NICTA

CAmkES Tutorial: Theory

14 August 2015

Ihor Kuz

Australian Government

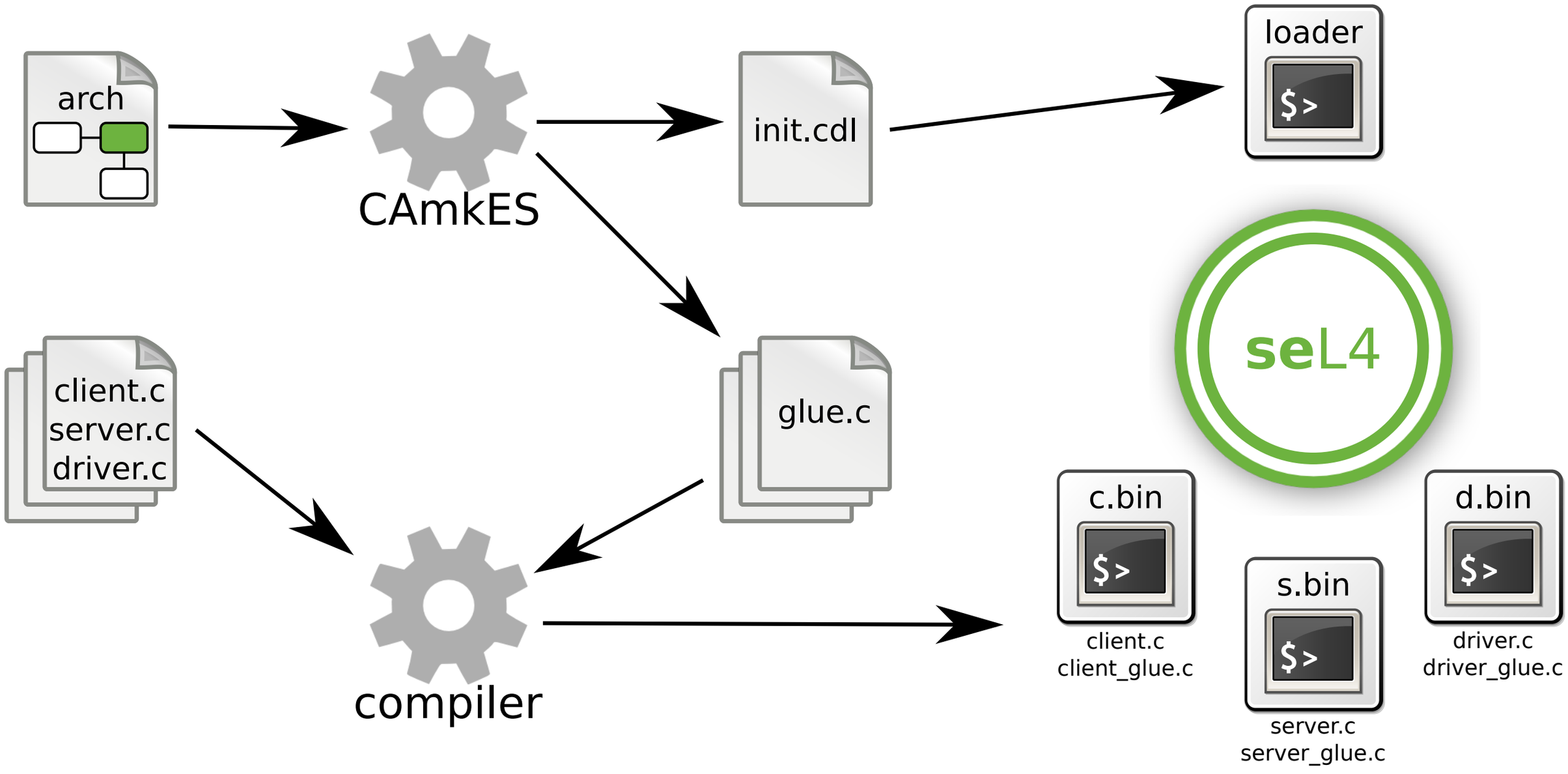
CAmkES

Component Architecture for micro-kernel based Embedded Systems

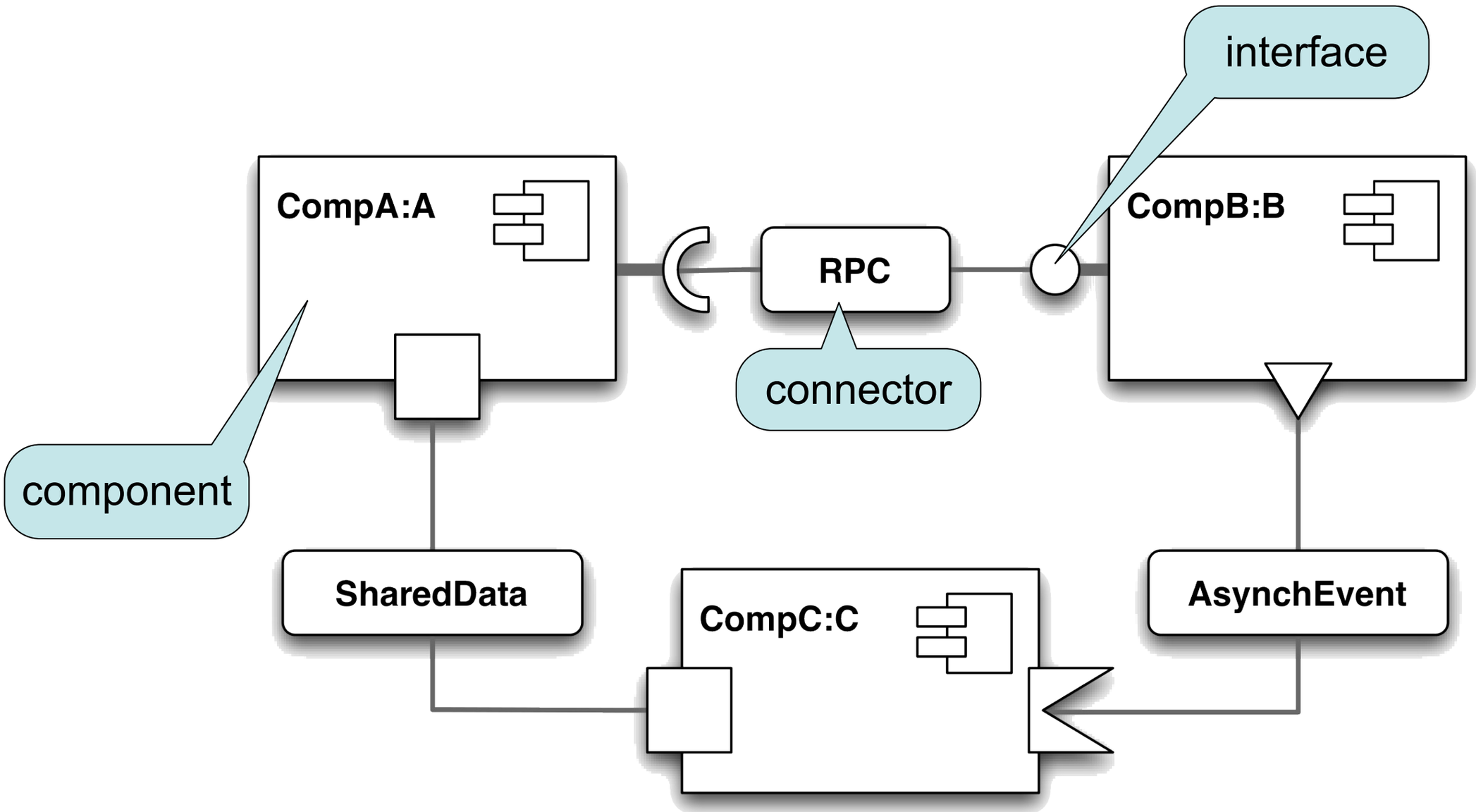
- Goal
  - Simplify *development* & *reasoning* for µk-based systems
- History
  - Originally on L4:Pistachio, OKL4. Rewritten for seL4
- Properties
  - Static: all components, connections defined at build time
  - Generated glue code
- Principles
  - Explicit architecture, Connectors as first class concepts
  - Don’t pay for what you don’t use

CAmkES in a nutshell

NICTA



Example System



Main Concepts

- Component
  - Component *Type* vs Component *Instance*
- Interface
  - RPC (Remote Procedure Call): synchronous comm
  - Event: notifications
  - Dataport: shared data
- Connector
  - *Connector* Type vs Connector Instance (*Connection*)
- Assembly
  - Composition
  - Configuration

Component

- ADL code

```
component Client {
        control;  // has thread of control
        uses Simple a;  // use an interface of another component
        provides Complex b; // implements and interface
        attribute int num_widgets; // config data for component
}
```

RPC Interfaces

- IDL code

```
procedure Simple {
    string echo_string(in string s);
    int echo_int(in int i);
    int echo_parameter(in int pin, out int pout);
};
```

- C code

```
char * b_echo_string(const char *s)
int b_echo_int(int i)
int b_echo_parameter(int pin, int *pout)
```

  - note: “b” is name of provider interface

Events

- ADL code
  - consumes Event ev;
  - emits Event ev;
- C code
  - ev_emit()
  - ev_wait()
  - ev_reg_callback(void (*callback)(void*), void *arg)

Dataports

- ADL code
  - dataport Buf d;
  - include <my_typedefs.h>;
  - dataport a_typedef_t d;
- C code
  - char d[PAGE_SIZE];
  - a_typedef_t d;
- dataport pointers
  - #include <camkes/dataport.h>
  - dataport_ptr_t dataport_wrap_ptr(void *ptr);
  - void *dataport_unwrap_ptr(dataport_ptr_t ptr);

Connectors

- Standard connectors
  - include <std-connectors.camkes>
- Connection
  - connection <Connector> <conn_name>(from comp.inf, to comp.inf);
  - e.g.: connection seL4RPC ab_conn(from a.i, to b.i);
- User-defined connectors
  - ADL spec

```
connector seL4RPCCallDataport {
    from Procedure user_inf template "seL4RPCCallDataport-from.template.c";
    to Procedure provider_inf template "seL4RPCCallDataport-to.template.c";
}
```

  - template code: generate glue code. uses python and C.

Assembly: Composition

- ADL code

```
assembly {
        composition {
                component Echo echo;
                component Client client;

                connection seL4RPC simple(from client.a, to echo.b);
        }
}
```

Assembly: Configuration

- ADL code

```
assembly {
        composition {
            …
        }
        configuration {
            client.num_widgets = 2;
            client.priority = 200;
    }
```

  - Component attributes: num_widgts
  - Infrastructure attributes: priority
- C code
  - variable with attribute name, contains value
  - do_something(num_widgets);

Composite Components <TODO>

- ADL code

```
component Outer {
    provides Simple s;
    composition {
        component InnerA a;
        component InnerB b;
        connection seL4RPC internal1(from a.i, to b.i);
        connection ExportRPC exp1(from a.s, to s);
    }
}
```

- C code
  - component “Outer” disappears