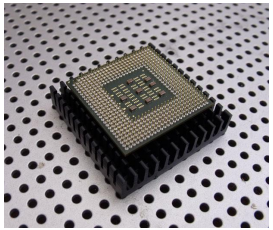
P4 and MACAW

COS 461 - Precept 10

Networking before the advent of programmable hardware

- Switching hardware influences network system.
  - What chips are available? What are the exposed primitives?
- Chip determines system capabilities.
  - Probably has a bunch of unnecessary stuff.
- Hampers improvement of systems.

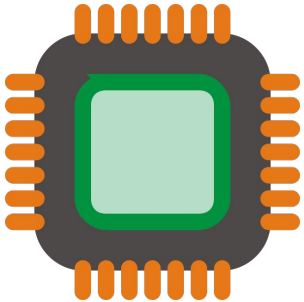


Dictates

Network switch functionality

What we want the network switch to do.

Programmable hardware and P4



P4 supported switch

Network switch functionality

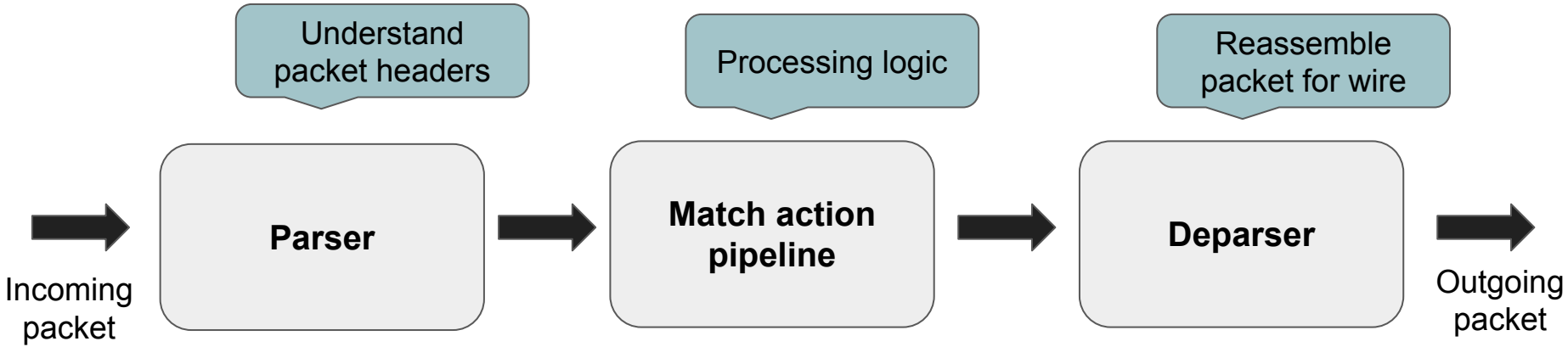
influences

What we want the network switch to do.

Examples of P4 applications

- Layer 4 load balancer
- In-network caching
- In-network telemetry
- In-network DNS
- Aggregation for MapReduce applications
- And so on..

P4 program architecture



P4 - example of the Parser state machine

```
state start {
        pkt.extract(p.ethernet);
        // These are set appropriately in the TopPipe.
        user_metadata.do_dns = 0;
        user_metadata.recur_desired = 0;
        user_metadata.response_set = 0;
                user_metadata.is_dns = 0;
                user_metadata.is_ip = 0;


        digest_data.flags = 0;
        digest_data.src_port = 0;
        digest_data.eth_src_addr = 0;


        transition select(p.ethernet.etherType) {
                        0x800: parse_ip;
                        default: accept;
                }
    }
```

```
state parse_ip {
        pkt.extract(p.ipv4);

                user_metadata.is_ip = 1;
                transition select(p.ipv4.proto) {
                        17: parse_udp;
                        default: accept;
                }
        }

        state parse_udp {
        pkt.extract(p.udp);


                transition select(p.udp.dport == 53 ||
p.udp.sport == 53) {
                        true: parse_dns_header;
                        false: accept;
                }
        }
```

P4 Hello World - connect port 1 to port 2

```
#include <core.p4>
#include <v1model.p4>
struct metadata {}
struct headers {}

parser MyParser(packet_in packet,
   out headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {

    state start { transition accept; }
}

control MyVerifyChecksum(inout headers hdr, inout metadata
meta) {   apply {  }   }

control MyIngress(inout headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
apply {
        if (standard_metadata.ingress_port == 1) {
            standard_metadata.egress_spec = 2;
        } else if (standard_metadata.ingress_port == 2) {
            standard_metadata.egress_spec = 1;
        }
    }
}
```

```
control MyEgress(inout headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    apply {  }
}

control MyComputeChecksum(inout headers hdr, inout metadata
meta) {
     apply { }
}

control MyDeparser(packet_out packet, in headers hdr) {
    apply { }
}

V1Switch(
   MyParser(),
   MyVerifyChecksum(),
   MyIngress(),
   MyEgress(),
   MyComputeChecksum(),
   MyDeparser()
) main;
```

Example program from P4 tutorials

MACAW

“Almost WiFi”

Review: MACA

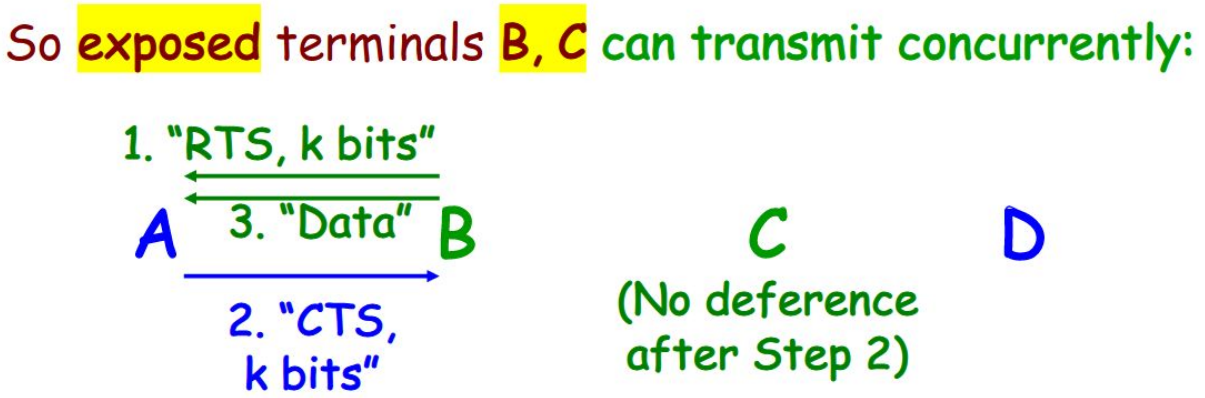
- Multiple Access with Collision Avoidance
- Method for solving Hidden and Exposed Terminal problems

– Solves hidden terminal problem

- When C hears CTS from B, C infers there is a hidden terminal (A) attempting to send to B. C defers to avoid collision.

Review: MACA

- Multiple Access with Collision Avoidance
- Method for solving Hidden and Exposed Terminal problems



- C hears RTS from B. C then waits until it should have heard CTS. If it doesn't, the receiver is a hidden terminal, and C can send to D without interfering with the transmission from B to A.