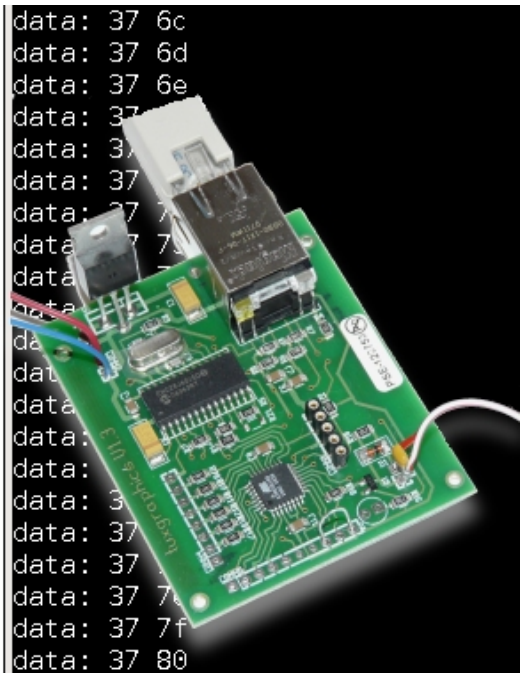
Fast realtime communication over ethernet



Abstract:

The tuxgaphics avr ethernet board can be used for very fast time critical communication. You can read sensor data or control io-ports with less than a millisecond delay.

The trick is to remove the TCP/IP overhead and use raw ethernet frames.

A Linux application using RAW network sockets is controlling this.

The idea

A few weeks ago I received from Rogier Schouten (http://www.rogiershikes.tk) a very interesting email.

Hi Guido,

I have a project, based on your "AVR-based ethernet device", that might interest you. To my surprise, Ethernet+AVR in hobby projects only seems to be used for web-based or network communications. In industry, Ethernet is more and

more used for fast I/O (i.e. on a cable with only master PC and slave devices). You can look up e.g. the "EtherCAT" standard if you like.

With this in mind, I changed your code for faster, possibly real-time communication between PC and AVR. I dropped support for higher-level protocols such as UDP, and in return there is faster communication, suitable for e.g. controlling a robot directly from the PC. It is a great replacement for e.g. the RS232 link that I used for such purposes. The code described here allows for sending a packet AND receiving the reply in under 320 microsec. This allows for about 3kHz updates. Using a real-time extension for linux (such as Xenomai+RTnet), this communication can be made real-time (i.e. supporting deadlines).

This is an application that I really missed on the tuxgraphics website, so I give you the code and you can do what you want with it.

Regards,
Rogier

I looked at Rogier's code (download at the end of this article) and what he did is this: He reduced the overhead for packet communication to almost zero by removing all the protocol layers except plain IEEE 802.3 ethernet and he minimized the communication between ethernet controller chip (enc28j60) and avr microcontroller. To use this you need a direct straight ethernet cable between avr ethernet board and a Linux PC (a direct cable and not a cross over cable is needed because the tuxgraphics ethernet board has a auto-MDIX jack).

His linux control application does not necessarily need a real-time extension for linux. Any linux PC will do. You just have to run it as root and you can still get communication packets flying between PC and avr ethernet board in fractions of milliseconds intervals. The ethernet cable can be 100m long.

Changes to run the data over a small switched LAN connection

Rogier's code is optimized for speed but can not be used with a LAN in-between. It has to be a dedicated cable. The reason is that reply packet is sent back to the broadcast address FF:FF:FF:FF:FF:FF. This was done because setting the correct address would cost time.

I wanted to see if it can be changed to add a bit more "networking overhead" and be able to run it over the normal office LAN. So the code presented as part of this article can be run safely on a LAN used by other PCs as long as the LAN is not carrying too much traffic and there must be only switches in the LAN connection between controlling PC and avr ethernet board. The advantage would be that a PC with only one network card can communicate in real-time with the avr ethernet board and still use the normal office network at the same time.

In the download section for this article you have the choice to select whatever is more suitable for your purpose. The original "rogier" code which is optimized for speed and the slightly slower code which can run over LAN switches.

Switches, Hubs and WIFI-routers

10 years ago Hubs where the state of the art interconnection points on a LAN. A Hub is just an amplifier and it does not look at the data in the packet. Today it is even difficult to buy a Hub. Most boxes, even for small office applications, are LAN-switches. A switch looks at the ethernet frame and sends it only out on the ports where the communicating equipment is connected. Switches should send any traffic for which they don't know where to send it too just to all ports but it seems that some manufactures assume that the only packet traffic is IP. They just drop packets when they don't know where to send them to.

Communication on a local LAN in the IP world starts with an ARP request. That is a "search for the MAC address" of a given IP address. The ARP request is sent to the broadcast MAC address FF:FF:FF:FF:FF:FF. The IP host owning the IP address replies with its own MAC address. By looking at the ARP requests and replies the LAN switches can learn where the hosts in the network are connected and to which ports to send the packets to.

In our case we don't use IP addresses. Everything is based on ethernet and Mac-addresses but to be able to work with those broken LAN switches (or optimized LAN switches as the manufacturer would probably call them) we need to let the LAN switch know what the MAC address of the ethernet board is before sending actual traffic. This is called gratuitous ARP request. This is a spontaneous ARP request and we fill in some dummy IP address (=never used) just to teach the switch what our MAC address is. The gratuitous ARP is sent at startup of the board as soon as the link comes up.

While working with this ethernet realtime communication software I learned also something about WIFI-routers. I always thought that they are just switches inside the LAN and the routing function comes only in place when sending data to the Internet. This is not the case. At least for my netgear WIFI router. It does not switch any traffic between wired LAN and WIFI LAN unless those are ethernet frames containing IP traffic. In other words plain ethernet communication does not work over WIFI networks of this kind. It works only over HUBs and LAN switches.

Summary: what is needed to run it over the LAN

In order to send plain ethernet frames over the LAN without flooding the network we need two things:

- Use only unicast traffic with proper source and destination MAC addresses (no FF:FF:FF:FF:FF:FF)
- Send a gratuitous ARP

The packet format

The eth_com_realtime-X.Y software, that's what I called the LAN capable code, uses the following packet format:

```
| - - - - - - | - - - - - - | - - | - - - |
 dest MAC       source MAC    len   data
```

In the example code we use 3 bytes of data. The actual ethernet frames are much longer but the extra dummy data is added automatically by ethernet chip to be 802.3 compliant and we don't have to worry about it.

What eth_com_realtime-X.Y does

The data is 3 bytes long. You can make it much longer if you want, up to 254 bytes but this is just an example application and you are expected to modify it as needed.

- Byte zero in the data field is a sequence number. The ethernet board just sends back what it gets from the Linux PC. This is to be able to detect missing frames.
- Byte one is used to set the 8 output pins of PORTD according to the bit pattern sent. You can use this to switch something on or off, e.g control motors, lights, ...
- Byte two is just a constant at this moment. Use it if you want.

The ethernet board replies back to each packets that it gets and fills the 3 bytes of data as follows:

- Byte zero is the unmodified sequence number
- Byte one represents the state of the IO-Pins on port C. You can use it to read up to 8 digital lines at once. Note: the DIP version of atmega168 has only 6 IO-Pins on port C.
- Byte two is just a counter to send some data.

The application on the linux PC sends 256 packets in a row and the bytes one and two as received are printed to the screen. The printing to screen is at the moment the bottle neck. Everything is much faster without printouts but it is only an example application showing that 16 bits of data (or more) can be sent and received at high speeds over an ethernet link.

Conclusion

I really like it that people take existing open source code and build new things with it.

This example shows how fast communication can be (very little delay). Web-servers are nice because to offer a user interface without the need for a special application software or driver but it is a slow way of communication with a max. 10 to 50 web pages per second. Using the same hardware but raw ethernet speeds of several thousand control packets per second are possible. You could read sensor data and control any hardware in real time.

References/Download

- Software download area: Download of eth-com code
- The avr ethernet board is available in our online shop: shop.tuxgraphics.org

---

2009-02-22, generated by tuxgrparser version 2.56