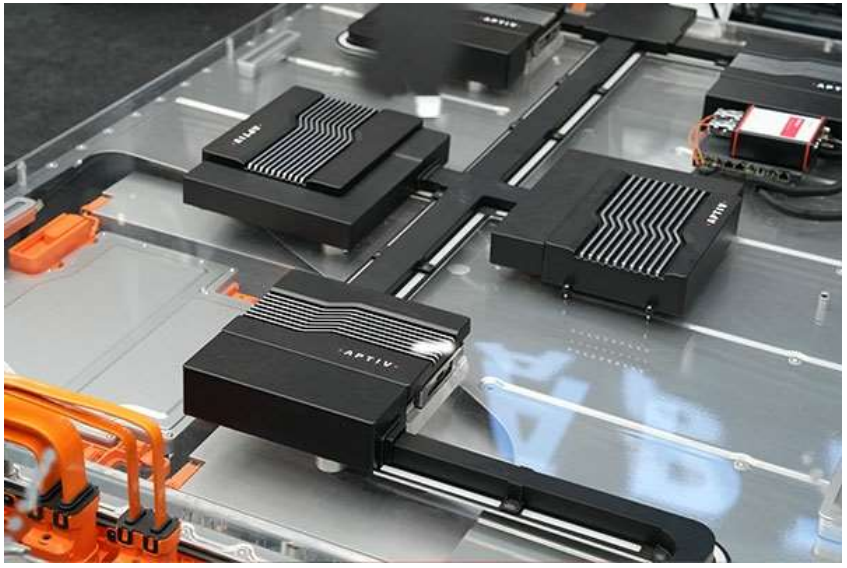
November 12th, 2024

Christof Petig
Staff SW engineer

COMPONENT MODEL IN SOFTWARE DEFINED VEHICLES

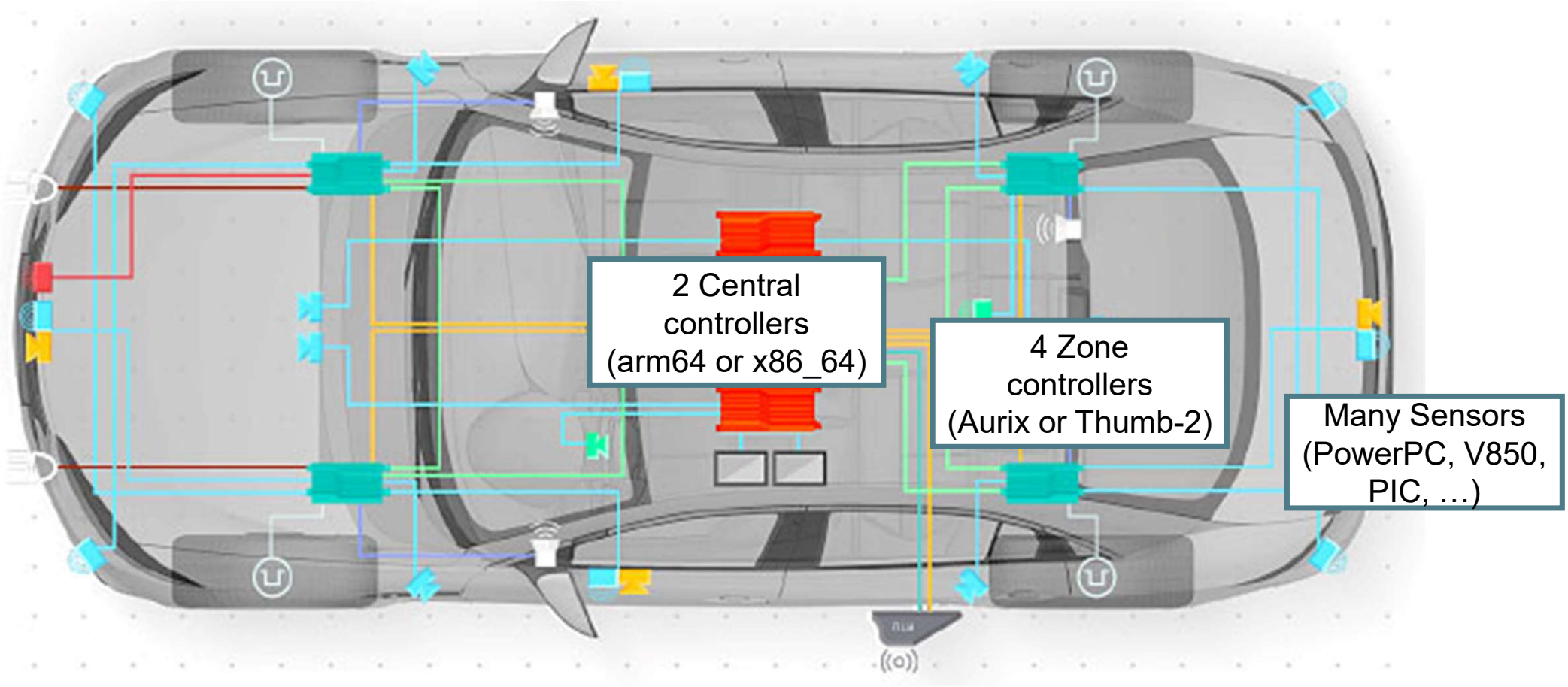


APTIV

Contents

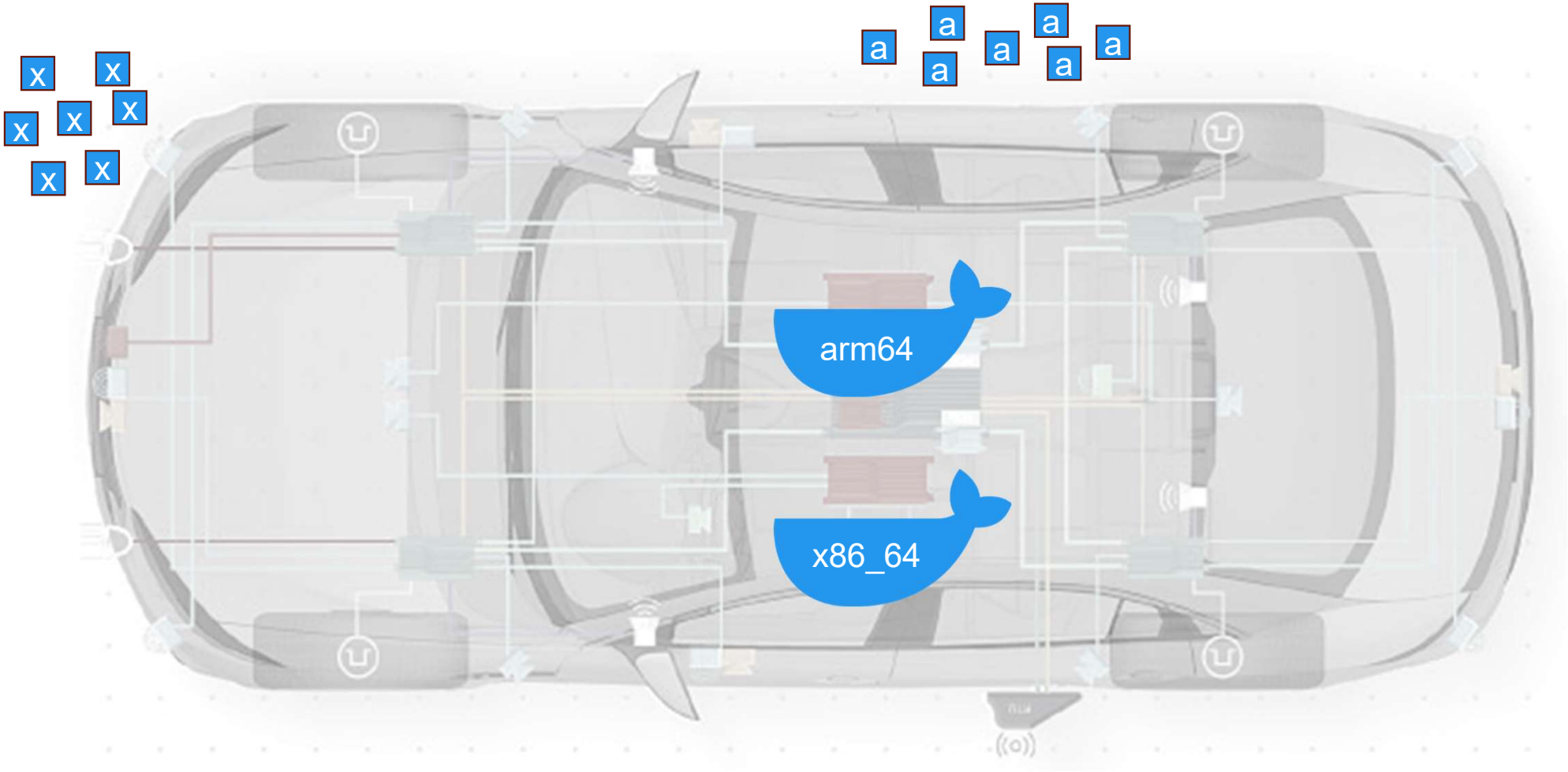
Wasm helps SDV

Zonal car architecture



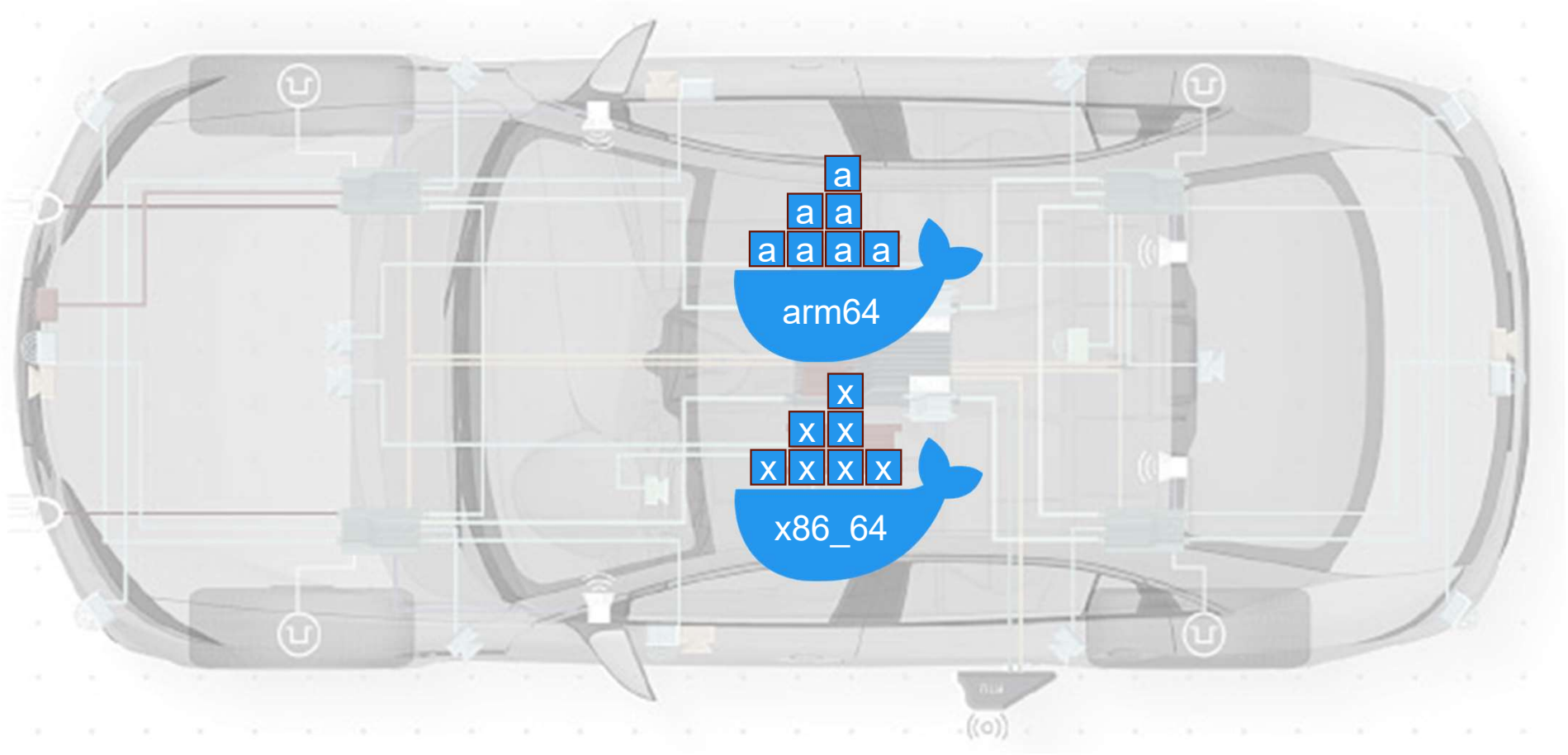
APTIV

Classic containers are arch. and OS specific

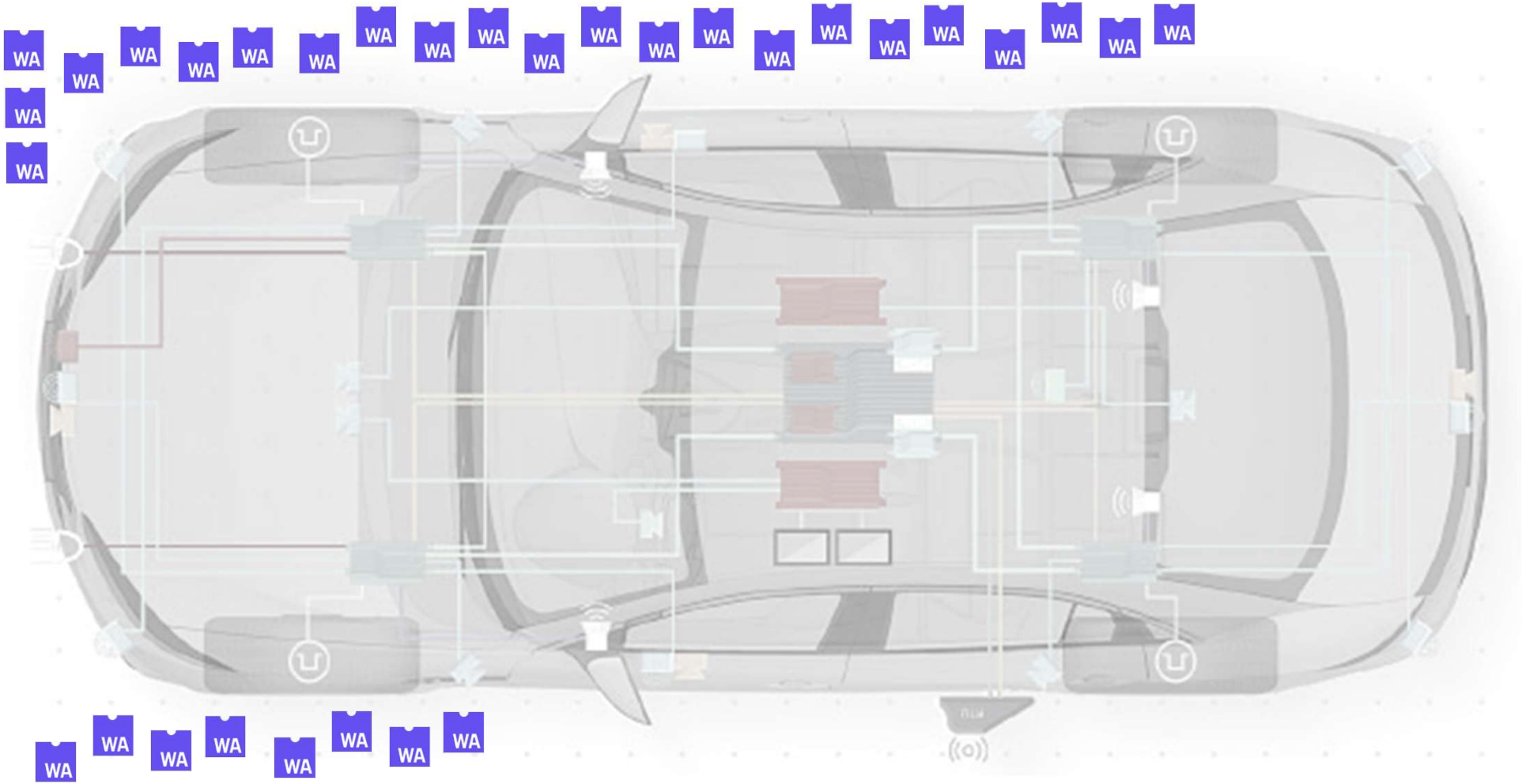


APTIV

Docker containers are arch. and OS specific



Wasm components fit anywhere

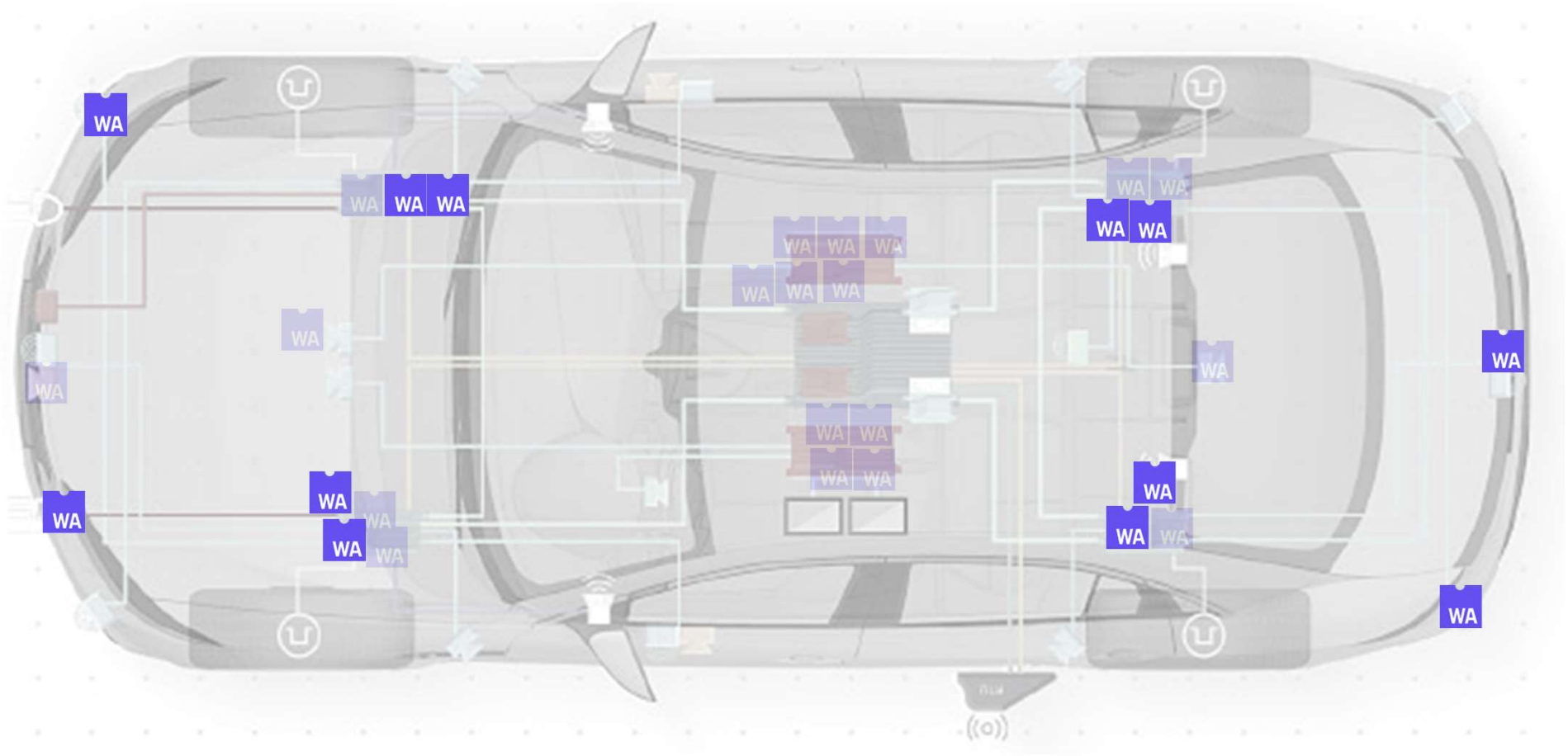


APTIV

Wasm components fit anywhere

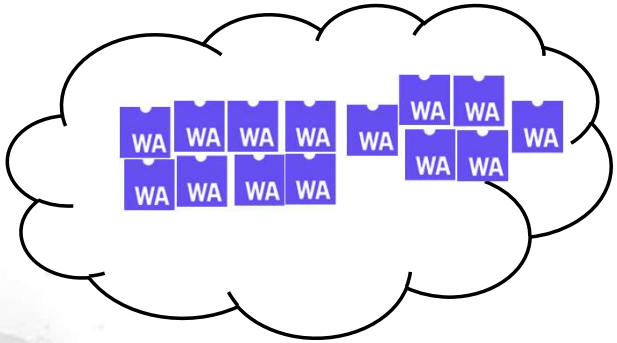
APTIV

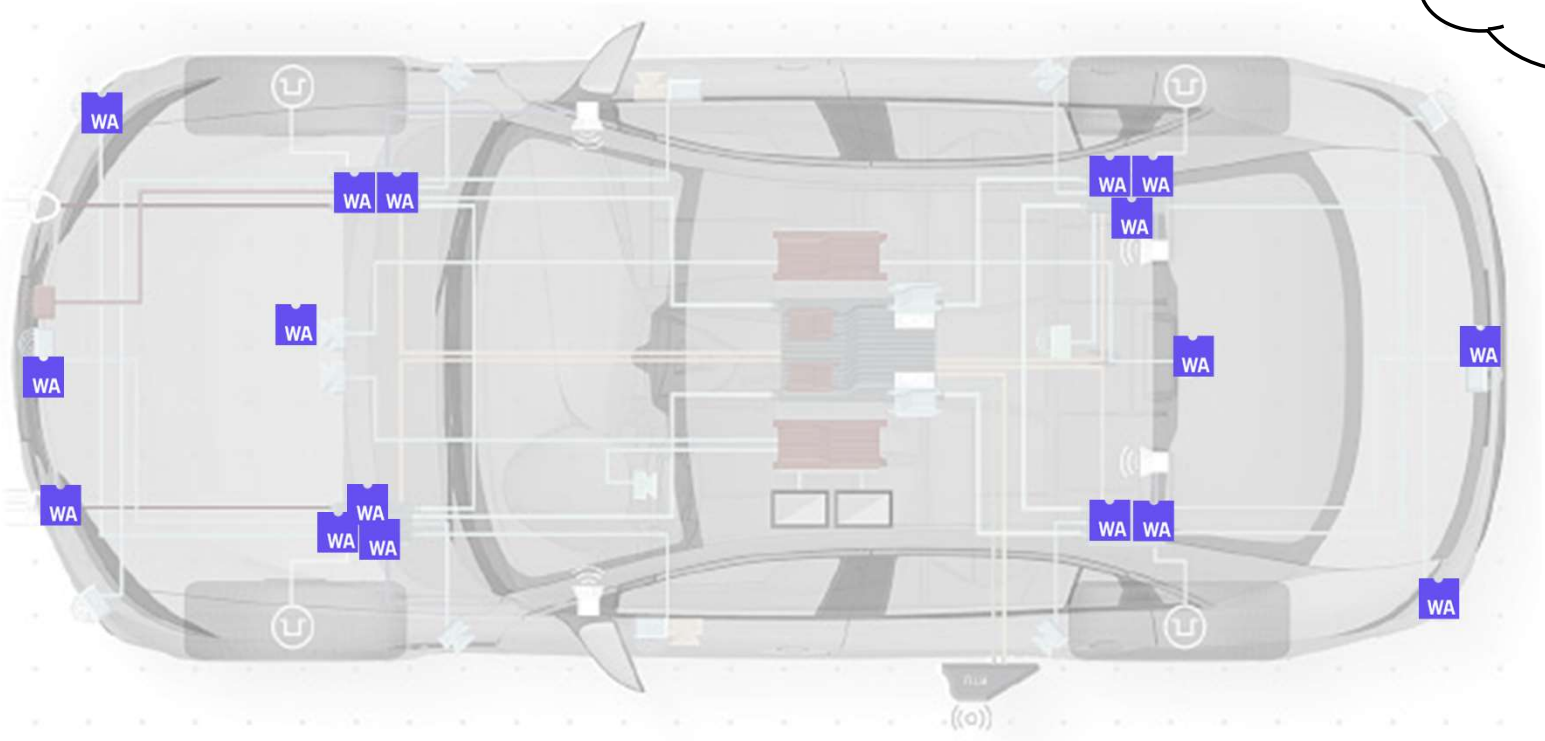
You can move them on powerdown



APTIV

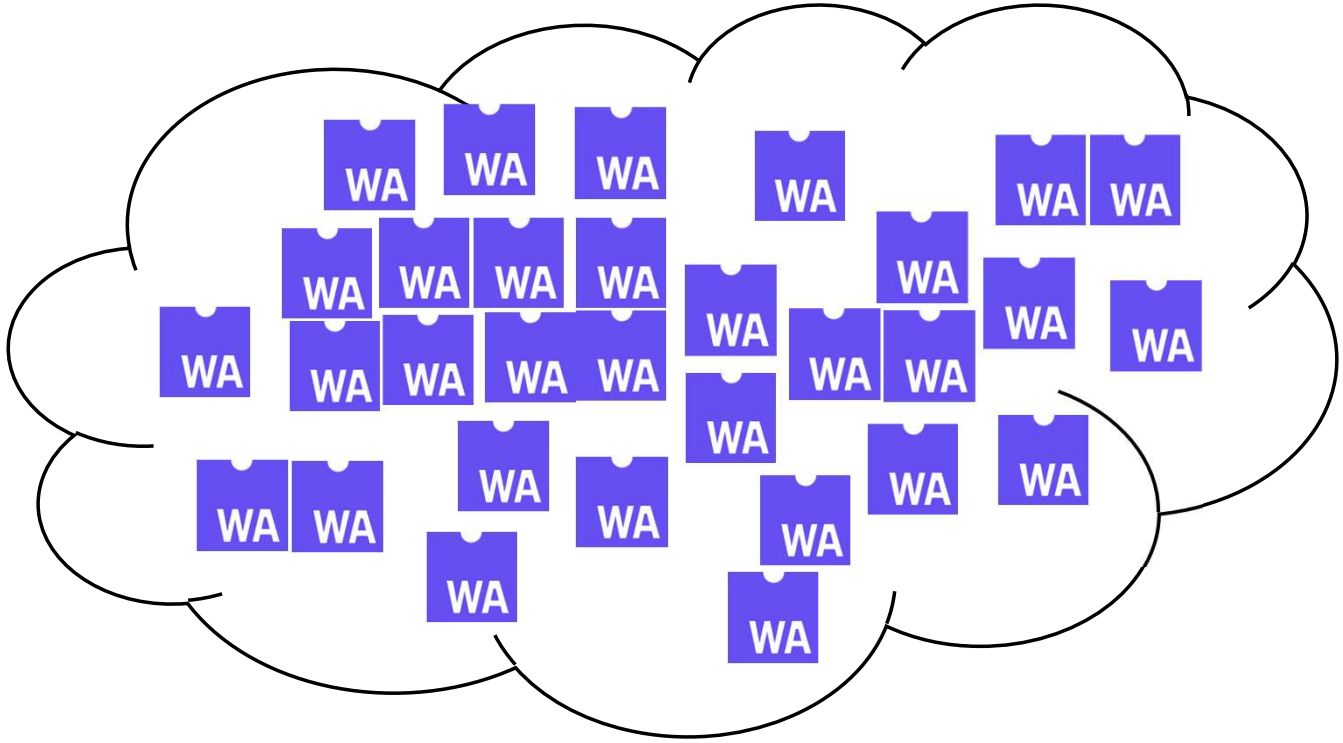
You can run components on the edge





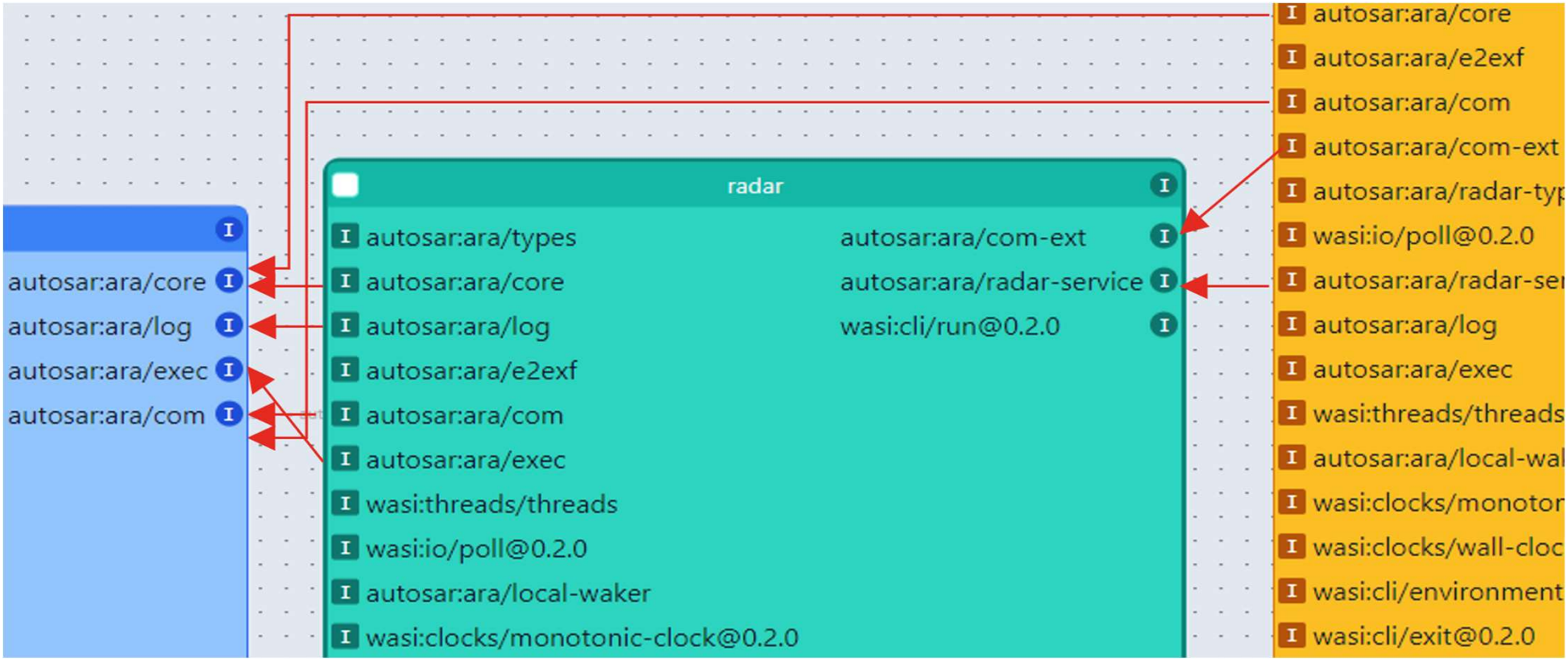
APTIV

… or fully simulate the vehicle

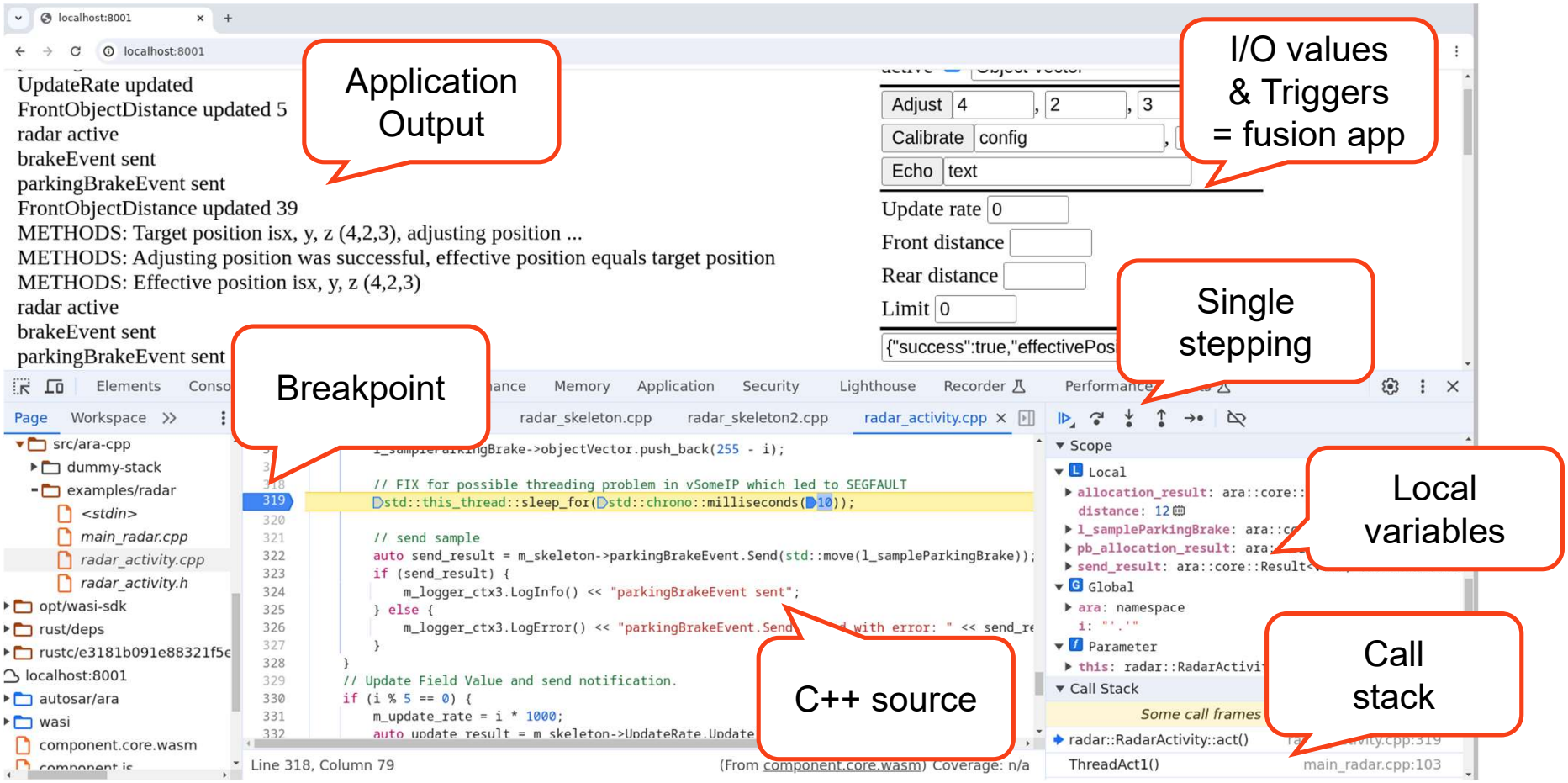


APTIV

Component Model offers graphical composition



… and debugging in the browser



APTIV

Enabled by the Component Model

- **Language neutral definition**
  Wasm Interface Types

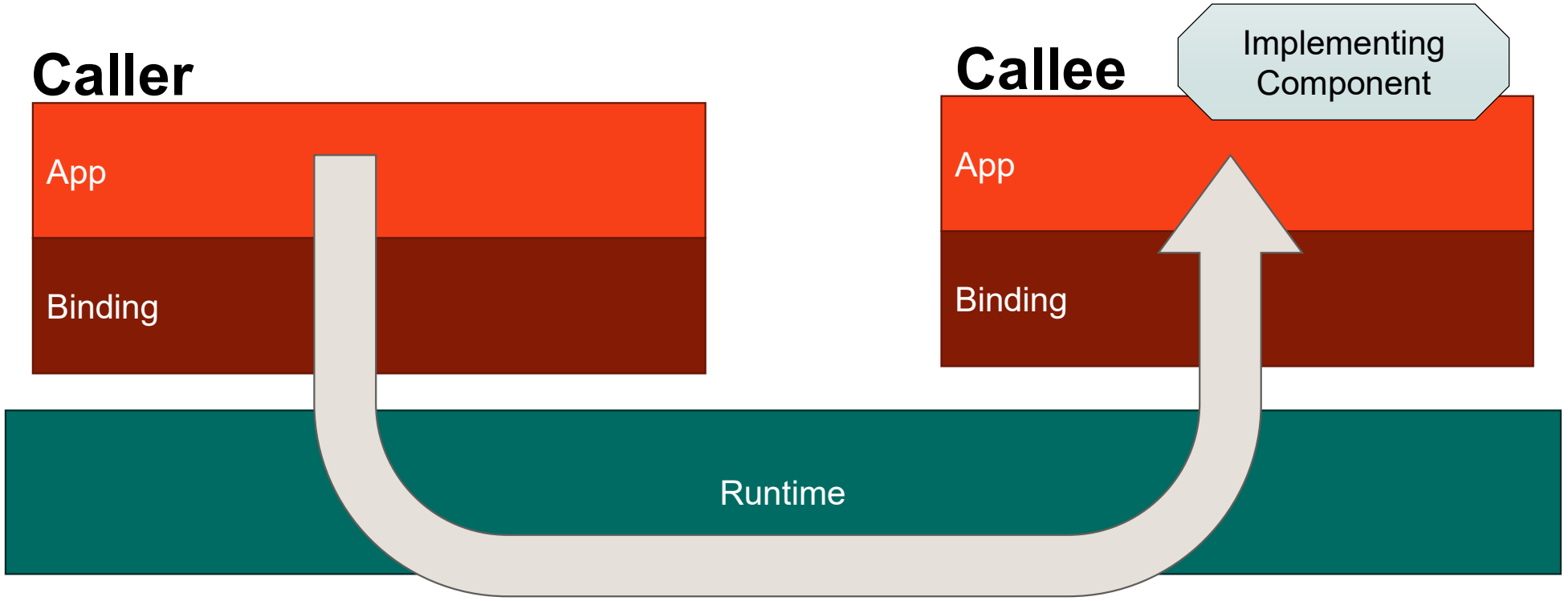
- **High level data types**
  Option, Result, String, Vector, Future, Stream

- **Standardized calling convention (canonical ABI) and system calls (WASI)**

Perfect match with AUTOSAR Adaptive Platform

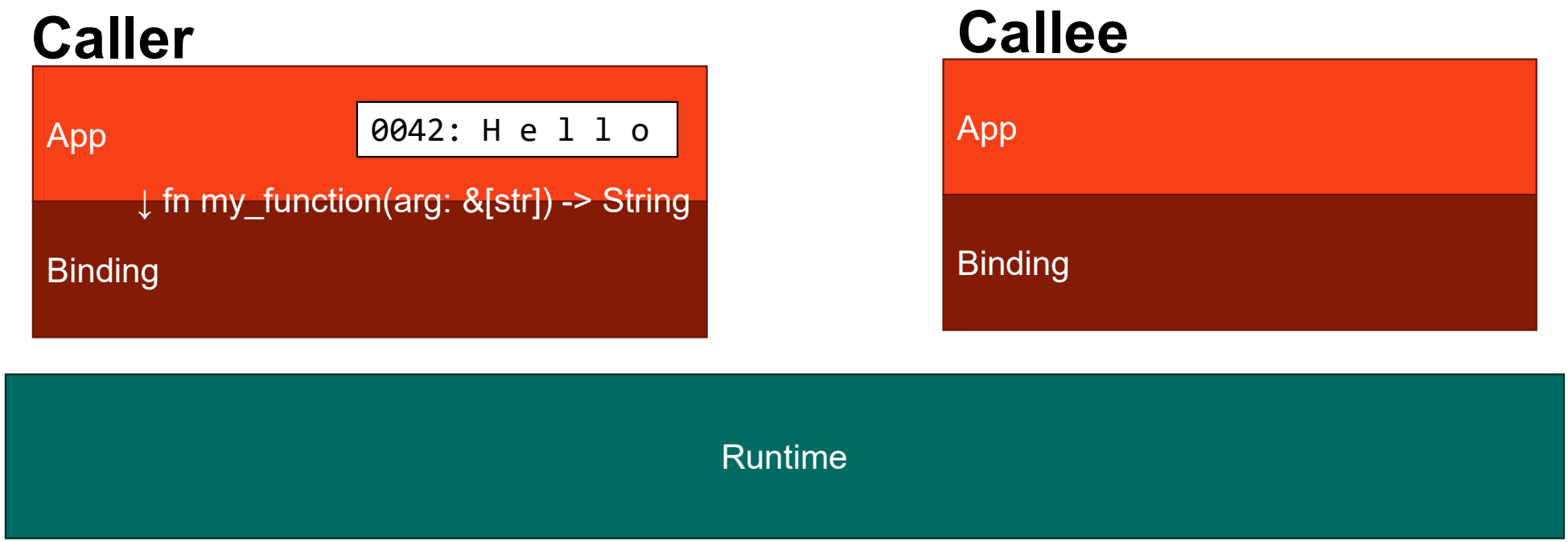
Lifting and Lowering: Canonical ABI

my-function: func(arg: **string**) -> **string**



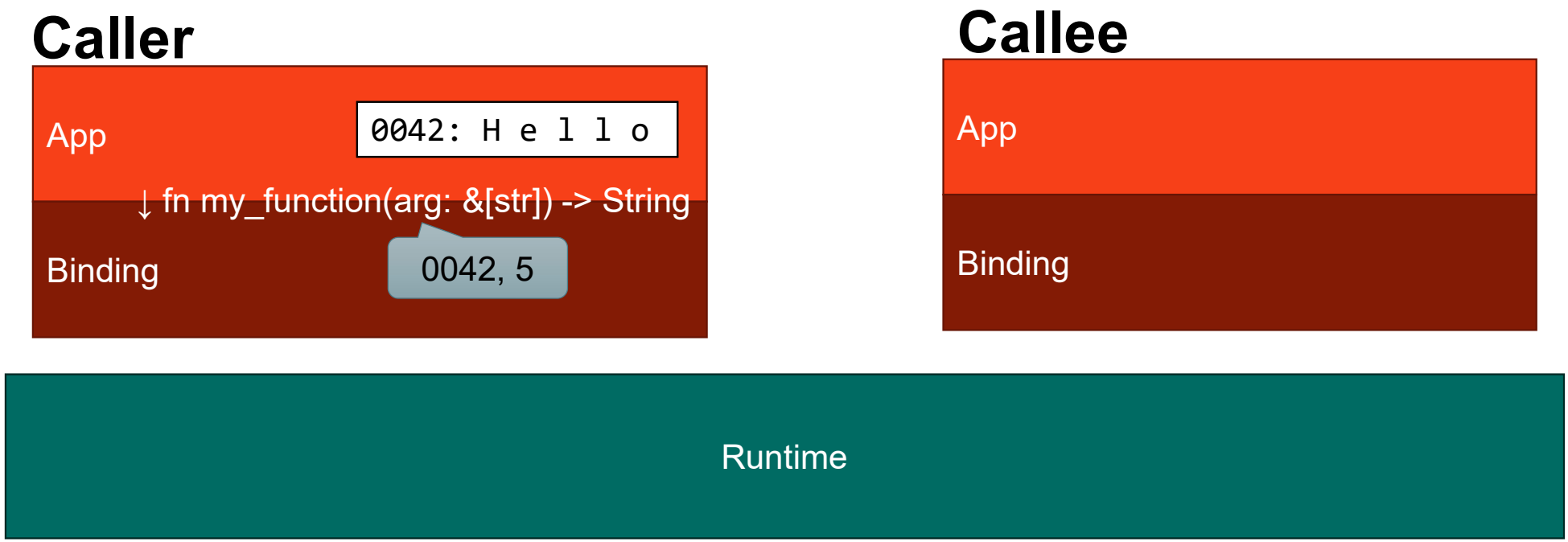
APTIV

App starts with a string in memory



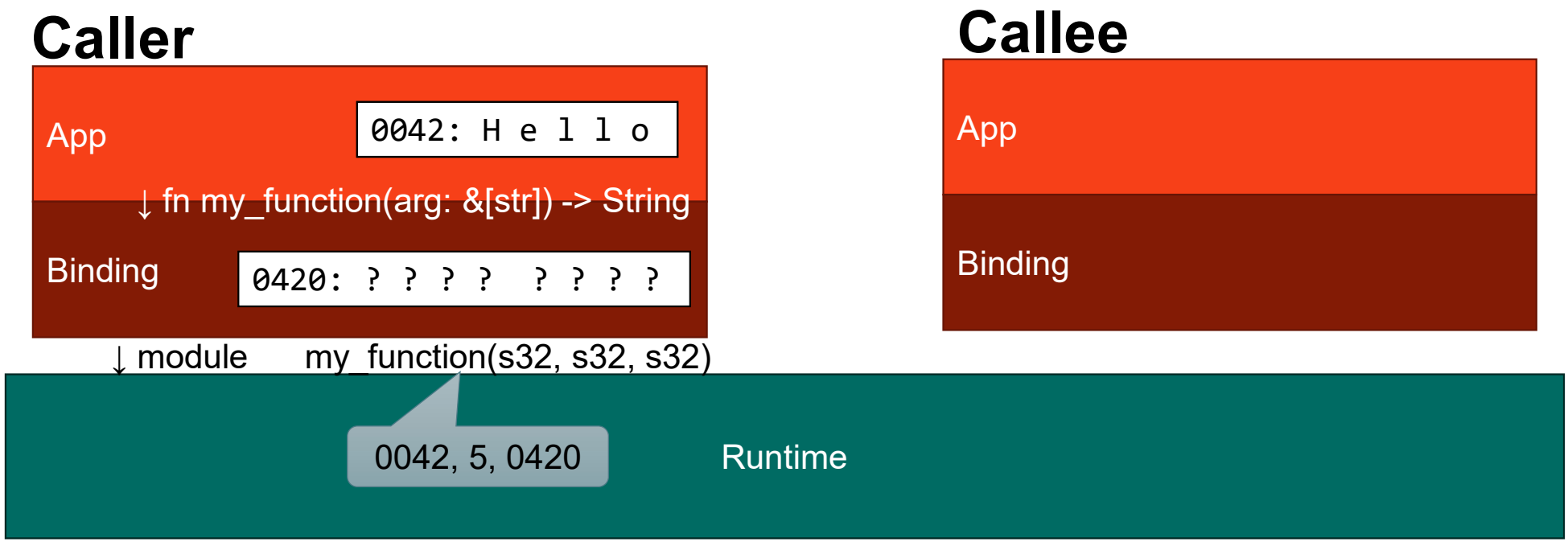
APTIV

App calls binding



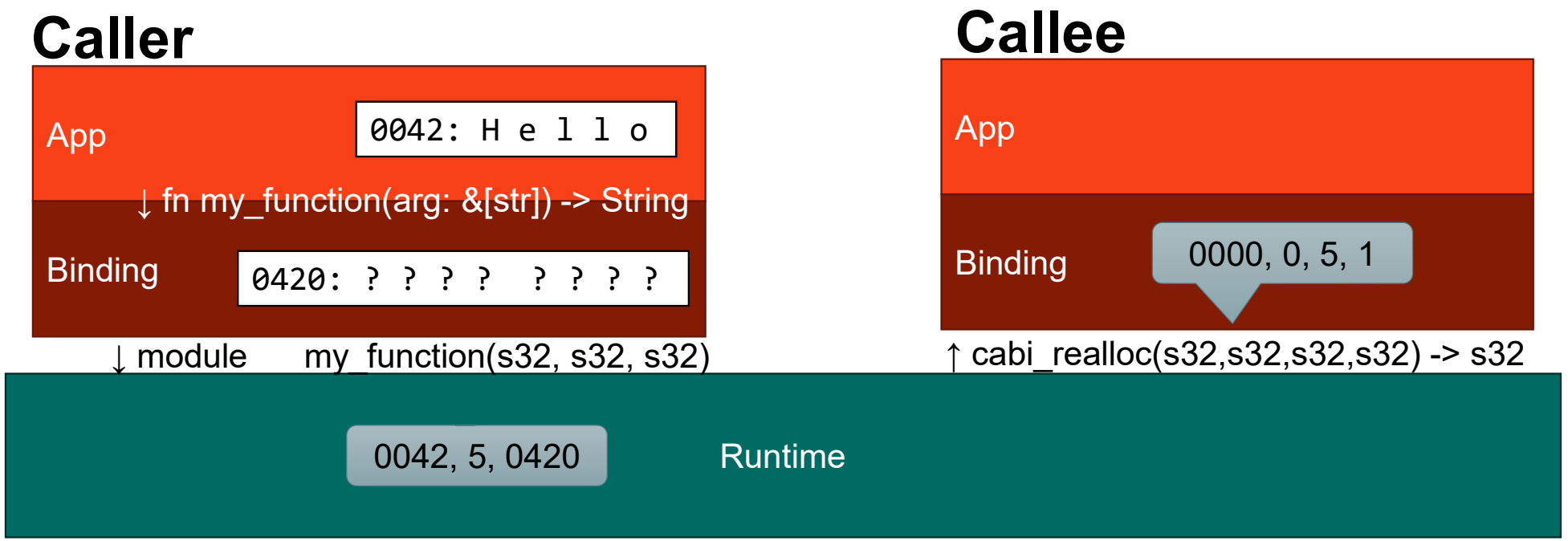
APTIV

Binding elects return area and calls runtime



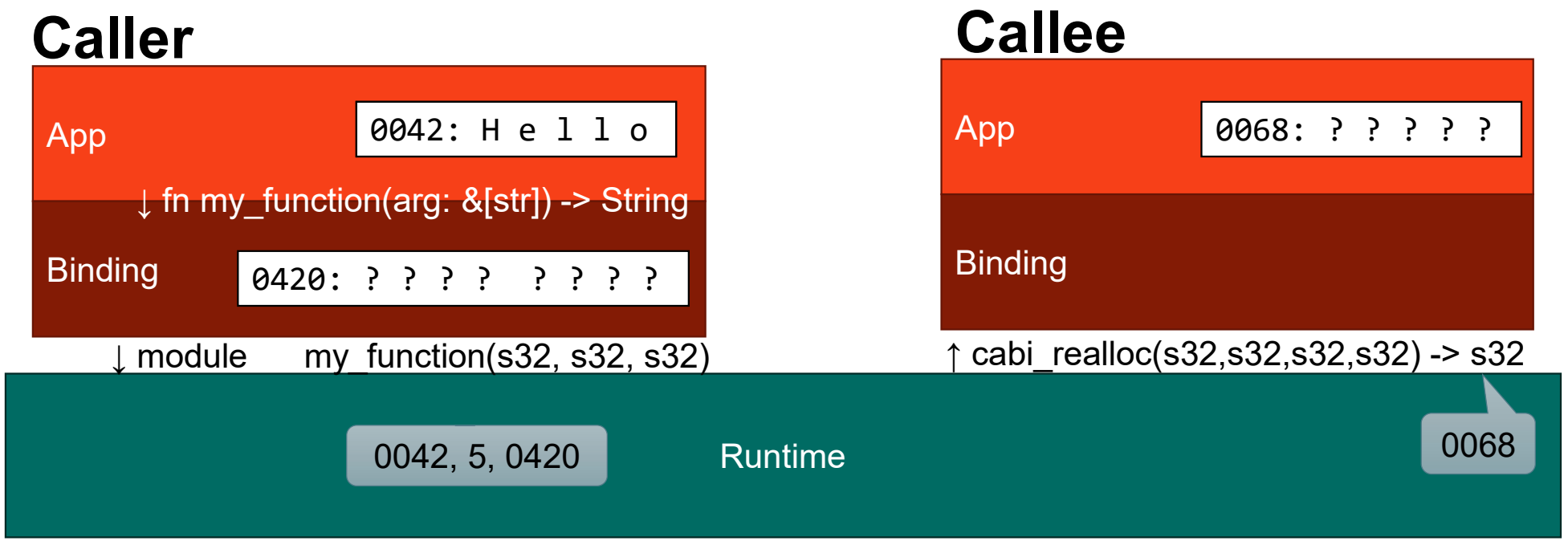
APTIV

Runtime allocates memory in callee



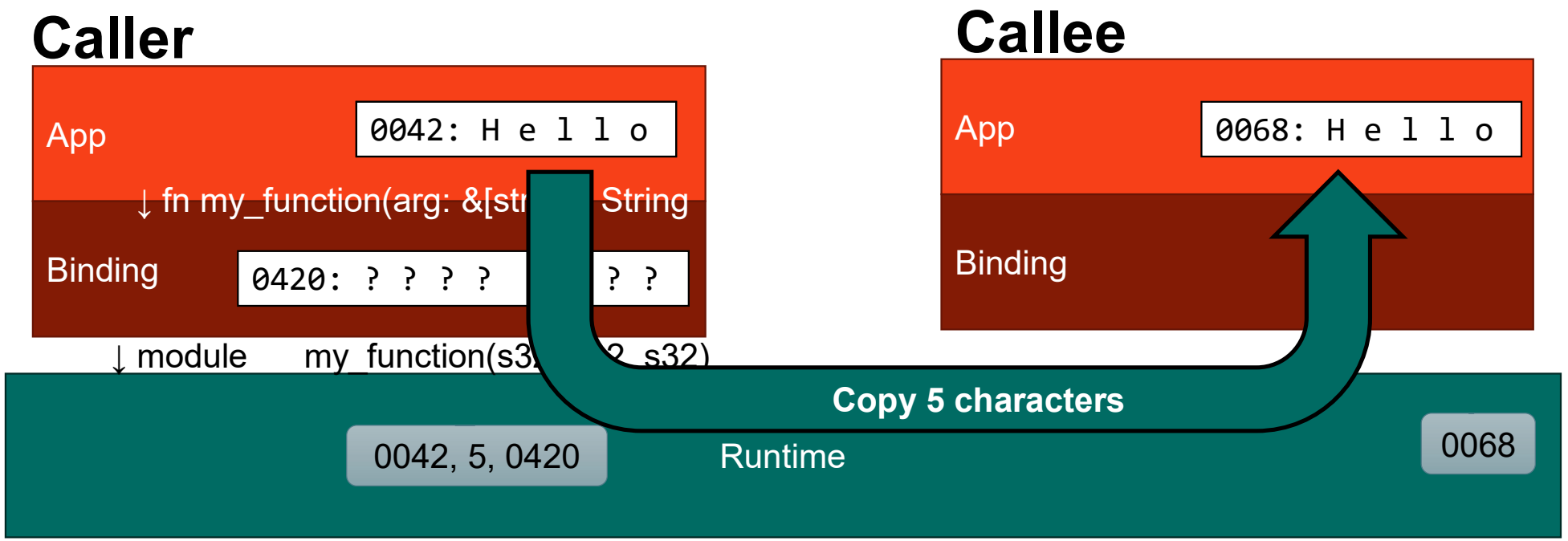
APTIV

Callee provides memory address



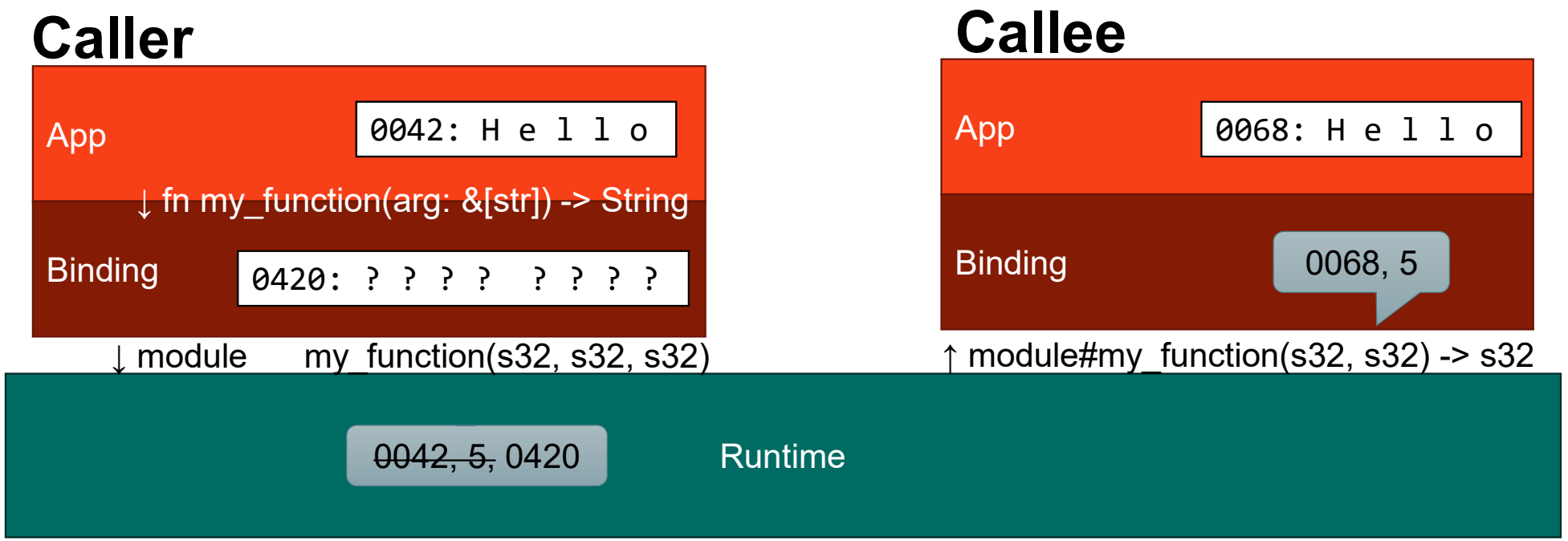
APTIV

Runtime copies five characters



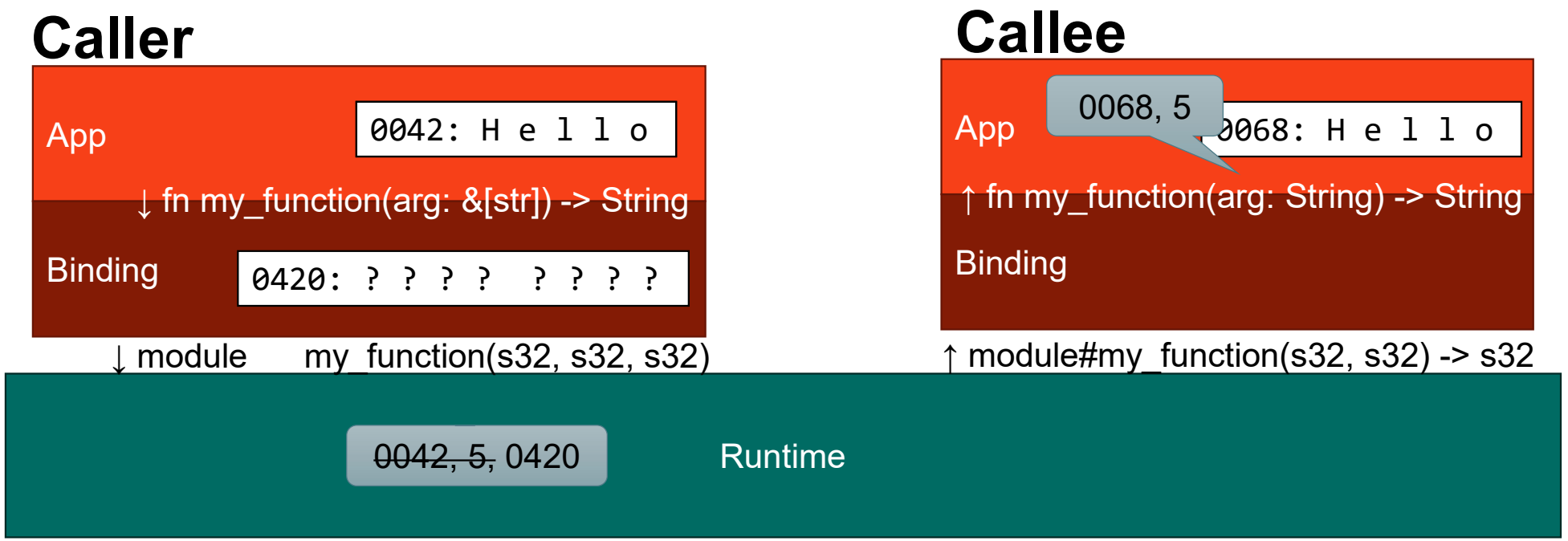
APTIV

Runtime calls binding



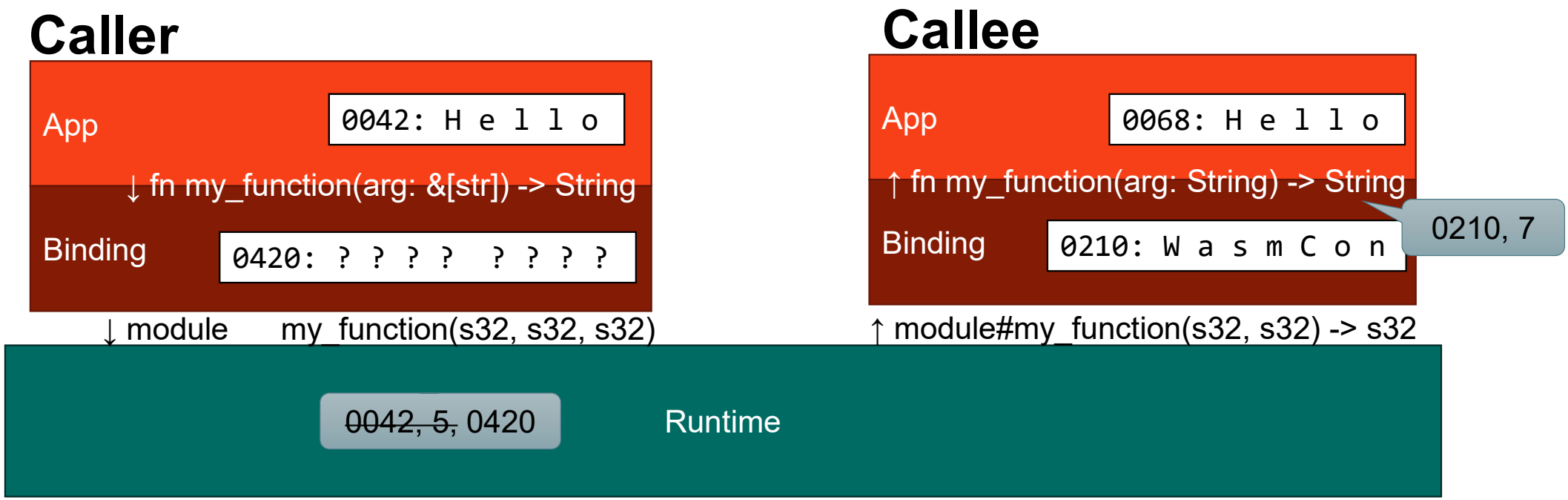
APTIV

Binding calls implementation



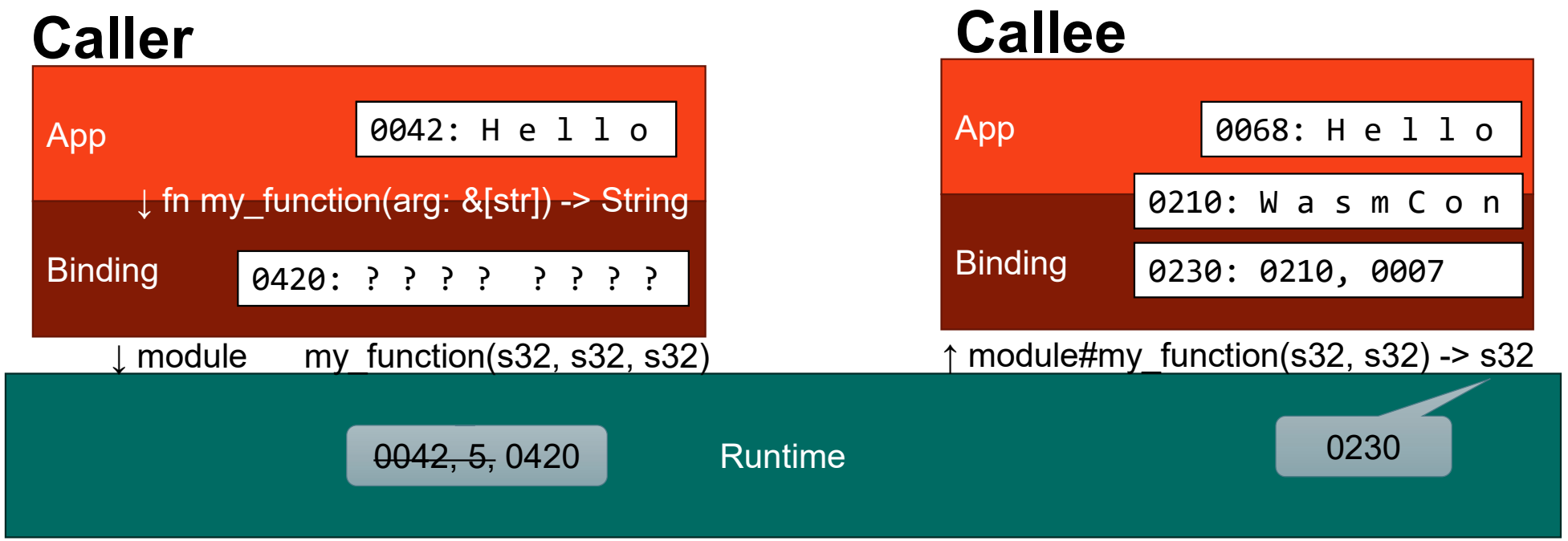
APTIV

Implementation returns String



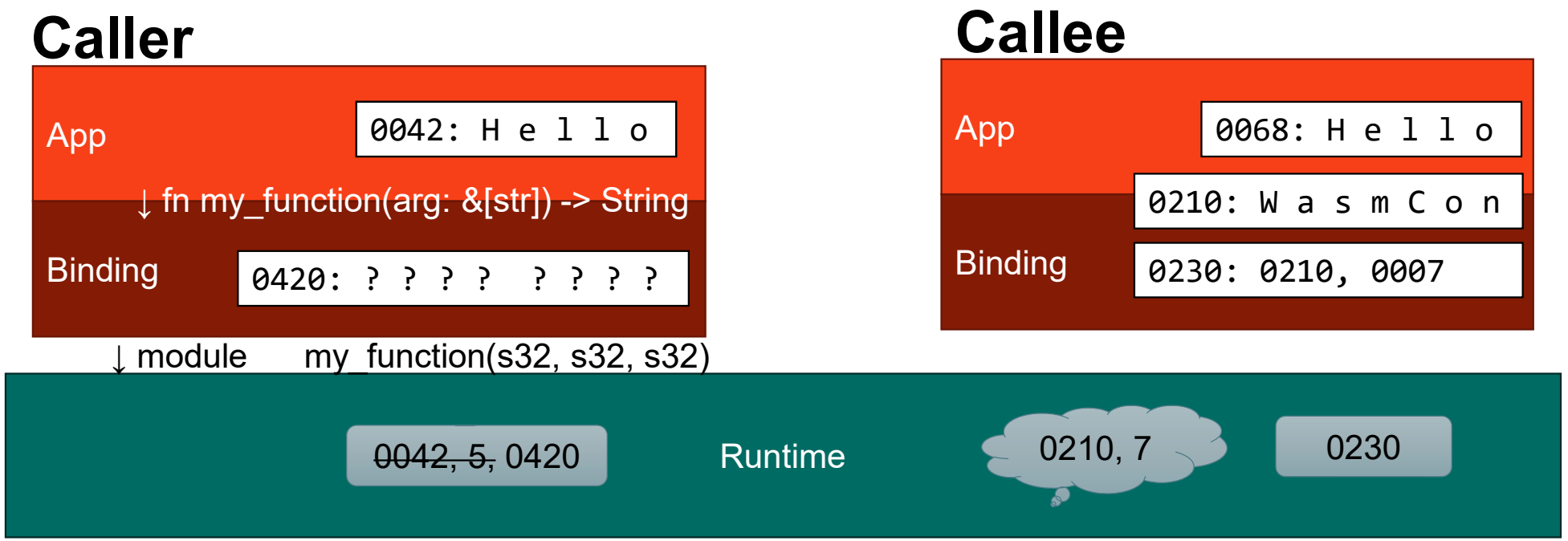
APTIV

Binding returns static return area



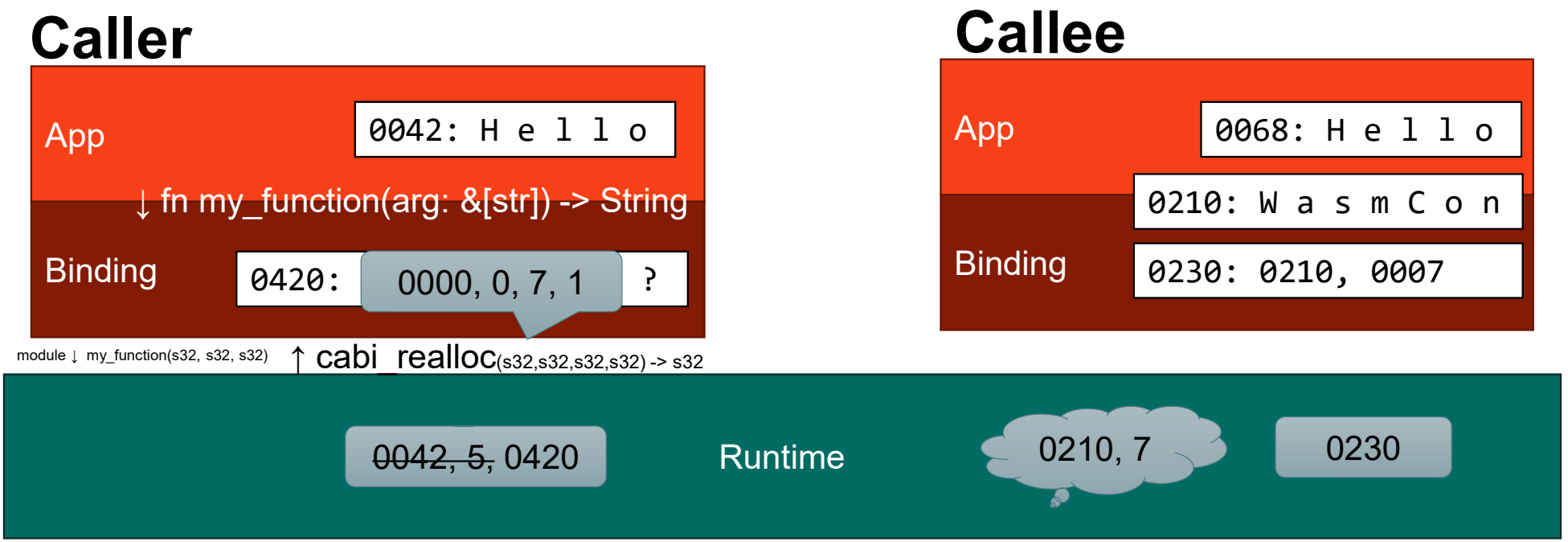
APTIV

Runtime reads address and lenght



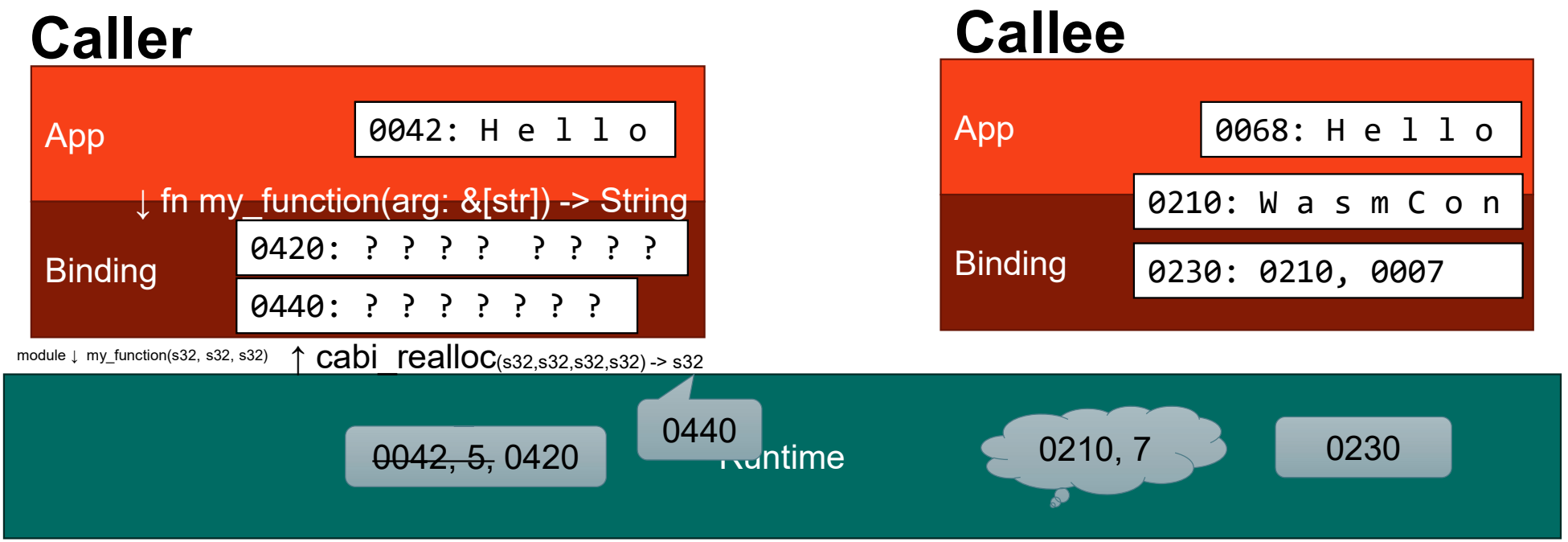
APTIV

Runtime allocates memory in caller



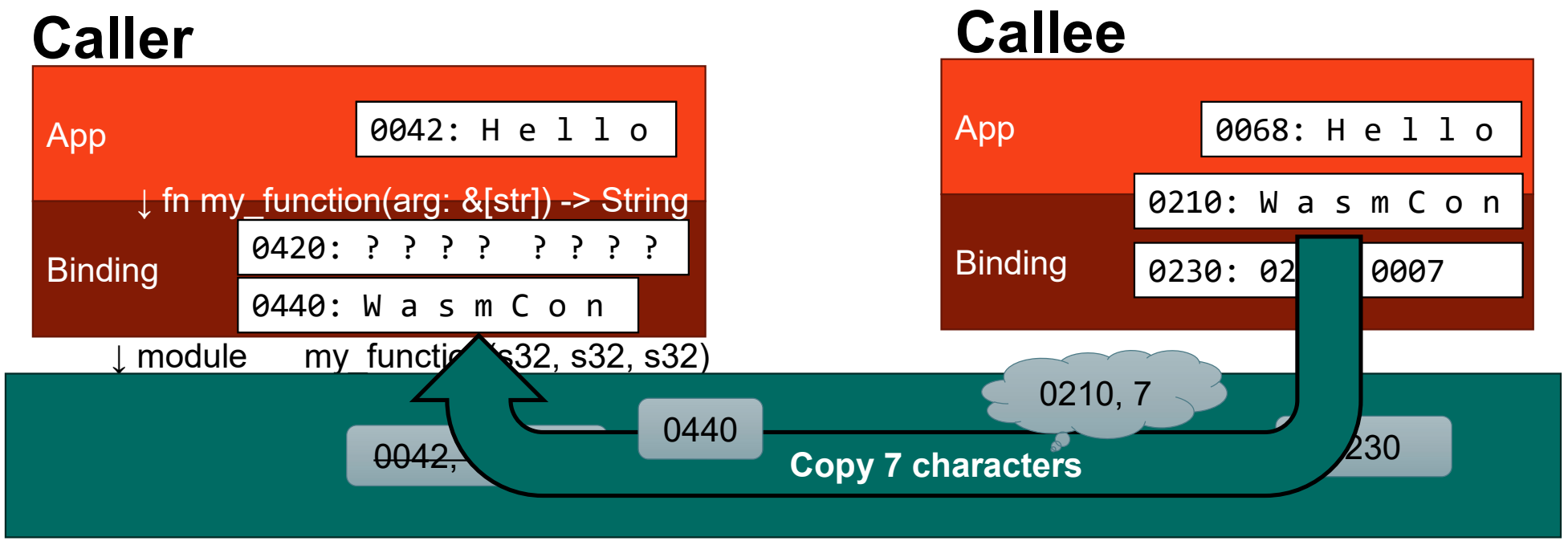
APTIV

Caller returns address



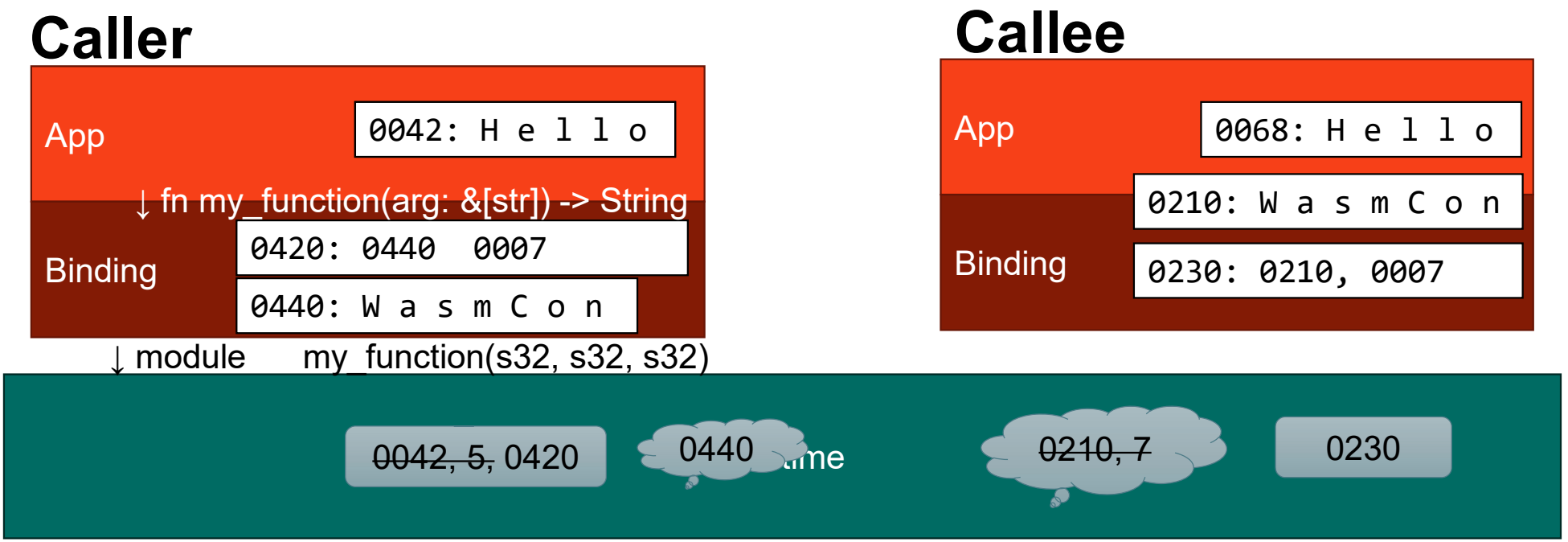
APTIV

Runtime copies seven characters



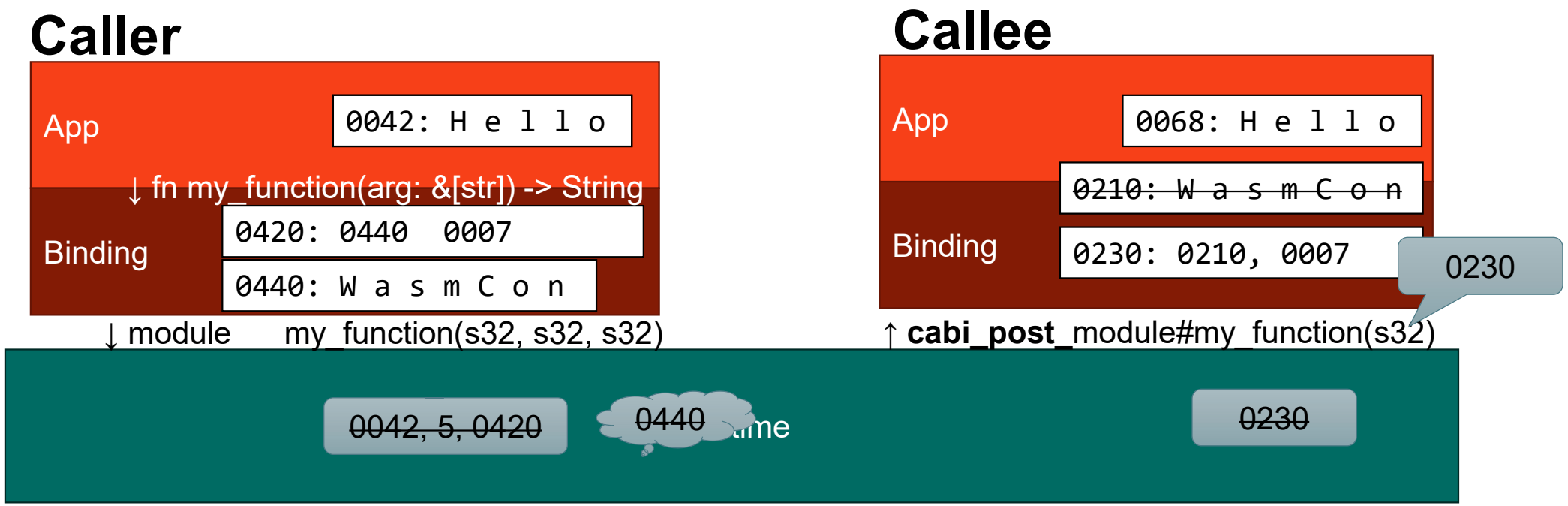
APTIV

Runtime fills return area in caller



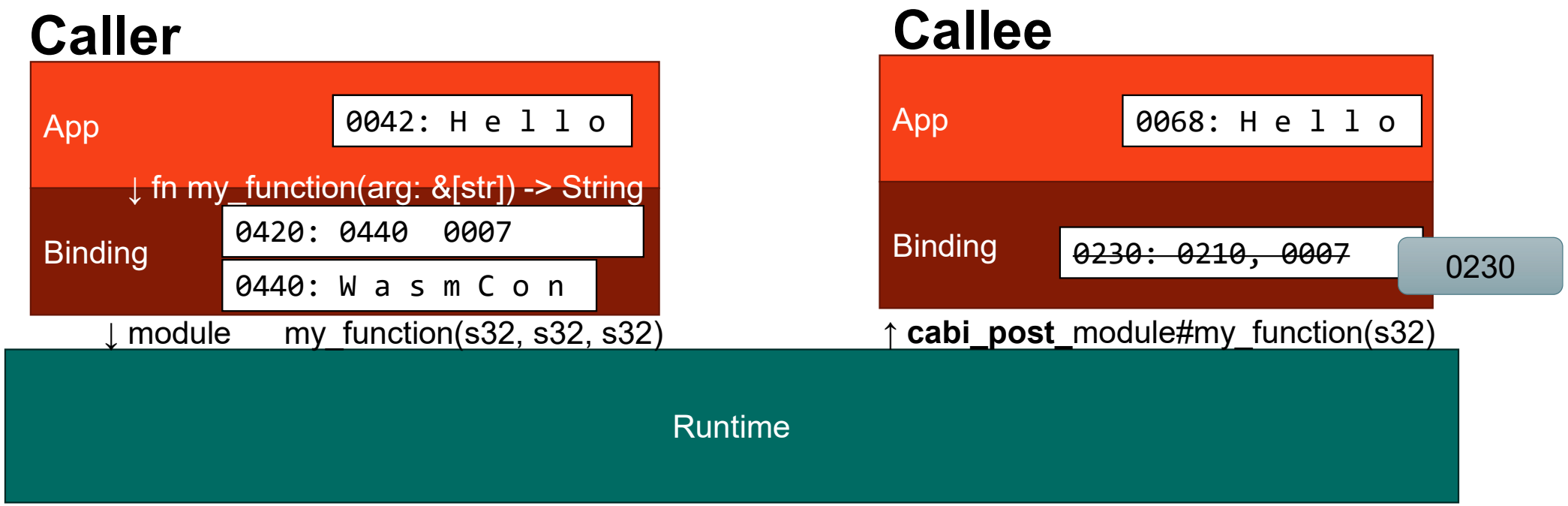
APTIV

Runtime cleans up



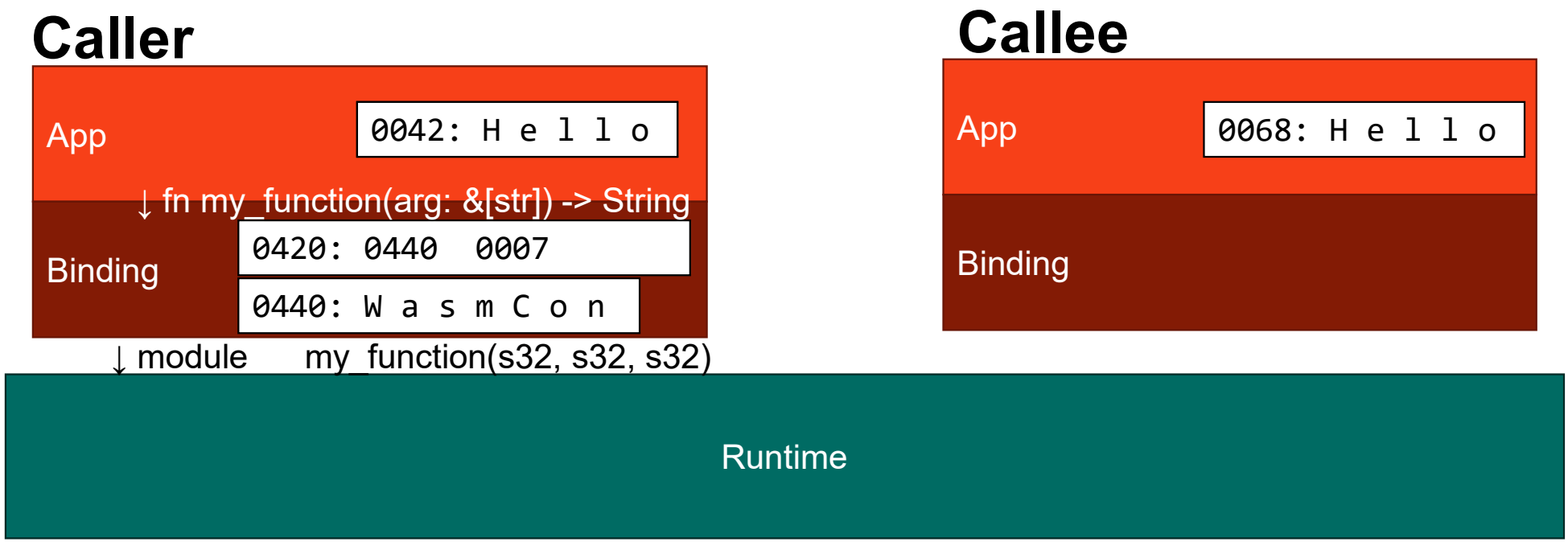
APTIV

Binding cleans up



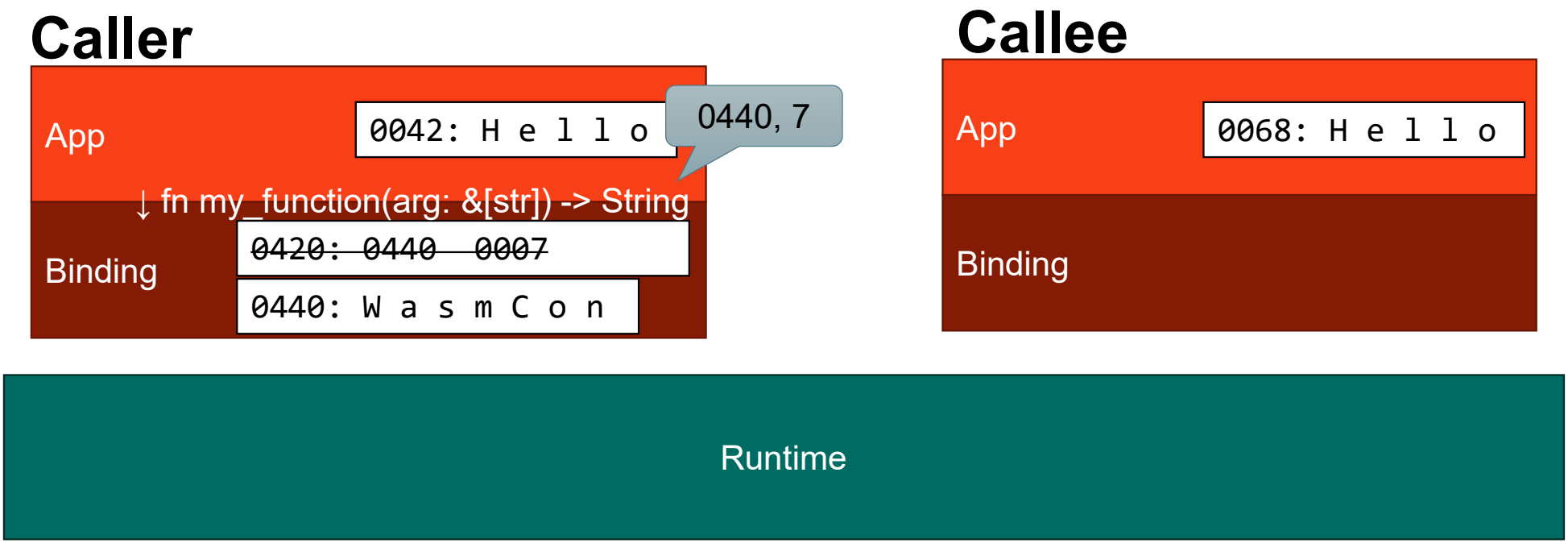
APTIV

Return to caller



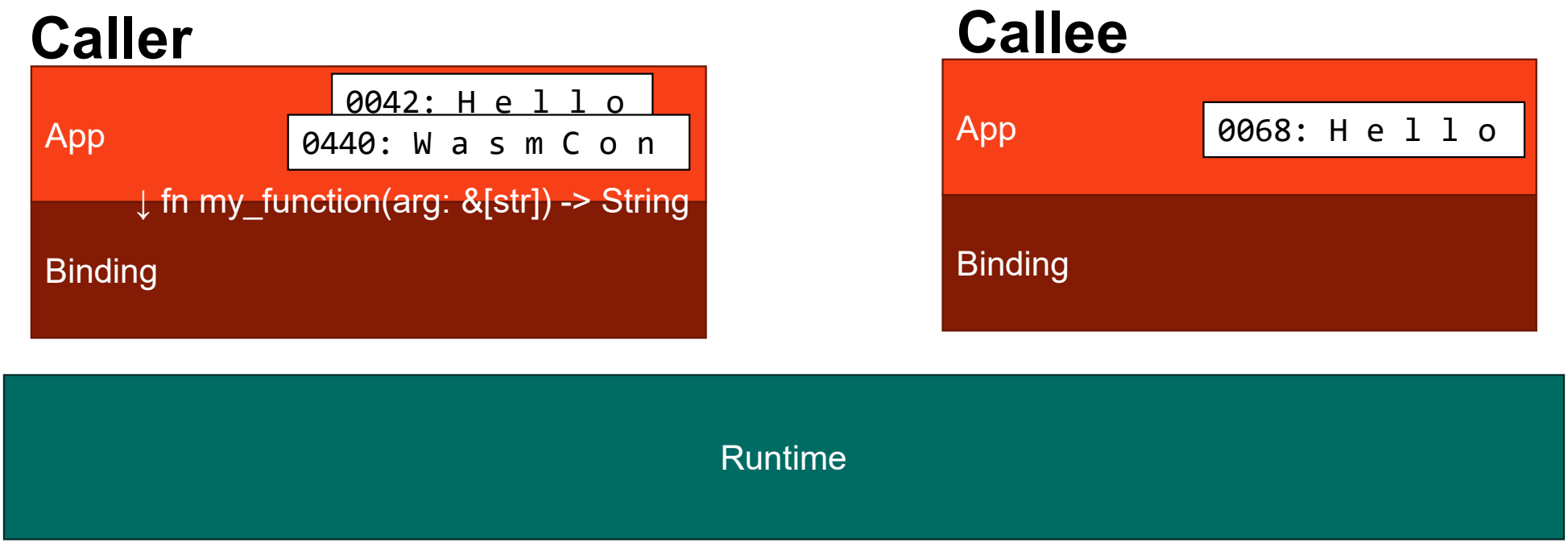
APTIV

Return to app



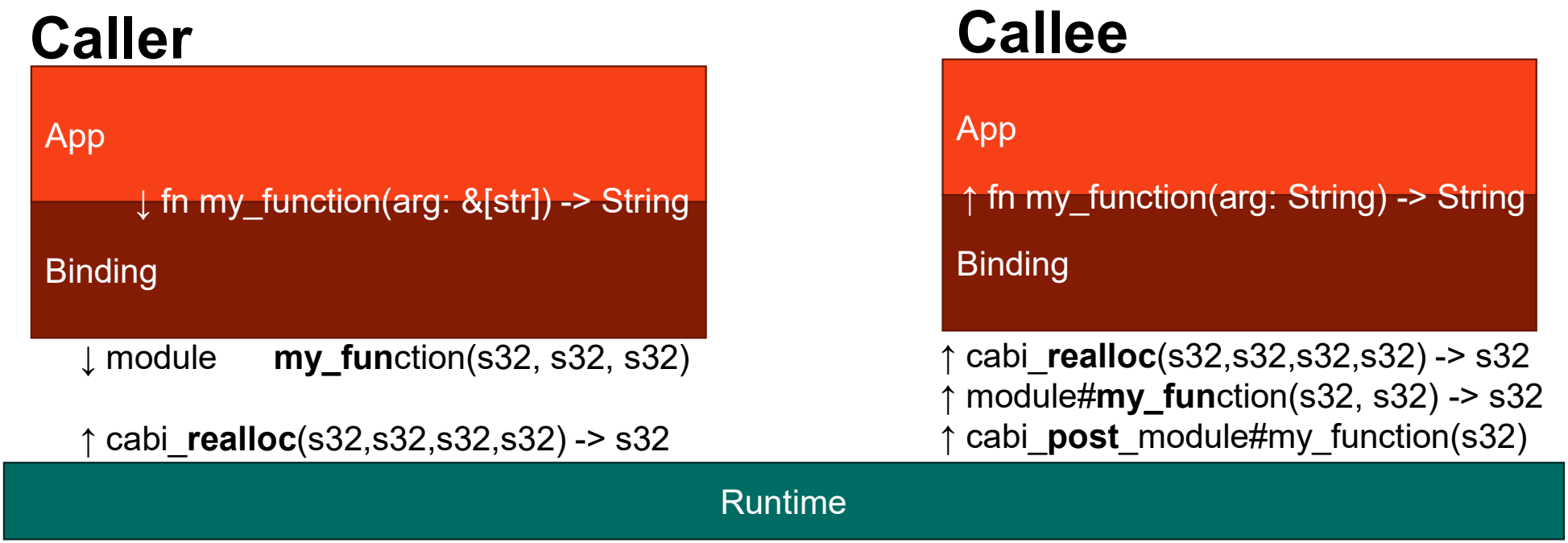
APTIV

Both apps own their strings



APTIV

Recap: Calls and symbols



APTIV

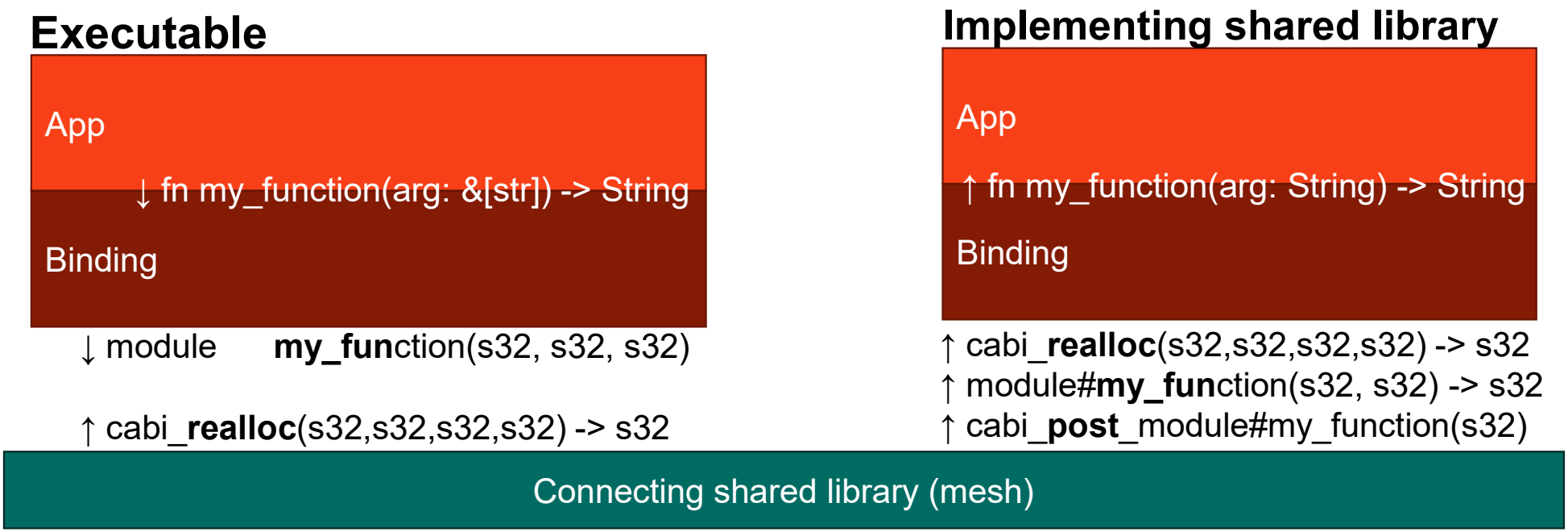
Native compilation

⚠ speed ~~sandbox~~

WIT for native binaries in SDV

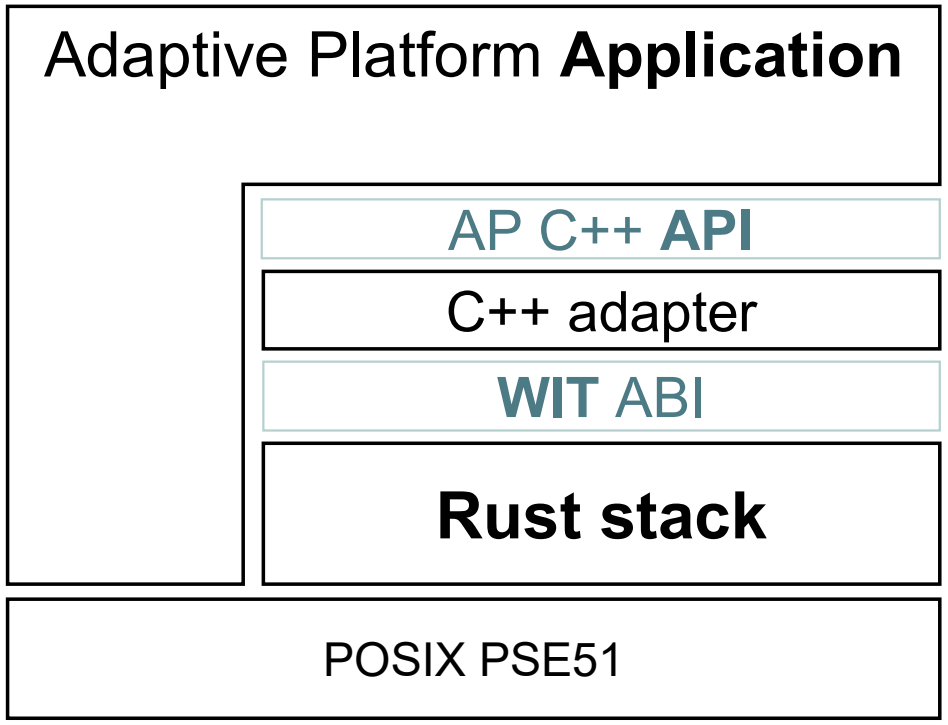
- **Native binaries maximize CPU efficiency**
  Unchanged debugging and deployment experience
- **Mixing languages with minimal overhead**
  Optimized ABI for shared everything
- **Transition to and from wasm by recompilation**
  Transparently move between native and wasm components

Native compilation



APTIV

The Component Model enables mixing languages



- **Unmodified C++ application**
- **Async methods and events via WASI 0.2**
- **Rust AUTOSAR adaptive stack**
- **Native binary**

Publisher subscriber demo application

```
FACT Info fusion alive
FACT Info not subscribed to brakeEvent yet
FACT Info Callback registered.
FACT Info brakeEvent subscription complete
FACT Info parkingBrakeEvent SetReceiveHandler() complete
FACT Info Callback registered.
FACT Info parkingBrakeEvent subscription c
FACT Info Subscribe to Update Rate Field
FACT Info updateRateSubscription()
FACT Info Callback registered.
FACT Info Subscribe to Front Object Dista
FACT Info frontObjectDistanceSubscription
FACT Info Callback registered.
FACT Info Testing Field Getter.
FACT Info FIELDS: Trying to get values for UpdateRate and RearObjectDistance
CTX3 Info FIELDS: Getting the field Update Rate value :87
FACT Verbose ParkingBrakeEvent E2E state:ok (NoData)
FACT Verbose Callback: ParkingBrakeEvent - Radar: NOT active Object:255
               E2E checkStatus:ok (NotAvailable)
CTX3 Info FIELDS: Getting the Rear Object Distance Distance :-1
FACT Verbose FIELDS:  Current Rear Object Distance is 65535
FACT Verbose FIELDS:  Current Update Rate is 87
FACT Info Testing Field Setter.
FACT Info FIELDS: Trying to set UpdateRate and ObjectDetectionLimit
FACT Info FIELDS: UpdateRate setting skipped
FACT Info FIELDS: ObjectDetectionLimit setting skipped
FACT Info Testing Synchronous Method Call.
CTX3 Info radar active
CTX3 Info brakeEvent sent
CTX3 Info parkingBrakeEvent sent
```

Application Output

`libwasi.so`: WASI clocks, poll

`libstack.so`: ara::core exec log

`libradar.so`: Publisher

`libfusion.so`: Subscriber

`target/…/main`: Calls 2×run

libradar.so symbols

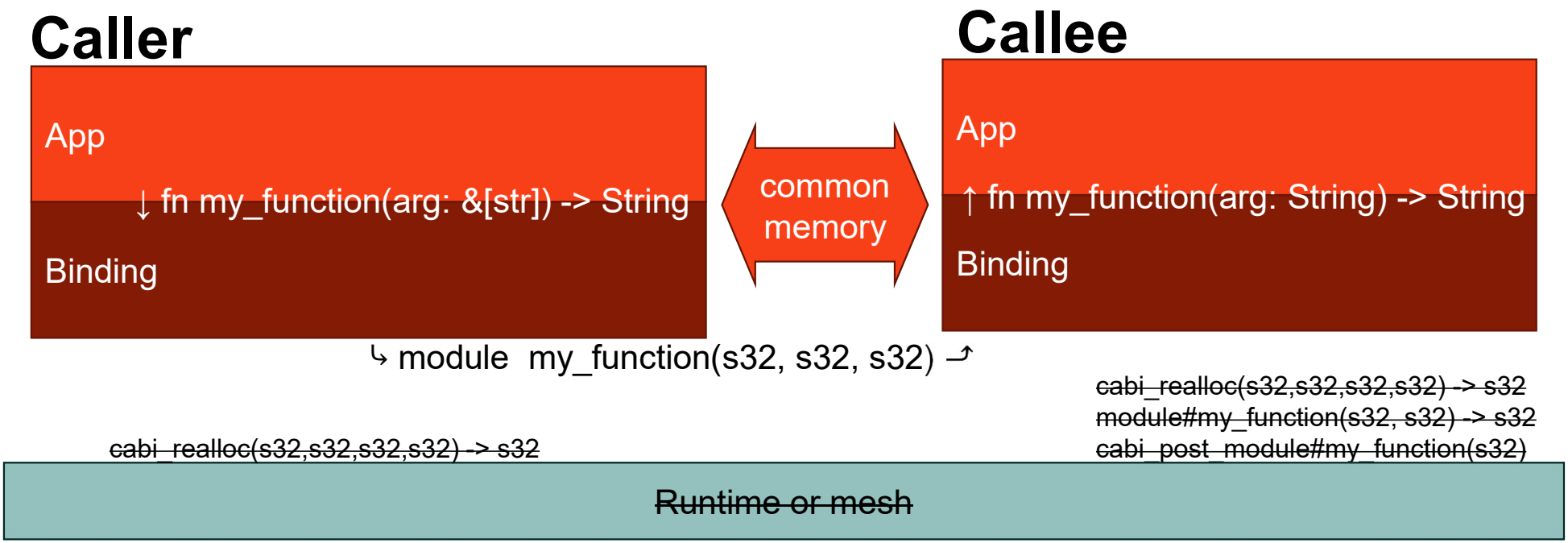
```
  D  *UND* autosar:ara/log‖[constructor]logger
  D  *UND* autosar:ara/log‖[method]logger.report
  D  *UND* autosar:ara/log‖[resource-drop]logger

  D  *UND* wasi:clocks/monotonic-clock@0.2.0‖now

g DF .text radar-wasi:cli/run@0.2.0#run

g DF .text autosar:ara/radar-service‖[method]instance.adjust
g DF .text autosar:ara/radar-service‖[method]future-result-adjust-output-error-code.subscribe
g DF .text autosar:ara/radar-service‖[static]future-result-adjust-output-error-code.get-value
```

Symbol mangling for native environment

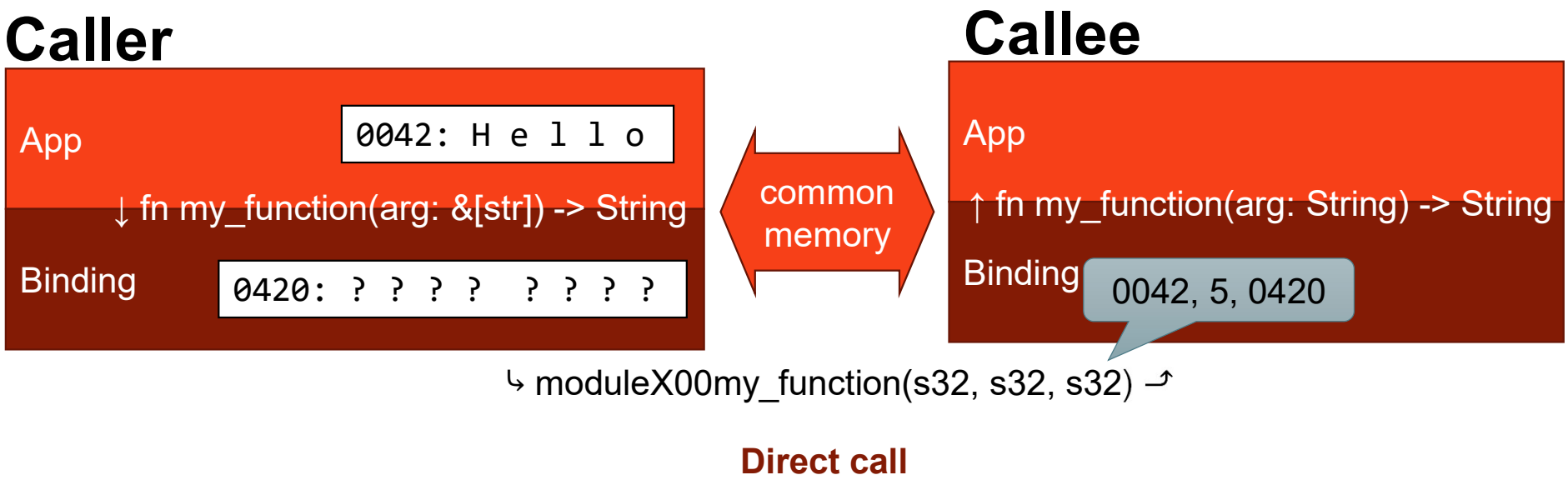
```
autosar:ara/radar-service‖[static]
future-result-adjust-output-error-code
.get-value

autosarX3AaraX2Fradar_serviceX00X5BstaticX5D
future_result_adjust_output_error_code
X2Eget_value
```

Symmetric ABI

Symmetric Idea: Can we remove the runtime?

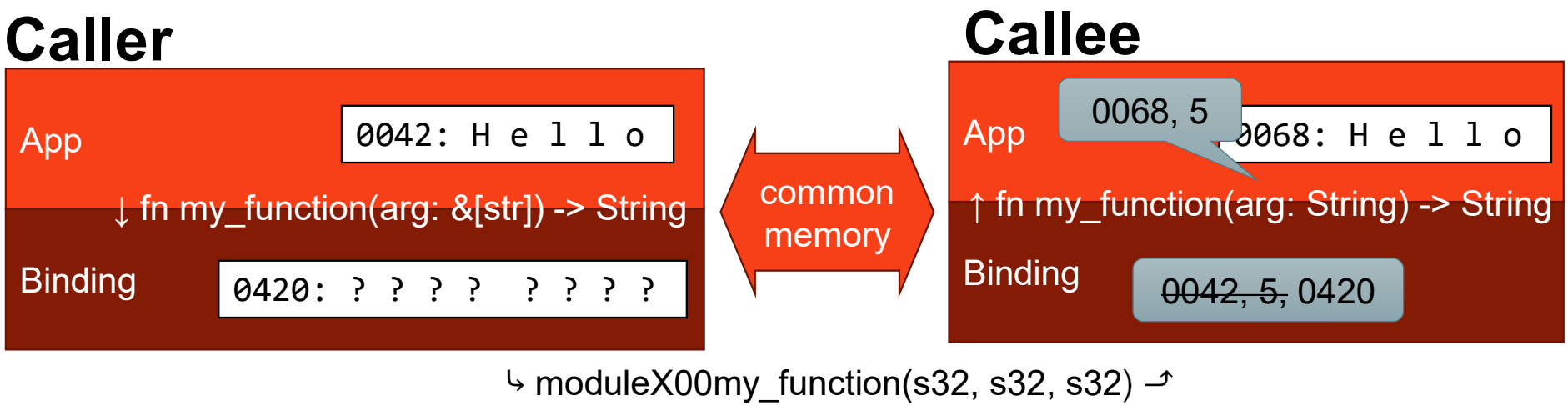


Symmetric: Caller binding directly calls callee

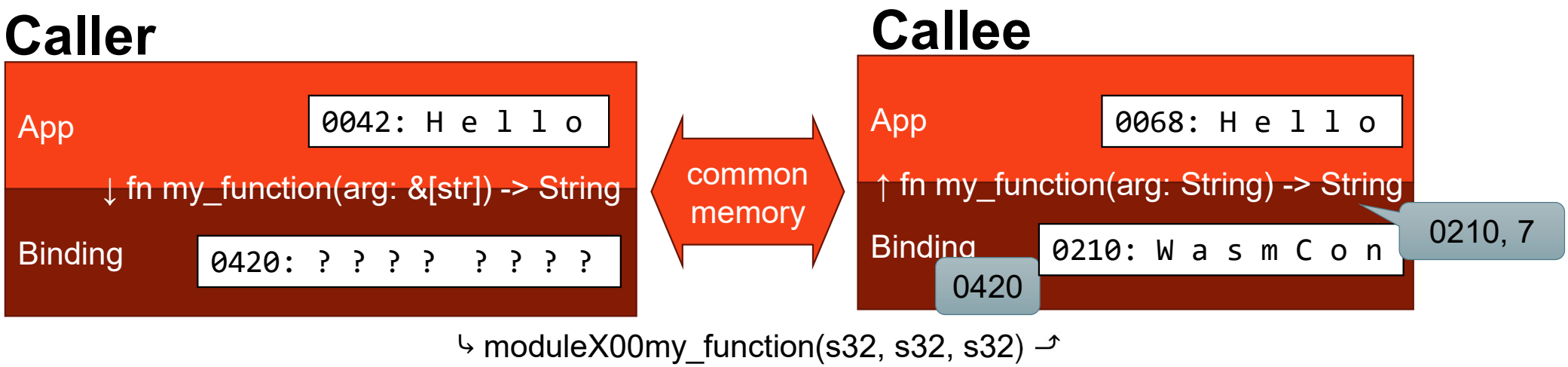


APTIV

Symmetric: Binding allocates, copies, and calls

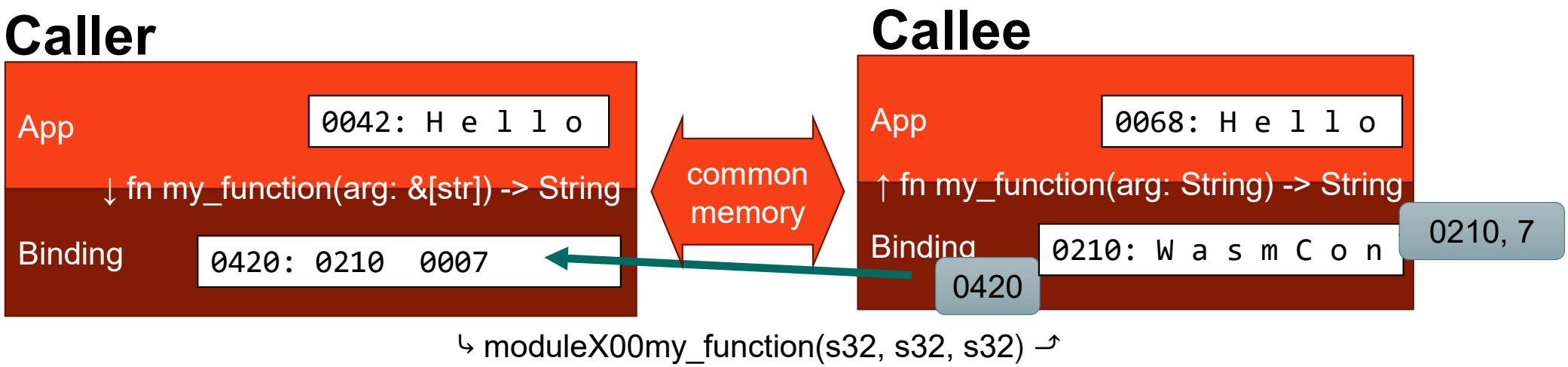


Symmetric: App returns string



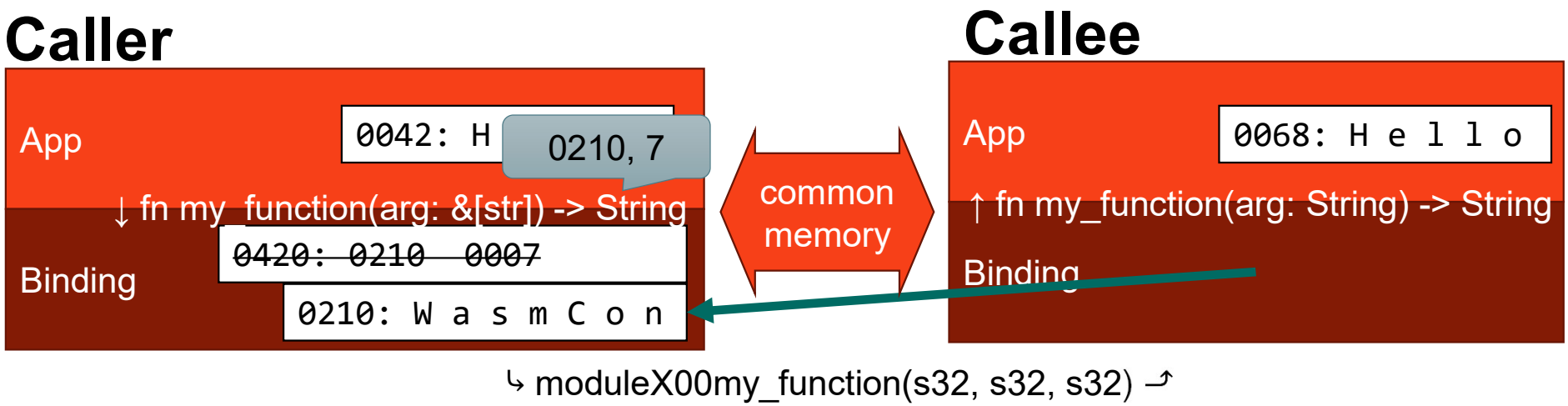
APTIV

Symmetric: Binding fills return area



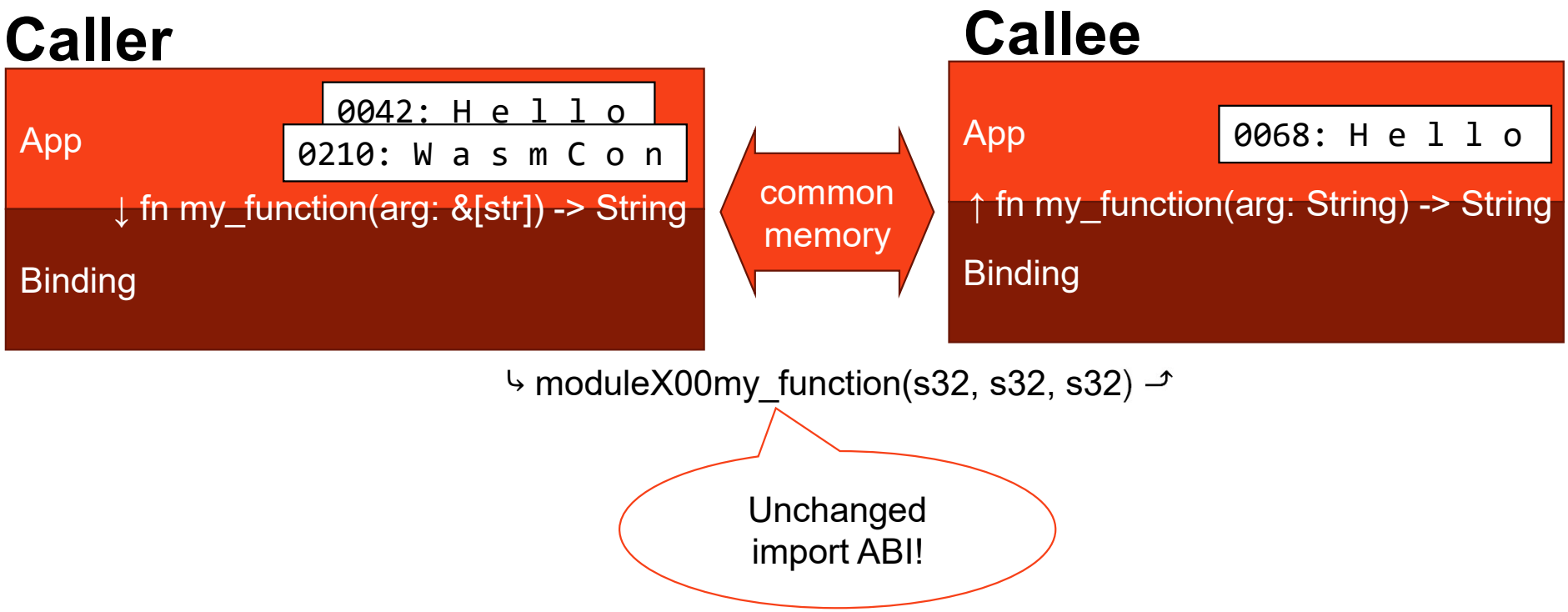
APTIV

Symmetric: Binding returns re-owned string



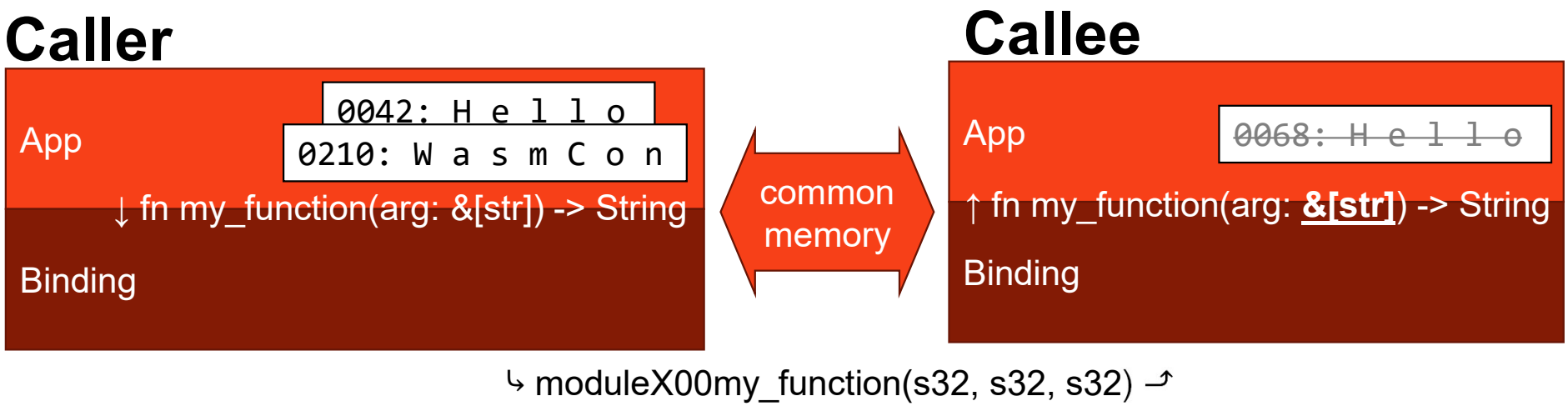
APTIV

Symmetric: Neither app sees a difference



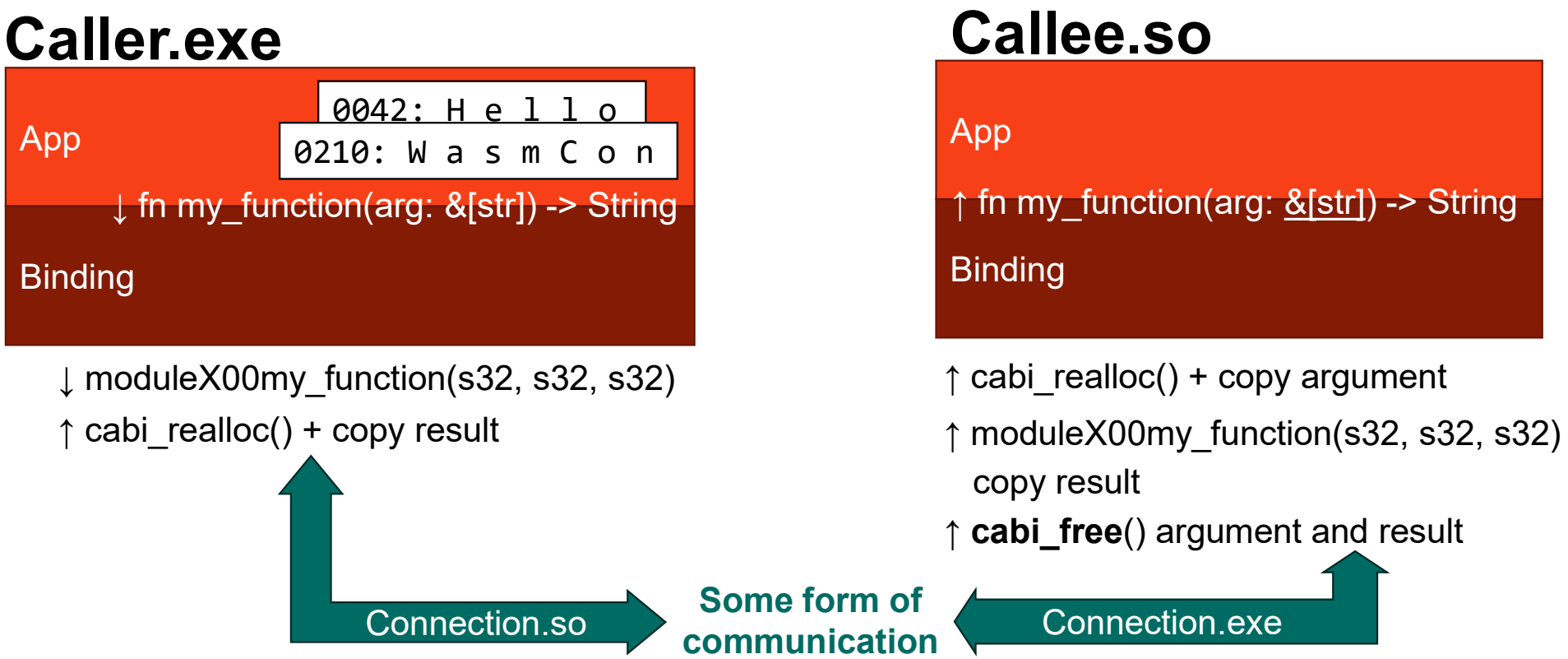
APTIV

Symmetric API: Simpler and no copy



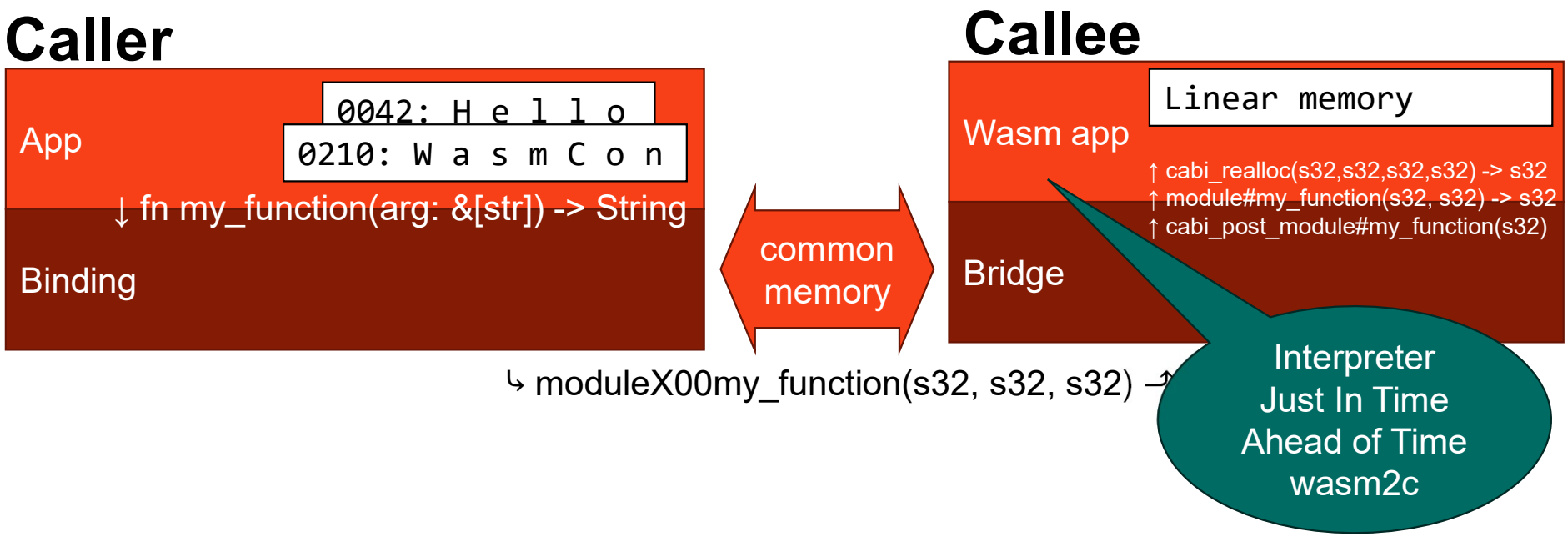
APTIV

Full insulation is a runtime option



APTIV

Symmetric: Wasm remains a transparent option



APTIV

Outlook

Further use for WIT without wasm

- Use WIT for zero-overhead plugins
  e.g. in wasmtime or embedded
- A stable binary interface between Rust modules
- No sandbox, plugins can use all of POSIX
  (both good and bad)

Additional topics

- **Multithreading & Exceptions: C++**
- **Symmetric resource ABI**
- **Async calls: WASI 0.3**
- **Functional safety: Zero-allocation**
- **Zero-copy sandbox**

Where to get more information

Fork of wasm-tools and wit-bindgen at https://github.com/cpetig/

Issue for Symmetric ABI https://github.com/WebAssembly/component-model/issues/386

Flat data types https://github.com/cpetig/flat-types-rust

TL;DR

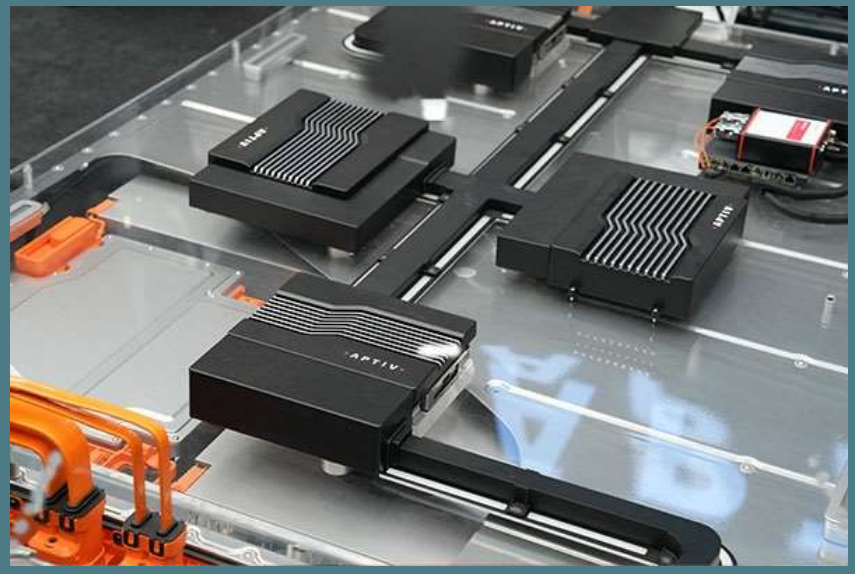
- **Symmetric ABI optimizes the component model for shared everything**

- **Mixing sandboxed and shared everything leverages their complimentary strengths**

Thank you!

Especially to all creators of WASI!

Questions?



Backlog

C++ challenges WASI 0.2

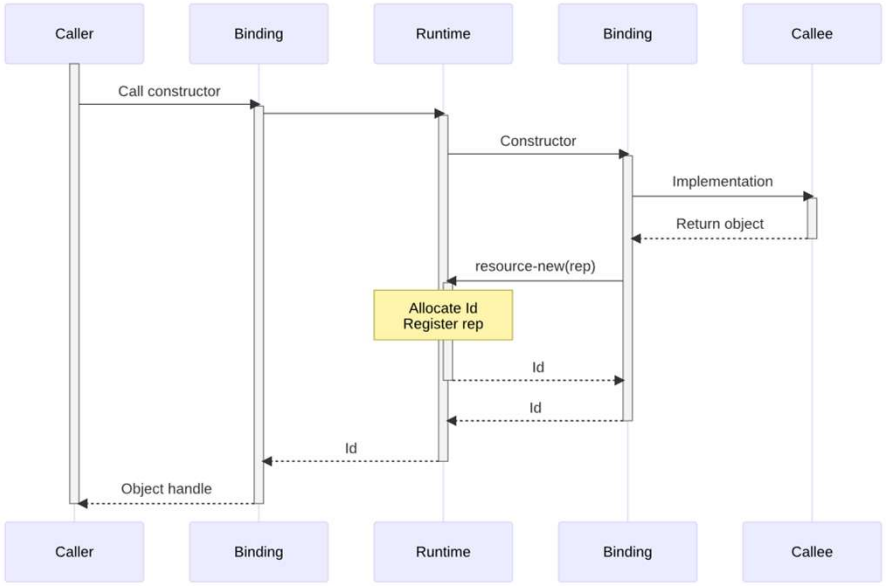
- **Async (promise & future)**
  needs multi-threading
  - **… or concurrency TS's .then() or C++-20's co_await**
- **Throw & catch is common**
  needs exceptions

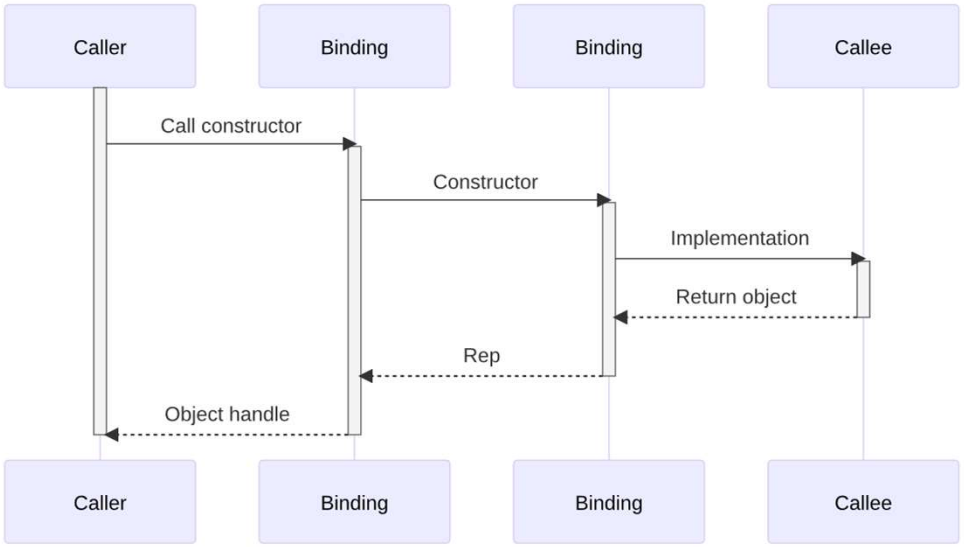
Symmetric: Resources and async

- **Resources: Use import ABI**
- Much simpler ABI, no resource-new/drop/rep
- A pointer sized resource ID obviates callee side resource-tables

- **Async: Use import ABI**
- Needs shared event & executor
- Event awaiting callee registers callback with executor

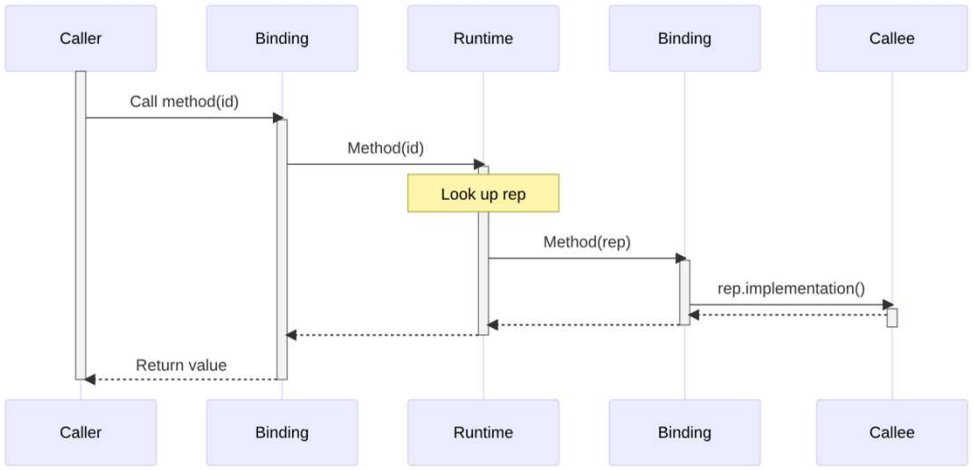
Symmetric Resources: Create

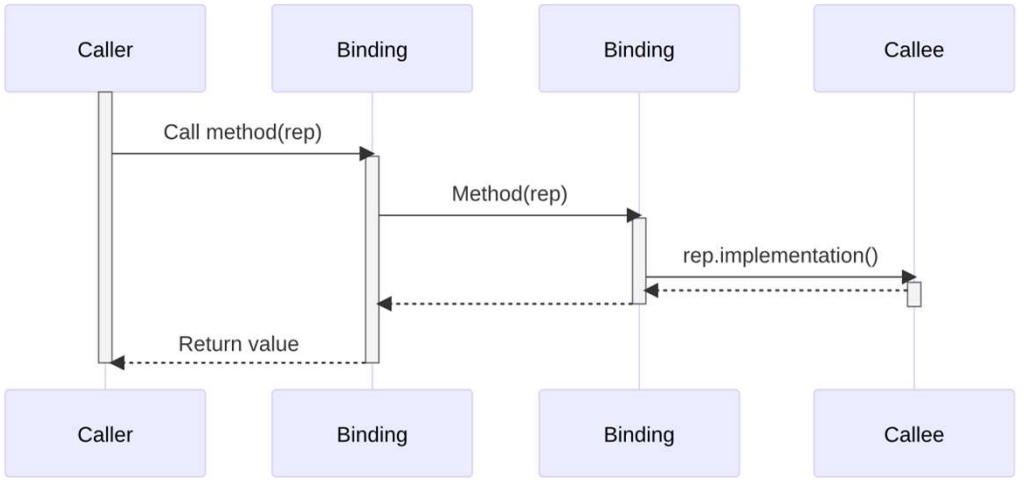




APTIV

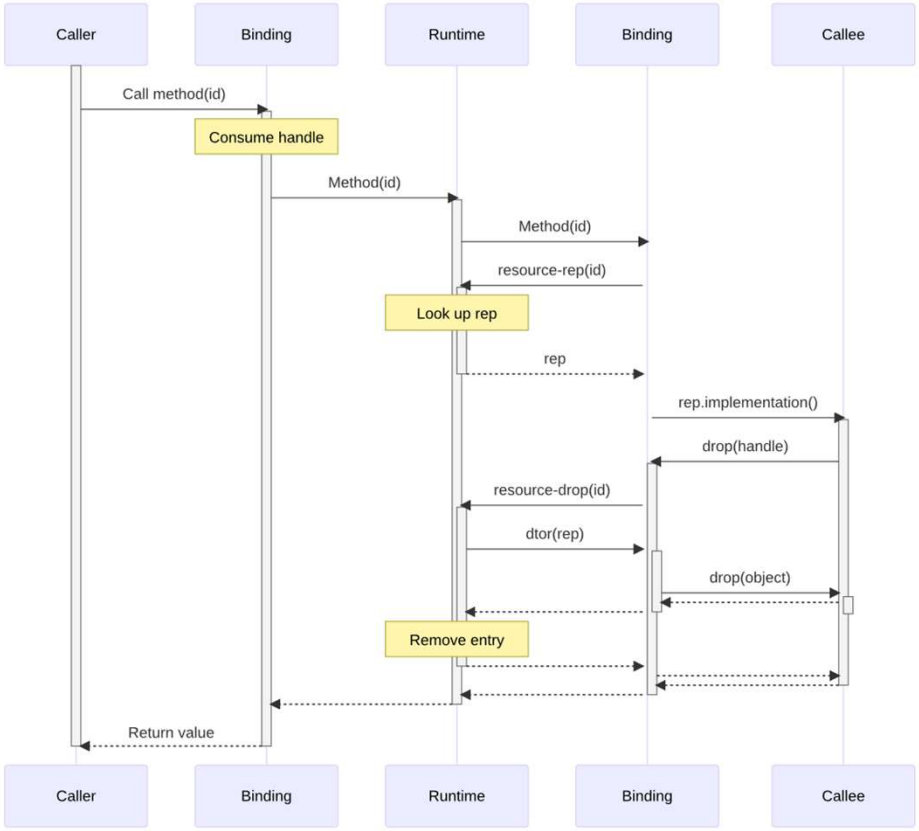
Symmetric Resources: Method

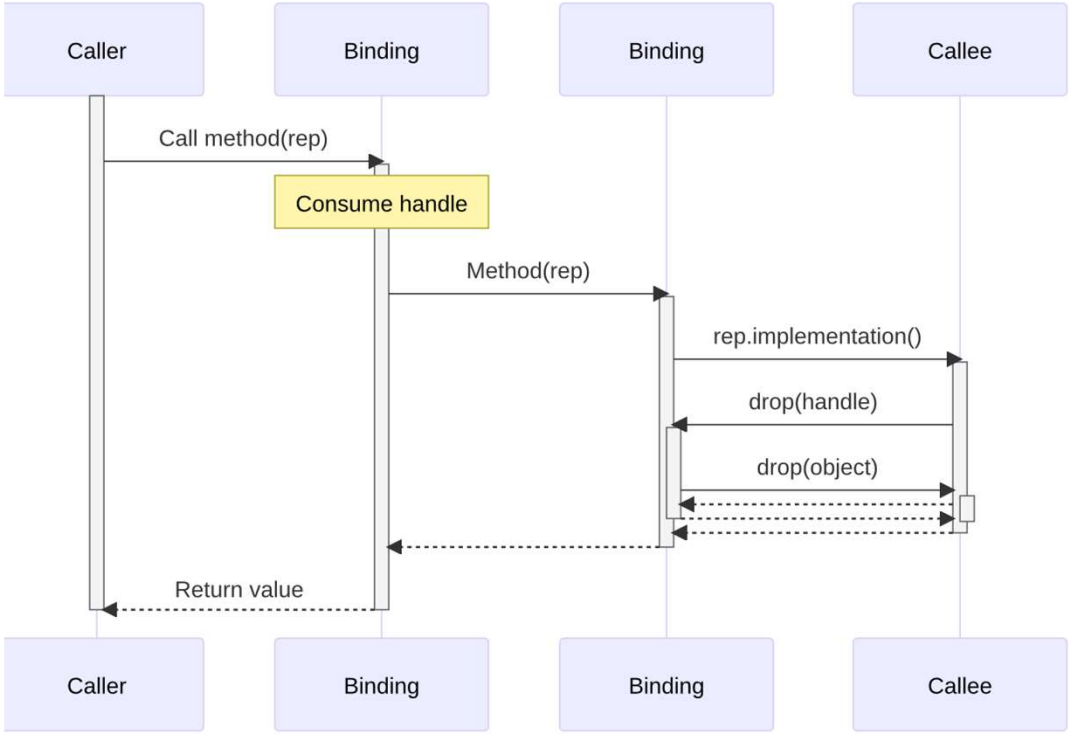




APTIV

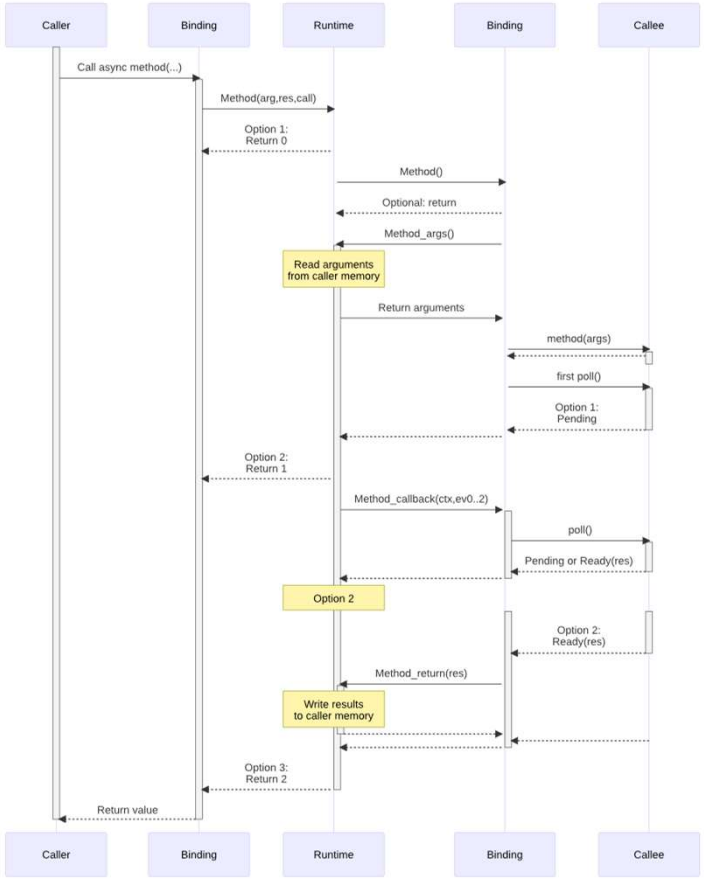
Symmetric Resources: Owned

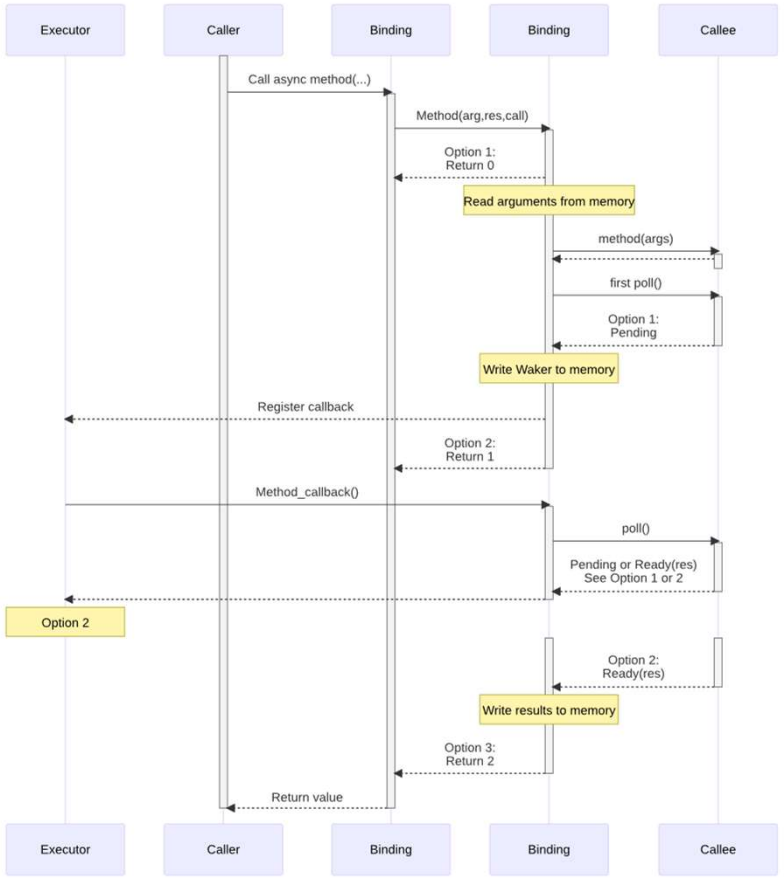




APTIV

Symmetric Async





APTIV

Functional safety

- **Sandboxing is freedom from interference**
- **Constant time allocations**
- **Deterministic allocator or no allocations at run-time**
- **Qualification of bindings code**

Zero-copy between sandboxes

- **Needed for AI and image processing**
- **Shared memory**
- **One writer exclusive or many readers per element**
- **Flat data structures: Position independent**
- **Multi-memory or memory mapping**
- **More versatile if caller provides buffer**

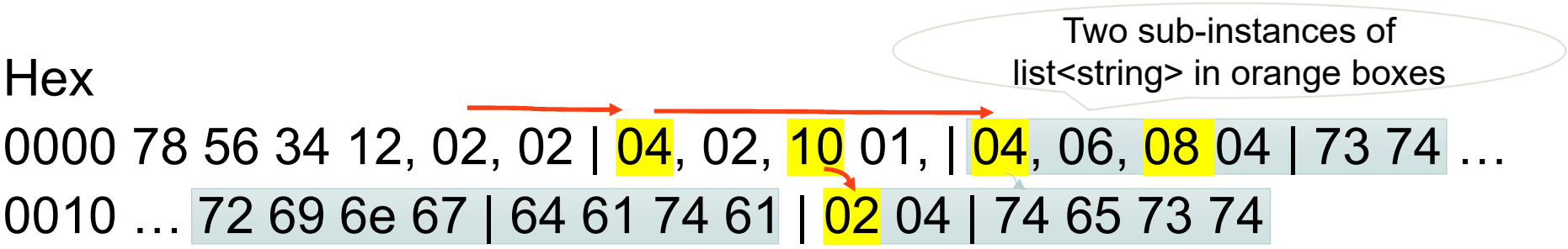
Zero-copy inter-process via shared memory

```
record data {
    id: u32,
    name: list<list<string>>,
}
```

Layout:

$id_{32}\ a_8\ l_8$ | $a_8\ l_8\ a_8\ l_8$ |$a_8\ l_8\ a_8\ l_8$ | “string” “data” | $a_8\ l_8$ | “test”

Hex



APTIV