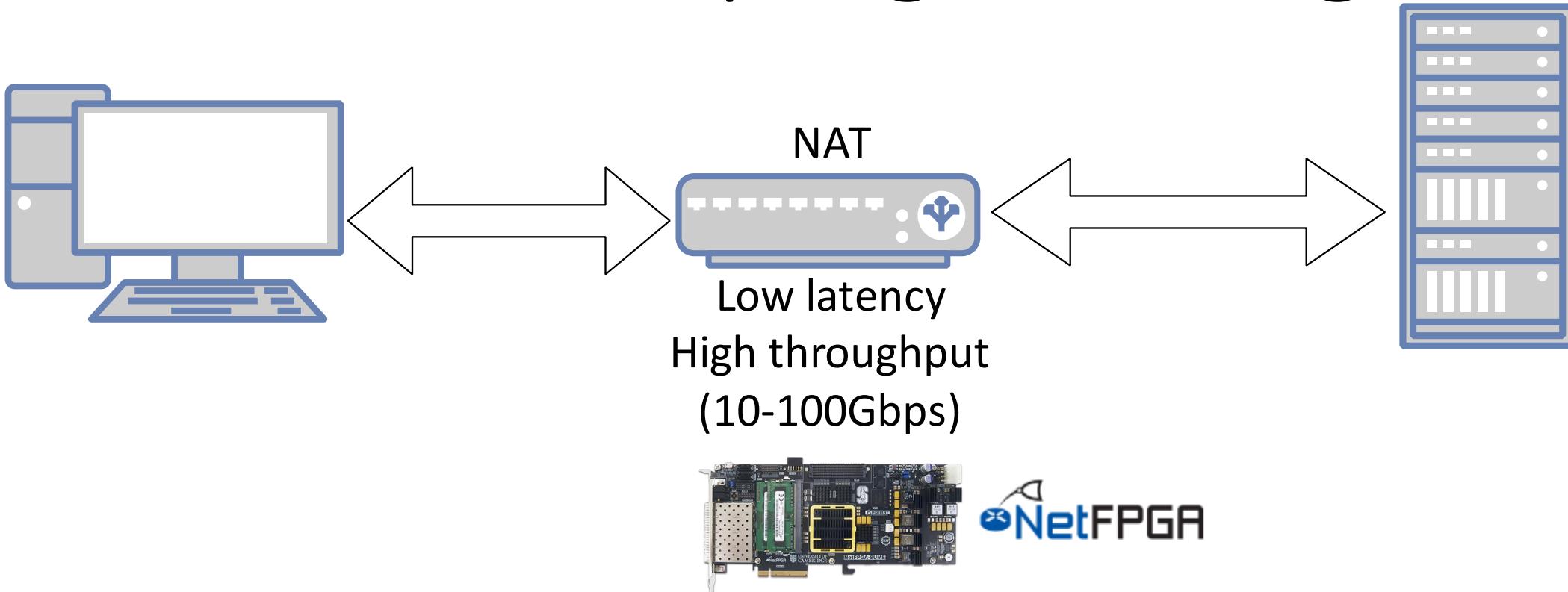
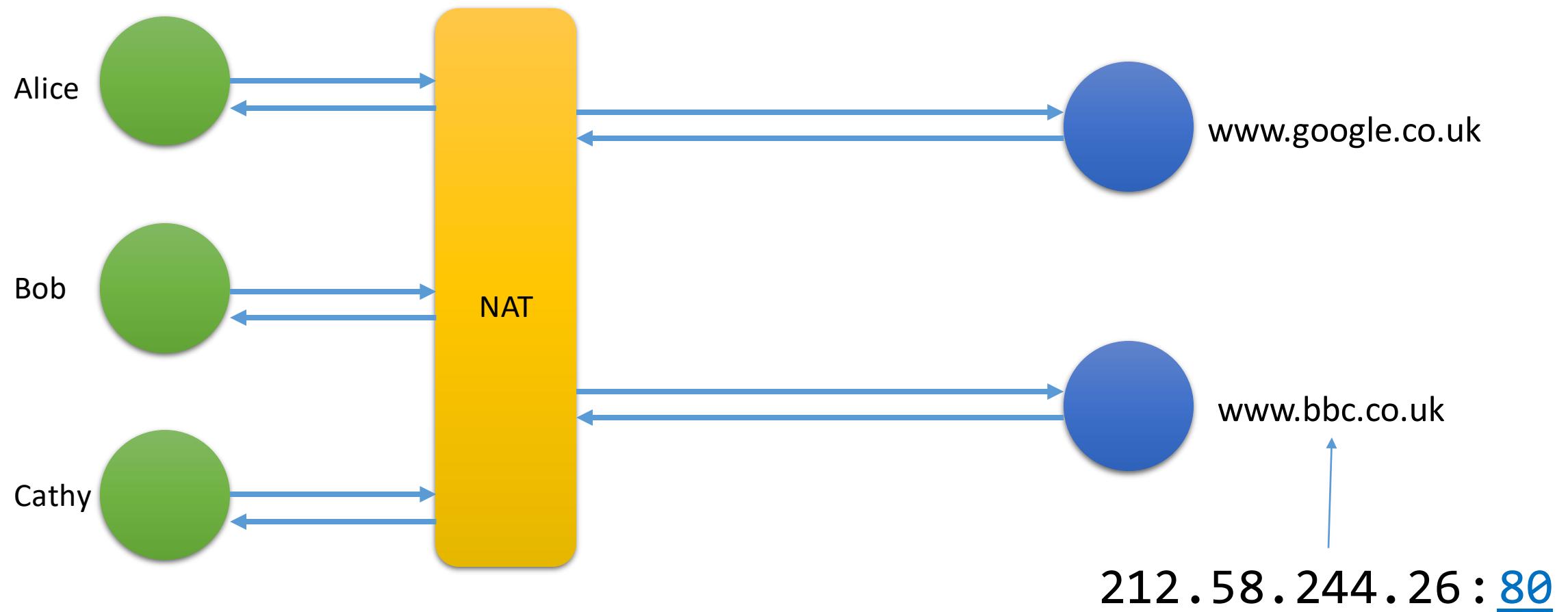


Using C# for high performance network programming

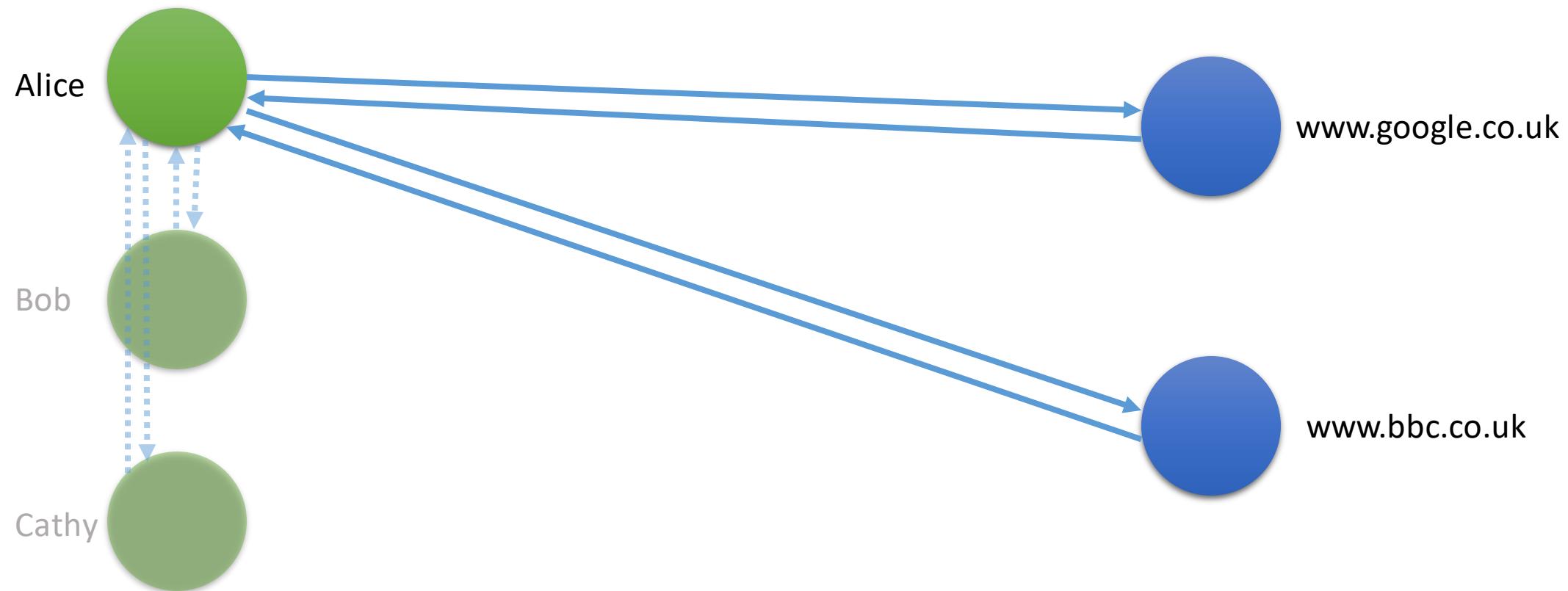


Jonny Shipton
CL, University of Cambridge

What is a NAT?



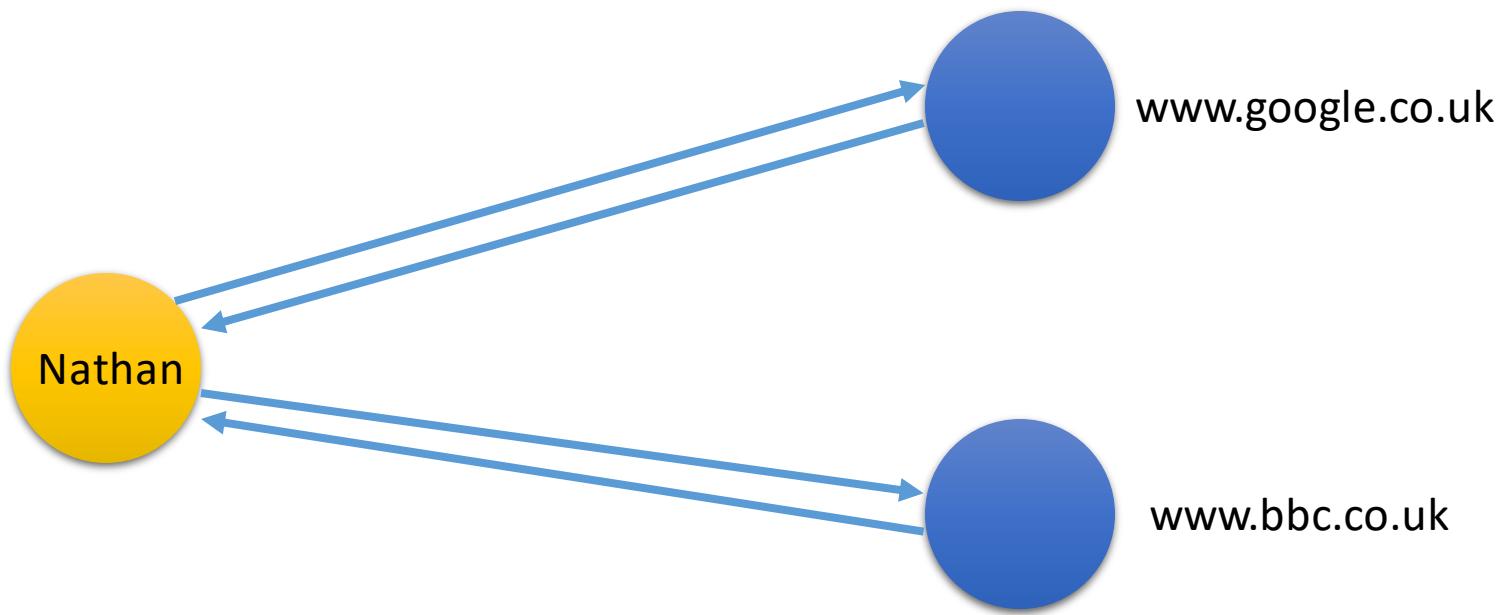
What is a NAT?



What is a NAT?

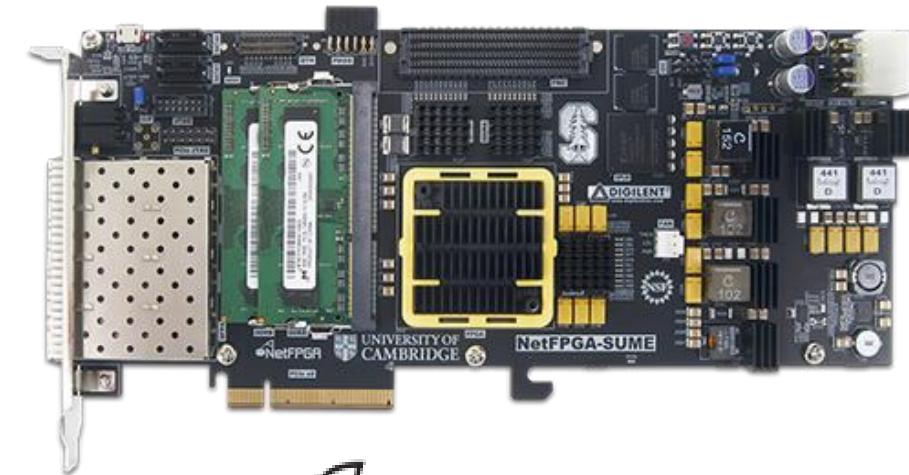


Server's POV



Why C#?

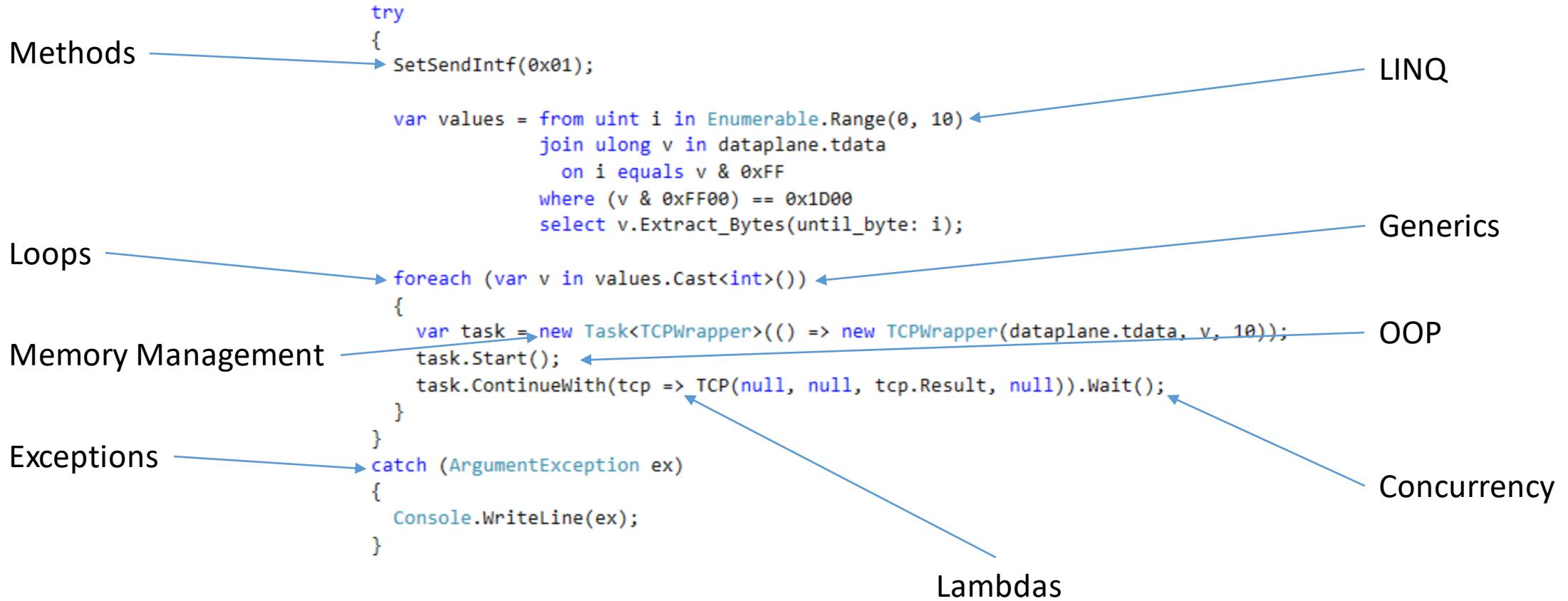
	Verilog	C#
Performance	✓✓✓	✓
Scalable to 100Gbps+	✓✓✓	✓
Skill base	Small	Large
Comfy language features	:(✓✓✓



 NetFPGA

The logo consists of a blue hexagon containing a white stylized 'X' or 'N' shape, followed by the word "NetFPGA" in a bold, sans-serif font.

Why C#?



Also iterators, properties, inheritance, libraries, dynamic variables, safe expressive type system, ...

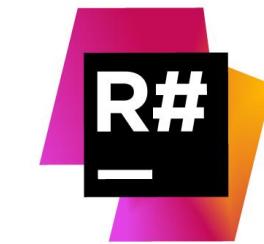
Why C#?

A screenshot of Microsoft Visual Studio showing a code editor with C# code for a NAT application. The code handles memory manipulation and offset calculations. Below the code are several debugging windows: Locals, Call Stack, and Diagnostic Tools (Events, Memory Usage, CPU Usage). The Locals window shows variables like data, offset, value, length, and mask. The Call Stack shows the call hierarchy from Main() to various methods in EmuUtil.cs.

A screenshot of Microsoft Visual Studio showing the Unit Test Sessions window. It displays a tree view of test results for NUnit. A specific test, "Test_Trending_NUnit(2)", is highlighted as failed. The error message indicates an expected string length of 6 but found 13 at index 5. Other tests in the same class are also listed with their outcomes (Success or Failed).

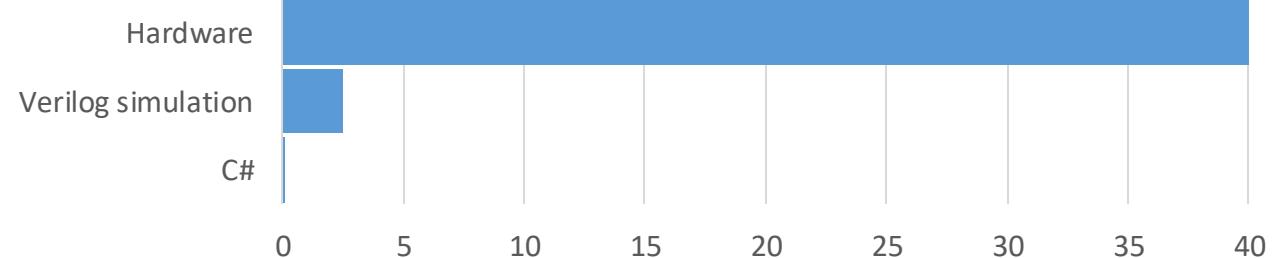


Visual Studio®

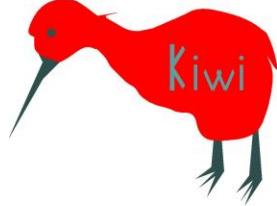


ReSharper

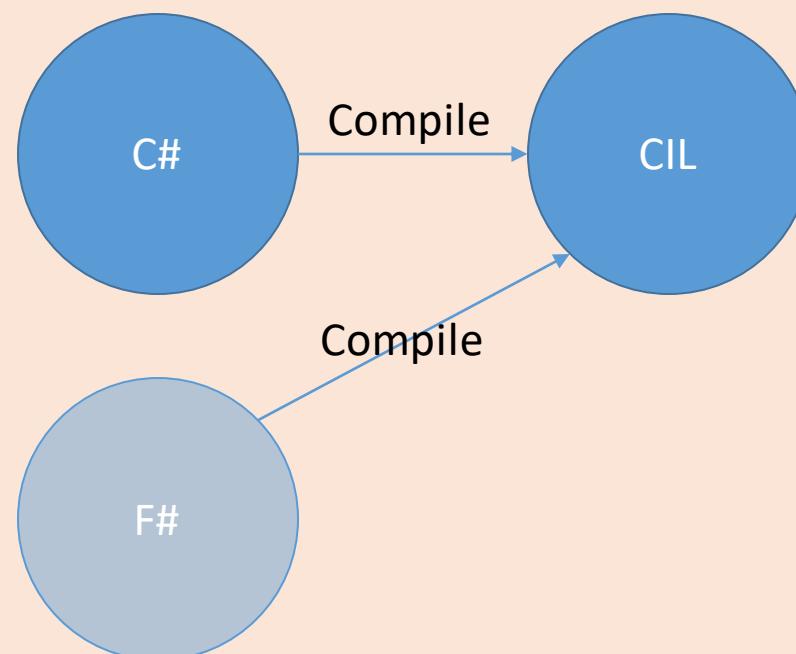
Time (mins) to compile and run



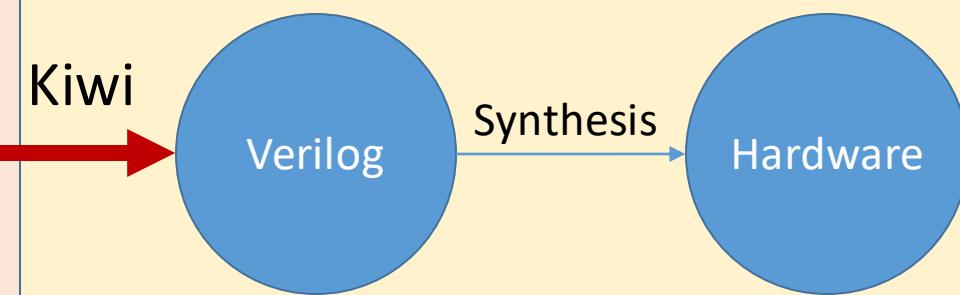
Compile C# to Verilog?!



Sequential/Synchronous Semantics



Concurrent Semantics



Look at the code

```
else if (eth.EtherType == EthernetWrapper.EtherType.ARP)
{
    if (arp.Operation == ARPWrapper.Op.Request
        && arp.HardwareType == ARPWrapper.HwType.Ethernet
        && arp.ProtocolType == (ushort)EthernetWrapper.EtherType.IPV4
        && arp.HardwareAddressLength == 6
        && arp.ProtocolAddressLength == 4)
    {
        Kiwi.Pause(); // Reduces Kiwi compile time
        if (arp.TargetProtocolAddress == myInsideIP || arp.TargetProtocolAddress == myOutsideIP) // Request for a MAC I own
        {
            uint intfIp = arp.TargetProtocolAddress;
            ulong intfMac = macOfIntf(dataplane.GetReceiveIntf());

            // Update arp packet to use as response
            arp.Operation = ARPWrapper.Op.Reply;
            arp.TargetHardwareAddress = arp.SenderHardwareAddress;
            arp.TargetProtocolAddress = arp.SenderProtocolAddress;
            arp.SenderHardwareAddress = intfMac;
            arp.SenderProtocolAddress = intfIp;

            // Update ethernet fields
            eth.DestinationMac = arp.TargetHardwareAddress;
            eth.SourceMac = intfMac;

            Kiwi.Pause();

            // Send reply
            SetSendIntf(dataplane.GetReceiveIntf());
            SendFrame(pkt_size);
        }
    }
}
```

Look at the code

The diagram illustrates annotations on a snippet of C# code. A blue arrow labeled "Property" points from the word "Operation" in the first line to the declaration of the property. Another blue arrow labeled "Object wrapping raw data array" points from the word "Operation" in the condition of the if-statement to the line "arp.Operation = ARPWrapper.Op.Reply;". A third blue arrow labeled "Intuitive enum" points from the word "Op" in the enum type to the same line.

```
if (arp.Operation == ARPWrapper.Op.Request ...  
...  
arp.Operation = ARPWrapper.Op.Reply;
```

```
public Op Operation { get { return (Op)data.Get16(Offset + 6); }  
                     set { data.Set16(Offset + 6, (ushort)value); } }
```

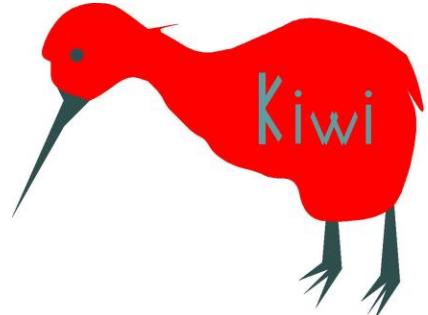
Limitations

- No truly dynamic allocation (yet)
 - Array sizes must be determined statically
 - Objects can be allocated and used within loops
- Library support less mature
- Complexity -> longer compile times
- No reflection, dynamic invoke, etc.
- Clock cycles are more precious, so we want to do more in each one
 - Asynchronous – Kiwi flattens code

Thanks to:

- Nik Sultana
- Salvator Galea
- David Greaves
- Networks-as-a-Service
(Naas-project.org)
- EPSRC
- Kynesim

Kiwi



Google “Kiwi compiler”

<http://www.cl.cam.ac.uk/~dij11/kiwi/>

<http://www.cl.cam.ac.uk/research/srg/han/hprls/orangepath/kiwic.html>



NetFPGA-SUME

<http://netfpga.org/site/#/systems/1netfpga-sume/details/>