- Simulator: a piece of Python code

..Simulator subscribes to intent changes and update the status on the model

Each mock runs in its own microservice ("digi")

- Isolation: separate dev./deployment lifecycle

Digibox

Design

Mock

Scene

Attach

Property

Interactive

Ensemble support

Reproducible

Customizable

Scalable

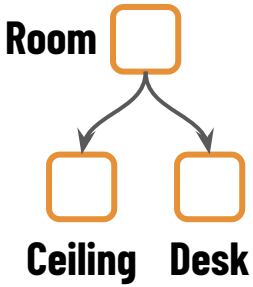
In the demo:

Scene

Room

Sensor: "ceiling"
Sensor: "desk"

Command:
attach



Goal: Ensemble Support

Digibox

Design

Mock

Scene

Attach

Property

Interactive

Ensemble support

Reproducible

Customizable

Scalable

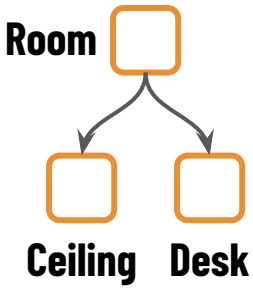
In the demo:

Scene

Room

Sensor: "ceiling"
Sensor: "desk"

Command:
attach



Goal: Ensemble Support

Scene: same components as a mock (model & simulator)

Attach(M, Sc):

allows Sc.sim to write to the mock's model M

Digibox

Design

Mock

Scene

Attach

Property

Interactive

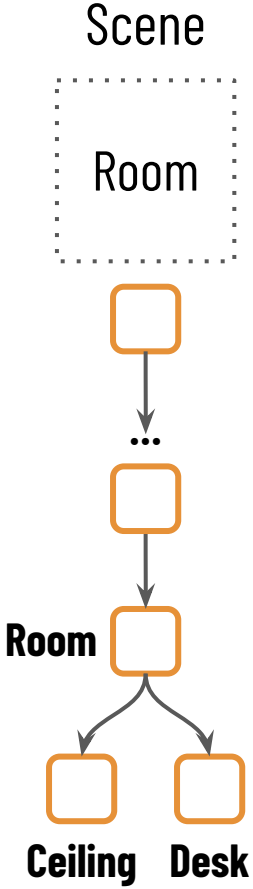
Ensemble support

Reproducible

Customizable

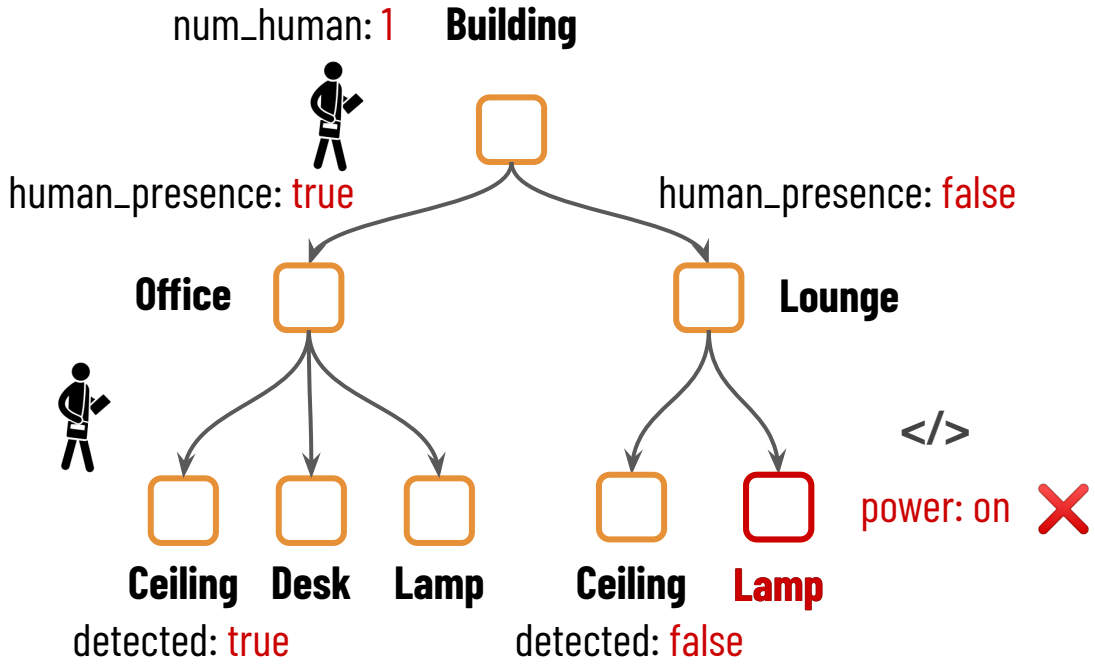
Scalable

In the demo:



Goal: Ensemble Support

Scene composition



Digibox

Design

Mock

Scene

Attach

Property

Interactive

Ensemble support

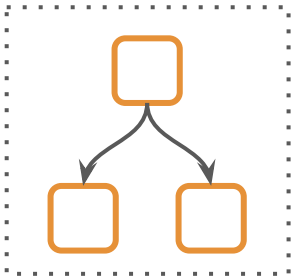
Reproducible

Customizable

Scalable

In the demo:

Take a snapshot of mocks/scenes



Command:

commit

push

Goal: Share and Reproduce

Digibox

Design

Mock

Scene

Attach

Property

Interactive

Ensemble support

Reproducible

Customizable

Scalable

See the paper for details:

Programming Mocks and Scenes

Creating, programming, and configuring digis

Integrating mocks with app frameworks

Implementation and Scalability

Kubernetes-based runtime

Scaling from a laptop to machine cluster in cloud

Sharing and Reproducing Scenes

Managing mocks and scenes with Git/GitOps

Logging and replaying traces

Digibox

Design

Mock

Scene

Attach

Property

Interactive

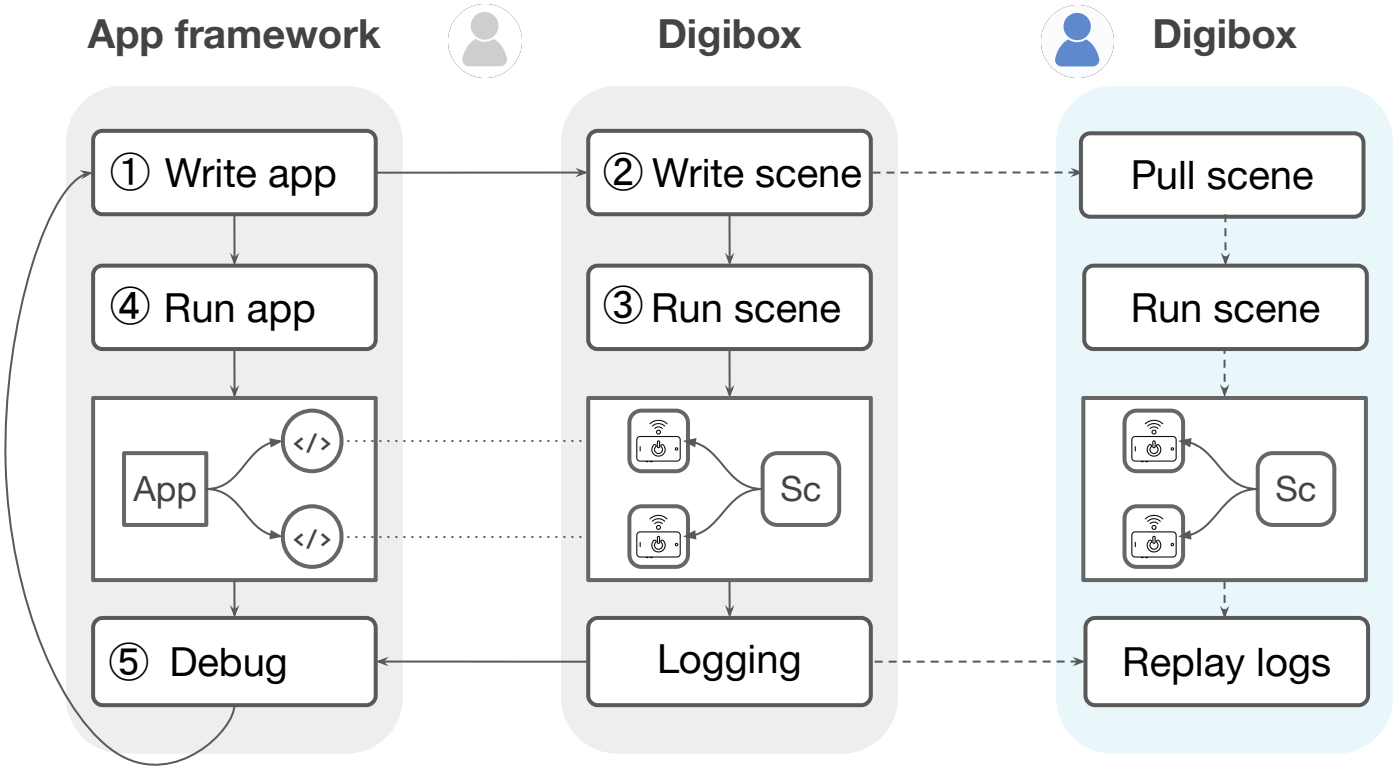
Ensemble support

Reproducible

Customizable

Scalable

Prototyping Workflow

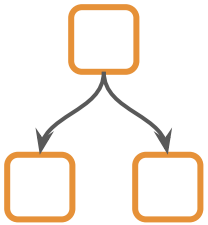


Goal: Accelerate IoT app prototyping (sim./testing)

Interactive, ensemble behaviors, reproducible, scalable

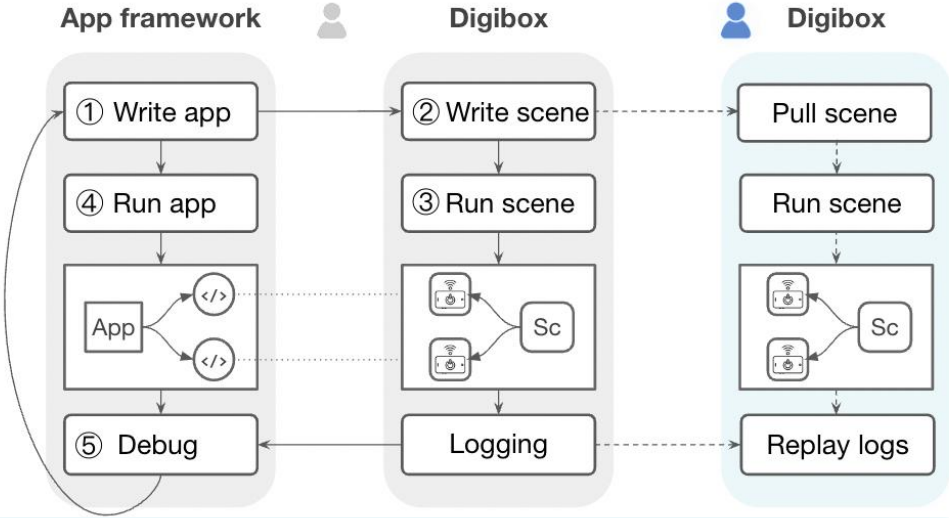
Digibox

Mock, Scene + Attach



digi.dev/digi

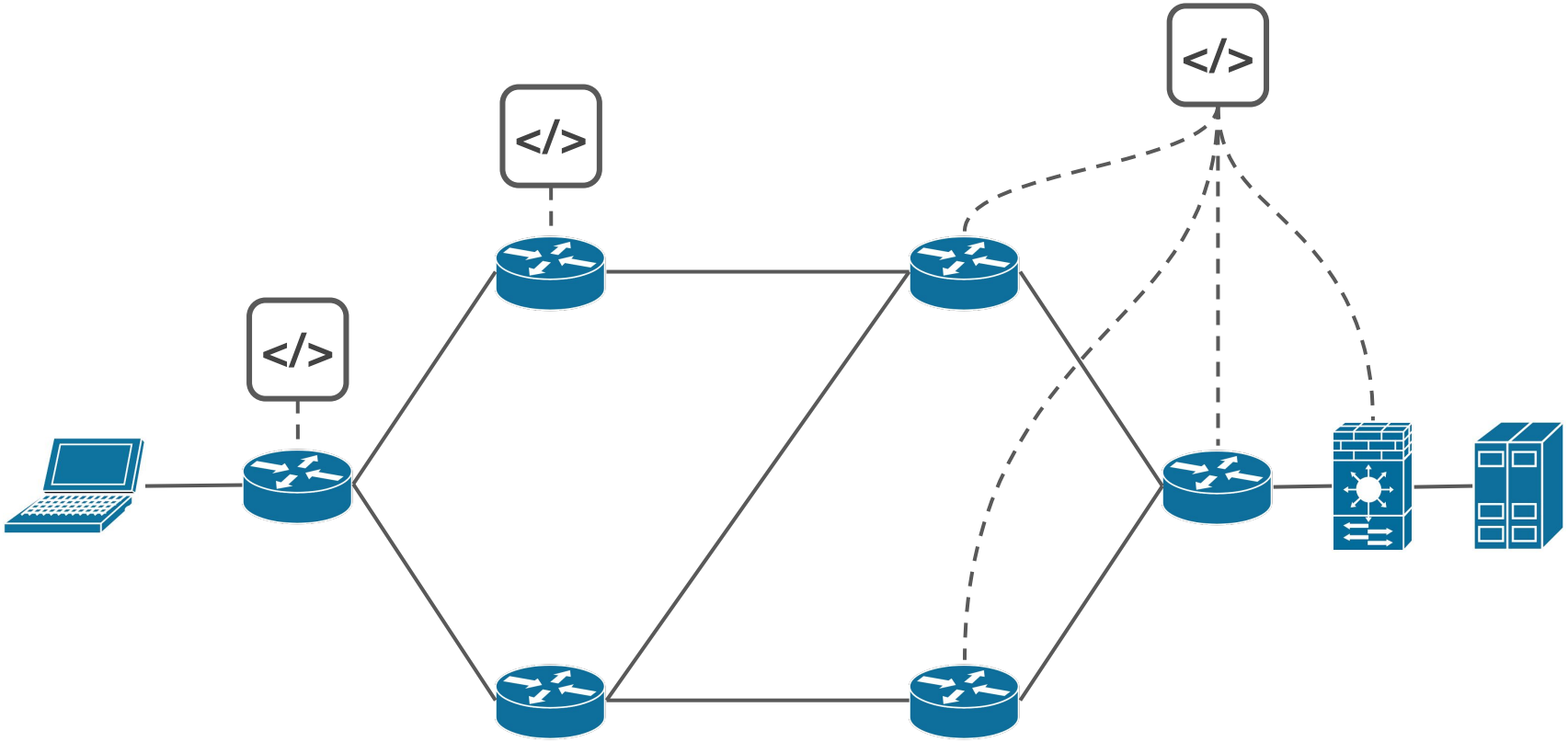
Prototyping Workflow



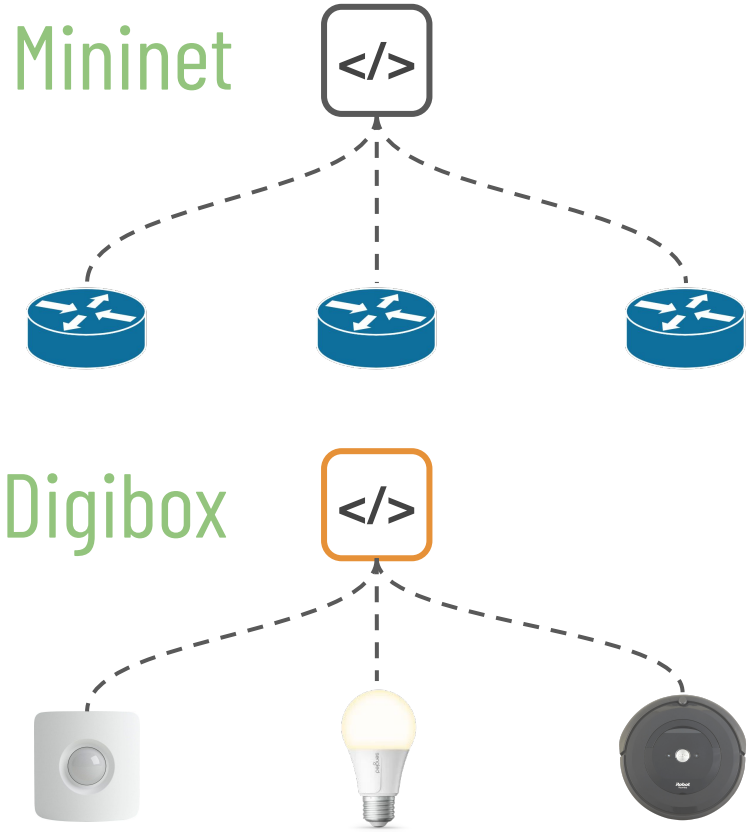
Looking ahead

"Net Apps"

Network Apps



Network Apps



Interacting with network devices
→ the physical world

Open Challenges

Prototyping environment:

High-fidelity simulation
Efficient, large-scale simulation
Supporting new applications

Goal: Accelerate IoT app prototyping (sim./testing)

Interactive, ensemble behaviors, reproducible, scalable

Digibox

Mock, Scene + Attach

Thank you!

digi.dev/digi

Prototyping Workflow