My Dream Ethernet Interface

Geoff Collyer

Plan 9 Foundation

ABSTRACT

Most existing Ethernet interfaces have lots of registers that I, as a system programmer, have no interest in. This paper is a thought experiment in designing my ideal minimal interface.

1. Requirements

I don't write diagnostics, so I only care about production use and prefer that the hardware come up in an immediately-usable state. Diagnostic registers and those that control fancy features for jumping on passing bandwagons should be controlled by other, disjoint registers. These are the functions that an operating system needs:

- establish buffer rings or NVME-style submit and completion queues for input and output;
- read and set MAC addresses;
- enable and disable interrupts for received packet(s);
- start and stop transmission and reception; and
- notification of link speed.

Note that none of this requires the use of proprietary techniques nor interfaces, nor does it require software fiddling with PHYs nor EPROMs nor S?R?G?MII nor DMA settings. Auto-negotiation of speed and link state, and the like should happen automatically and without intervention from the operating system.

The interface should be cache coherent (including DMA) with all CPUs in the system.

Errors are generally uninteresting; Ethernet is not guaranteed reliable and higher-level protocols such as TCP and UDP will have to detect errors or loss.

Multicast filtering should be performed internally and not require the operating system to know the hash function used, if any. Broadcast packets should always be accepted (ARP must work).

If IP checksum offloading is performed, it *must* be thoroughly validated by automated means, such as the Spin verifier.[1] A zero checksum must be understood as 'no checksum'. Offloading must be optional. When it's wrong, it breaks things, and Intel has got this wrong in the past.

2. Strawman Proposal

Byte order of registers and descriptors is assumed to be little-endian, since that seems to be what we are cursed with. Integer types are those of Plan 9.

Each transmit and receive buffer needs a descriptor:

```
struct Desc {
    uvlong address;
    ushort length;
    ushort bytesread;
    uint   flags; /* bits: rx filled, tx sent */
};
```

The registers interface can be fairly brief:

```
struct Etherifc {
    uchar  magic[4];  /* "eth" */
    ulong  control; /* bits: rx go, tx go, speed, link, */
           /* promisc, rx intr enable, one-shot intr */
    uvlong rxring;       /* base address of rx Descs */
    ulong  rxrlen;
    uvlong txring;       /* base address of tx Descs */
    ulong  txrlen;
    uchar  macs[64][6];  /* many [um]*cast addresses */
};
```

`macs[0]` should contain the default MAC address and all others should be zero at reset.

References

1. Gerard J. Holzmann, *Using SPIN*.