IP Network Stack in Ada 2012 and the Ravenscar Profile

Stéphane Carrez

Ada Europe 2017

Ada Embedded Network Stack

- Project Presentation
- Implementation Details
- EtherScope use case
- Difficulties and solutions with Ravenscar profile

Project Presentation

- IPv4 network stack written in Ada 2012
- Runs on bare metal ARM boards
- Small footprint: 50 Kb
- Several standard protocols: ARP, IP, UDP, DHCP, DNS, NTP
- Open source license: Apache License 2.0

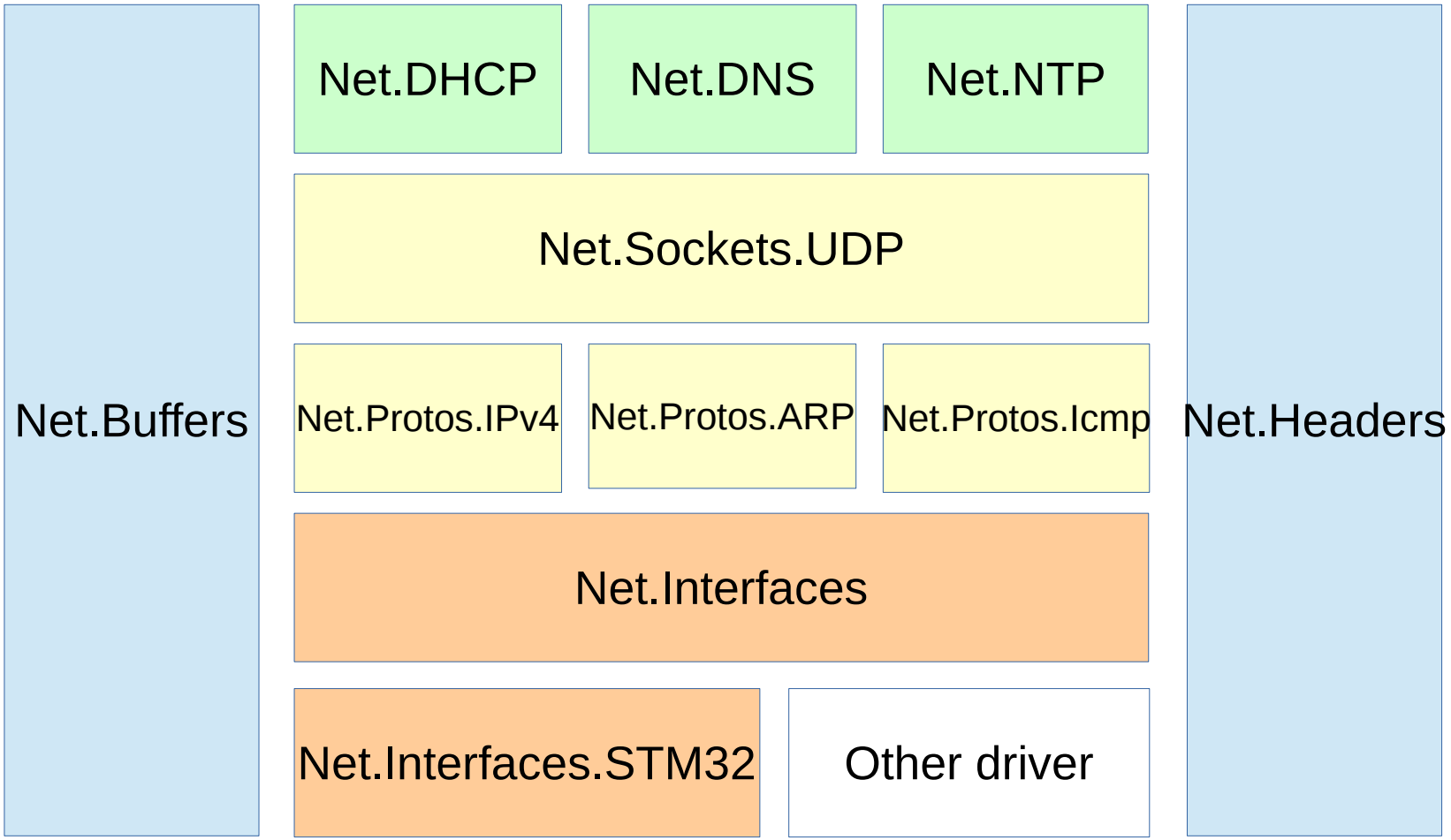
Motivations

- Boards need to interact with the network
- Objects have to be connected
- Get a reliable, safe and secure network stack
- Created for the EtherScope MakeWithAda project

Implementation Goals

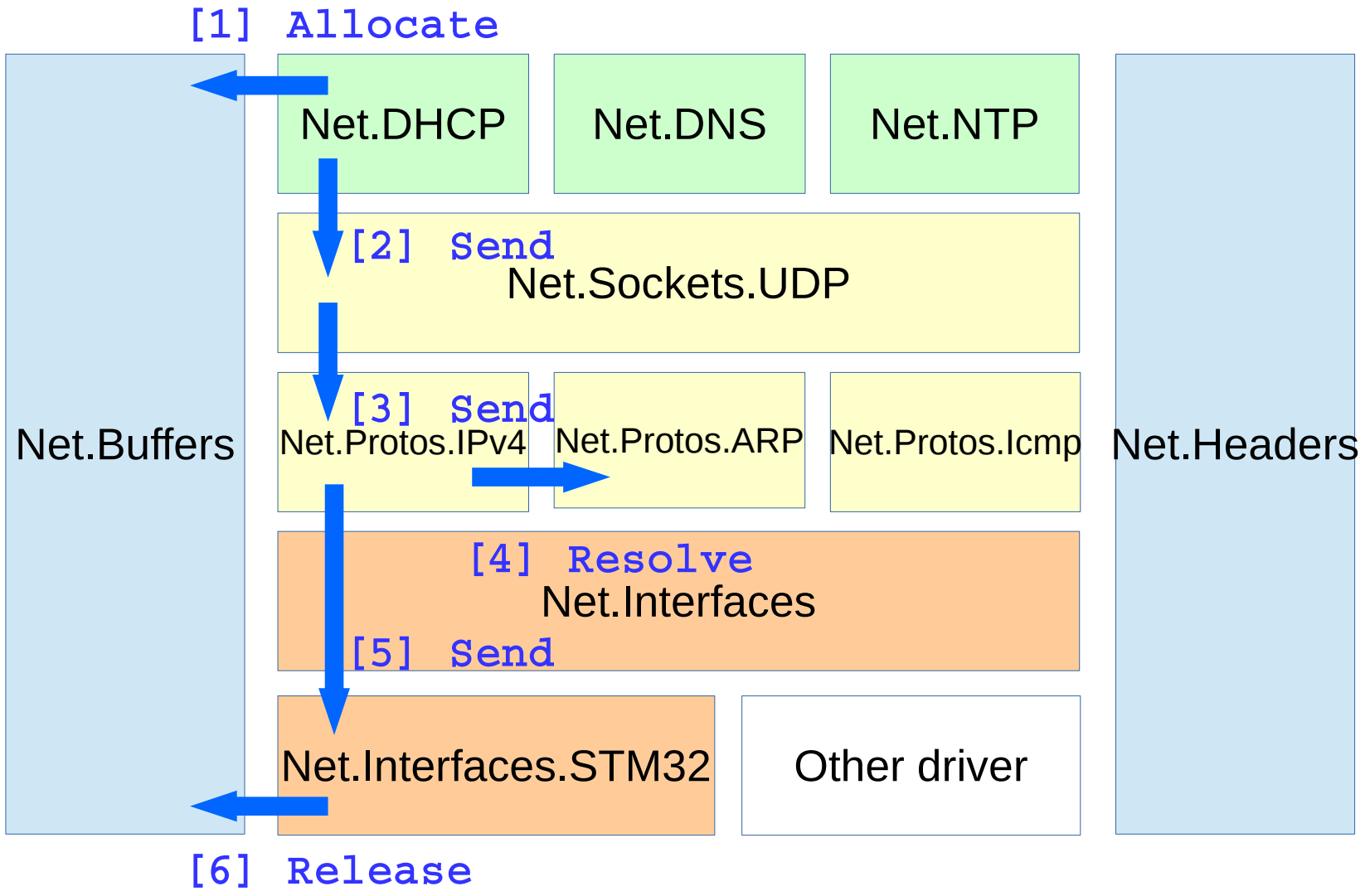
- Ada 2012 implementation with Ravenscar sfp profile
- Avoid memory copies when sending or receiving
- Leave the task model to the application
- Promote asynchronous programming models
- Blocking operations for receiving and sometimes for sending

Architecture & Ada Package

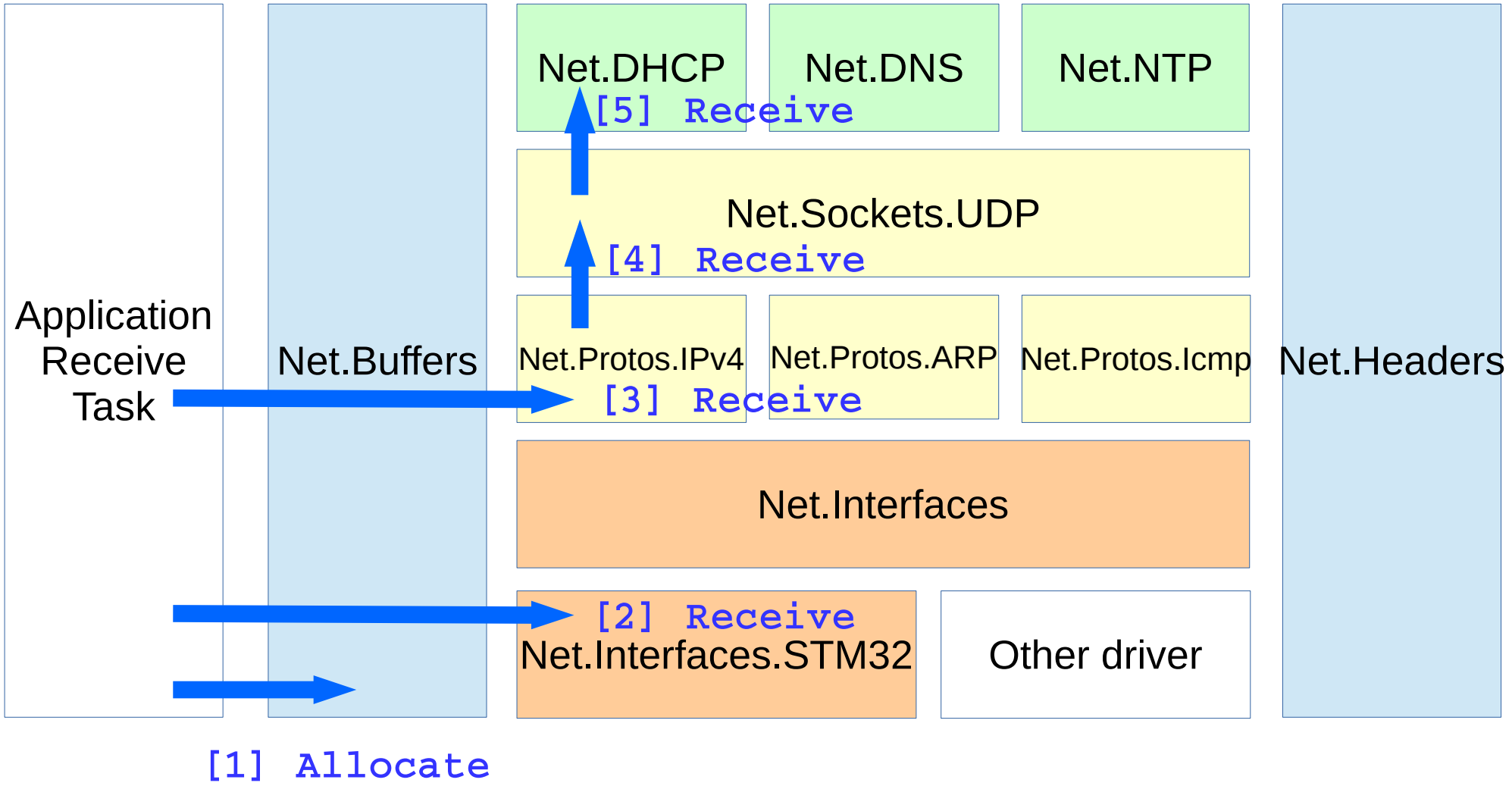


https://github.com/stcarrez/ada-enet

Sending a packet



Receiving a packet



Ethernet Driver & Max_Protected_Entries => 1

- Represented by an abstract tagged type: `Net.Interfaces.Ifnet_Type`
- Defines 3 abstract operations: `Initialize`, `Send`, `Receive`
- Concrete implementation: `Net.Interfaces.STM32.STM32_Ifnet`
- STM32 Ethernet driver uses interrupts to send and receive packets
- Transmit and receive queues controlled by two protected objects

```
protected Transmit_Queue is
   entry Send (Buf : in out Buffer_Type);
   procedure Transmit_Interrupt;
   procedure Initialize;
private
   Tx_Ready : Boolean := False;
   ...
end Transmit_Queue;
```

```
protected Receive_Queue is
   entry Wait_Packet (Buf : in out Buffer_Type);
   procedure Receive_Interrupt;
   procedure Interrupt;
   procedure Initialize;
private
   Rx_Available : Boolean := False;
   ...
end Receive_Queue;
```

https://github.com/stcarrez/ada-enet

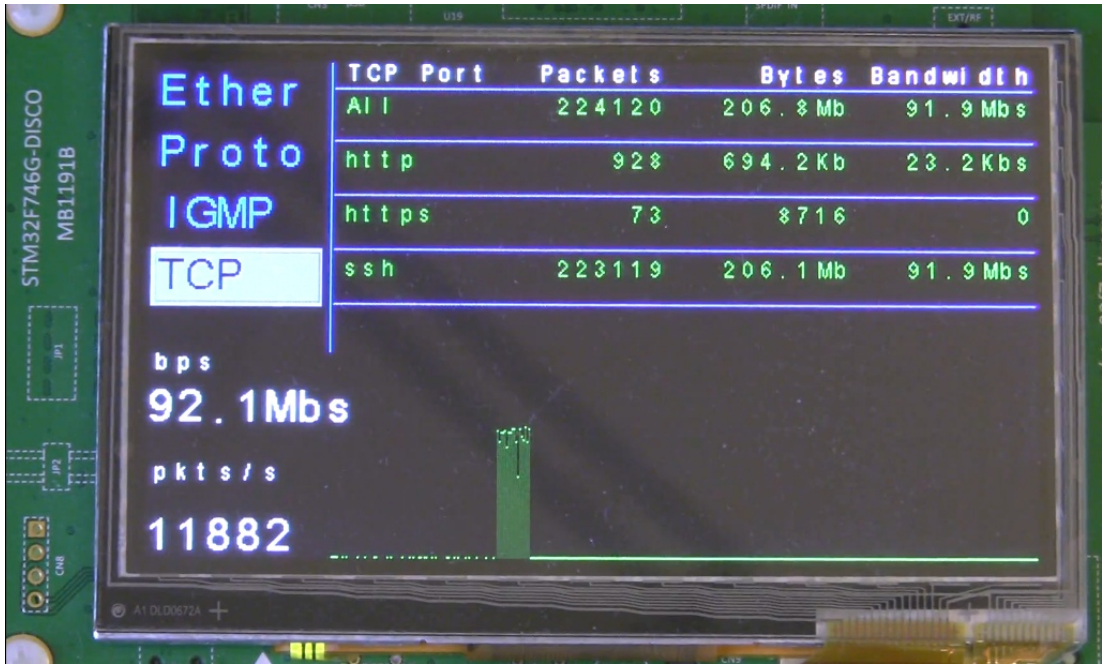
Network housekeeping & No_Relative_Delay

- Need to manage ARP timeouts and ARP queries
- Need to manage the DHCP state machine
- Can be implemented as specific tasks
- Can be integrated in application's main loop

```
   Deadline : Ada.Real_Time.Time;
begin
   loop
      Net.Protos.Arp.Timeout (Ifnet);
      Dhcp.Process (Deadline);
      delay until Deadline;
   end loop;
end;
```

EtherScope example

- EtherScope is a simple network protocol analyzer
- It receives packets, analyzes them, displays results
- Realtime analysis up to more than 12000 packets/sec



- Ada 2012
- Runs on STM32F746 board

https://github.com/stcarrez/ada-enet

EtherScope example

- Main loop waits for touch panel events and refresh the display periodically
- Receiver task loops to receive packets and analyze them
- Realtime pressure on the receiver task only

```
Ifnet.Initialize;
Set_True (Ready);
loop
  if Button_Pressed then
     Update_Display;
  end if;
  ...
  if Refresh_Deadline <= Now then
     Update_Display;
  end if;
  ...
  delay until Next_Deadline;
end loop;
```

```
with Ada.Synchronous_Task_Control;
...
Ready : Suspension_Object;
task body Controller is
  Packet : Net.Buffers.Buffer_Type;
begin
  Suspend_Until_True (Ready);
  Net.Buffers.Allocate (Packet);
  loop
    Ifnet.Receive (Packet);
    EtherScope.Analyze.Base.Analyze (Packet);
  end loop;
end Controller;
```

https://github.com/stcarrez/ada-enet

Difficulty: No Random Numbers

- Random number generators are used by DHCP and DNS
- No `Ada.Numerics.Discrete_Random` package in Ravenscar sfp

```
with Ada.Numerics.Discrete_Random;
package Rand is new
  Ada.Numerics.Discrete_Random (Uint32);

R : Rand.Generator;

function Random return Uint32 is
begin
   return Rand.Random (R);
end Random;
```

https://github.com/stcarrez/ada-enet

Solution: No Random Numbers

- Use hardware support on STM32 board
- Use `STM32.RNG.Interrupts` package from Ada_Drivers_Library

```
with STM32.RNG.Interrupts;
procedure Initialize is
begin
   STM32.RNG.Interrupts.Initialize_RNG;
end Initialize;
function Random return Uint32 is
begin
   return STM32.RNG.Interrupts.Random;
end Random;
```

```
protected body DHCP_State_Machine
is
   procedure Make_Request is
   begin
      XID := Random;
      ...
   end Make_Request;
end DHCP_State_Machine;
```

https://github.com/stcarrez/ada-enet

Difficulty: pragma Detect_Blocking

- A protected operation must not call a protected entry
- Program_Error is raised when a protected operation calls a protected entry

raise `Program_Error`

```
protected body DHCP_State_Machine
is
   procedure Make_Request is
   begin
      XID := Random;
      ...
   end Make_Request;
end DHCP_State_Machine;
```

Solution: pragma Detect_Blocking

- Could be detected by static analysis of the complete program
- Call blocking operations outside of protected types

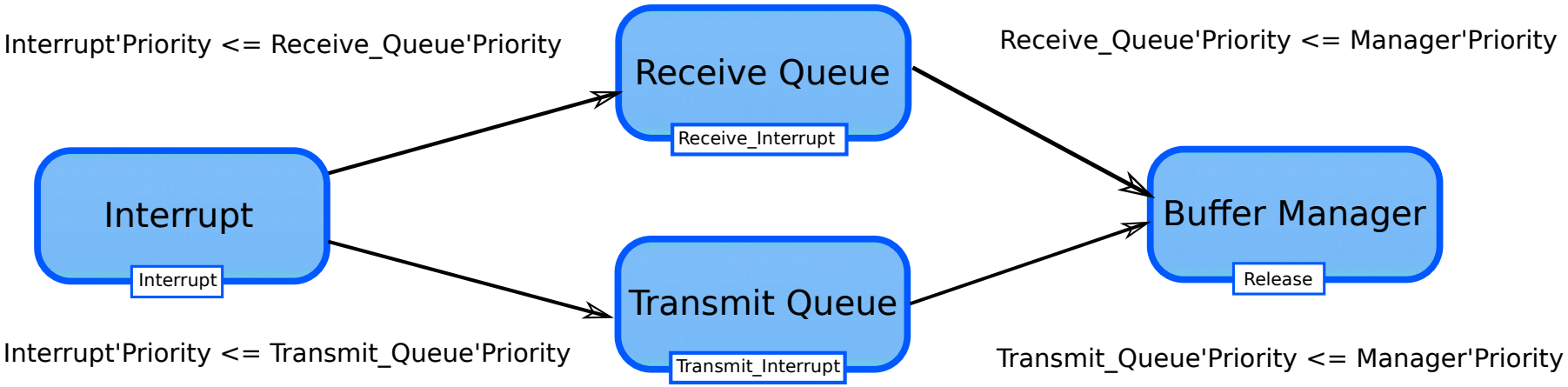
```
protected body DHCP_State_Machine is
  procedure Make_Request (Id : in Uint32) is
  begin
    XID := Id;
    ...
  end Make_Request;
end DHCP_State_Machine;
...
DHCP_State_Machine.Make_Request (Id => Random);
```

Difficulty: pragma Locking_Policy(Ceiling_Locking)

- A task of high priority must not access a protected object of lower priority
- Program_Error is raised when ceiling priorities are not respected

Solution: pragma Locking_Policy(Ceiling_Locking)

- Static analysis of the complete program



Call graph of protected objects

```
package Net is
 Network_Priority : constant System.Interrupt_Priority
                                    := System.Interrupt_Priority'First;
package Net.Interfaces.STM32 is
  protected Transmit_Queue with Priority => Net.Network_Priority is ...
  protected Receive_Queue with Priority => Net.Network_Priority is ...

package Net.Buffers is
  protected Manager with Priority => Net.Network_Priority is ...
```

https://github.com/stcarrez/ada-enet

Difficulty: memory management

- 340 Kb of SRAM can be used for data, tasks, stack
- 8 Mb of SDRAM but needs controller initialization
- Memory allocation with '`new`' is limited to 24 Kb of SRAM
- No `System.Storage_Pools` with Ravenscar sfp profile

```
package Net.Buffers is
   type Buffer_Type is tagged limited private;
   ...
private
   type Buffer_Type is tagged limited record
      Kind : Packet_Type := RAW_PACKET;
      Size : Uint16 := 0;
      Pos  : Uint16 := 0;
      Packet : Packet_Buffer_Access;
   end record;
   ...
end Net.Buffers;
```

- How to allocate `Buffer_Type` from SDRAM?

https://github.com/stcarrez/ada-enet

~~Solution~~: memory management

- No good solution to use the SDRAM memory
- Can use SDRAM for buffers only (display, network buffers)
- Could initialize the SDRAM controller from bootloader or setup code (in `Setup_Pll`)

```
Addr : System.Address;
Size : Uint32 := 10 * Net.Buffers.NET_ALLOC_SIZE;
...
  Addr := STM32.SDRAM.Reserve (Amount => HAL.UInt32 (Size));
  Net.Buffers.Add_Region (Addr => Addr, Size => Size);
```

Conclusion

- Ada concurrency model helps having a clear design
- Ada pre/post conditions increases robustness
- Ada reduces debugging significantly
- But, having the sources is key to understand problems
- AdaCore's “Ada Drivers Library” is a killer