seL4 Developers Day

Beginner CAmkES Development

Ihor Kuz, Stephen Sherratt

October 2015

Agenda and Exercises

Agenda

- CAmkES overview
- Building and running
- RPC
- Dataports
- Events
- Attributes

Exercises

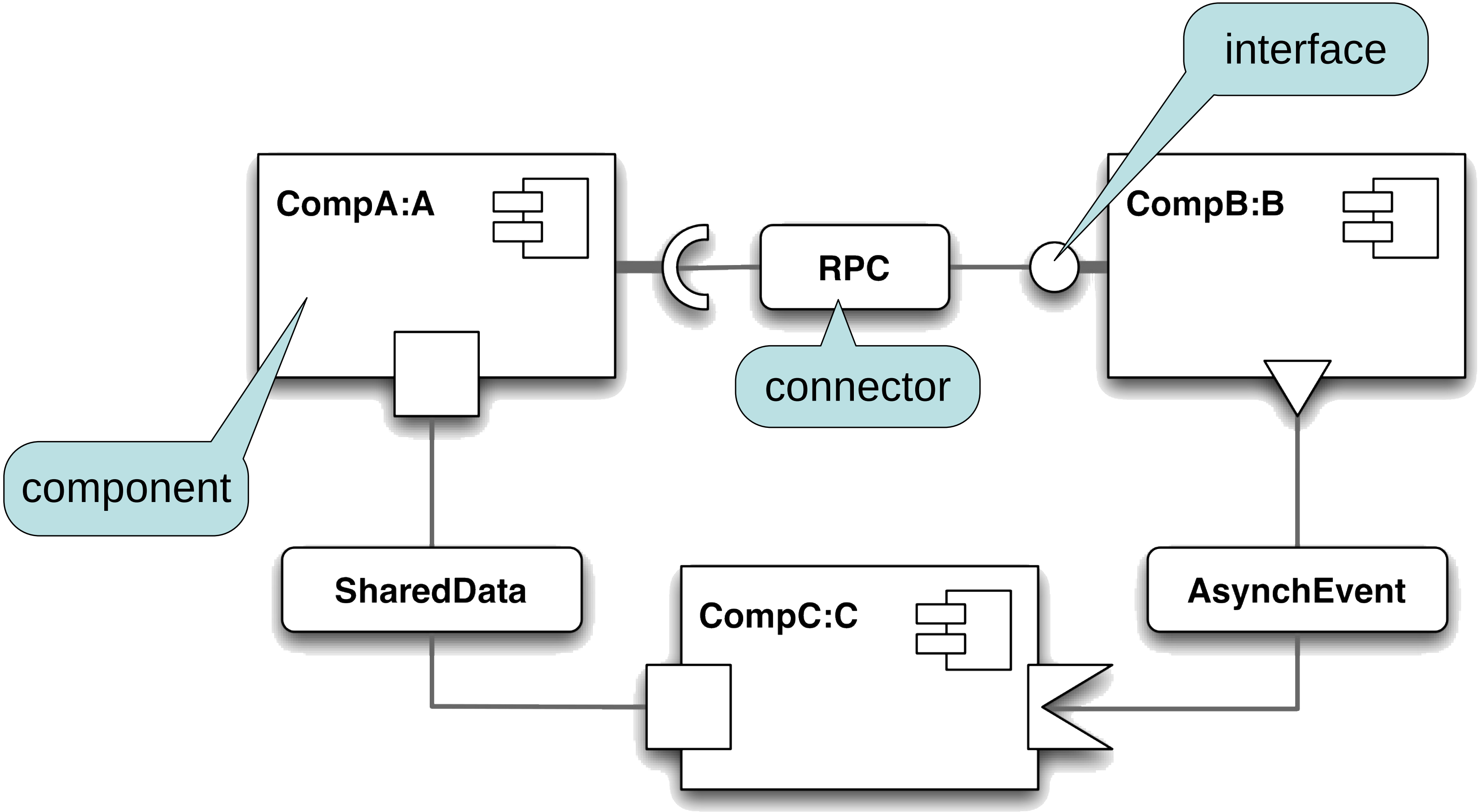
- Hello-camkes-0
  - Simple hello world
- Hello-camkes-1
  - Using RPC
- Hello-camkes-2
  - Using dataports
  - Using events
  - Using attributes

CAmkES

Component Architecture for micro-kernel based Embedded Systems

- Goal
  - Simplify development & reasoning for microkernel-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

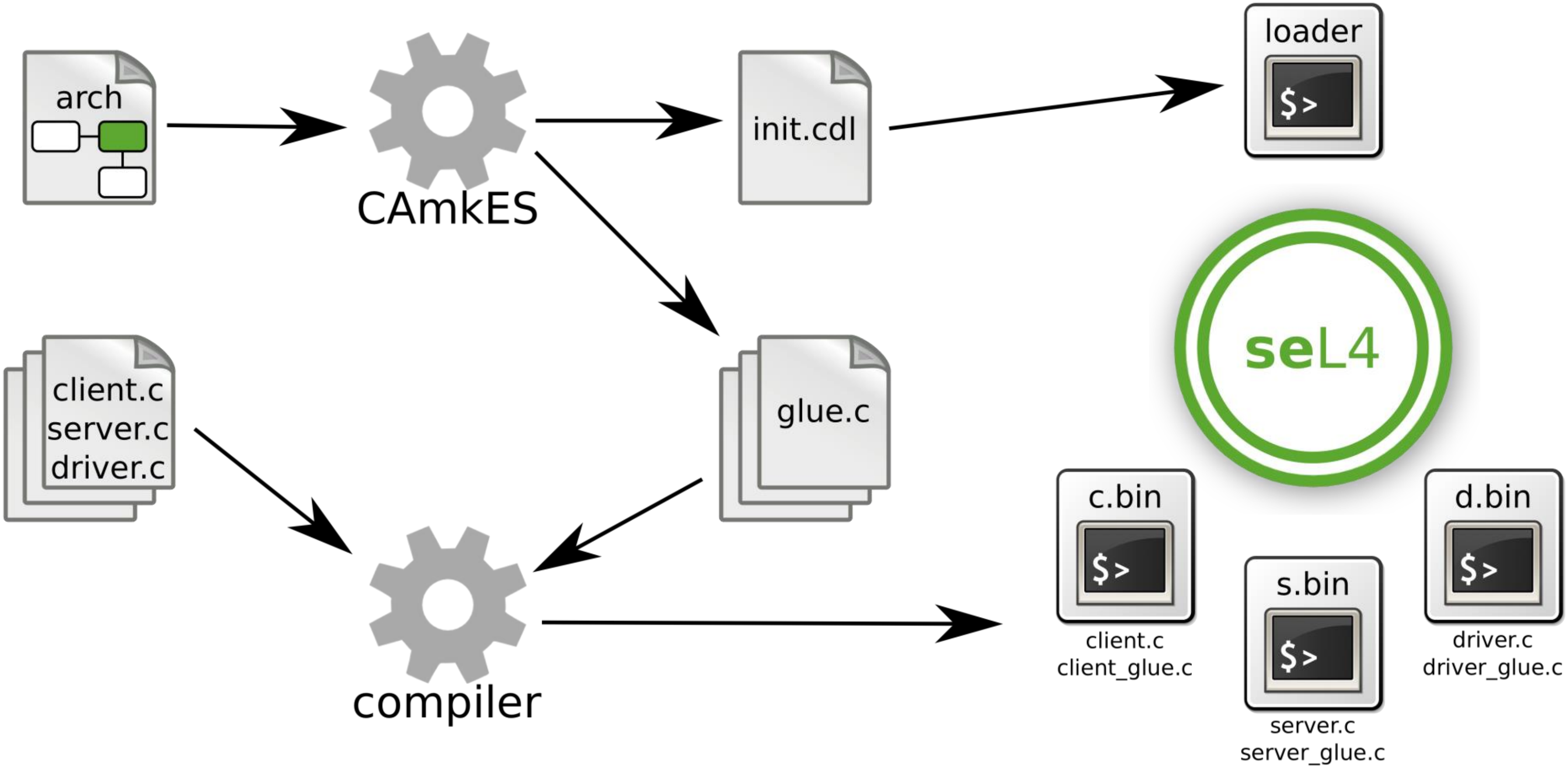
Example system



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
- ADL and IDL
  - Architecture Description Language, Interface Definition Language

CAmkES in a nutshell



Documentation: https://github.com/seL4/camkes-tool/blob/master/docs/index.md

Hands-on: hello-camkes-0

- Getting code
  - If you haven't already
  - Note: this is different than the seL4 tutorial

```
mkdir camkes-tutorials; cd camkes-tutorials
repo init -u http://github.com/sel4-projects/sel4-tutorials-manifest
       -m camkes-tutorials.xml
repo sync
```

- Working directory
  - `apps/hello-camkes-0/`
    - `hello-camkes-0.camkes`
    - `components/`
      - `Client/: Client.camkes, src/ include/`
- `make arm_hello-camkes-0_defconfig` (can also use ia32); `make`
- Run using qemu: `qemu-system-arm -M kzm -nographic -kernel images/capdl-loader-experimental-image-arm-imx31`

CAmkES

- ADL
- RPC and IDL

Component

- ADL code example

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

- https://github.com/seL4/camkes-tool/blob/master/docs/index.md#creating-an-application

Connectors

- Standard connectors
  - In ADL: `include <std-connectors.camkes>`
- Connection
  - connection <Connector> <conn_name>(from <comp>.<inf>, to <comp>.<inf>);
- Examples
  - RPC connection
    - `connection seL4RPC ab_r(from a.i, to b.i);`
    - `connection seL4RPCCall ab_r(from a.i, to b.i);`
  - Dataport connection
    - `connection seL4SharedData ab_d(from a.d,to b.d);`
  - Event connection
    - `connection seL4Async ab_e(from a.e, to b.e);`

Assembly

- ADL code example

```
assembly {
  composition {
    // component instances
    component Client client;
    component Server server;

    // connections between components
    connection seL4RPC simple(from client.a, to server.b);
  }
}
```

RPC Interfaces

- IDL code example

```
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 example
  - Client (`uses Simple a`) - generated

    ```
    char * a_echo_string(const char *s);
    int a_echo_int(int i);
    int a_echo_parameter(int pin, int *pout);
    ```

  - Server (`provides Simple b`) – must be manually written

    `char * b_echo_string(const char *s) { … }`, etc.

Hands-on: hello-camkes-1

- Edit files in: `apps/hello-camkes-1`
- Edit `hello-camkes-1.camkes`
  - TODO 1: component instances
  - TODO 2: connections
- Edit `interfaces/HelloSimple.idl4`
  - TODO 3: RPC functions
- Edit `components/Client/src/client.c`
  - TODO 4: invoke the RPC function
- Edit `components/Echo/src/echo.c`
  - TODO 5: implement the RPC function
- Build and run

CAmkES Events

Event interfaces

- ADL code example

```
consumes Event ev;
emits Event ev;
```

- `Event` is the event type
  - Can be anything you choose – not really used currently
  - Matching events (consumes and emits) should have the same type
- C code example

```
ev_emit()
ev_wait()
ev_reg_callback(void (*callback)(void*), void *arg)
```

- https://github.com/seL4/camkes-tool/blob/master/docs/index.md#an-example-of-events

Hands-on: hello-camkes-2 (part 1)

- Edit files in: `apps/hello-camkes-2`
- Edit `components/Client/Client.camkes`
  - TODO 1: the event interfaces
- Edit `components/Echo/Echo.camkes`
  - TODO 3: the event interfaces
- Edit `hello-camkes-2.camkes`
  - TODO 5: Event connections
- Edit `components/Client/src/client.c`
  - TODO 10: emit event
  - TODO 11: wait to get an event back
  - TODO 14: emit event
  - TODO 15: wait to get an event back
- Continued next slide …

Hands-on: hello-camkes-2 (part 1)

- Edit `components/Echo/src/echo.c`
  - TODO 17: fix function name
  - TODO 18: register the first callback handler
  - TODO 21: register the second callback
  - TODO 22: notify the client
  - TODO 24: register the original callback handler
  - TODO 25: notify the client
- Build and run

CAmkES Dataports

Dataport interfaces

- ADL code example
  - Buf: untyped, Frame sized shared buffer

```
dataport Buf d;
```

  - C typedef: typed shared buffer, sized according the type

```
include <my_typedefs.h>;
dataport a_typedef_t dt;
```

- C code example

```
char d[PAGE_SIZE]; // generated
a_typedef_t *dt; // generated
```

- Dataport pointers: to pass pointers between components

```
#include <camkes/dataport.h>
dataport_ptr_t dataport_wrap_ptr(void *ptr);
void *dataport_unwrap_ptr(dataport_ptr_t ptr);
```

- https://github.com/seL4/camkes-tool/blob/master/docs/index.md#an-example-of-dataports

Hands-on: hello-camkes-2 (part 2)

- Edit files in: `apps/hello-camkes-2`
- Edit `components/Client/Client.camkes`
  - TODO 2: the dataport interfaces
- Edit `components/Echo/Echo.camkes`
  - TODO 4: the dataport interfaces
- Edit `hello-camkes-2.camkes`
  - TODO 6: Dataport connections
- Edit `components/Client/src/client.c`
  - TODO 9: copy strings to an untyped dataport
  - TODO 12: read the reply data from a typed dataport
  - TODO 13: send the data over again
- Continued next slide …

Hands-on: hello-camkes-2 (part 2)

- Edit `components/Echo/src/echo.c`
  - TODO 19: read some data from the untyped dataport
  - TODO 20: copy modified data
  - TODO 23: read some data from the dataports
- Build and run

CAmkES Attributes

Attributes

- Three kinds:
  - Component Configuration
    - Used to configure properties of components (e.g. priority, stack size, etc.)
  - Connection Configuration
    - Used to configure properties of connections (e.g. endpoint badge)
  - Component
    - Defined as attribute in component definition
    - Available to component code at runtime
- ADL code example:

```
Component Client {
  …
  // data used by component
  attribute int num_widgets;
}
```

Assembly: configuration

- ADL code example

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

- Component configuration attribute: `priority`
  - Sets priority of all threads in `client` component
- Component attribute: `num_widgets`
  - C code example
    - variable with attribute name, contains value
    - `do_something(num_widgets);`

Hands-on: hello-camkes-2 (part 3)

- Edit files in: `apps/hello-camkes-2`
- Edit `hello-camkes-2.camkes`
  - TODO 7: set component priorities
  - TODO 8: restrict access to dataports
- Edit `components/Client/src/client.c`
  - TODO 16: test the read and write permissions on the dataport.
- Try to add a component attribute and access it in component code
  - No TODOs for this – you're on your own!

What’s Next?

Advanced CAmkES

- Hierarchical components
  - https://github.com/seL4/camkes-tool/blob/master/docs/index.md#hierarchical-components
- N-to-1 connections
  - https://github.com/seL4/camkes/tree/master/apps/multiclient
- User–defined connectors
  - https://github.com/seL4/camkes/tree/master/apps/mutex
- Hardware components and device drivers
  - hello-camkes-timer
  - https://github.com/seL4/camkes-tool/blob/master/docs/index.md#hardware-components