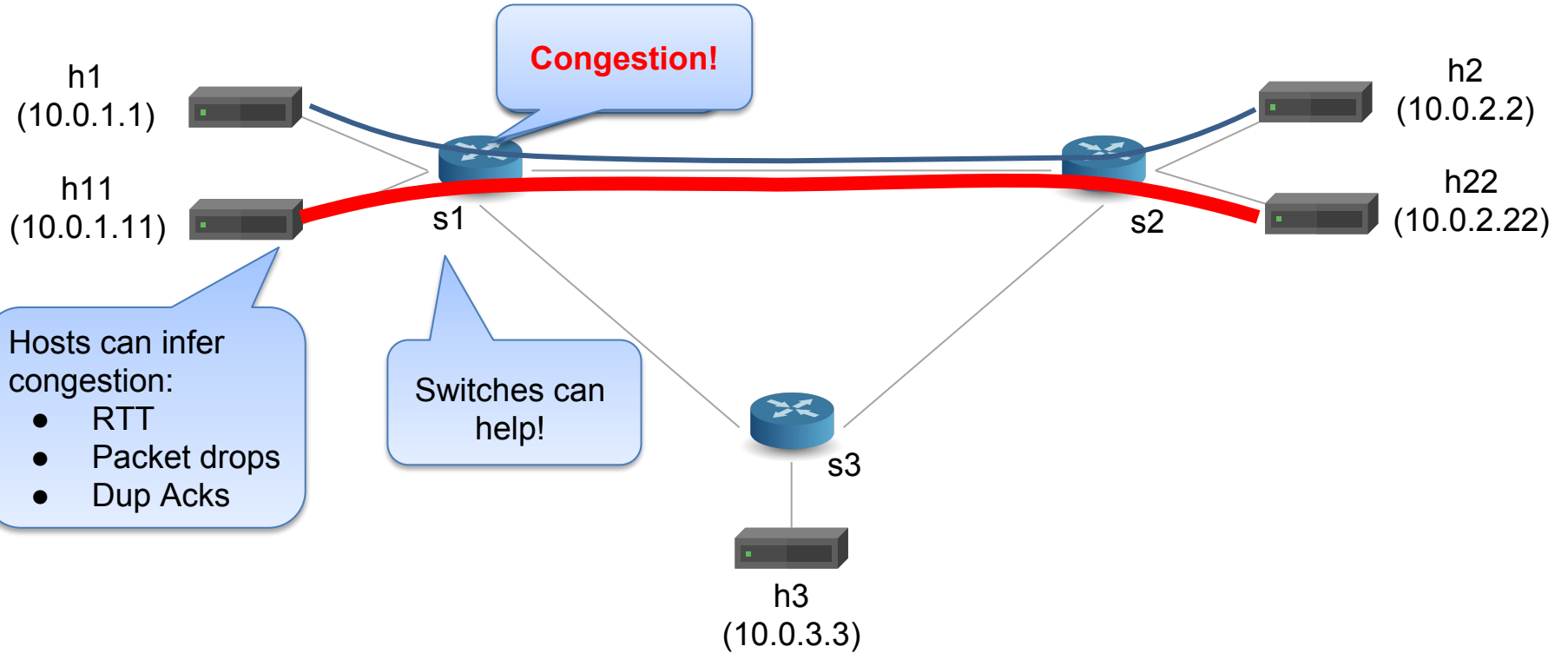


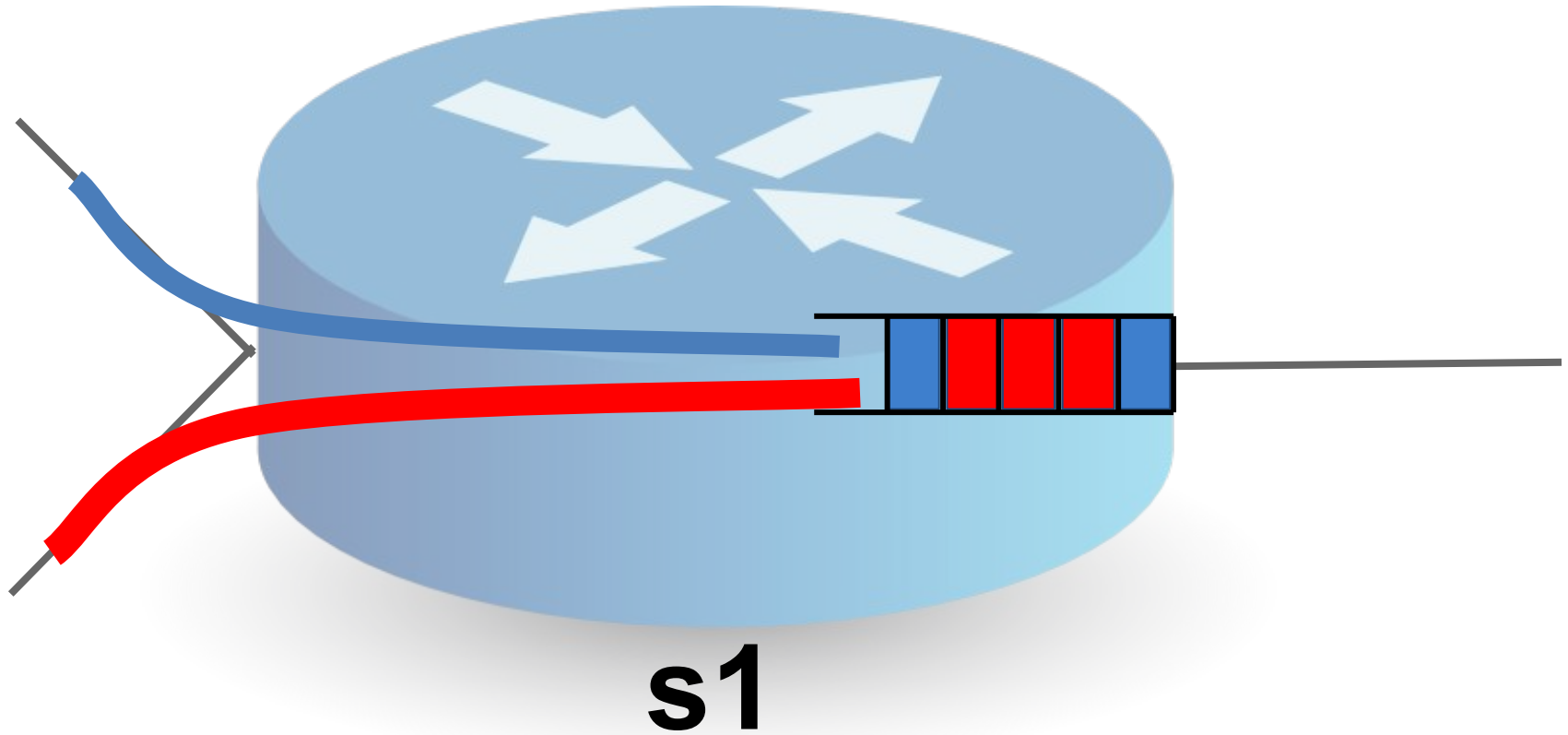
Lab 3: Monitoring & Debugging



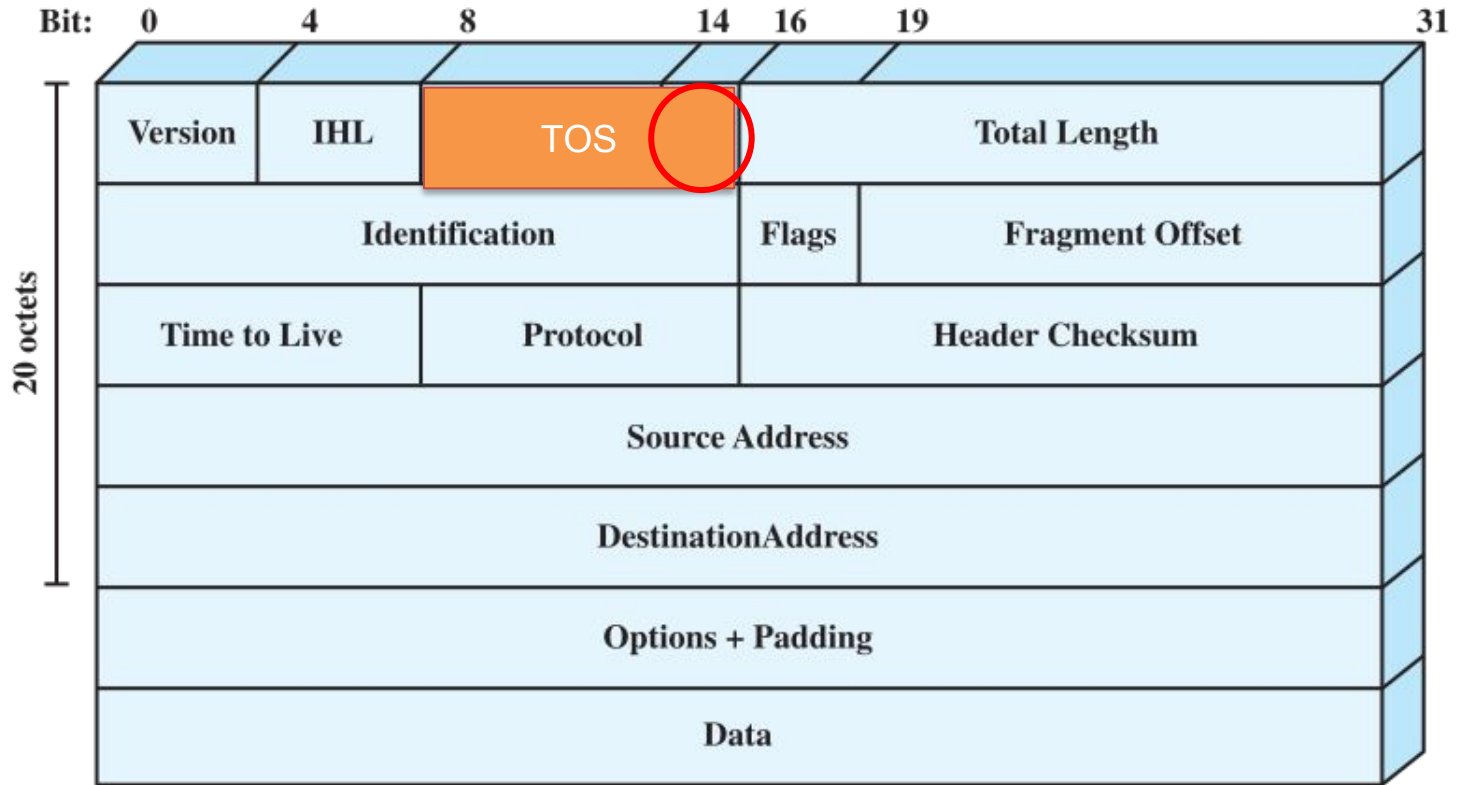
Monitoring & Debugging



Monitoring & Debugging



Explicit Congestion Notification



Explicit Congestion Notification

- **Explicit Congestion Notification**

- 00: Non ECN-Capable Transport, Non-ECT
- 10: ECN Capable Transport, ECT(0)
- 01: ECN Capable Transport, ECT(1)
- 11: Congestion Encountered, CE

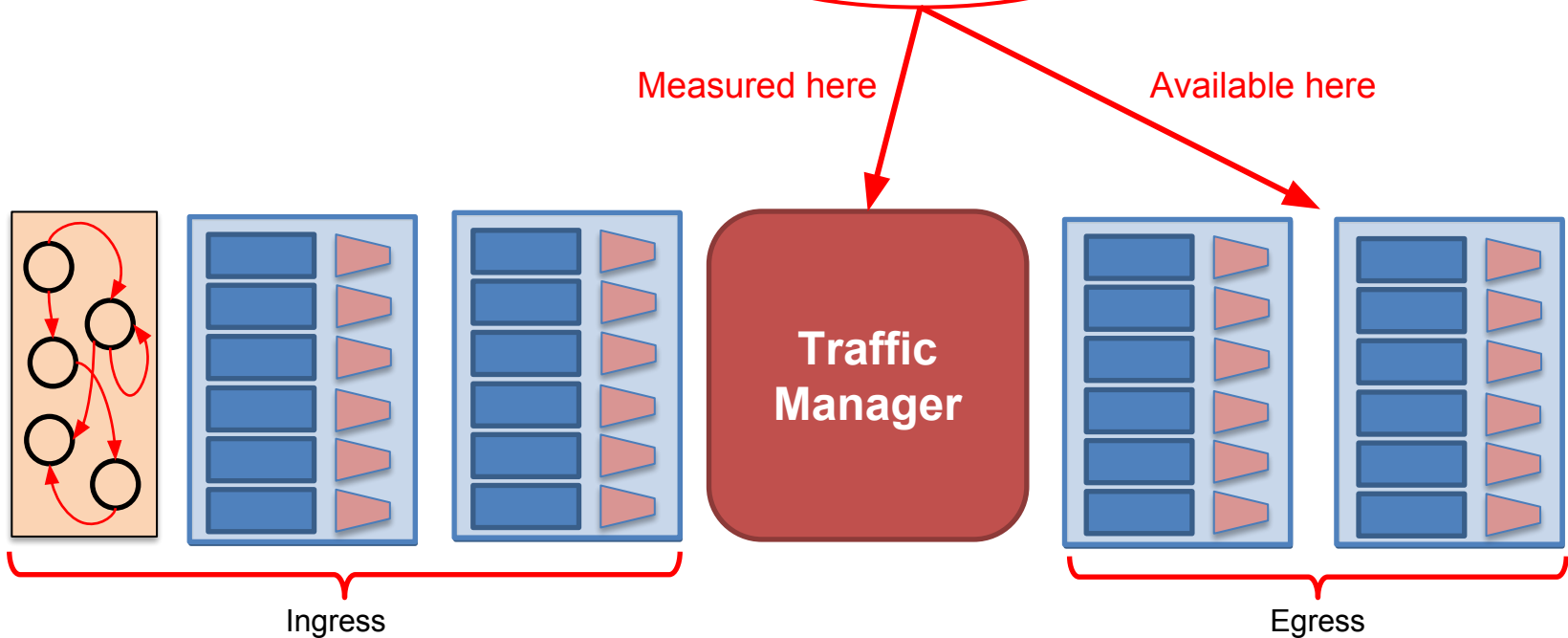
- **For packets originating from ECT, ECN-capable switches set the CE bit upon congestion**

- E.g., observed queue depth > threshold



Explicit Congestion Notification in P4

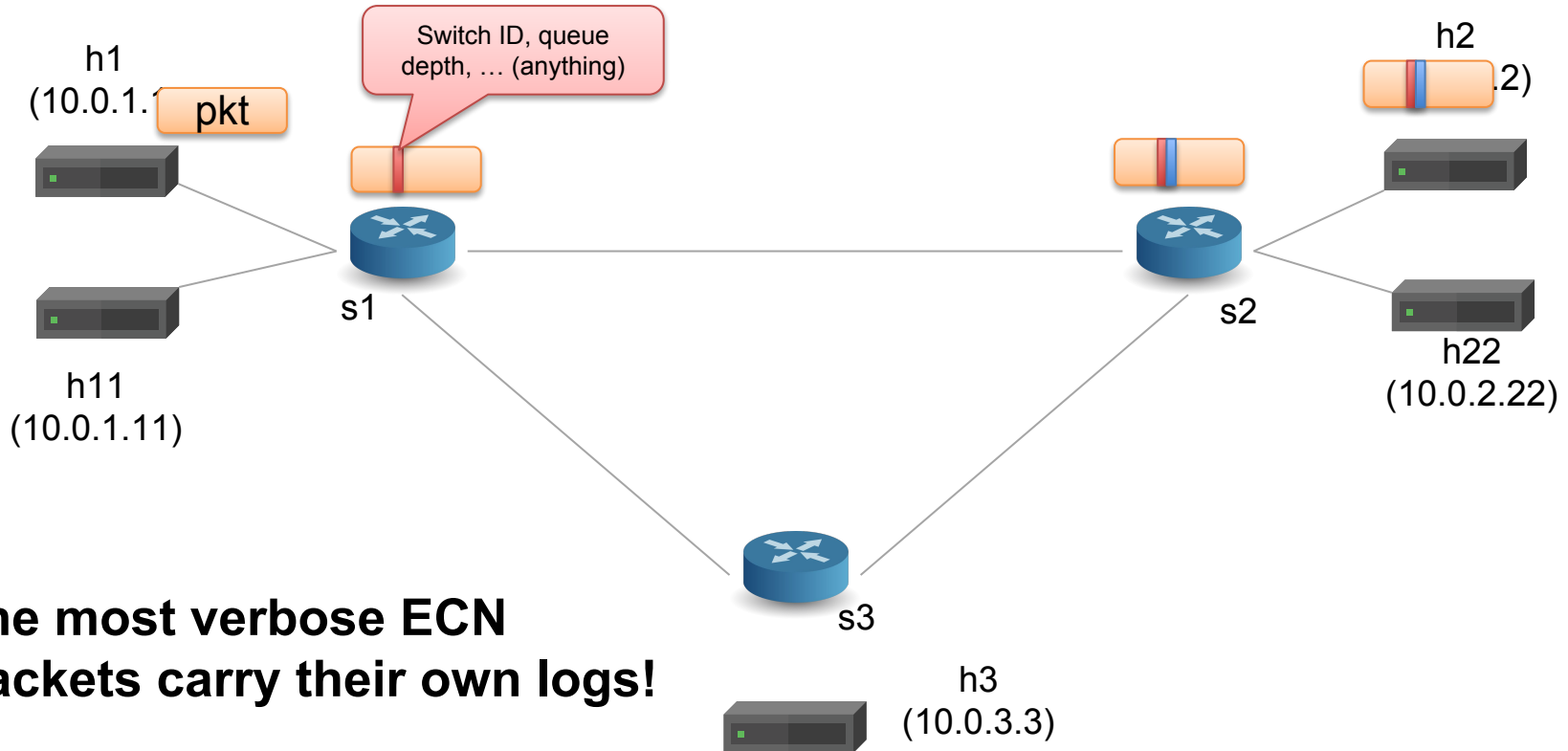
- The standard data for the V1Model includes the queue depth:
bit<19> standard_metadata.enq_qdepth



Coding Break



Multi-Route Inspection



**The most verbose ECN
Packets carry their own logs!**

Multi-Route Inspect: Packet Format

```
header mri_t {  
    bit<16>  count;  
}
```

```
header switch_t {  
    switchID_t  swid;  
    qdepth_t    qdepth;  
}
```

```
struct headers {  
    ethernet_t    ethernet;  
    ipv4_t        ipv4;  
    ipv4_option_t  ipv4_option;  
    mri_t         mri;  
    switch_t[MAX_HOPS] swtraces;  
}
```

- **Header validity operations:**

- `hdr.setValid(): add_header`
- `hdr.setInvalid(): remove_header`
- `hdr.isValid(): test validity`

- **Header Stacks**

- `hdr[CNT] stk;`

- **Header Stacks in Parsers**

- `stk.next`
- `stk.last`
- `stk.lastIndex`

- **Header Stacks in Controls**

- `stk[i]`
- `stk.size`
- `stk.push_front(int count)`
- `stk.pop_front(int count)`

Header verification

```
/* Standard errors, defined in core.p4 */
```

```
error {  
    NoError,           // no error  
    PacketTooShort,   // not enough bits in packet for extract  
    NoMatch,          // match expression has no matches  
    StackOutOfBounds, // reference to invalid element of a header stack  
    OverwritingHeader, // one header is extracted twice  
    HeaderTooShort,   // extracting too many bits in a varbit field  
    ParserTimeout     // parser execution time limit exceeded  
}
```

```
/* Additional error added by the programmer */
```

```
error { IPv4BadHeader }
```

```
...
```

```
state parse_ipv4 {  
    packet.extract(hdr.ipv4);  
    verify(hdr.ipv4.version == 4, error.IPv4BadHeader);  
    transition accept;  
}
```



Coding Break

