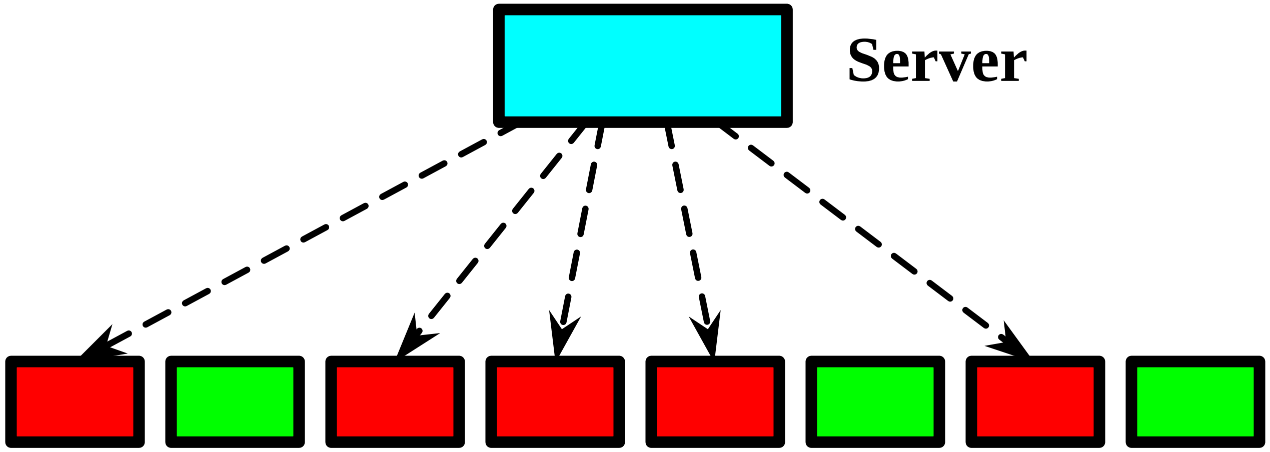
# Efficient Operating System Support for Group Unicast

Martin Karsten, Jialin Song,
Michael Kwok, Tim Brecht

University of Waterloo

## Problem

**Some apps send same data to multiple receivers**



Sending to group of red hosts/users

## Example Applications

- Distributed Virtual Environments
  - Multiplayer on-line games
  - Computer Supported Cooperative Work (CSCW)
- Audio/Video conferencing
- Chat room servers
- Streaming media servers
- Multicast overlay networks

## Example Applications

- Distributed Virtual Environments
  - Multiplayer on-line games
  - Computer Supported Cooperative Work (CSCW)
- Audio/Video conferencing
- Chat room servers
- Streaming media servers
- Multicast overlay networks

**Many/most of these use UDP**

**How to efficiently send to a group using UDP?**

University of Waterloo

## Example Applications

- Distributed Virtual Environments
  - Multiplayer on-line games
  - Computer Supported Cooperative Work (CSCW)
- Audio/Video conferencing
- Chat room servers
- Streaming media servers
- Multicast overlay networks

**Many/most of these use UDP**

**How to efficiently send to a group using UDP?**

**Other Transport Protocols $\Rightarrow$ Future Work**

University of Waterloo

## Possible Approaches / Related Work

- IP Multicast [Deering 88]
  - difficulties in wide spread deployment (not feasible)
- Multicast Overlay networks [Lots of Research]
  - implemention requires group communication
- Common Approach: User-level unicast (user-groupcast)

University of Waterloo

## Possible Approaches / Related Work

- IP Multicast [Deering 88]
  - difficulties in wide spread deployment (not feasible)
- Multicast Overlay networks [Lots of Research]
  - implemention requires group communication
- Common Approach: User-level unicast (user-groupcast)

```
for (i=0; i<GRPSIZE; i++) {
  fds[i] = socket(PF_INET, SOCK_DGRAM, 0);
}
for (i=0; i<GRPSIZE; i++) {
  bytes += send(fds[i], buf, bytes);
}
```

University of Waterloo

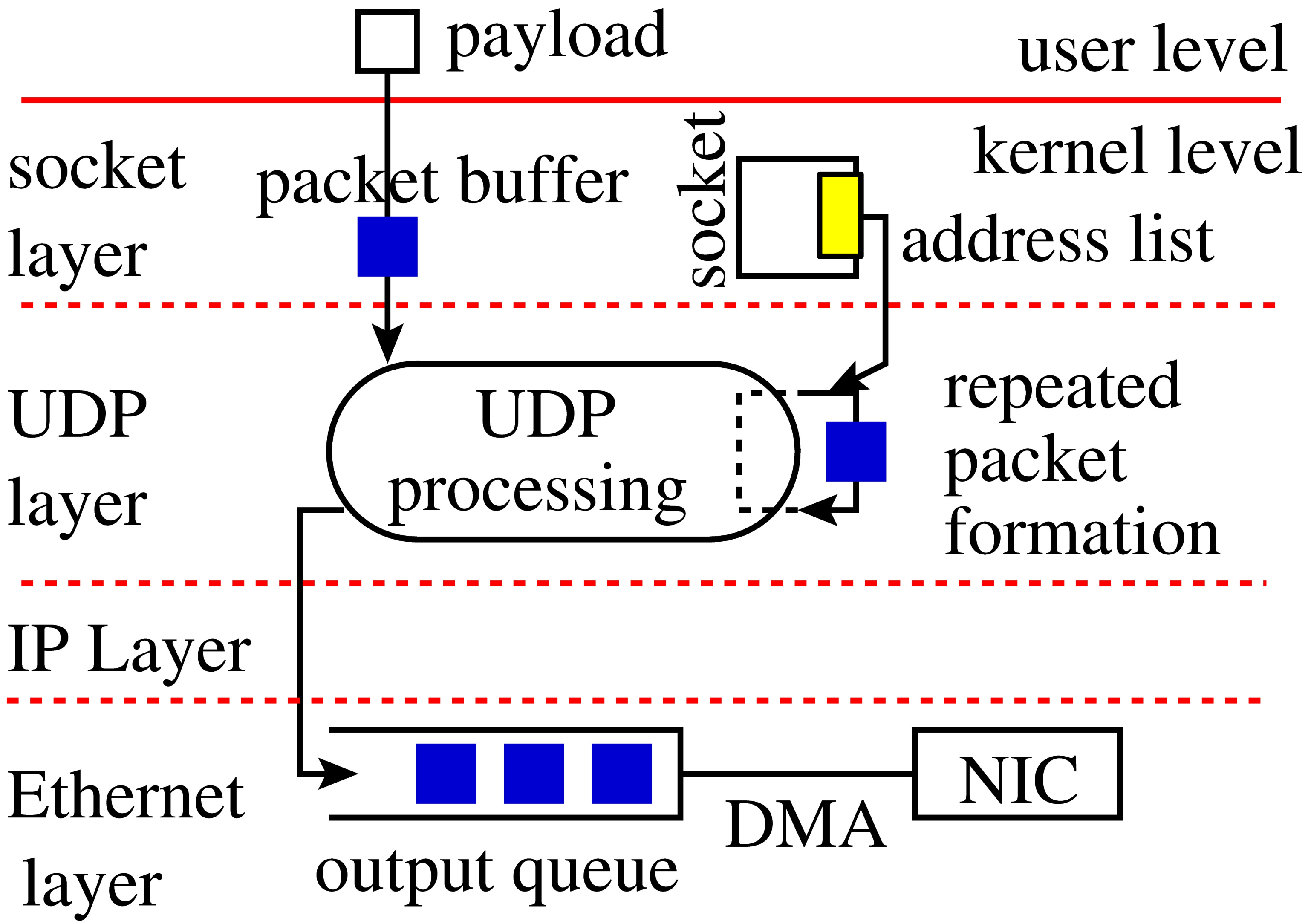
# Kernel-Level Group Unicast (kernel-groupcast)

```
grp = socket(PF_INET, SOCK_DGRAM, 0);
setsockopt(grp, SOL_SOCKET,
   SO_SETGRP, addrs,
   GRPSIZE * sizeof(struct sockaddr_in));

bytes = send(grp, buf, bytes);
```

University of Waterloo

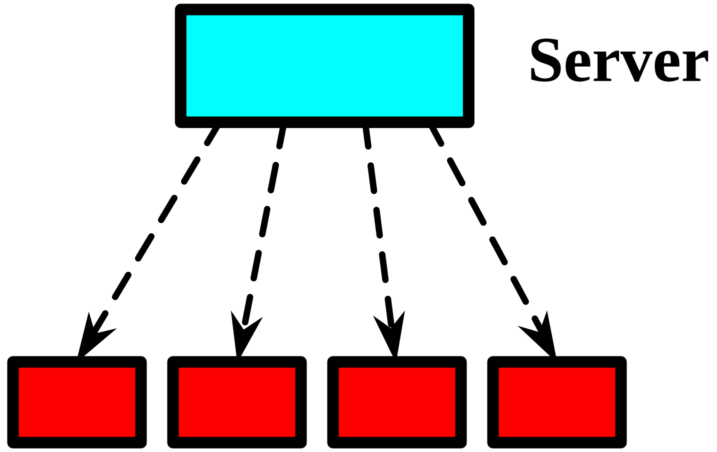
# Implementation Overview



University of Waterloo

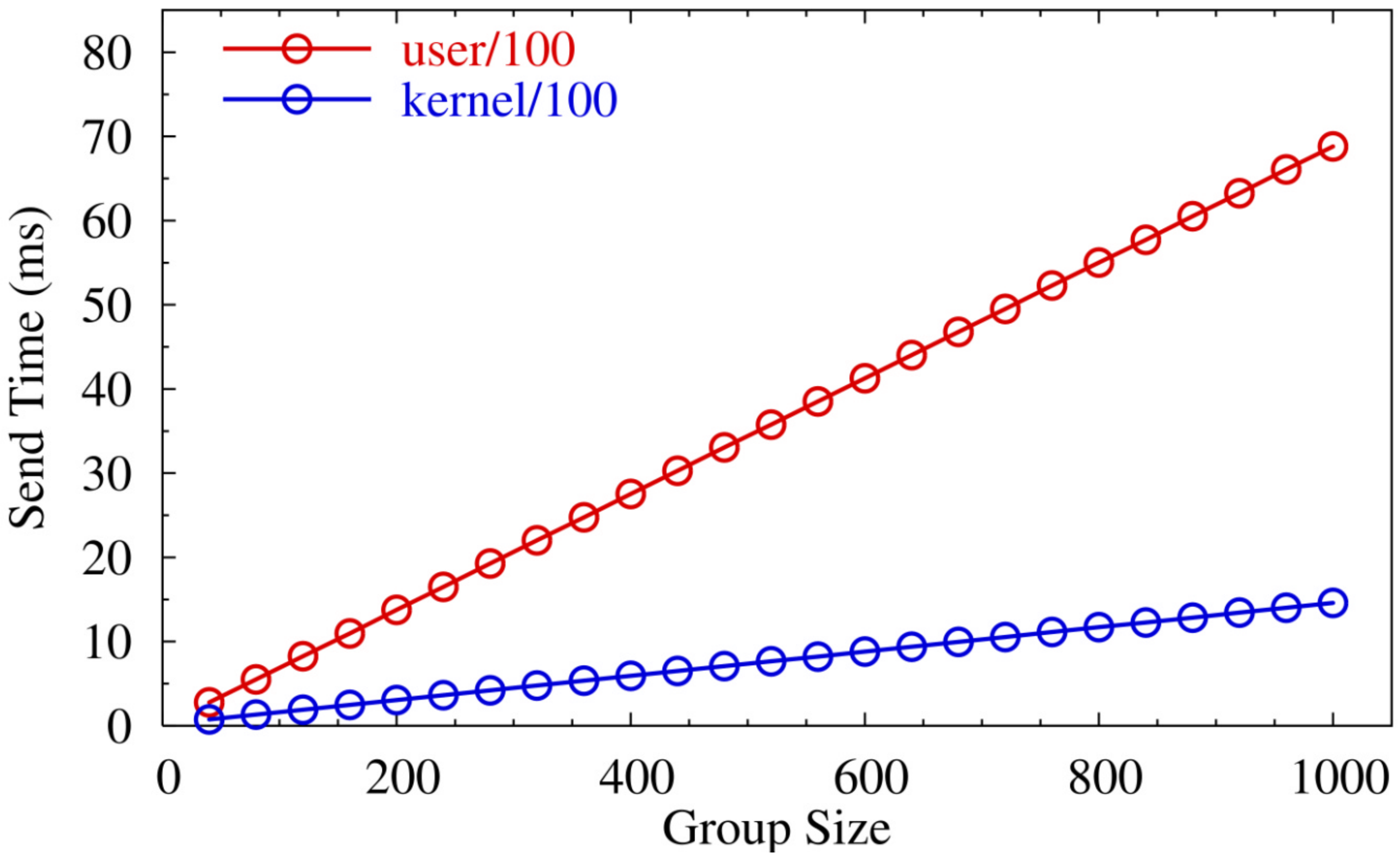
# Experimental Environment

- **Server:** 400 MHz Pentium II, 2 x e1000 Gbps enet
  - **FreeBSD 5.2.1, Fedora Core 2 with 2.6.8 kernel**
- **Switch:** HP Procurve Gbps switch: 24 ports
- **Clients:** 550 MHz Pentium III

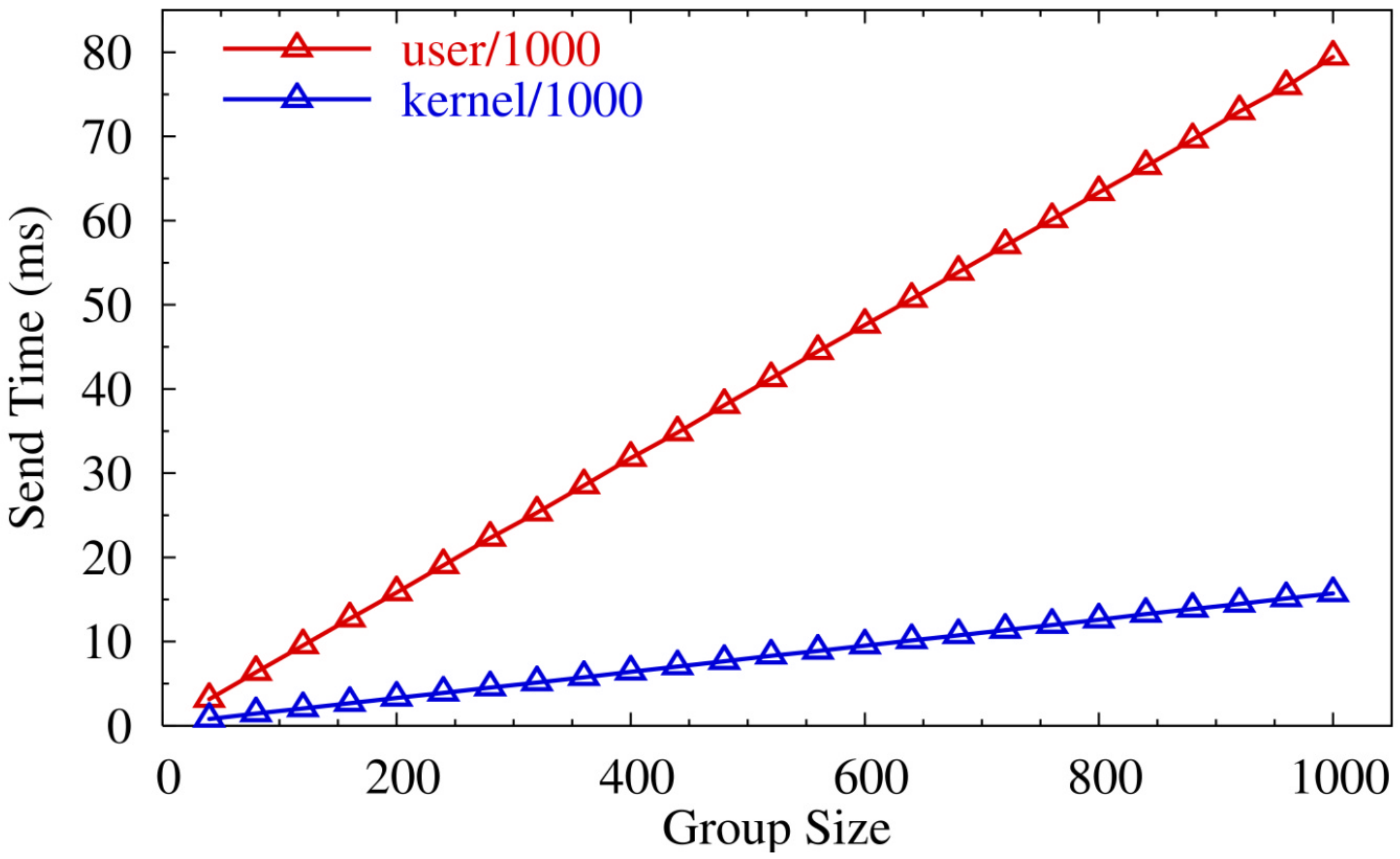


**Deliberately set up so that sender is bottleneck**

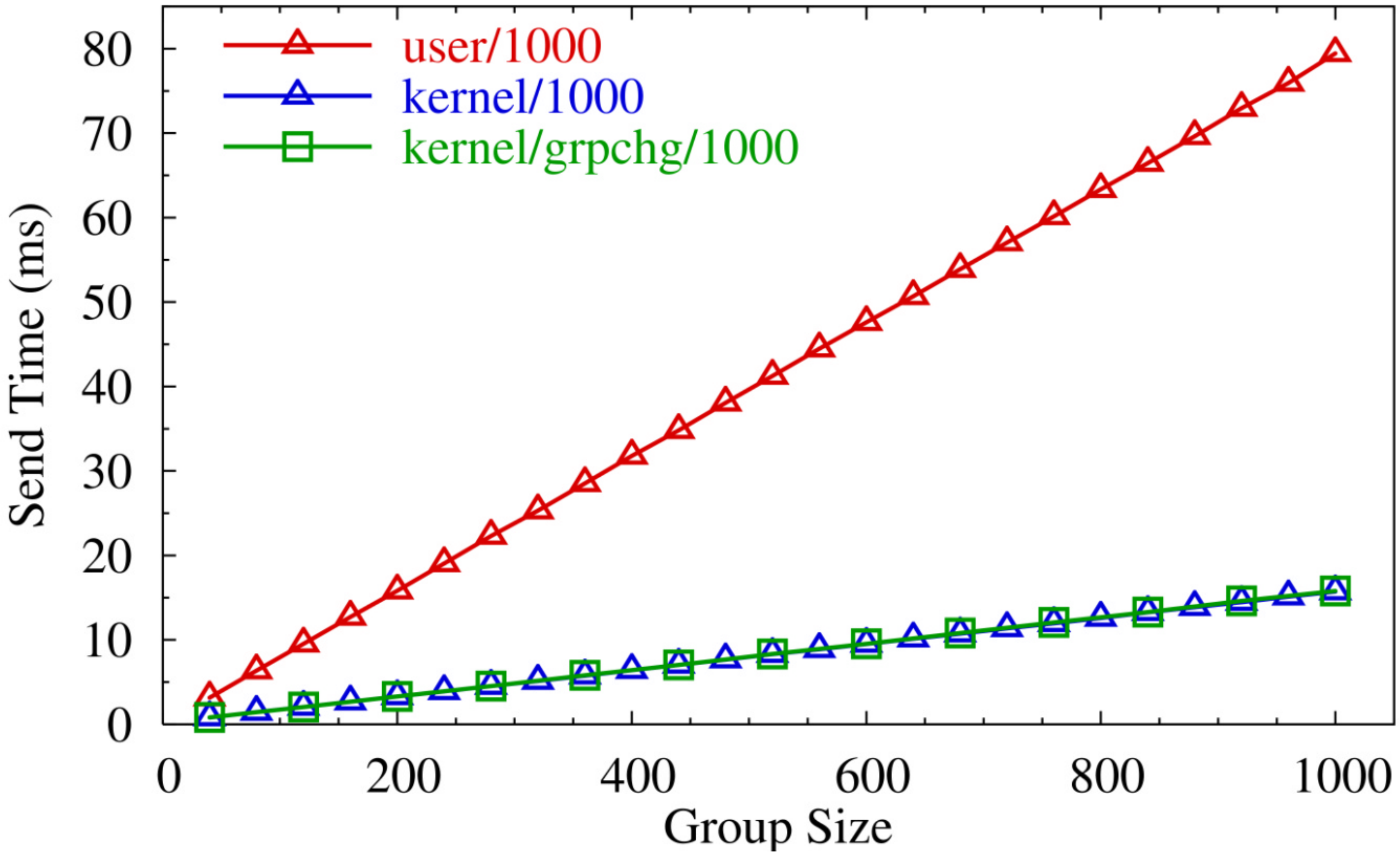
# FreeBSD Micro-benchmark: 100 bytes



# FreeBSD Micro-benchmark : 1000 bytes

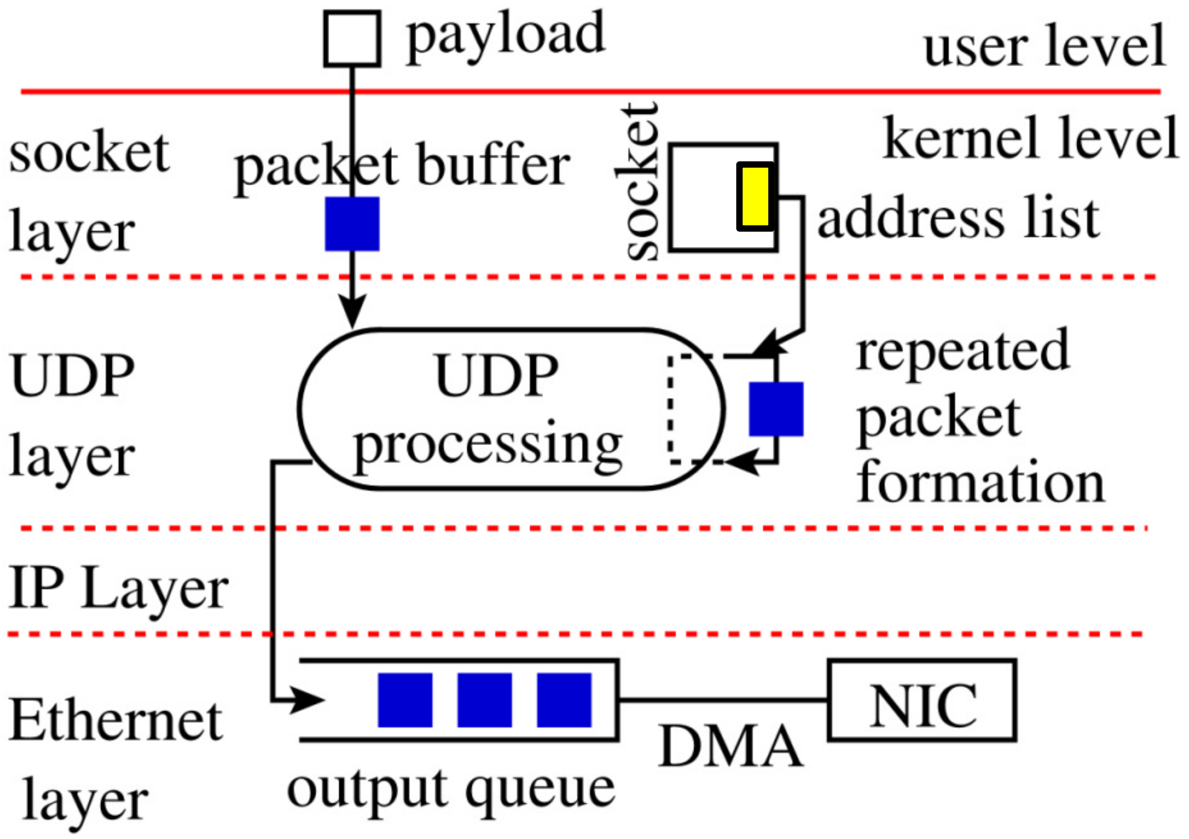


## FreeBSD Micro-benchmark: with grp change



University of Waterloo

# Software Slicing: User-groupcast



start of memcpy

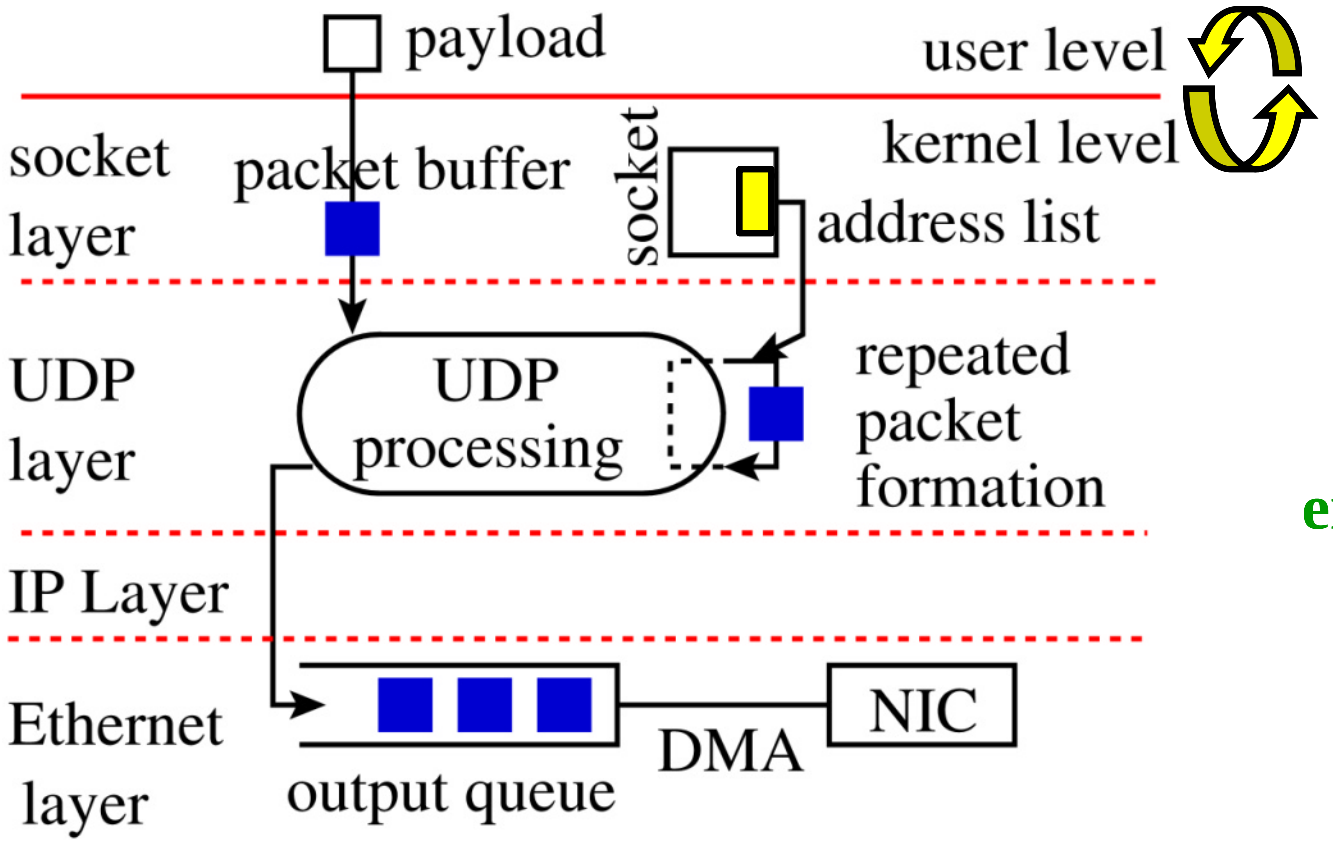
end of memcpy

end UDP / start IP

start of driver

complete send

# Software Slicing: User-groupcast



start of memcpy

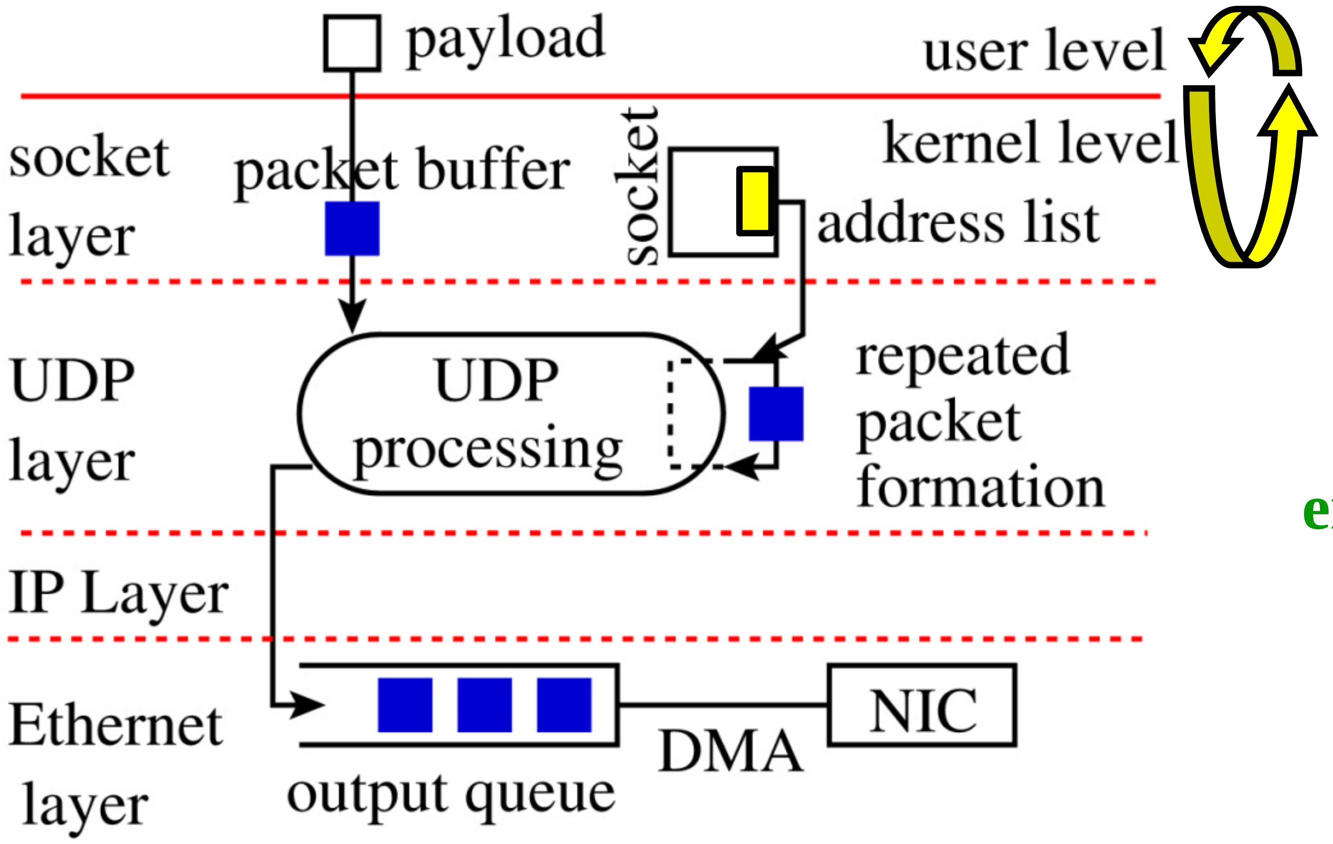
end of memcpy

end UDP / start IP

start of driver

complete send

# Software Slicing: User-groupcast



start of memcpy

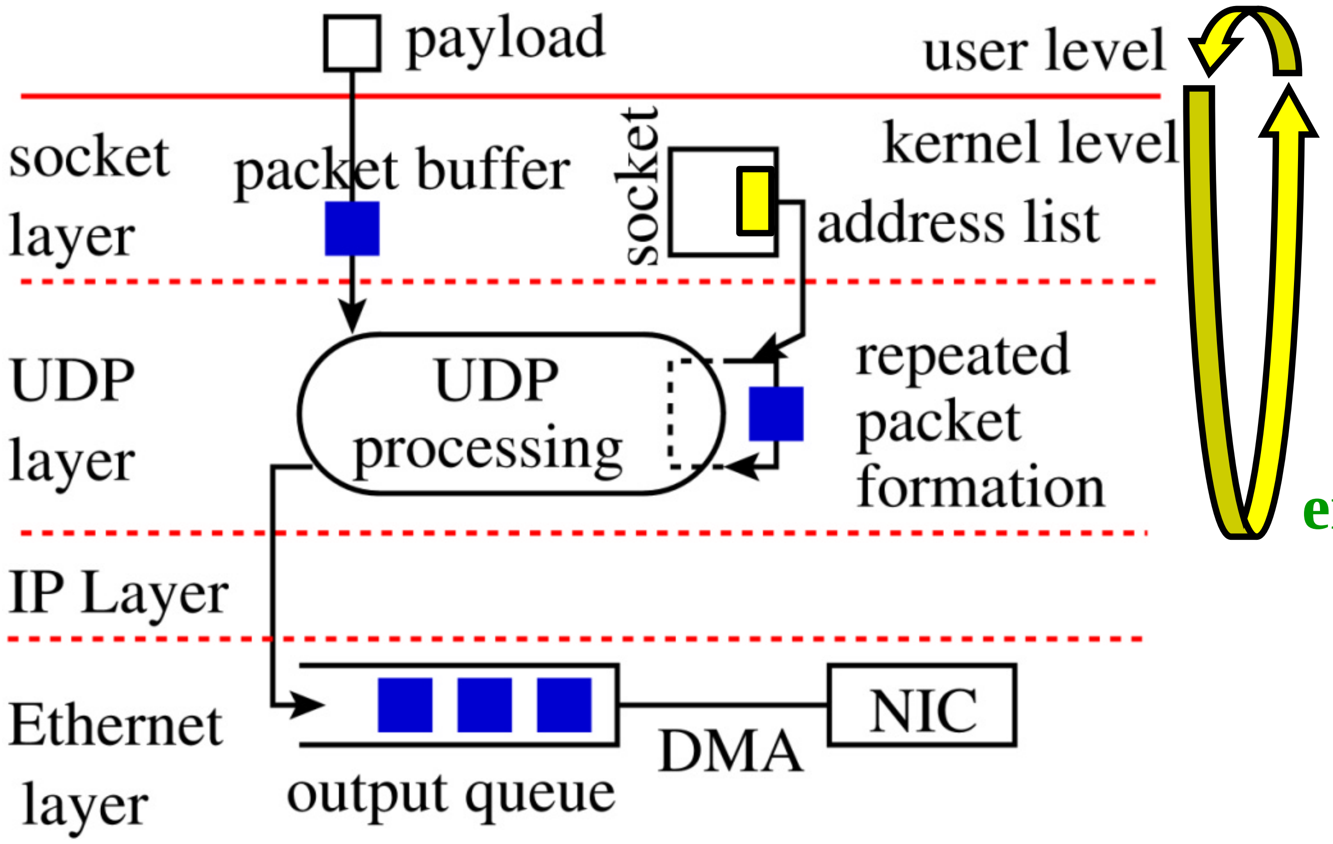
end of memcpy

end UDP / start IP

start of driver

complete send

# Software Slicing: User-groupcast



start of memcpy

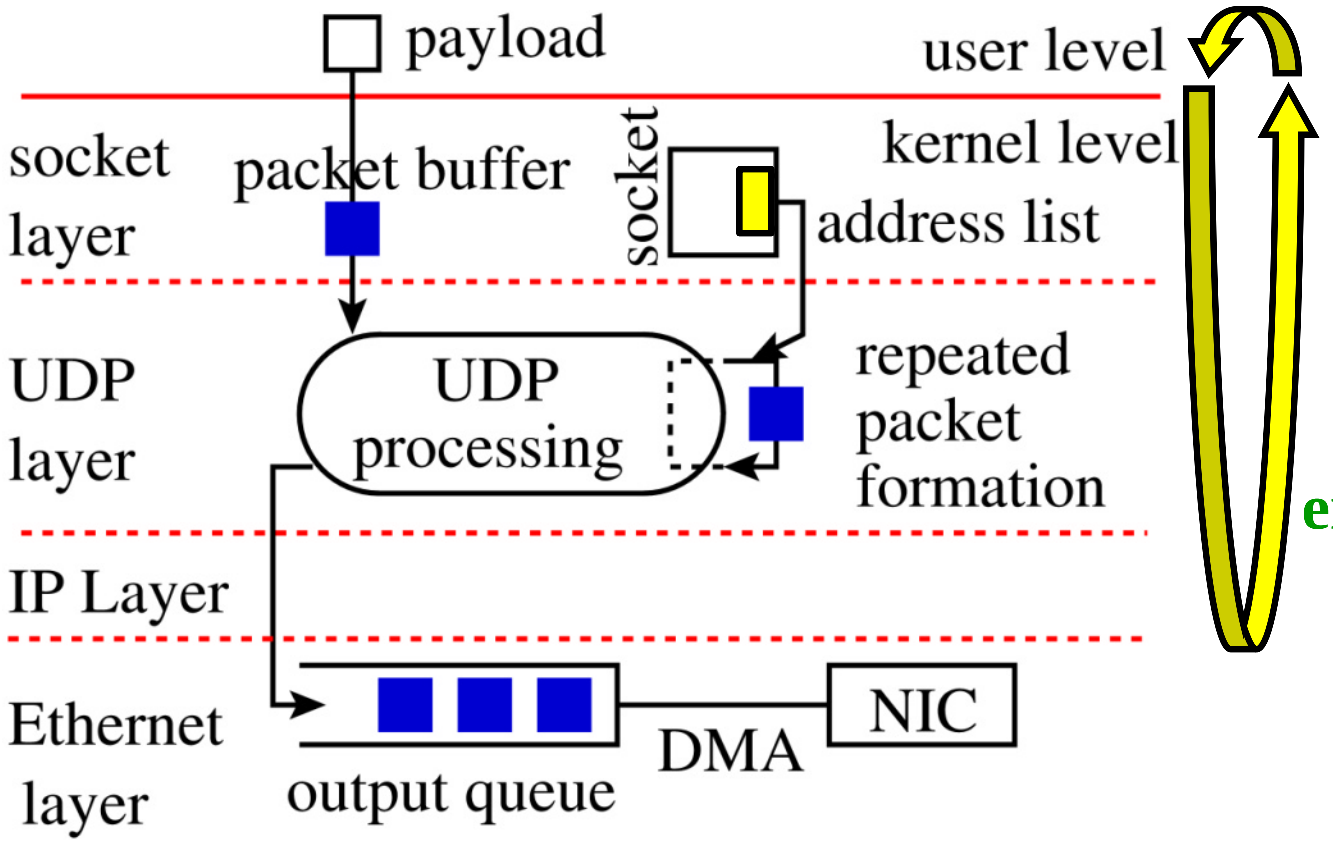
end of memcpy

end UDP / start IP

start of driver

complete send

# Software Slicing: User-groupcast



start of memcpy

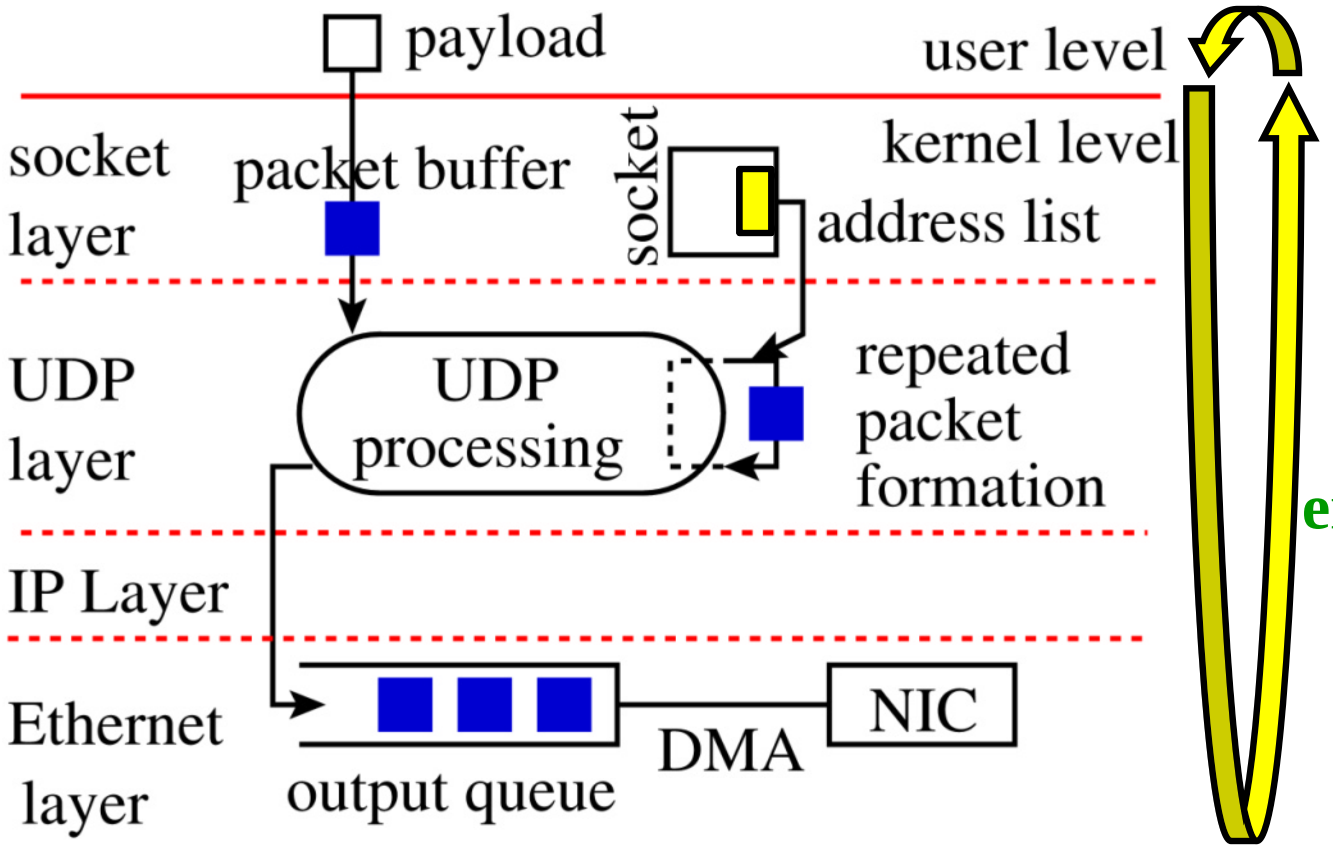
end of memcpy

end UDP / start IP

start of driver

complete send

# Software Slicing: User-groupcast



start of memcpy

end of memcpy

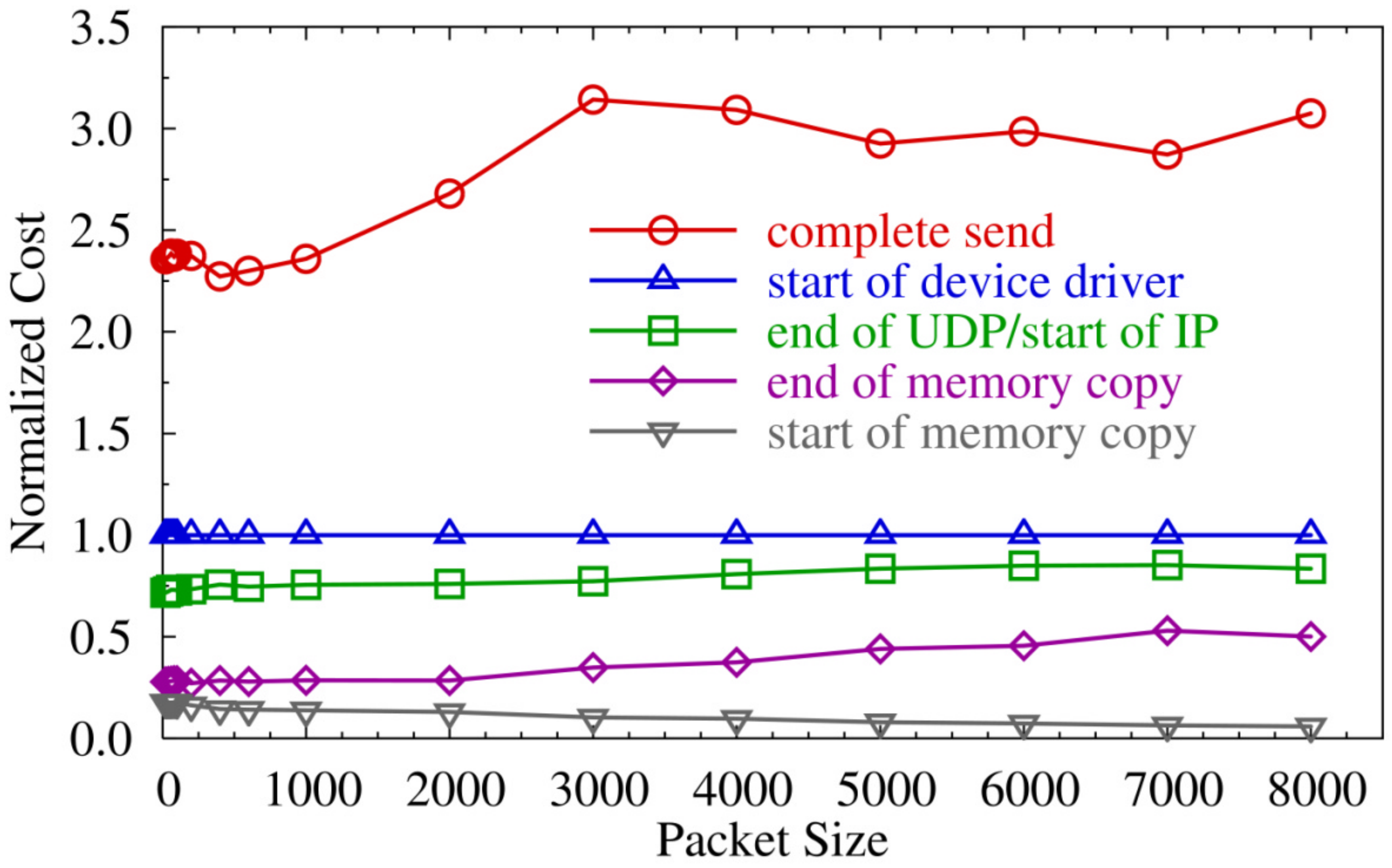
end UDP / start IP

start of driver

complete send

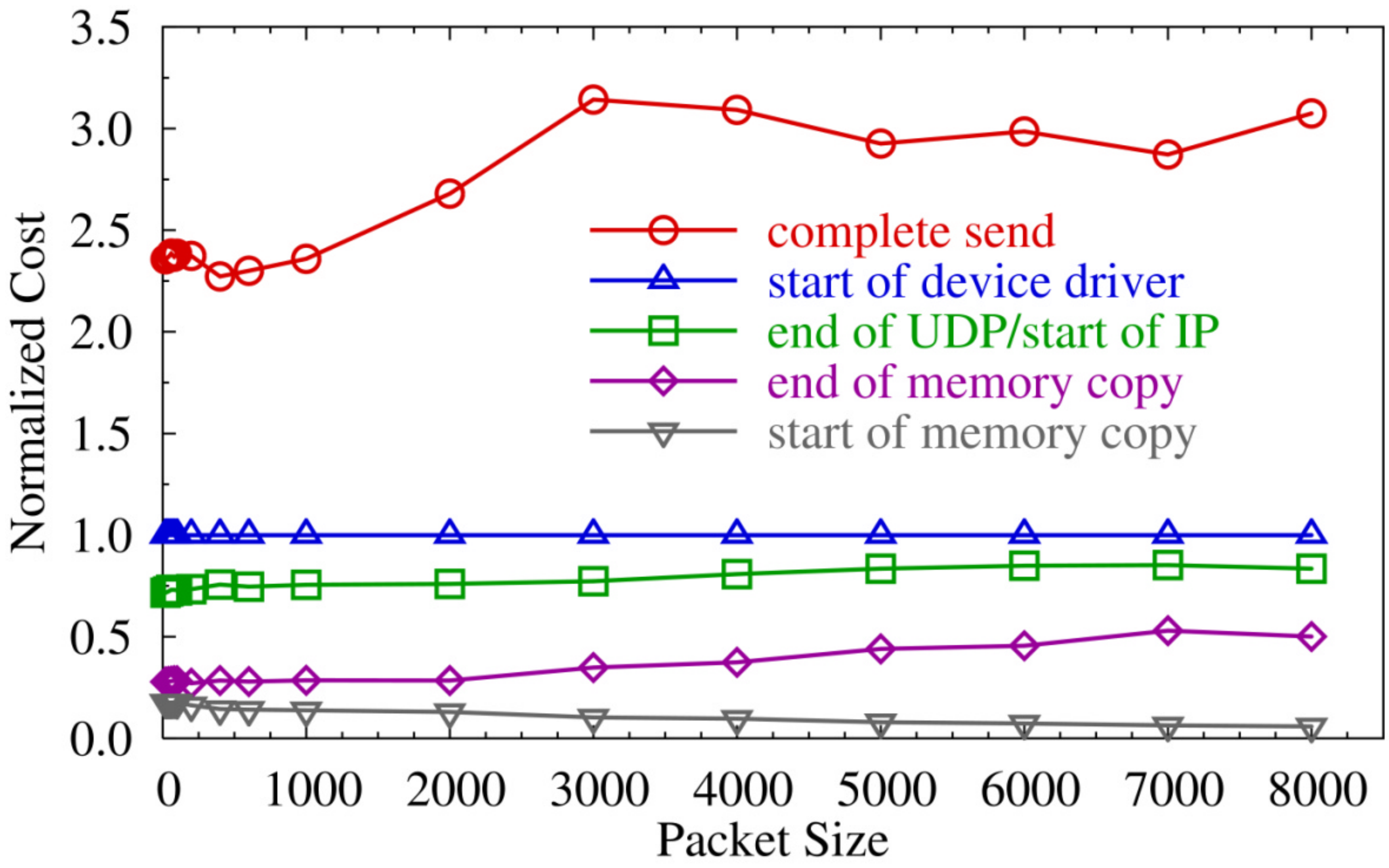
University of Waterloo

# User-Level Send Cost Breakdown: FreeBSD



University of Waterloo

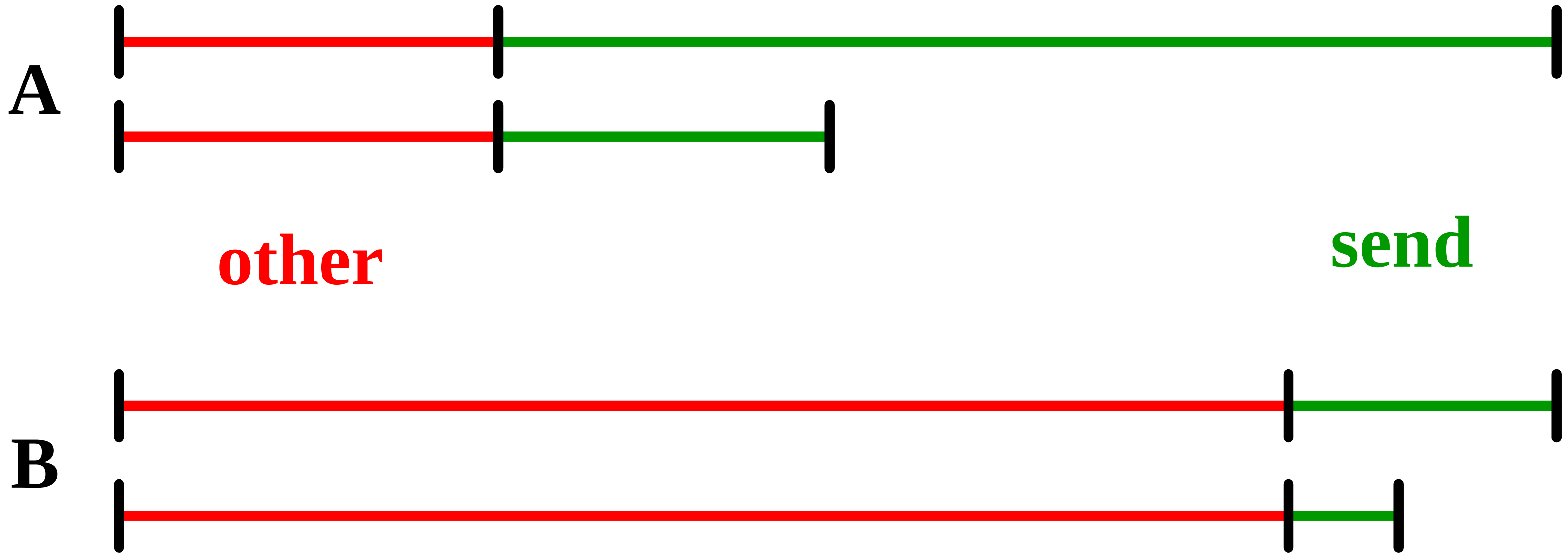
# User-Level Send Cost Breakdown: FreeBSD



**kernel or user-groupcast ~1 interrupt per system call**

**`ether_output()` 10 x faster: kernel-groupcast**

# Is this Important/Relevant to Applications?



- decrease latencies
- increase number of users / recipients

## Increase Group Size: (100 bytes, 33.3 ms)

| *N* | User *send* | Send *fraction* | Increase *factor* | Increase *N'* |
|---|---|---|---|---|
| 40 | 2.78 | 0.08 | 1.06 | 42 |
| 120 | 8.26 | 0.25 | 1.24 | 148 |
| 240 | 16.50 | 0.50 | 1.63 | 390 |
| 360 | 24.76 | 0.75 | 2.39 | 861 |
| 480 | 33.06 | 0.99 | 4.52 | 2170 |

**Your Mileage May (Will) Vary**

University of Waterloo

## Summary

- Kernel-groupcast
  - – OS interface and mechanism for group unicast
  - – relative minor modifications to FreeBSD and Linux
  - – significantly decrease time for group sends
- Does not reduce data sent
  - – improves server efficiency (efficient group unicast)
- Main source of improvement not reduced mem copy
  - – tight kernel loop
    - reduced interrupts
    - improved cache utilization

University of Waterloo

## Future Work

- Detailed breakdown of network I/O cost components
  - better understanding
  - on a variety of hardware platforms
- Better models for expected scalability
- Variety of apps and interaction with kernel-groupcast
  - library to work with existing interfaces?
- Apply kernel-groupcast to other transport protocols

University of Waterloo

## **The End**

University of Waterloo