MIRAGE OS

http://openmirage.org/

Reconstructing library operating systems

Virtualization is useful in many situations, but adds yet another layer to an already highly-layered software stack now including: support for old physical protocols (e.g. disk standards developed in the 80s such as IDE); irrelevant optimisations (e.g. disk elevator algorithms on SSD drives); backward-compatible interfaces (e.g. POSIX); user-space processes and threads (in addition to VMs on a hypervisor); managed code runtimes (e.g. OCaml, .NET or Java) which all sit beneath your application code.

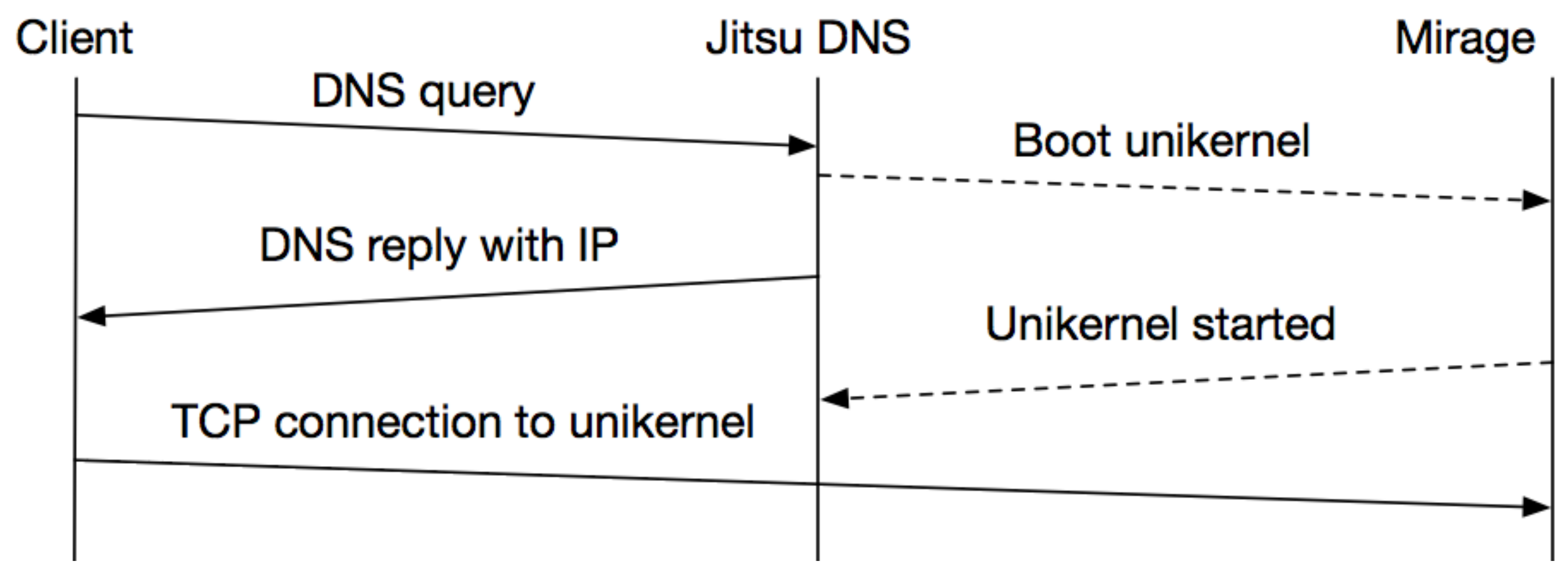
Are we really doomed to adding new layers of indirection and abstraction every few years, leaving future generations of programmers to become virtual archeologists as they dig through hundreds of layers of software emulation to debug even the simplest applications?

Mirage OS is a **library operating system** that constructs **unikernels** for **secure**, **high-performance** network applications across a variety of **cloud computing** and **mobile** platforms. Code can be developed on a normal OS such as Linux or MacOS X, and then compiled into a fully-standalone, specialised unikernel that runs under the Xen hypervisor.

Mirage is based around the **OCaml** language, with syntax extensions and 50 + libraries which map directly to operating system constructs when being compiled for production deployment. As such, Mirage includes **clean-slate functional implementations** of protocols ranging from **TCP/IP**, **DNS**, **SSH**, **TLS**, **Openflow**, **HTTP**, **XMPP** and **Xen** inter-VM transports.

INTRODUCING MIRAGE OS 2.0
These slides were written using Mirage on OSX:
They are hosted in a 938kB Xen unikernel written in statically type-safe OCaml, including device drivers and network stack.
Their application logic is just a couple of source files, written independently of any OS dependencies.
Running on an ARM CubieBoard2, and hosted on the cloud.
Binaries small enough to track the entire deployment in Git!
MacBook Air

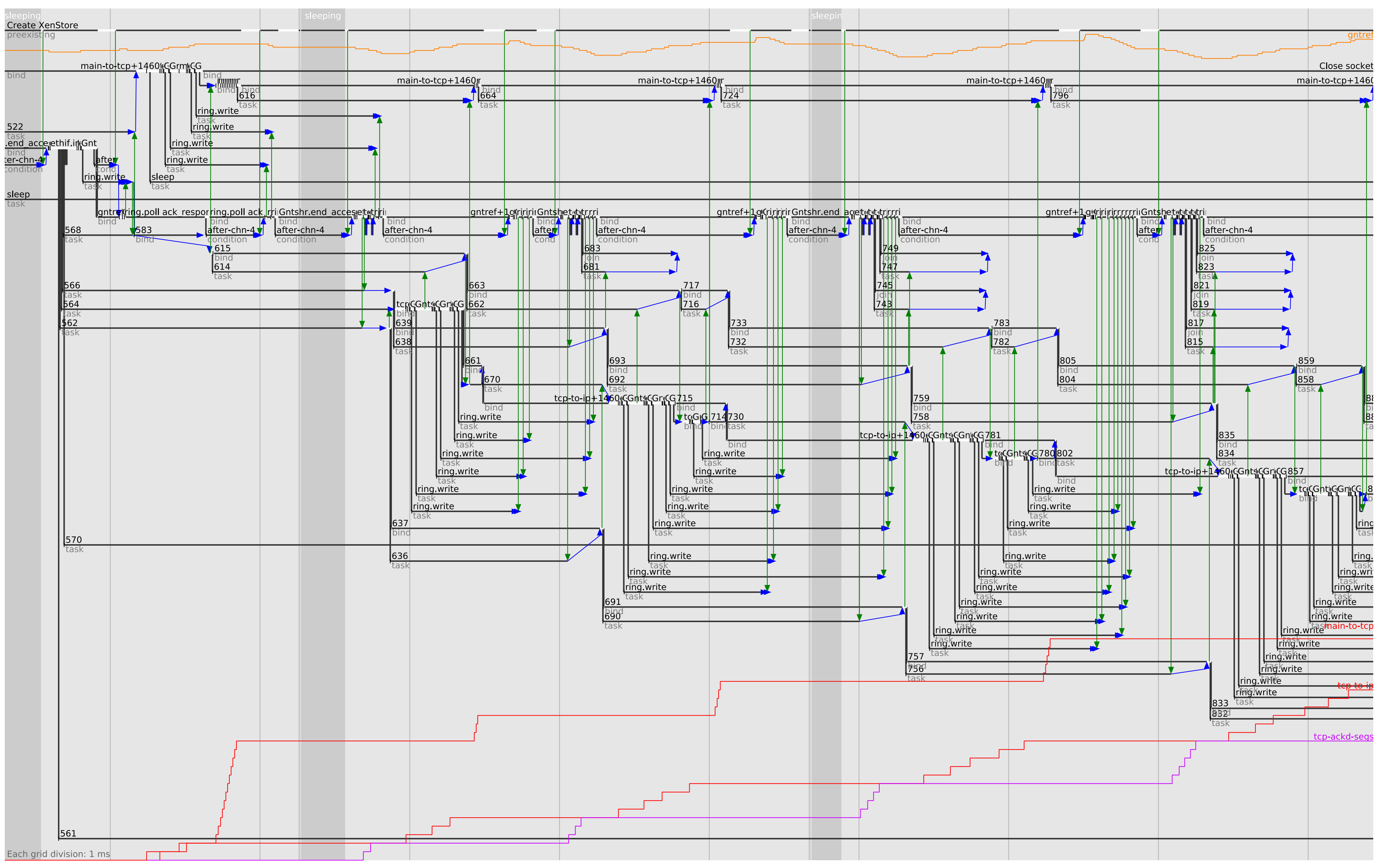
A Mirage web server unikernel running as a Xen guest on an ARM CubieTruck serves up a slide deck.



Example: Jitsu

Just-in-time summoning of unikernels (Jitsu) is a DNS server that saves resources by starting Mirage instances on demand.

Mirage boots in less than 10 ms on x86 and is up and running before the client has received the DNS response and opened a TCP connection.



Example: Tracing

The trace on the left shows a unikernel running on an ARM board transmitting TCP packets.

Horizontal black lines are Lwt threads/ promises, green arrows indicate a promise being resolved and blue arrows a result being read.

At the bottom, the three lines show how many TCP packets have been buffered for sending, transmitted, and ack'd by the remote end.

Having everything in a single language makes whole-stack tracing easier than in traditional systems.

UNIVERSITY OF CAMBRIDGE

Technical Contact:
Anil.Madhavapeddy@cl.cam.ac.uk