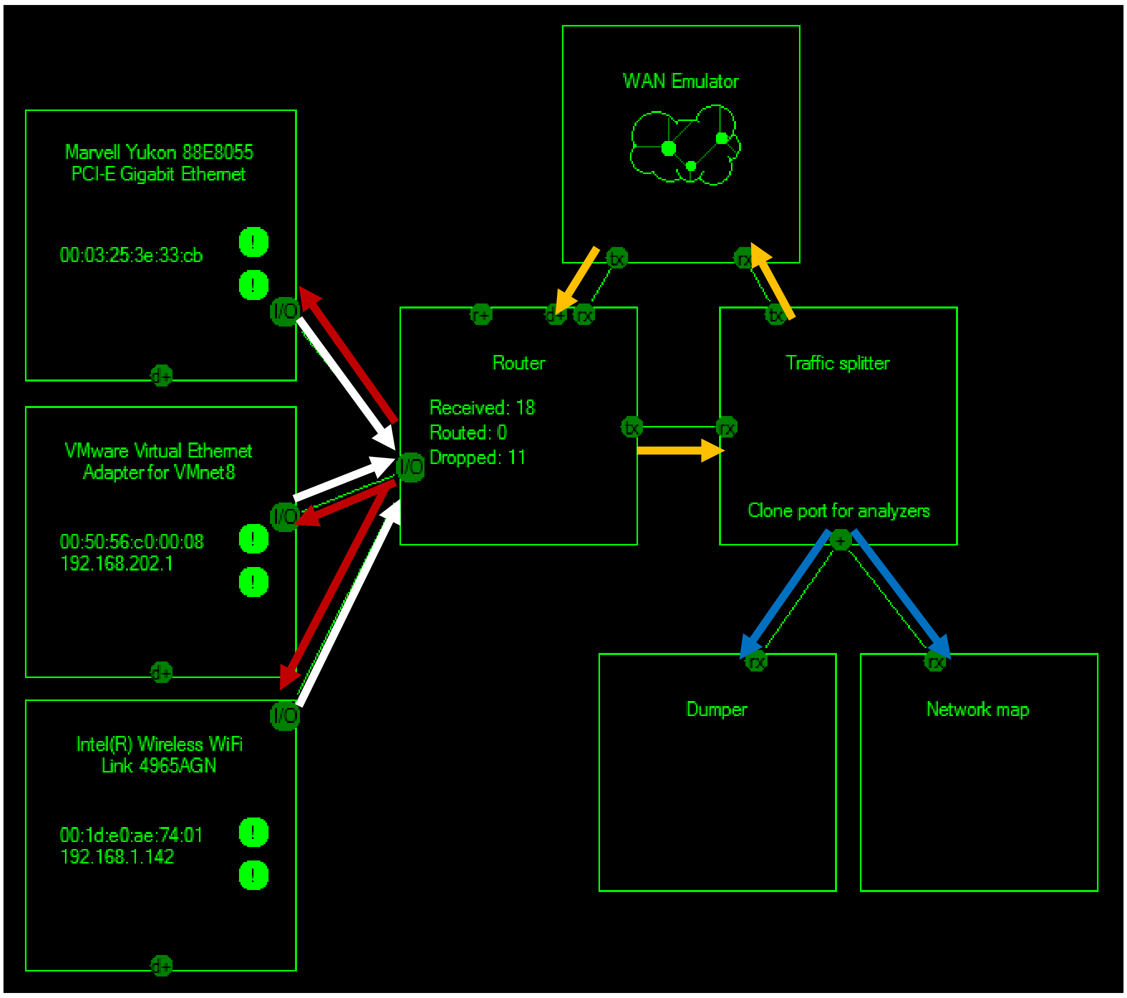
EXTENDING THE ROUTER

After understanding frames and traffic handlers it is very important to understand the flow of traffic. This document will explain how to extend a simple system like a router with some analyzers or traffic modifiers. Also this document will describe how this systems work and how the traffic will flow.

UNDERSTANDING TRAFFIC FLOWS

The Image below illustrates this flowing of traffic. Can you find the components we used in our router? There are several new components now. For this example we will add a traffic modifier, the WAN Emulator, which will simulate some WAN features like traffic loss and delay jitter. But you can also use another traffic modifier instead. Further we will add a few traffic analyzers, including a dumper, which will save all traffic for us and a network map, which will find several information about hosts around us for us.

As you can see in the Image below, traffic flows from the interface to the router (white arrows) and the router forwards the joined traffic streams to the traffic splitter (orange arrow). The traffic splitter then clones the frame once. This is very important to keep in mind as this is the reason why traffic analyzers must not modify any property of the frame because the traffic splitter forwards the one cloned frame to all traffic splitter simultaneously and modifying a frame while another traffic splitter can reads or also modifies it could cause racing conditions and serious program errors. The other instance will be forwarded to the next traffic modifier connected with the out port of the traffic splitter, even if it is the router like in the first lesson (YourFirstRouter). Because the frame is cloned, the next traffic analyzer can modify the frame without affecting any of the traffic analyzers.



The WAN emulator then will forward the frame to the router which will send it out to the according interface (white arrow). The only section where a single frame is sent to multiple handlers is when the traffic splitter sends frames to traffic analyzers (blue arrows). At each other circumstance (red, white and orange arrows), the frames are only edited or viewed by one traffic handler at one time. Keeping this in mind, it is possible to build very complex scenarios including conditional splitters, which split up traffic streams according to frame properties and other components.

BUILDING AN EXTENDED ROUTER

Adding new components to an existing system is not very difficult. If you worked through the first How-to-guide, you have already tried to link components together.

Adding new components to our existing router is easy. You just have to create the new traffic handles, then link them into the graph and then update the shutdown routine. Maybe we could also add some event handlers to generate some output.

```
LibCapDumper lcpDumper = new LibCapDumper();
NetMap nmMap = new NetMap();
WANEmulator wanEmulator = new WANEmulator();
```

```
lcpDumper.Start();
nmMap.Start();
wanEmulator.Start();
```

Add this two code snippets to your router’s code at the corresponding places (where the traffic handlers are created and where they are started). After creating and starting the handlers, we have to set some handler specific properties and link some events to get some output.

```
//Set traffic dumper properties
lcpDumper.StartLogging(Path.Combine(System.Environment.CurrentDirectory,
"Dump " + DateTime.Now.ToLongDateString()), false);
```

The above code segment should be pasted anywhere while setting properties or linking the graph. It causes the LibCapDumper, a traffic dumper which dumps traffic to a file readable by Wireshark, to start logging in the given file (CurrentDirectory\Dump + Current Date).
You can also set some settings for the WAN Emulator and play around a little bit.

Also it could be very useful to add an event handler to the NetMap to see some output when new hosts are found.

```
nmMap.HostInformationChanged += new
NetMap.HostChangedEventHandler(nmMap_HostInformationChanged);
```

```
static void nmMap_HostInformationChanged(HostInformationChangedEventArgs
args, object sender)
{
    Console.Write("Host found: " + args.Host.Name + " ");
    if (args.Host.IPAddresses.Count > 0)
    {
        Console.Write(args.Host.IPAddresses[0].ToString() + " ");
    }
    Console.WriteLine(args.Host.Type.ToString());
}
```

This code snippet will notify us about found host information trough the console.

The next step is to link the traffic handlers accordingly together. We have to set the WAN emulator as the output handler of the traffic splitter and the router as the output handler of the WAN emulator. Further we have to add the dumper and the network map to the analyzer list of the traffic splitter, as you can see in the image one page before.

```
//Let the router forward traffic from the interfaces to the traffic
splitter
rRouter.OutputHandler = tsSplitter;
//Let the traffic splitter forward received traffic to the WAN emulator
tsSplitter.OutputHandler = wanEmulator;
//Let the WAN emulator forward received traffic back to the router
wanEmulator.OutputHandler = rRouter;
//Let the traffic splitter clone each frame and send it to the traffic
dumper and the NetMap
tsSplitter.AddTrafficAnalyzer(nmMap),
tsSplitter.AddTrafficAnalyzer(lcpDumper);
```

The only thing which is left to do now is completing the shutdown routine. Simply add the Cleanup() and Stop() statements like described in YourFirstRouter. This is left as an exercise to you. If you get stuck, you can always see the example project.

Now it’s time to test our extended router.