Learning gem5 – Part II

Modifying and Extending gem5

Jason Lowe-Power

http://learning.gem5.org/

https://faculty.engineering.ucdavis.edu/lowepower/

A simple SimObject

http://learning.gem5.org/book/part2/helloobject.html

gem5’s coding guidelines

Follow the style guide (http://www.gem5.org/Coding_Style)

- Install the style guide when scons asks
- Don’t ignore style errors

Use good development practices

- Historically mercurial queues
- Now: ***git branches***

Adding a new SimObject

Step 1: Create a Python class

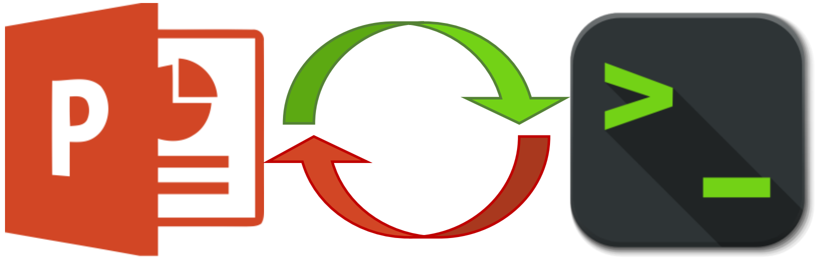
Step 2: Implement the C++

Step 3: Register the SimObject and C++ file

Step 4: (Re-)build gem5

Step 5: Create a config script

Switch!



Step 1: Create a Python class

HelloObject.py

```
from m5.params import *
from m5.SimObject import SimObject

class HelloObject(SimObject):
    type = ‘HelloObject’
    cxx_header = ‘learning_gem5/hello_object.hh’
```

m5.params: Things like `MemorySize`, `Int`, etc.

Import the objects we need

type: The C++ class name

cxx_header: The filename for the C++ header file

Step 2: Implement the C++

hello_object.hh

```
| #include "params/HelloObject.hh"
| #include "sim/sim_object.hh"
| class HelloObject : public SimObject
| {
|   public:
|     HelloObject(HelloObjectParams *p);
| };
```

params/*.hh generated automatically. Comes from Python SimObject definition

Constructor has one parameter, the generated params object.

Step 2: Implement the C++

hello_object.cc

```
HelloObject::HelloObject(HelloObjectParams *params)
    : SimObject(params)
{
    std::cout << "Hello World! From a SimObject!" << std::endl;
}
HelloObject*
HelloObjectParams::create()
{
    return new HelloObject(this);
}
```

HelloObjectParams: when you specify a **Param** in the Hello.py file, it will be a member of this object.

You must **define** this function (you’ll get a linker error otherwise). This is how Python config creates the C++ object.

Step 3: Register the SimObject and C++ file

SConscript

```
| Import(*)
| SimObject(‘Hello.py’)
| Source(‘hello_object.cc’)
```

Import: SConscript is just Python… but weird.

SimObject(): Says that this Python file contains a SimObject. Note: you can put pretty much any Python in here

Source(): Tell scons to compile this file (e.g., with g++).

Step 4: (Re-)build gem5

Step 5: Create a config script

```
| ...
| system.hello = HelloObject()
| ...
```

Instantiate the new object that you created in the config file (e.g., simple.py)

```
> build/X86/gem5.opt configs/learning_gem5/hello.py

...

Hello world! From a SimObject!

...
```

Simple SimObject code

gem5/src/learning_gem5/part2/hello_object.cc

gem5/src/learning_gem5/part2/hello_object.hh

gem5/src/learning_gem5/part2/HelloObject.py

gem5/configs/learning_gem5/part2/hello_run.py

Debug support in gem5

http://learning.gem5.org/book/part2/debugging.html

Adding debug flags

SConscript

Switch!

```
DebugFlag('Hello')
```

hello_object.cc

```
DPRINTF(Hello, "Created the hello object");
```

Declare the flag: add the debug flag to the SConscript file in the current directory

DPRINTF: macr statements in g

Hello: the debug flag declared in the SConscript. Found in "debug/hello.hh"

Debug string: Any C format string

Debugging gem5

```
> build/X86/gem5.opt --debug-flags=Hello configs/tutorial/hello.py
...
          0: system.hello: Hello world! From a debug statement
```

debug-flags: Comma separated list of flags to enable. Other options include --debug-start=<tick>, --debug-ignore=<simobj name>, etc. See gem5.opt --help

Event-driven programming

http://learning.gem5.org/book/part2/events.html

Simple event callback

```
| class HelloObject : public SimObject
| {
|  private:
|    ...
|    void processEvent();
|    EventFunctionWrapper event;
|
|  public:
|    HelloObject(HelloObjectParams *p);
|    void startup();
| };
```

EventFunctionWrapper: Convenience class for simple events.

processEvent: Callback function to run when event fires.

startup: Called after all SimObjects instantiated. Schedule local events here.

Simple event callback

```
| void
| HelloObject::processEvent()
| {
|     timesLeft--;
|     DPRINTF(Hello, "Hello world!"
|                   " Processing the event! %d left\n", timesLeft);
|     if (timesLeft <= 0) {
|        DPRINTF(Hello, "Done firing!\n");
|     } else {
|        schedule(event, curTick() + latency);
|     }
| }
```

schedule: Put an event instance on the event queue. An absolute tick used for when the event is processed.

curTick: Returns the current simulator time. Useful for relative time computations.

Event SimObject code

http://learning.gem5.org/book/_downloads/hello_object1.hh

http://learning.gem5.org/book/_downloads/hello_object2.cc

SimObject parameters

http://learning.gem5.org/book/part2/parameters.html

Adding parameters

```
class HelloObject(SimObject):
   type = 'HelloObject'
   cxx_header = "learning_gem5/hello_object.hh"

   time_to_wait = Param.Latency("Time before firing the event")
   number_of_fires = Param.Int(1, "Number of times to fire the event before "
                                  "goodbye")
```

Param.<TYPE>: Specifies a parameter of type <TYPE> for the SimObject

Param.<TYPE>(): First parameter: default value. Second parameter: “help”

Going further: **More parameters**

http://learning.gem5.org/book/part2/parameters.html

Included types (e.g., MemorySize, MemoryBandwidth, Latency)

Using a SimObject as a parameter

SimObject-SimObject interaction

src/learning_gem5/part2/hello_object.cc & hello_object.hh
src/learning_gem5/part2/goodbye_object.cc & goodbye_object.hh
src/learning_gem5/part2/HelloObject.py & GoodbyeObject.py

Questions?

We covered

- How to build a SimObject
- How to schedule events
- Debug statements in gem5
- Adding parameters to SimObjects

MemObjects

http://learning.gem5.org/book/part2/memoryobject.html

MemObject

Object that is part of gem5’s memory system

both classic caches and Ruby are MemObjects

Allowed to have MasterPorts and SlavePorts

Packets

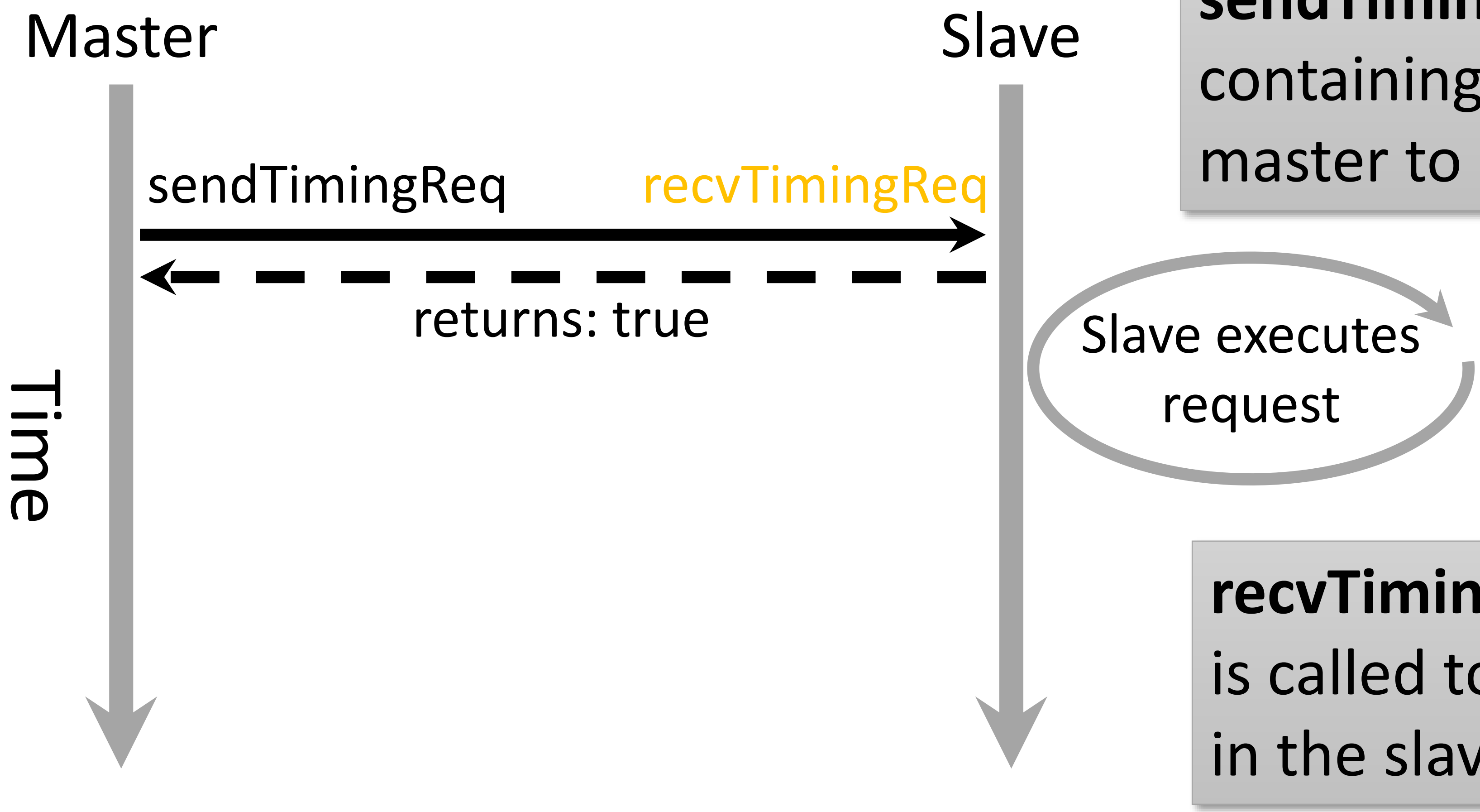
Unit of transfer between MemObjects

Packets pass between Master and Slave ports

Packets have

- Request
- Command
- Data
- Much more…

Master and slave ports



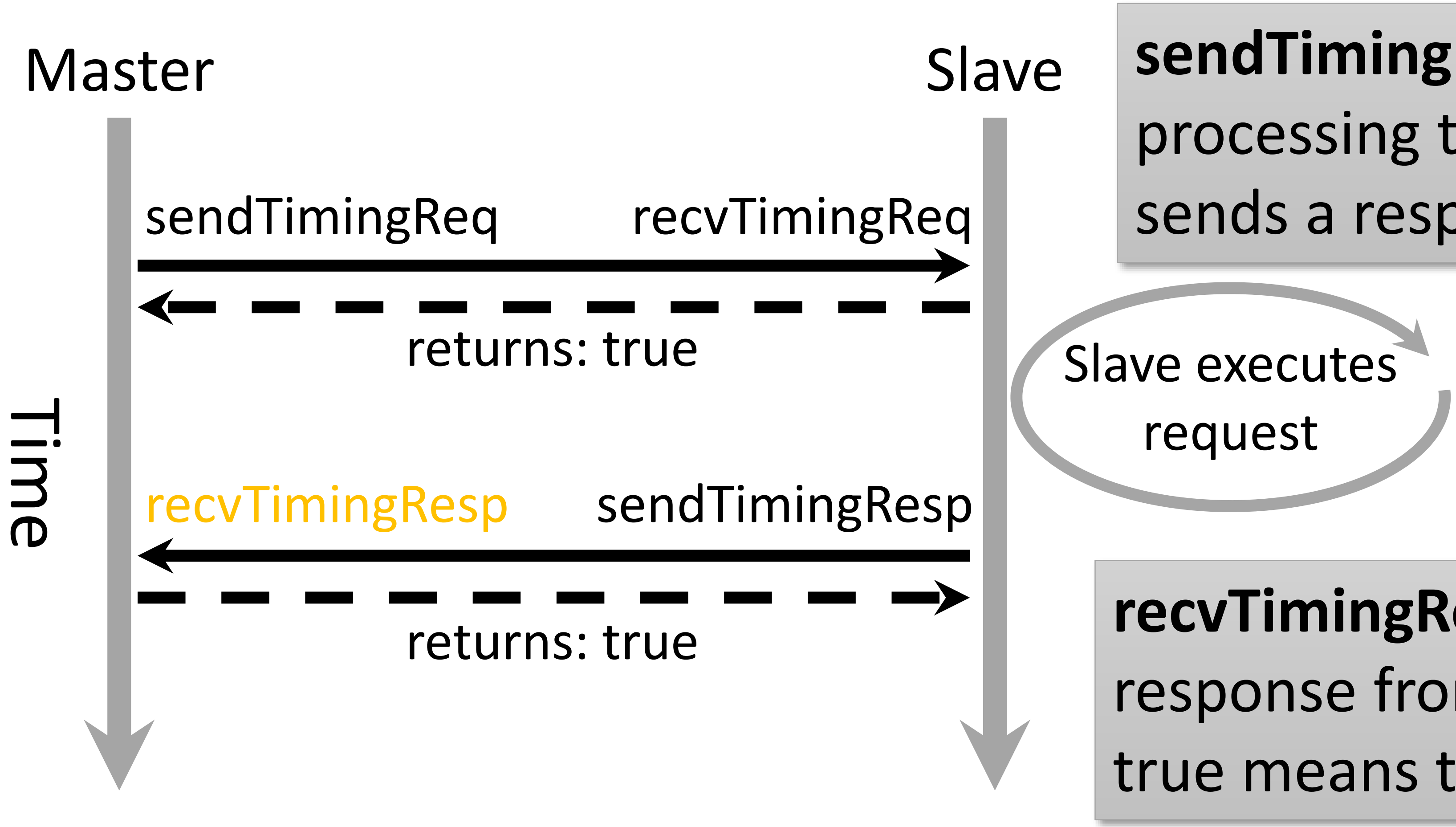
sendTimingReq: send a Packet containing a request from a master to a slave

return true: The slave can handle the request.

recvTimingReq: function that is called to handle the request in the slave port.

© Jason Lowe-Power <jason@lowepower.com>

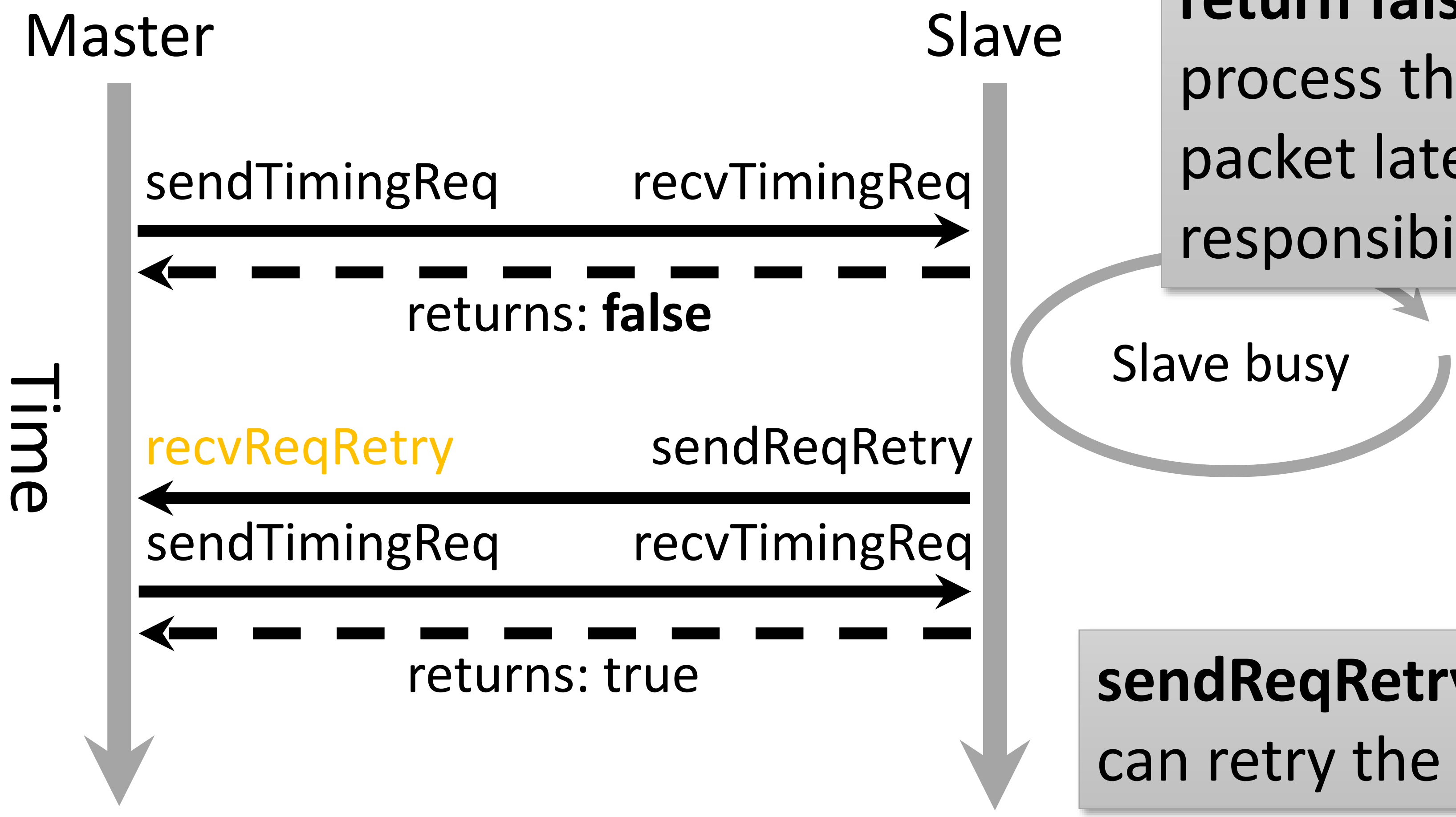
Master and slave ports



sendTimingResp: The slave finishes processing the request, and now sends a response (same packet).

recvTimingResp: Handles the response from the slave. Returning true means the packet is handled.

Master and slave ports

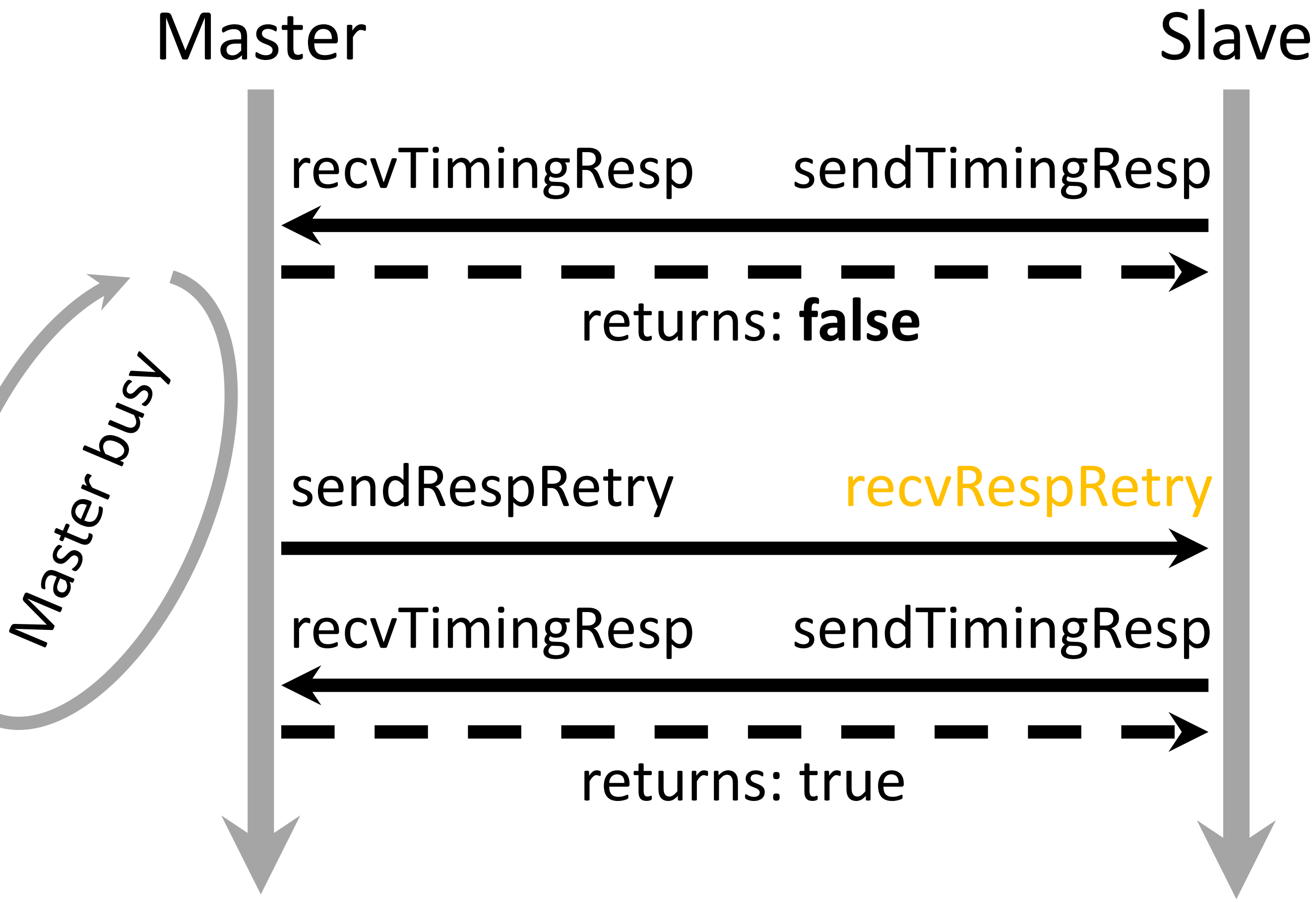


return false: Slave cannot currently process the Packet. Resend the packet later. The **Master's** responsibility to track Packet.

recvReqRetry: Can now retry the request by calling sendTimingReq.

sendReqRetry: Tell the master it can retry the stalled Packet.

Master and slave ports



return false: Master cannot currently process the Packet. Resend the packet later. The **Slave's** responsibility to track Packet.

sendRespRetry: Slave can now retry the response.

Master and slave port interface

Master

`recv Timing Resp`

`recv Req Retry`

`recv Range Change`

Slave

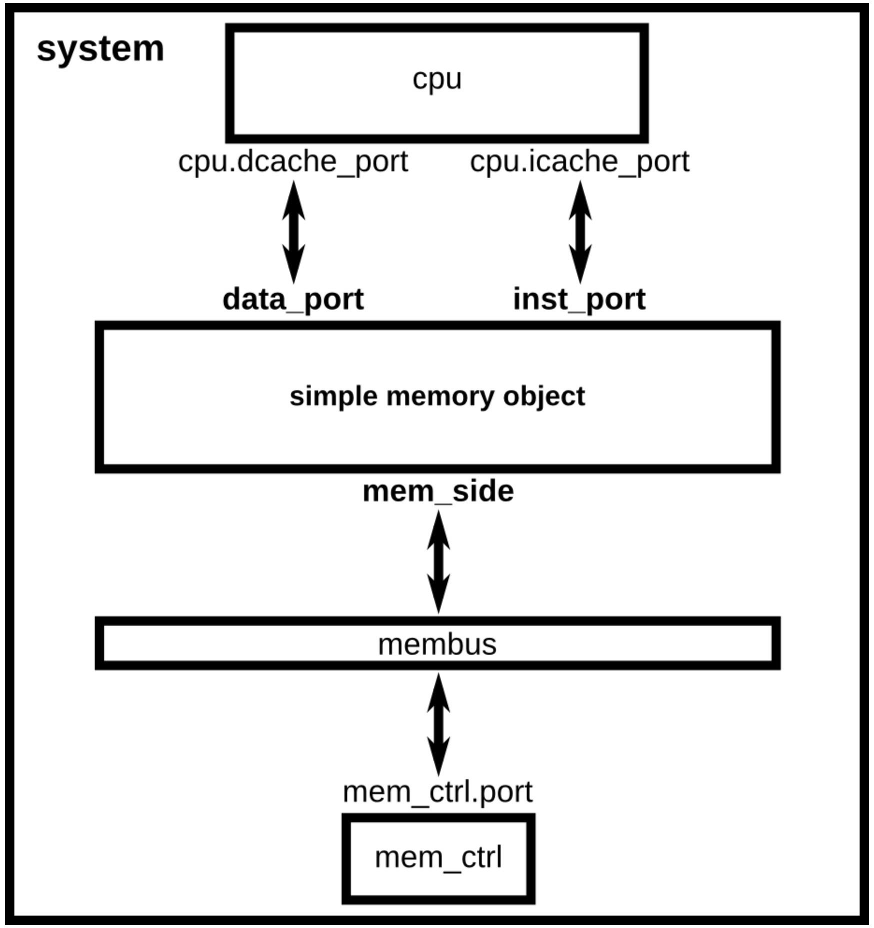
`recv Timing Req`

`recv Resp Retry`

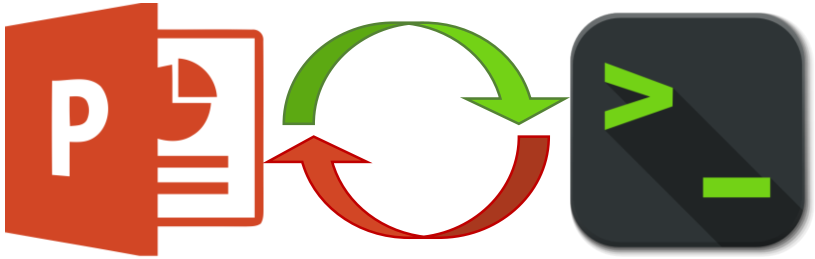
`recv Functional`

`get Addr Ranges`

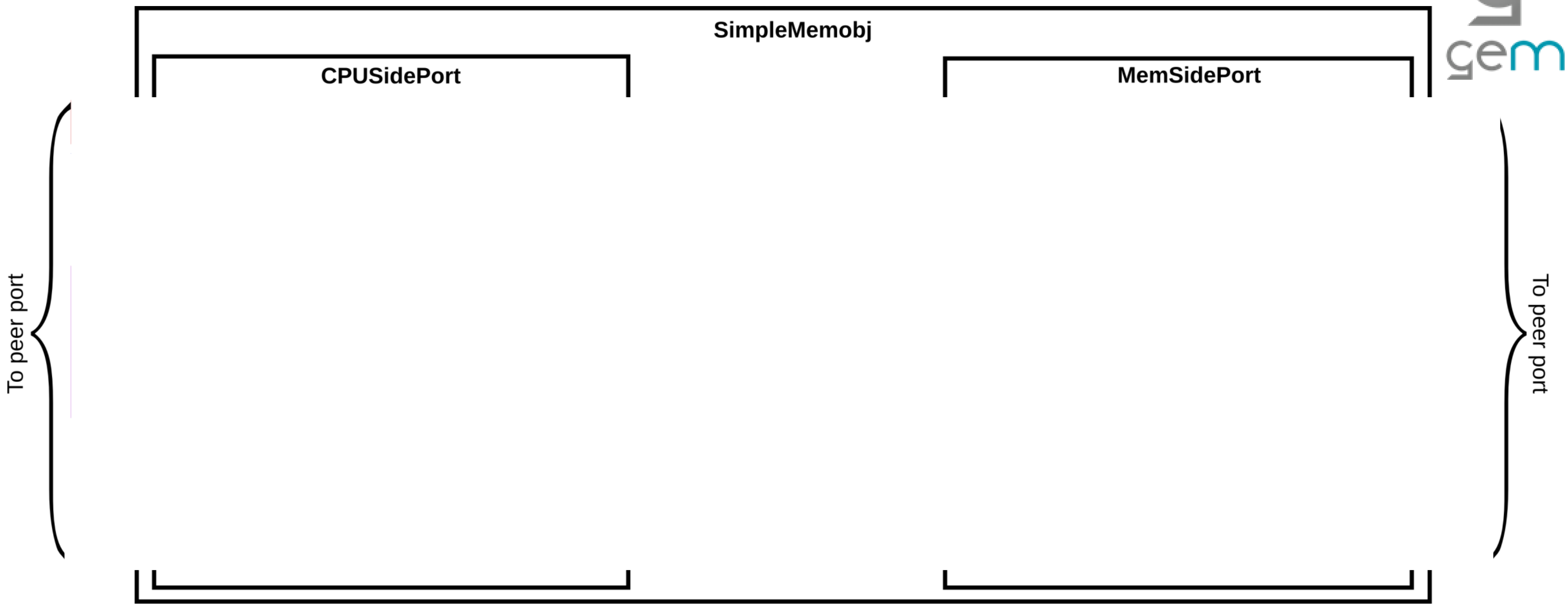
Simple MemObject



Switch!



Overview of SimpleMemobj



SimpleCache

http://learning.gem5.org/book/part2/simplecache.html

Cache: A first “real” object

How to model…

- Data storage
- Tags
- Associativity
- Data access latency
- Blocking?

std::map

Make an event

Could implement MSHRS…

Design

Handle request -> `accessTiming` with a delay

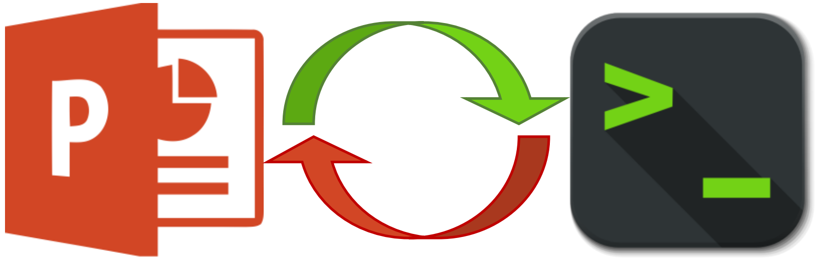
`AccessTiming`

- ->`accessFuntional` to check for hit/miss
- ->if hit, reply
- ->if miss, upgrade request and send read

Handle response

- -> `insert` new data (evict if needed)
- -> `accessFunctional` to read/write
- -> reply

Switch!



More on events

```
schedule(new EventFunctionWrapper(
        [this, pkt]{ accessTiming(pkt); },
        name() + ".accessEvent",
        true),
    clockEdge(laten
```

Anonymous function to execute

Local variables to "capture"

Delete this object after executing event

Packet construction

Many different packet constructors

See `src/mem/packet.hh` for details

```
Packet(Request, command)
```

```
Packet(Request, command, block size)
```

Make a packet that is block aligned (overrides request address)

```
createRead/createWrite(Request)
```

Should probably use these convenience functions

Packets data allocation

Dynamic data: Will be deleted when the packet is deleted

`packet->allocate()`: Allocates dynamic data

Static data: Give packet a pointer to the data. It will not delete it.

`SenderState`: Can be used to store “local” information

Packets: To delete or not to delete

Do **not** delete to send a response

Call packet->makeResponse()

Do delete if you are the final sink for the packet

E.g., a memory write

Do delete if you initiated the request and then received the response

Complete code available

Statistics

Better flow control

Code to make it work with O3CPU

Much more: http://learning.gem5.org/book/part2/simplecache.html

Questions?

We covered

- How to make a MemObject
- gem5 packets
- The master – slave API in gem5
- “Real” cache example