Is It OK to Hijack TCP?

John Ousterhout
Stanford University

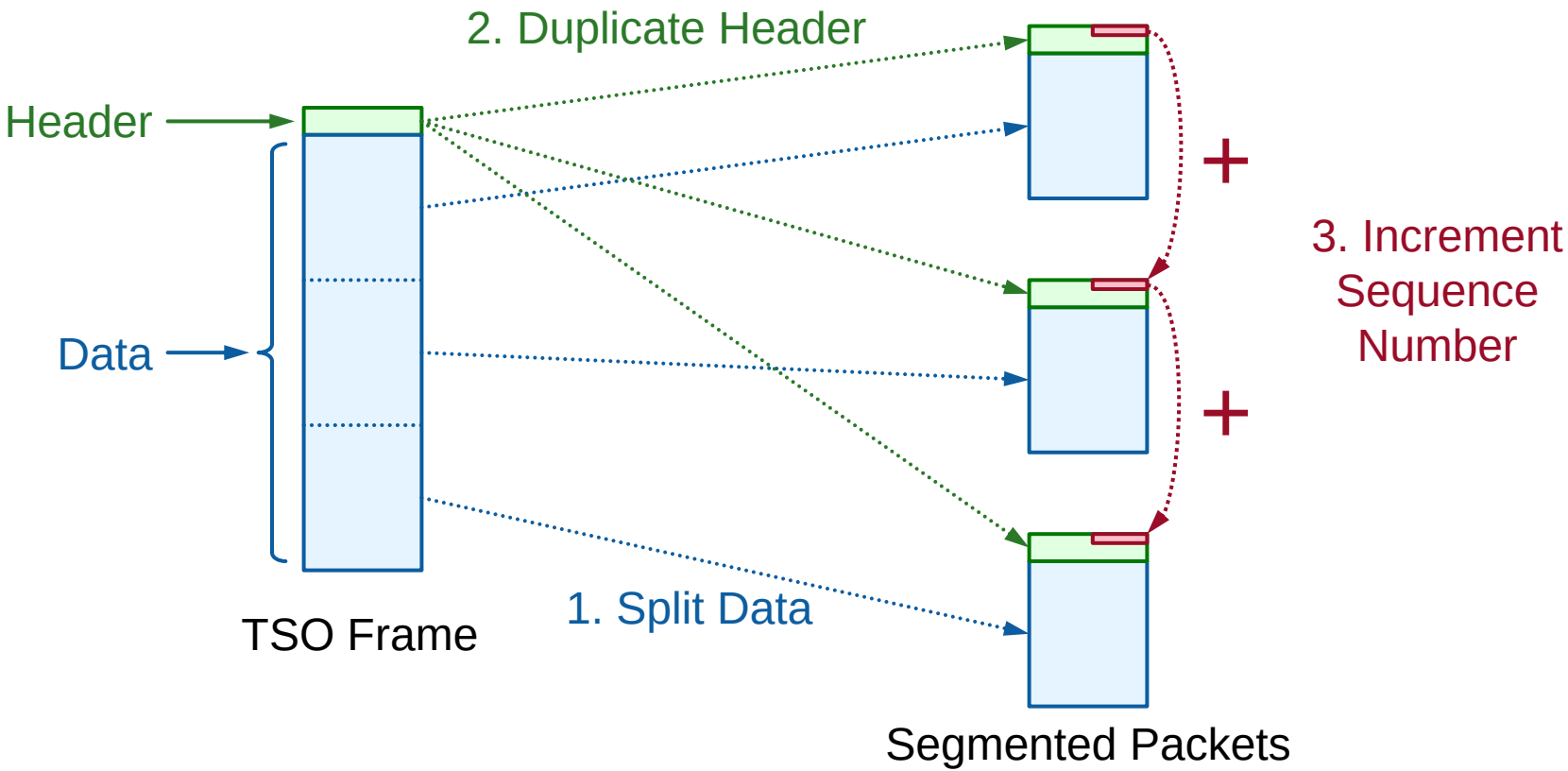
Introduction

- **Homa: alternate transport protocol to TCP**
  - >10x latency reductions for short messages under high load in datacenters
- **TCP's hardware support (TSO, RSS) creates a built-in advantage**
- **Difficult for Homa to take advantage of these**
- **Solution: encapsulate Homa packets as TCP packets**

Goal for this talk: get feedback:

- **Is this a terrible idea?**
- **Is there a better way to achieve the same goal?**

TCP Segmentation Offload



Segmentation Offload, cont'd

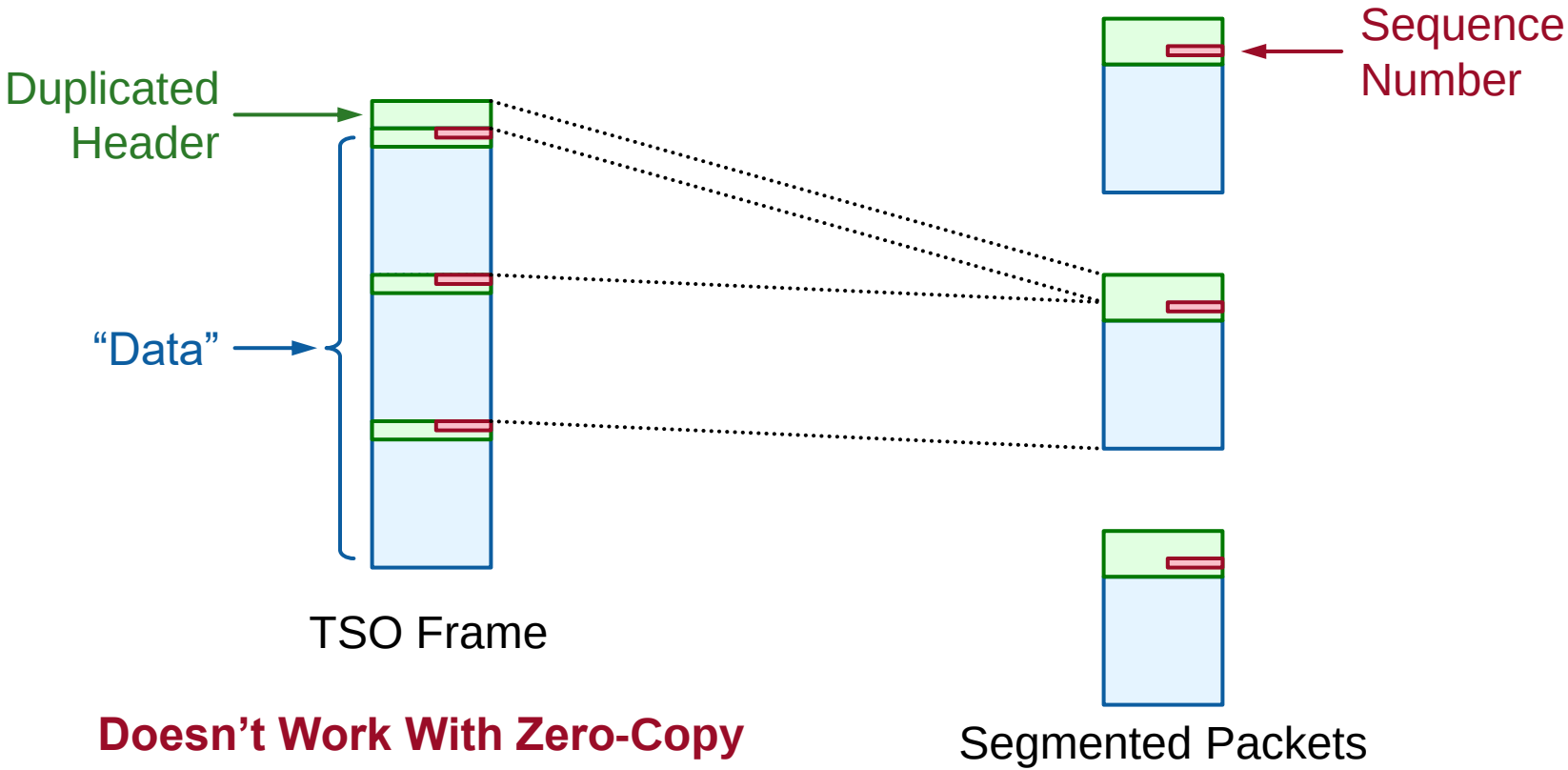
Essential for high performance:

- **Reduce traversals of the networking stack**

Support for protocols other than TCP varies between NICs:

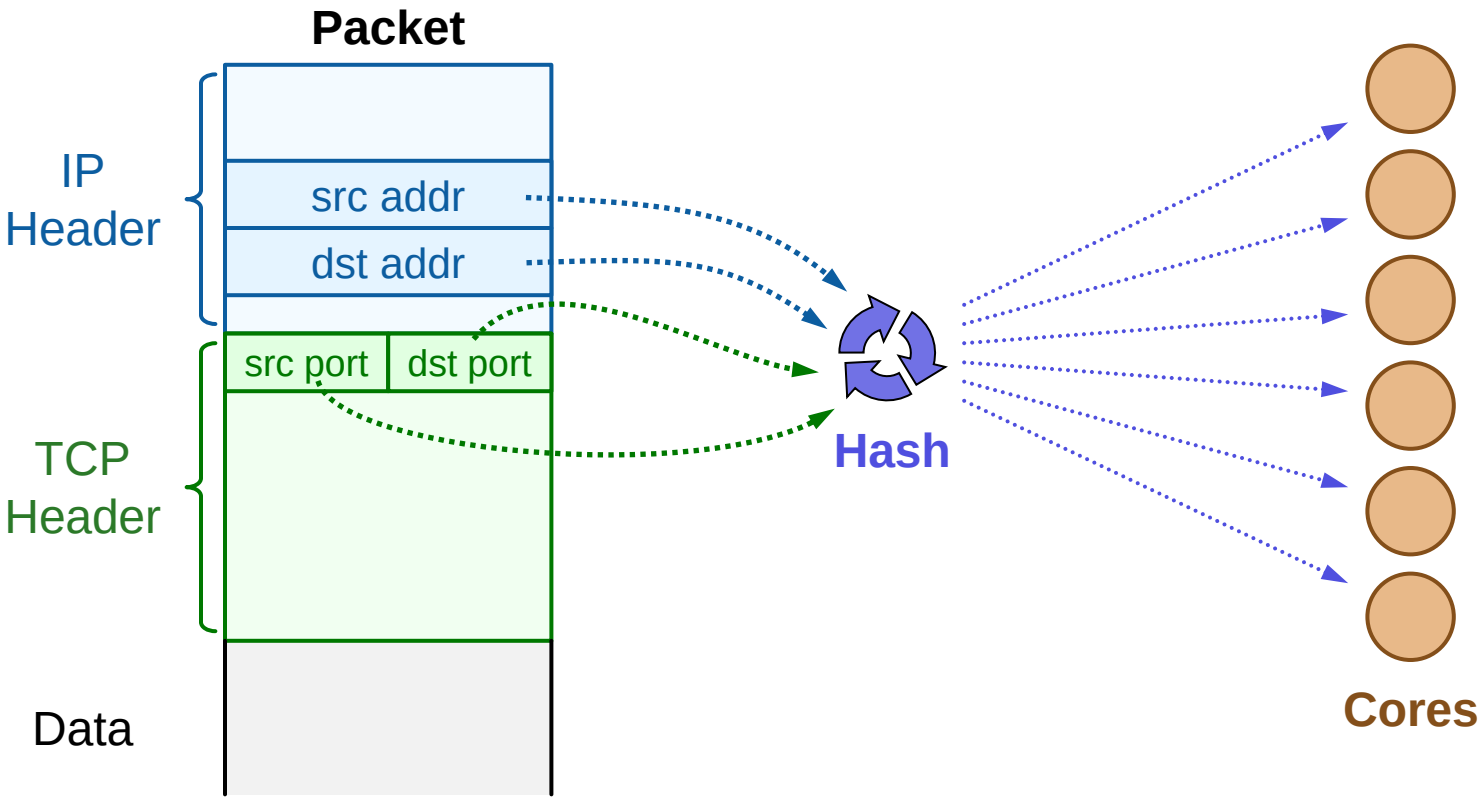
- **No support for non-TCP protocols?**
- **Can support non-TCP protocols, but requires customization?**
- **Partial support for non-TCP protocols:**
  - E.g. Mellanox ConnectX-5: supports non-TCP protocols, but doesn't increment sequence number

TSO with Mellanox NICs:

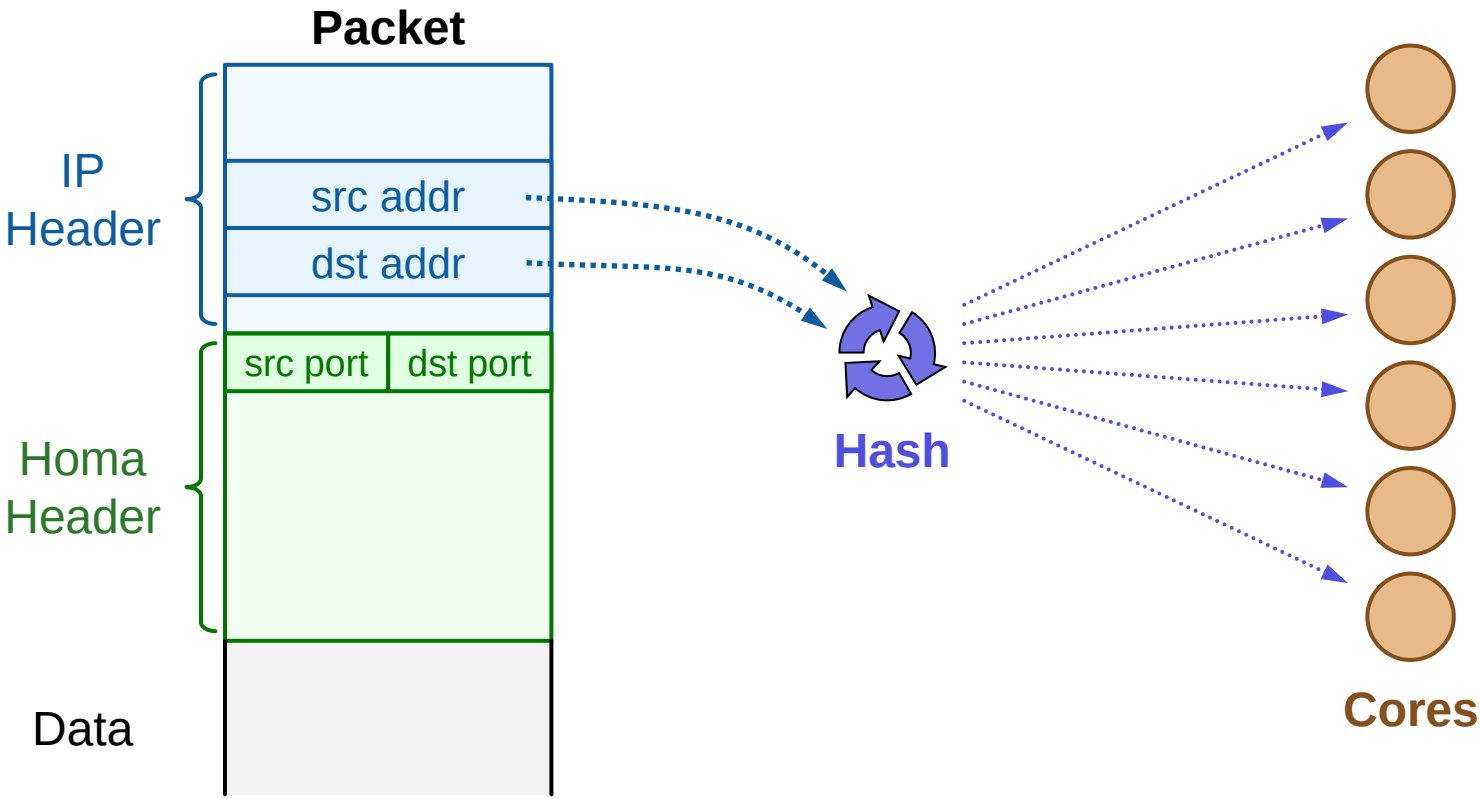


Doesn't Work With Zero-Copy

Receive-Side Scaling (RSS)



RSS With Homa



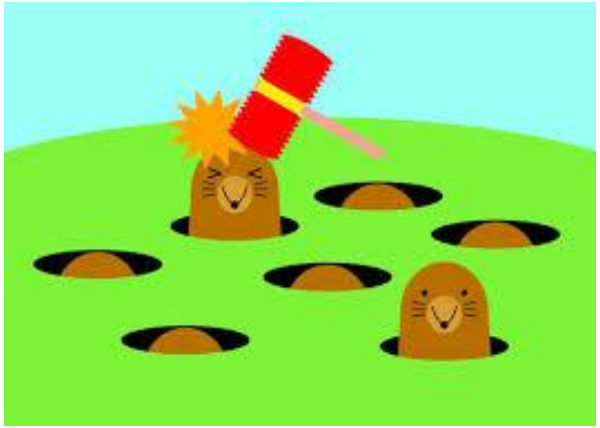
RSS With Homa, cont'd

- **All incoming packets from a single peer will be processed on a single NAPI core**
- **Core saturation limits throughput:**
  - 61 Gbps for Homa in 2-node throughput test with 100 Gbps links

Whack-A-Mole

Customize NICs to support Homa?

- Too many NICs
- New NICs, feature changes
- Very difficult for me to keep up
- Additional installation step for Homa users



Are all NICs suitably customizable?

- Without proper support, Homa will perform poorly compared to TCP

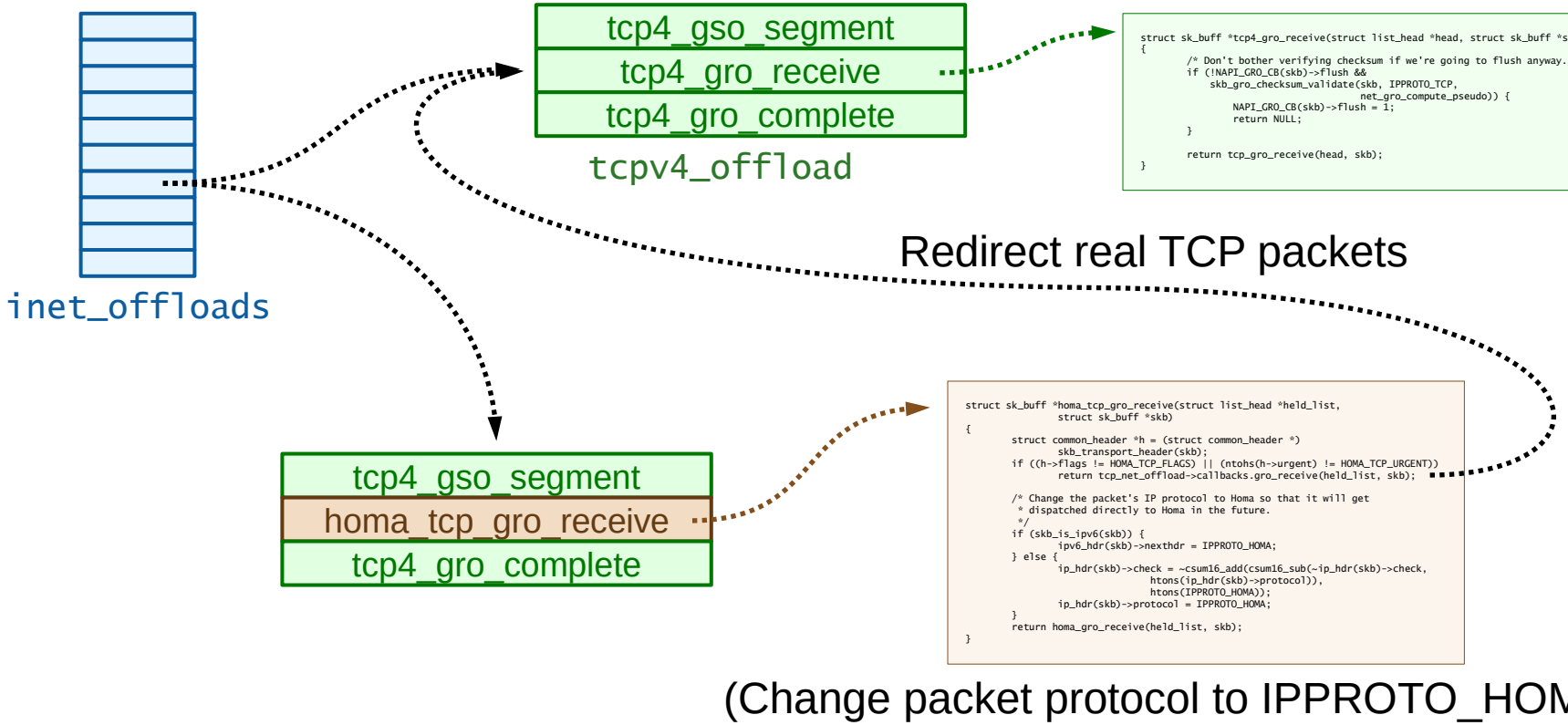
Hijack TCP

- **The only way to get the same hardware support as TCP is to be TCP:**
  - Homa packets must have valid TCP headers
  - Send Homa packets with IPPROTO_TCP
- **Challenges:**
  - Setting the protocol in the IP header
  - Stealing back Homa packets on the receiver
  - Distinguishing Homa-over-TCP packets from real TCP packets

Transmitting as TCP

- IP protocol field is set from `sock->sk_protocol`
- Homa sets this to IPPROTO_TCP during socket creation (app requests type IPPROTO_HOMA)
- Risk of unintended consequences?
- Is there a better way to set the outgoing IP protocol?

Reclaiming Homa Packets

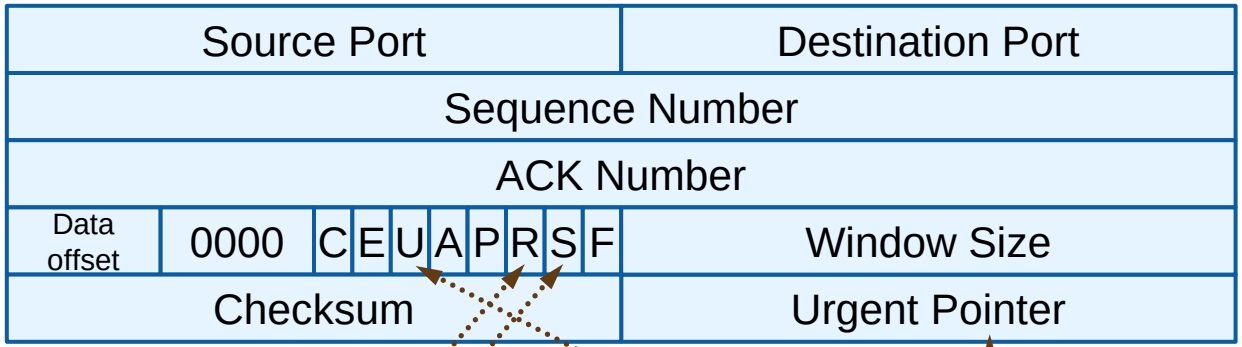



```
struct sk_buff *tcp4_gro_receive(struct list_head *head, struct sk_buff *skb)
{
        /* Don't bother verifying checksum if we're going to flush anyway. */
        if (!NAPI_GRO_CB(skb)->flush &&
            skb_gro_checksum_validate(skb, IPPROTO_TCP,
                                      net_gro_compute_pseudo)) {
                NAPI_GRO_CB(skb)->flush = 1;
                return NULL;
        }

        return tcp_gro_receive(head, skb);
}
```

```
struct sk_buff *homa_tcp_gro_receive(struct list_head *held_list,
                struct sk_buff *skb)
{
        struct common_header *h = (struct common_header *)
                skb_transport_header(skb);
        if ((h->flags != HOMA_TCP_FLAGS) || (ntohs(h->urgent) != HOMA_TCP_URGENT))
                return tcp_net_offload->callbacks.gro_receive(held_list, skb);

        /* Change the packet's IP protocol to Homa so that it will get
         * dispatched directly to Homa in the future.
         */
        if (skb_is_ipv6(skb)) {
                ipv6_hdr(skb)->nexthdr = IPPROTO_HOMA;
        } else {
                ip_hdr(skb)->check = ~csum16_add(csum16_sub(~ip_hdr(skb)->check,
                                htons(ip_hdr(skb)->protocol)),
                                htons(IPPROTO_HOMA));
                ip_hdr(skb)->protocol = IPPROTO_HOMA;
        }
        return homa_gro_receive(held_list, skb);
}
```

Identifying Homa Packets

TCP Header



RST and SYN both 1

“Magic” value for Urgent Pointer, but URG==0

Use header option instead?

Combination of values TCP would never use?

Status

- **Easy to implement (2 days)**
- **Appears to be working**
- **2-node throughput increased from 61 Gbps to 96 Gbps**

Conclusion

- **TCP's hardware support entrenches TCP:**
  - Any proposed replacement suffers nonlevel playing field
- **One possible solution: use TCP packets for other protocols**
- **Linux kernel implementation appears straightforward**

Discussion Questions

- **How bad of an idea is this?**
  - Potential problems that I haven't foreseen
- **Would this approach be accepted for upstreaming?**
- **Are there better ways to implement TCP hijacking?**
  - e.g., don't modify sock->sk_protocol?
- **Is there a better way to accomplish the same goal?**