Microkernels, Genode and Gapfruit OS

Timo Nicolai and Alice Domage

2023

Gapfruit AG

Content

Alice Domage

- Information Technologies Expert @ Epitech Paris
- Operating systems engineer @ Gapfruit
  - Framework updates and upstream contributions
  - Involved in platform support for various hardwares and architectures including x86_64 and arm_v8a
  - Involved in creation and maintenance of native, and Linux ported, device drivers

Timo Nicolai

- M.Sc. Information System Technology @ TU Dresden
- Previously intern @ Kernkonzept
  - Worked on Fiasco.OC/L4Re
  - Extended `uvmm` hypervisor
- Now operating systems engineer @ Gapfruit
  - Integrating Gapfruit OS with Azure IoT
  - Involved in designing and implementing our PKI

What is a Microkernel?

- Kernel only implements minimal functionality
- Drivers, resource management etc. implemented in userspace
- Kernel provides *mechanisms* instead of *policy*

Advantages of Microkernels

- **Trustworthiness**
  - Small trusted code base
  - Exploited drivers do not compromise kernel
- **Reliability**
  - Misbehaving components can be restarted without rebooting
- **Resource Management**
  - No “overpromising” of resources
  - Resource trading

Examples of Microkernel Systems

- **MINIX** [1]
- **GNU Hurd** [2]
- **Zircon/Fuchsia** [3]
- **Redox** [4]
- **L4 Family**

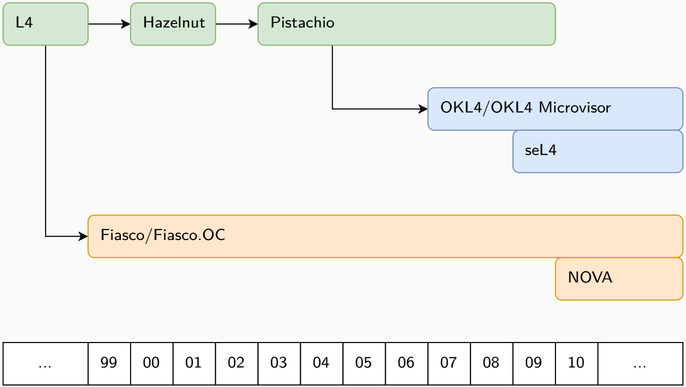
(L4) Microkernel History: Predecessors

- **RC 4000 Multiprogramming System** [5] (Per Brinch Hansen, 1969)
  - Possibly the first microkernel system
- **Mach** [6] (Richard Rashid and Avie Tevanian, 1985)
  - UNIX compatible
  - Basis of GNU Hurd
  - Painfully slow IPC
- **L4** [7] (Jochen Liedtke, early '90s):
  - Handwritten in Assembly
  - Fast IPC: synchronous, uses registers, cache friendly

(L4) Microkernel History: Recent Developments

- **University of Karlsruhe** (Jochen Liedtke)
  - *L4Ka::Hazelnut* [8]: Written in C++
  - *L4Ka::Pistachio* [9]: Cross-platform
- **UNSW/NICTA** (Gernot Heiser)
  - *OKL4*, *OKL4 Microvisor* [10] (now @ *General Dynamics*)
  - *seL4* [11]: Formally verified, fast and suitable for realtime
- **TU Dresden** (Hermann Härtig)
  - *Fiasco*/*Fiasco.OC* [12] (now @ *Kernkonzept*): L4Linux
  - *NOVA* [13] (now @ *BedRock Systems*): Focus on virtualization

(L4) Microkernel History: Recent Developments



Based on [14]

(L4) Microkernel History: The Third Generation

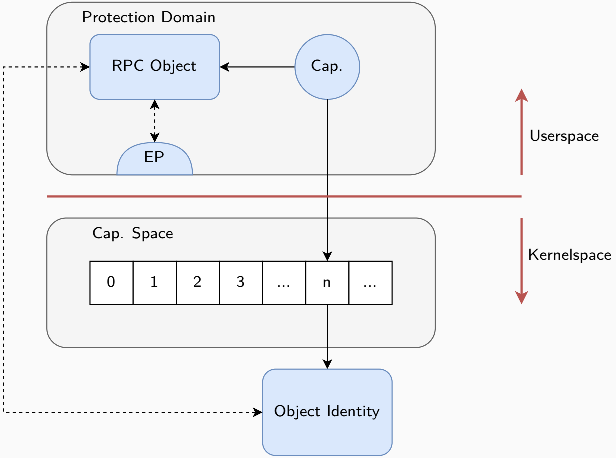
- Capability-based security
- Virtualization
- Formal verification

- Open source, developed at *Genode Labs* (+ *Sculpt* [15])
- Operating system *framework* [16]
- Includes the *HW* microkernel
- Runs on top of Linux, HW, Fiasco.OC, NOVA, OKL4, seL4

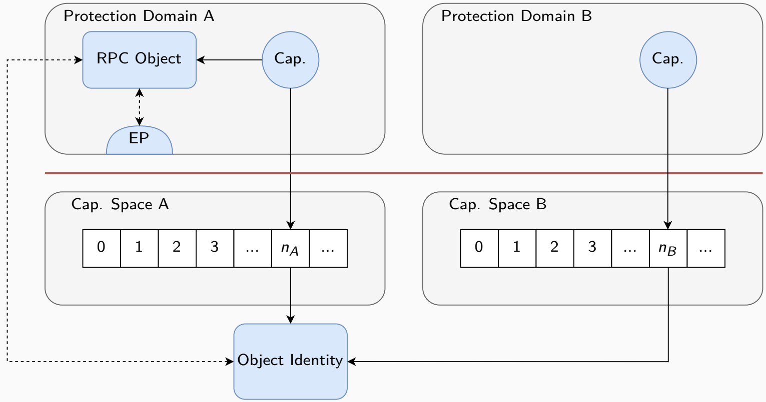
Genode: Capabilities

- Components need to provide/consume services
- Capabilities:
  - Live inside *protection domains*
  - Combine "pointers" and "access rights" to RPC objects
  - Are used to invoke methods of RPC objects

Genode: Capability Creation



Genode: Capability Creation



Genode: Capability Invocation

- Terminology: *client* (**B**) and *server* (**A**)
- RPC is mediated by kernel:
  1. Thread in **B** *calls* method of RPC object using its capability, providing client-local name, opcode, and arguments
  2. Kernel checks validity of capability
  3. Kernel finds corresponding RPC object in **A** and wakes up entrypoint thread
  4. Entrypoint thread in **A** finds and invokes the method of the RPC object
  5. Entrypoint thread in **A** may produce return value and *reply*
  6. Return value is passed back, if necessary, to calling thread which is woken up

Genode: Capability Delegation

- Need to be able to *delegate* capabilities between PDs
- Capabilities can be passed as RPC arguments/return values
- Kernel will create capabilities in receiving PD as necessary
- So, in order to transmit capabilities we require capabilities???

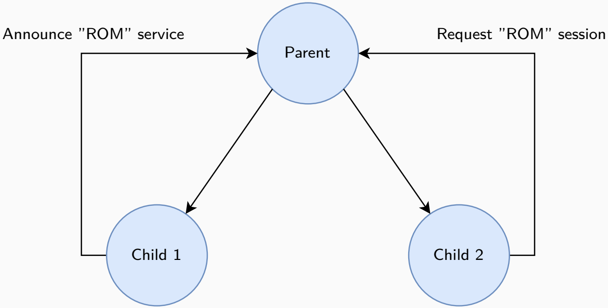
Genode: The Component Hiearchy: Parents and Children

- Every component except *core* has exactly one parent
- Parent provides children with resources and controls them
- Children start out with a single capability to parent
- Children have to trust parent but not the other way around

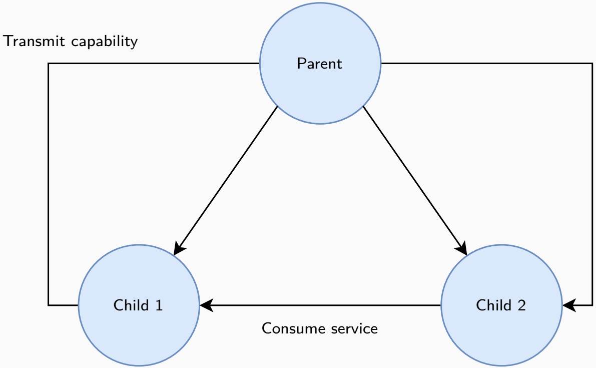
Genode: The Component Hiearchy: Services and Sessions

- Children announce *services* via RPC objects implementing *root interface*
- Other children can perform *session requests*
- Parent can modify these and deny them, handle them, forward them to a child/its own parent
- Once a session is established, communication is direct
- Clients need to trust servers but not the other way around
- Clients can share resources when initiating a session

Genode: The Component Hiearchy: Services and Sessions



Genode: The Component Hiearchy: Services and Sessions



Genode: The C++ Runtime

- Most native Genode components don't use the STL
- Genode supports exceptions
- Genode comes with its own minimal C++ runtime

Genode: A Simple Genode Component

Let's try to build this:

```
timer tick 1
timer tick 2
timer tick 3
timer tick 4
timer tick 5
```

Genode: A simple Genode component

```
/* component which outputs a message 5 times and then exits */
class Main {
private:
    Genode::Env &_env;

    /* timer */
    Timer::Connection _timer { _env };
    /* timer tick counter */
    Genode::size_t _timer_ticks { 0 };
    /* timer trigger callback */
    Genode::Signal_handler<Main> _timer_handler { _env.ep(), *this, &Main::_handle_timer };

    void _handle_timer() {
        if (++_timer_ticks > 5)
            _env.parent().exit(0);

        Genode::log("timer tick ", _timer_ticks);
    }

public:
    explicit Main(Genode::Env &env) : _env { env} {
        /* initialize timer trigger handler */
        _timer.sigh(_timer_handler);
        /* trigger timer periodically (once every second) */
        _timer.trigger_periodic(1'000'000);
    }
};

/* component entry point */
void Component::construct(Genode::Env &env) {
        static Main main { env };
}
```

Gapfruit OS: Purpose

- Trustworthy/reliable OS for IoT edge gateways
- Collect data from devices and bring it to the cloud
- Control devices via the cloud

Gapfruit OS: Challenges

- Focus on integrity, stability, reliability and uptime
- Minimize human interaction with the system (*Zero-Touch*)
- Deploy devices and establish cloud connectivity automatically

Gapfruit OS: Solutions

- $>$ 99% reduction of attack surface
- Strong control over service dependency and delegation
- Able to recover malfunctioning components (via *heartbeats*)
- Updating e.g. drivers without downtime is trivial
- Use TPM-backed PKI for automatic mass deployment

Gapfruit OS: Azure Example

Example *Device Twin*:

```
"properties": {
  "desired": {
    "dynamic": {
      "deploy": [
        {
          "start": {
            "name": "http_server",
            "pkg": "gapstage/pkg/http_server/2023-01-01"
          }
        },
        {
          "start": {
            "name": "mqtt_bridge",
            "pkg": "gapstage/pkg/mqtt_bridge/2023-02-01"
          }
        }
      ]
    }
  }
}
```

The End

Thank you for listening, are there any questions?

References

[1] *MINIX 3*. https://www.minix3.org/.

[2] *GNU Hurd*. https://www.gnu.org/software/hurd/.

[3] *Fuchsia*. https://fuchsia.dev/.

[4] *Redox OS*. https://www.redox-os.org/.

[5] Per Brinch Hansen. *RC 4000 Software: Multiprogramming System*. Regnecentralen. 1969.

[6] Richard Rashid, Avadis Tevanian, and Michael Young. “Mach: A New Kernel Foundation for UNIX Development”. In: *USENIX Summer Conference*. 1986.

[7] Jochen Liedtke. “Improving IPC by Kernel Design”. In: *SIGOPS Oper. Syst. Rev.* 1993.

[8] *L4Ka::Hazelnut*. https://www.l4ka.org/57.php.

[9] *L4Ka::Pistachio*. https://www.l4ka.org/65.php.

[10] Gernot Heiser and Ben Leslie. “The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors”. In: *Proceedings of the First ACM Asia-Pacific Workshop on Workshop on Systems*. 2010.

[11] Gerwin Klein et al. “SeL4: Formal Verification of an OS Kernel”. In: *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*. 2009.

[12] *The L4Re Microkernel*. https://os.inf.tu-dresden.de/fiasco/.

[13] *NOVA Microhypervisor*. https://hypervisor.org/.

[14] Gernot Heiser and Kevin Elphinstone. “L4 Microkernels: The Lessons from 20 Years of Research and Deployment”. In: *ACM Trans. Comput. Syst.* (2016).

[15] *Sculpt OS*. https://genode.org/download/sculpt.

[16] *Genode Foundations*. https://genode.org/documentation/genode-foundations/index.