Tiny Nets: Project Report One
A Small Network with Industrial Dreams

Jake Read, Dougie Kogut, Nick Selby, Patrick Wahl

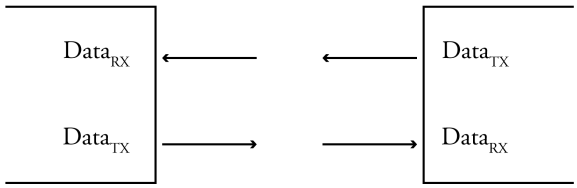
October 2017

1 TinyNet Implementation

Here we will briefly describe how TinyNet is implemented physically, how it structures packets, and how switches in TinyNet route messages.

1.1 Our PHY

We use UART Hardware to link nodes. UART is a ubiquitous micro-controller peripheral - at least one port can be found on almost any micro-controller on the market today, and often many more are present. The micro-controllers we have begun building with have five ports, and micro-controllers in the same family have up to nine ports, allowing for the development of radically interconnected graphs. By using UART Hardware, we can bring nearly any embedded system onto the network with only two micro-controller pins and a software library. UART is full-duplex, and requires little configuration.



UART Hardware:

Ubiquitous microcontroller peripheral. Uses 8-bit words on a full-duplex connection, typically modulated at the microcontroller's logic level.

1.2 Our Packet

As opposed to Ethernet's minimum size of 672 bits, we implement a packet with a 48 bit minimum. This is of primary importance for driving message delivery times down, and we expect to approach Ethernet's delivery time without any of the purpose built ICs and electronics that are found in Ethernet switches and endpoints.
For Ethernet, the Time to Transmit a message,

$$T_m = \frac{bits/packet}{bitrate}$$

$$T_{m_{ethernet}} = \frac{672}{100*10^6} = 6\mu s$$

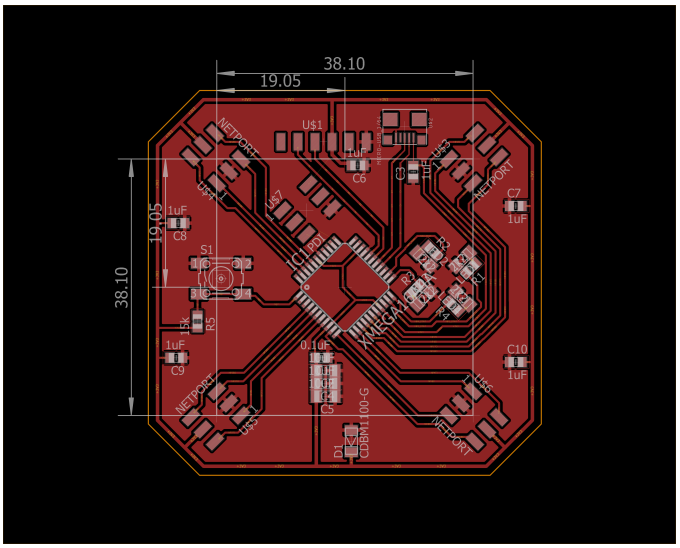


Figure 1: A network switch, implemented on an XMega Microprocessor, $5 total hardware cost.

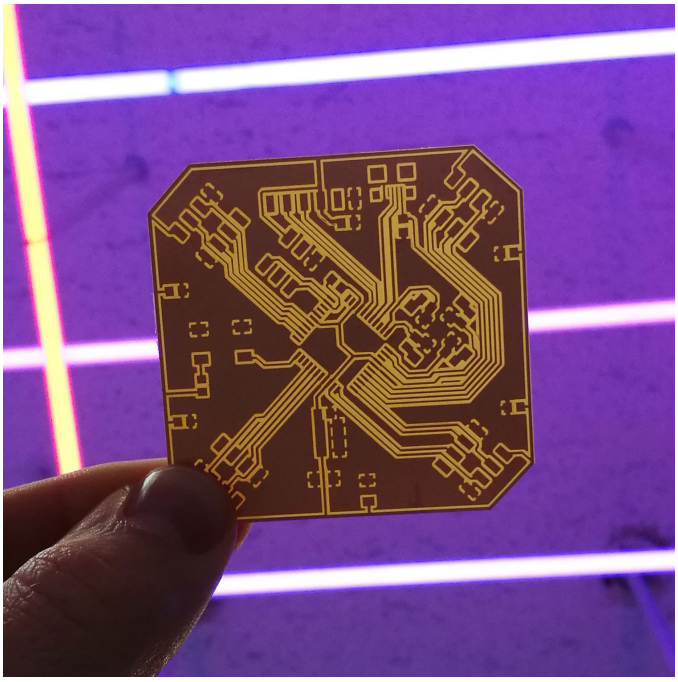


Figure 2: The board, fabbed.

We use UART Hardware with a 2Mbps delivery rate, that we may be able to increase to 4Mbps.

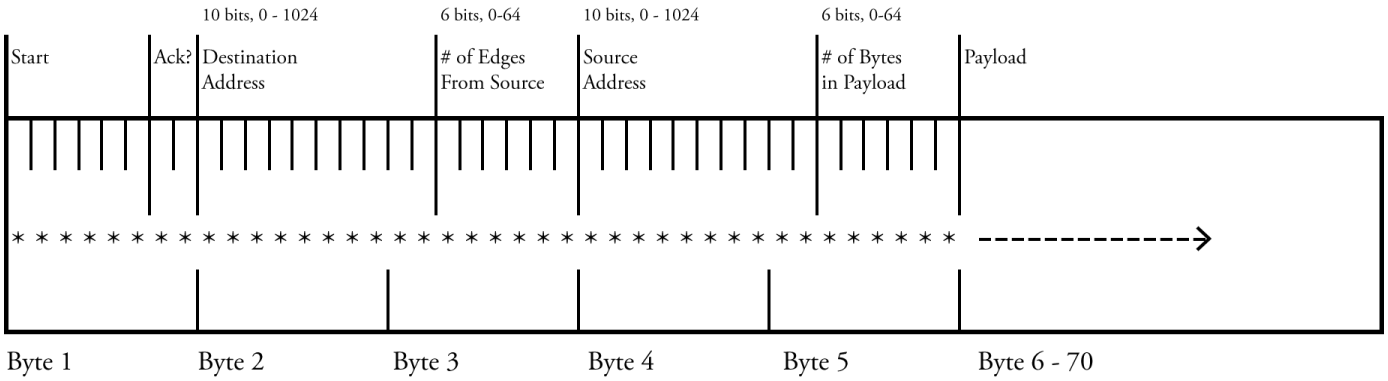


Figure 3: A TinyNet Packet. A Start Byte, Destination and Source Addresses, as well as the Number of Edges the Packet has Crossed prior to arriving at the switch, the Number of Bytes in the Payload, and of course the Payload itself.

$T_{m_{uart}} = \frac{48}{2*10^6} = 24\mu s$

It is clear to see that if time were spent to engineer a higher bitrate into the physical layer, a new network architecture using radically smaller packets could deliver a major improvement in delivery time. We are interested in using FPGA technology in order to implement higher bitrates on our network.

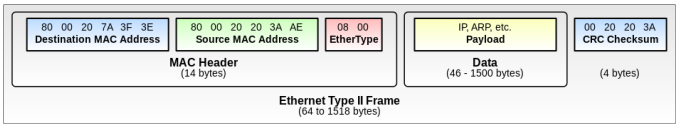


Figure 4: An Ethernet Packet.

Our Packet contains destination and source information, a payload, and critically, 6 bits describing the number of edges it has crossed before arriving at the current switch. These additional 6 bits are critical in allowing the switch to make informed decisions about where to deliver the packet next, as will be discussed in the following subsection.
push addresses to 16 bits, do one byte for all routing information.

1.3 Our Routing Protocol

Switched Ethernet's Spanning Tree Protocol is designed to eliminate all but one possible route to any given endpoint. While this increases network simplicity, it does not allow for any dynamic routing, and promotes 'star' topologies rather than truly distributed networks. When an Ethernet switch is occupied with an incoming transmission, other nodes that share the switch as a gateway to the network must wait before they are passed along.

We see this as a critical limitation to deterministic message delivery. In order to switch packets in a dynamic manner, we push a small amount of routing information into the packet itself, in the form of a count of the edges that that packet has traversed. The Edge Count value begins at zero when the packet departs from it's source, and each switch in series increments the value by one.

To begin explaining how we expect to route packets, we offer this simple example network.

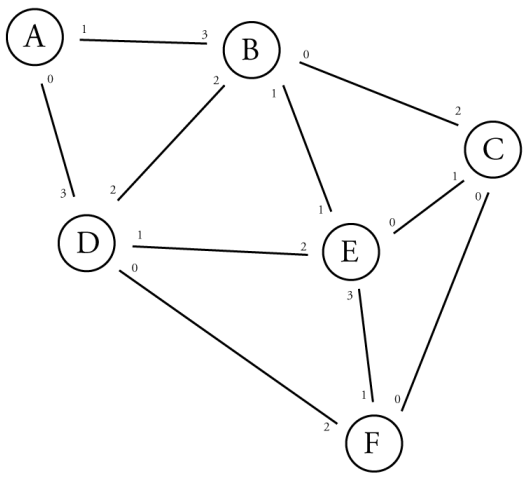


Figure 5: A example TinyNet graph. Switches have addresses (A, B, ... F) and Ports (E_0, ... E_4).

1. **Routing Tables:**
   Much like Ethernet, switches in a TinyNet build routing tables. Each switch has a table with a list

An Ideal Address Table from Node E's Memory

Address	Port	Min. Edges	
0			
.			
.			
.			
A	0	3	
	1	2	
	2	2	
.	3	3	
.			
.			
B	0	2	
	1	1	
	2	2	
.	3	3	
.			
.			
C	0	1	
	1	2	
	2	3	
.	3	2	
.			
.			
.			
.			
.			

Figure 6: An ideal table for Node E from Figure **??**.

of addresses, a list of the ports that packets from those addresses have appeared on, and data about the particular connection on that port - namely, how many edges a packet crossed before it arrived on that port. Switches pull this data out of packets they see during operation. An example of an ideal routing table for our example graph is show in Figure 6

2. **Port Selection:**
   Data in this table is used when a switch receives a packet. For example, if Switch E has previously received a packet from A on Port E_1 with an edge count of 2, as well as a packet from A on Port E_0 with an edge count of 3, it can select port E_1 as the ideal port to transmit a packet that is destined for A.

3. **Back Pressure Routing:**
   Channels are all full-duplex. When a switch begins transmitting, the recipient, upon receiving the start frame, can deliver an 'ok' or a 'busy' response. If the transmitter sees a busy response, or does not receive an 'ok', it can terminate transmission of the rest of the message and return to its lookup table to determine where the next-best switch to transmit on would be. It performs this lookup-and-test repeatedly, until it finds an available port. In this way, messages dynamically find a way through the network, rather than simply waiting for a switch to become available.

4. **Flood Forwarding and Discovery:**
   In the case that a switch receives a packet with a destination address that is not yet contained in its table, it forwards the packet to all other neighbouring ports, so long as the packet has not already travelled over a certain number of edges (this prevents packets ringing through the network).

5. **Acks:**
   When packets contain an Ack flag (01) as opposed to a no-ack flag (11), they return to their sender with an 'is-ack' flag (00) and no payload. Ack Requests and Flood Packets allow new nodes to discover network topology, by sending flood packets with ack-requests, and recording the responses.