# Making Reliable More Reliable:

## Expanding MINIX 3 with Link Aggregation

Lazar Stričević
University of Novi Sad

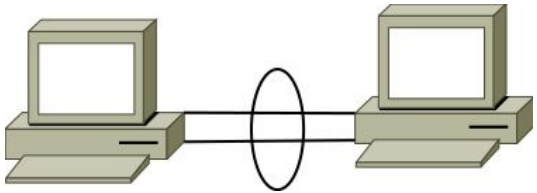
MINIXCon 2016

# Contents

- A Word About Link Aggregation
- Existing OS-level Implementations
- MINIX3 Implementation of Link Aggregation
- Performance Tests

# What is Link Aggregation (LA)?

A set of methods for using multiple parallel links between a pair of devices as if they were a single higher-performance communication channel.



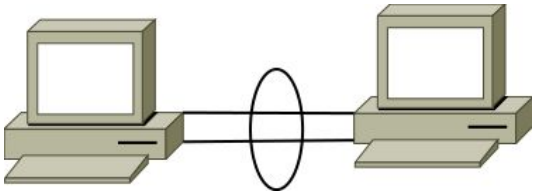
Different names: bonding, trunking, teaming, bundling, ...

# Reasons for LA

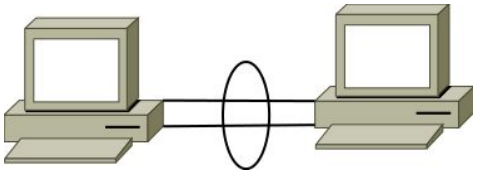
Goal: increase the **availability** and **throughput** of the channel without changing the networking technology.

Availability / Fault Tolerance

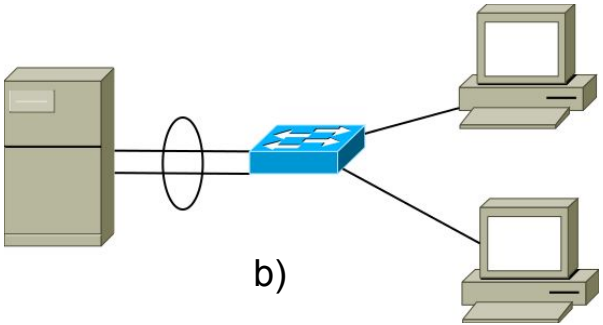
Throughput / Load Balancing



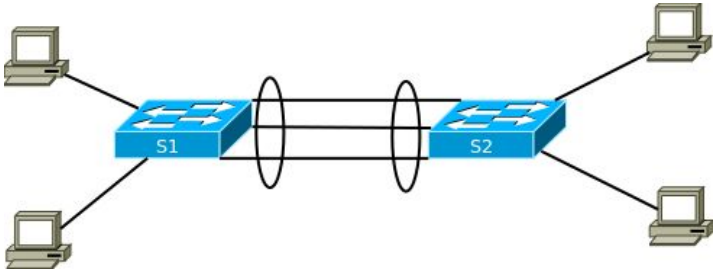
# Typical LA Setup



a)



b)

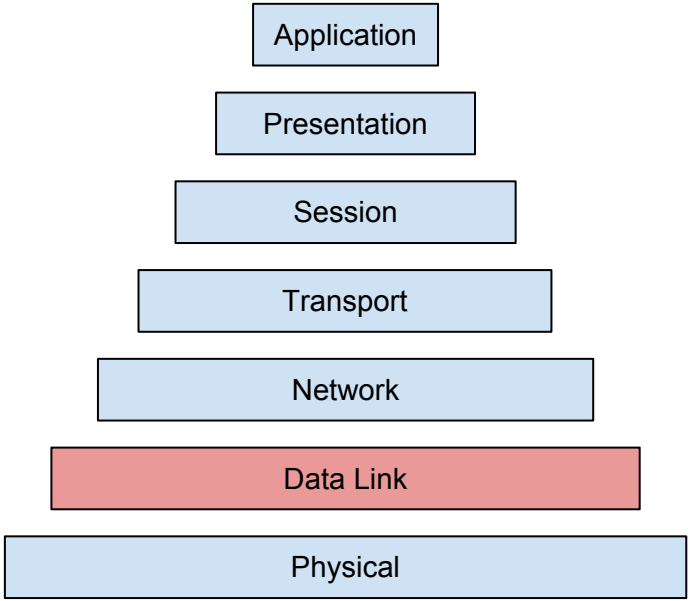



c)

# LA in OSI

Can be implemented at different layers of OSI (Open Systems Interconnection) model...

… but it is mostly done at the Data Link layer.



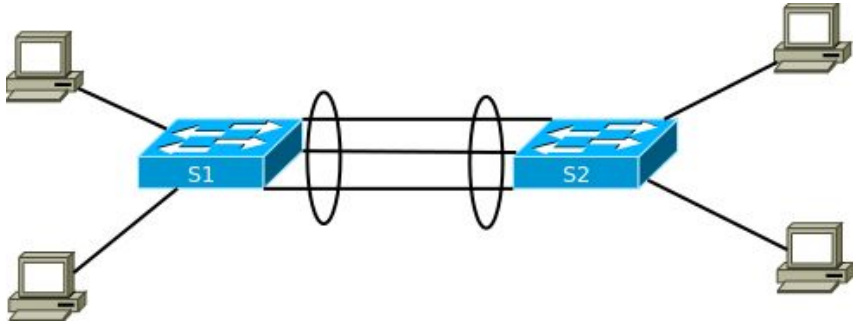
# Elements of LA

Frame forwarding

Link monitoring

# Aggregation forwarding modes

- active-backup / failover
- balance-rr / roundrobin
- balance-xor / loadbalance (fec)
- 802.3ad / lacp
- ...

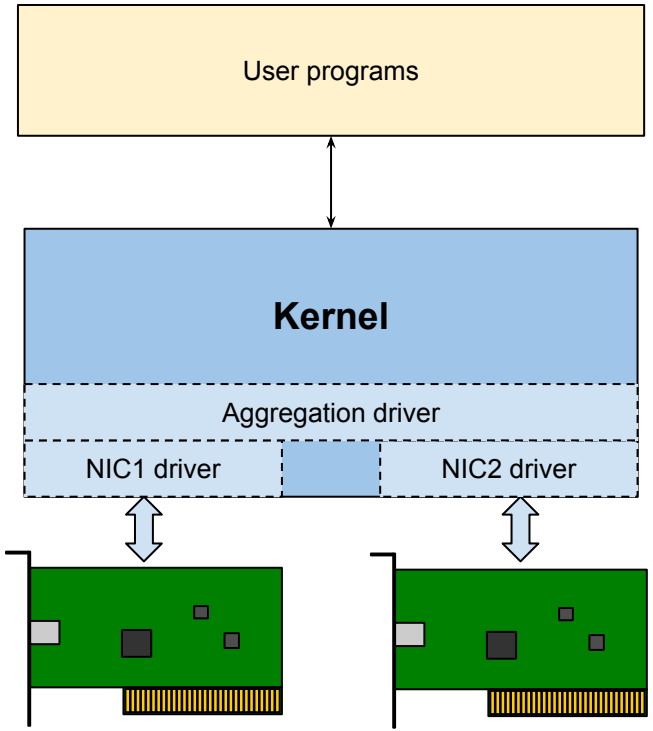


# Link monitoring modes

- MII monitoring: watching controller's MII registers
  - \+ fast
  - \- monitor monitors only the link carrier state
- ARP monitoring: send ARP frame to selected target(s), wait for reply
  - \+ more reliable
  - \- slower than MII monitoring
  - \- uses bandwidth
- Aggregation protocol (e.g. LACP)
  - \+ specifically made for LA
  - \+ not only link monitoring (support for dynamic aggregation)
  - \- also uses bandwidth

# Ethernet LA implementation on monolithic kernel

Virtual aggregation interface: “enslaves” some of the physical NIC drivers



# Existing OS-level LA implementations

## Unix-like OSes

- Linux (bond)
- BSD (lagg, trunk, agr)
- XNU (bond)

## Windows

- Windows Home/Professional
  - no OS support for LA
  (various NIC device drivers implement LA e.g. by Intel)
- Windows Server
  (2012 and later)

# LA on MINIX3? Why?

“One of the main goals of MINIX3 is reliability.”
*www.minix3.org*

One of the main goals of LA is to increase the availability of the communication channel.

MINIX3 + LA = More reliable/available MINIX 3

# MINIX implementation of LA

Requirements:

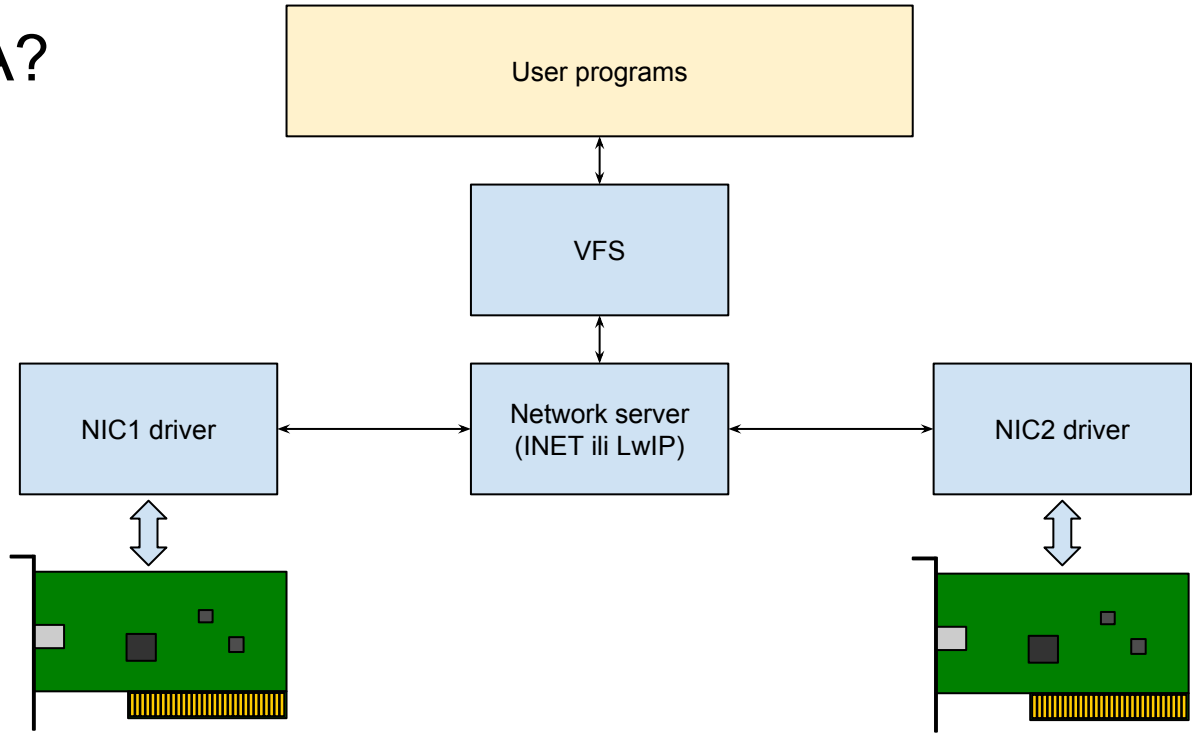
1. Increase availability of the network connection with:
   a. fault tolerance (failover mode)
   b. some of the other aggregation modes?

2. Low impact on performance (low overhead)

3. Minimal change to MINIX3

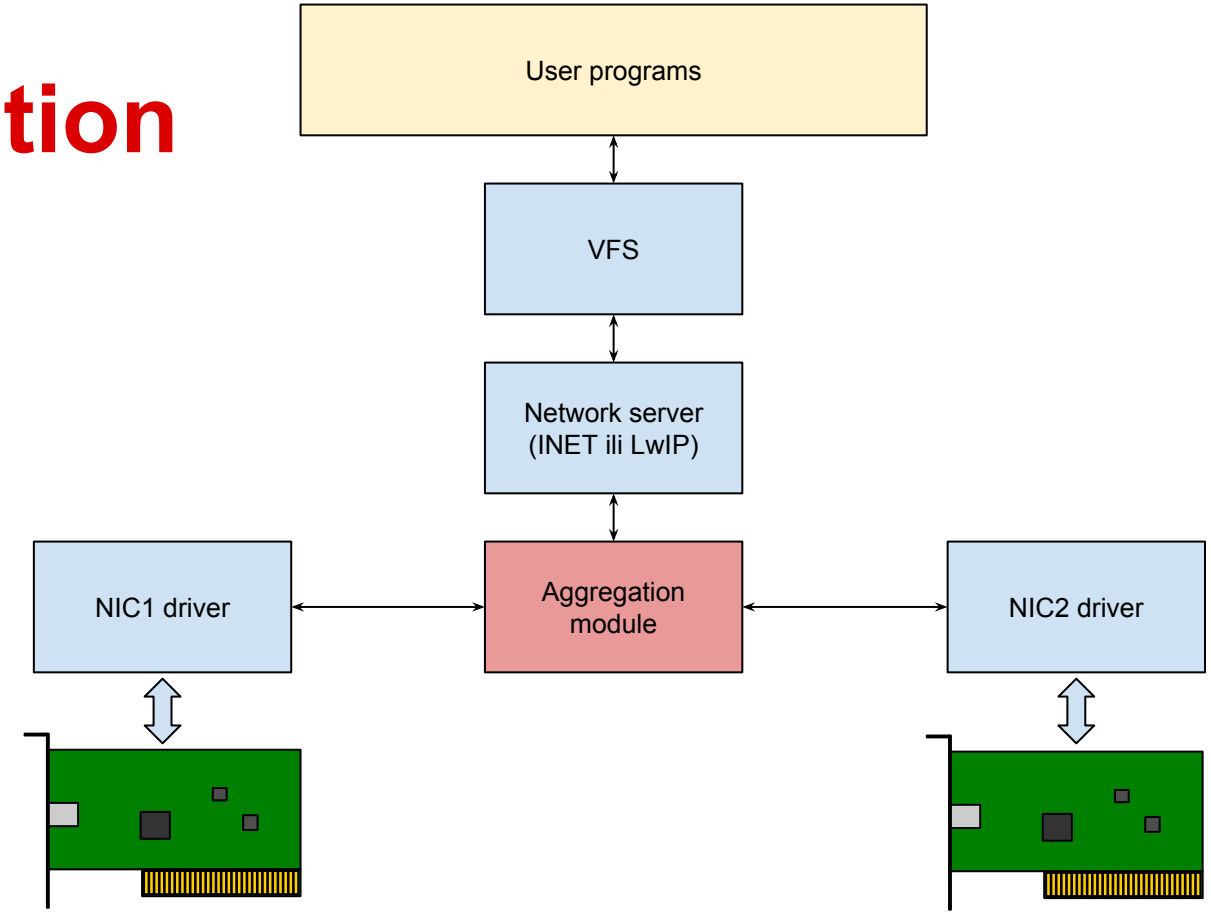
# MINIX3 networking subsystem

Where to put LA?



# MINIX3 LA Implementation

Insert aggregation driver between INET and the real drivers...

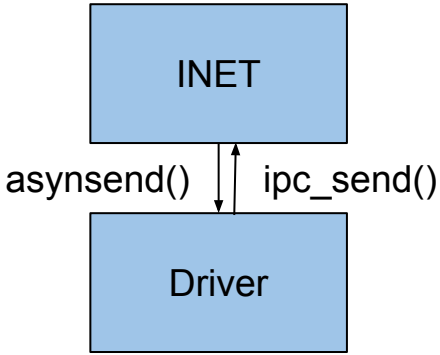


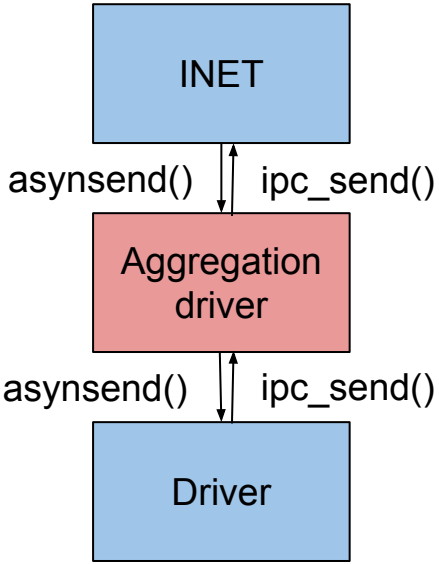
# Insertion of the LA driver

- Subscribe to **`"drv.net.*"`** events
- React to DS_DRIVER_UP event - get endponit
- Register at DS as a regular network driver

# LA driver/server IPC

- Driver to the server (INET): ipc_send()
- Server to the (NIC) driver: asynsend()

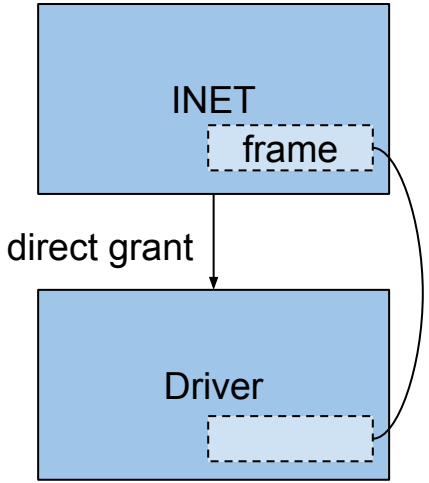


# MINIX3 Frame Send & Receive

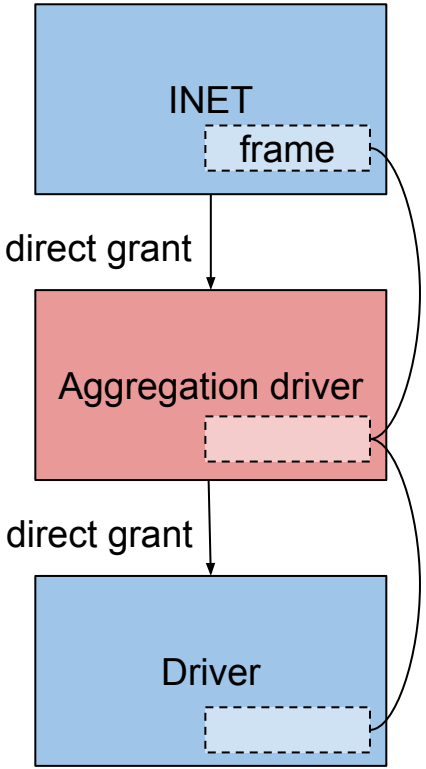
1. INET sends the grant to the NIC driver

2. NIC driver copies the frame from/to INET using the grant



# MINIX3 LA Frame Send & Receive

1. INET sends the grant to the LA driver
2. LA driver copies the frame from/to INET using the grant
3. LA driver sends the grant to the NIC driver
4. NIC driver copies the frame from/to LA driver using the grant

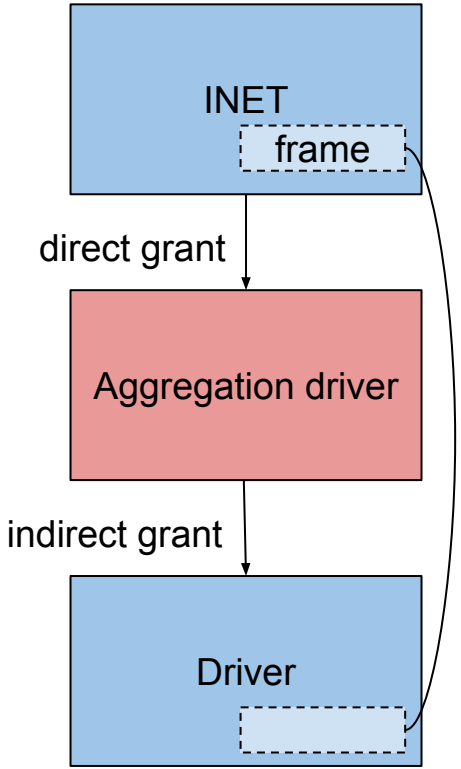


# MINIX3 LA Frame Send & Receive (2)

1. INET sends the grant to the LA driver

2. LA driver converts grant to indirect and sends it to the NIC driver

3. NIC driver copies the frame from/to INET using the indirect grant



# MINIX3 LA: Link Monitoring

- MII monitoring: MINIX3 network drivers cannot report MII status, so: no MII monitoring

- ARP monitoring:
  1. If there are no frames received within **`arp_interval`**, send ARP request frame to specified IP target
  2. Still no frames received within **`arp_interval`** - link is dead, switch to next live backup link, i.e. time for dead link detection: 2 x **`arp_interval`**

# MINIX3 LA: Configuration

/etc/inet.conf

```
eth0 lnprox 0 { default; } ;
eth1 rtl8169 0 { } ;
eth2 rtl8169 1 { } ;
```

/etc/rc.net

```
ifconfig -I /dev/ip0 -n
255.255.255.0 -h 192.168.1.15
add_route -g 192.168.1.1
```

/etc/system.conf

```
service lnprox
{
    uid 0;
};
```

/etc/lnprox.conf

```
slave1=rtl8169_0
slave2=rtl8169_1
aggmode=failover  #slave1 is active
arp_interval=1
arp_iptarget=192.168.1.1
```

# Performance/Throughput Test

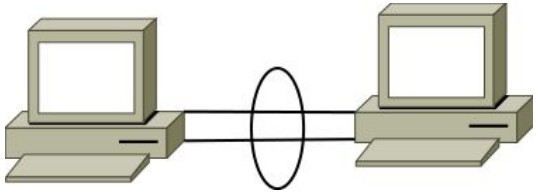
Question:

How much overhead is introduced because of the aggregation driver?

# Performance/Throughput Test

Setup:

- 2 computers
  (Intel E2160 CPU)
- 100BASE-TX NICs
  (Realtek 8139)
- 1000BASE-T NICs
  (Realtek 8169)



# Performance/Throughput Test

Type of throughput test:

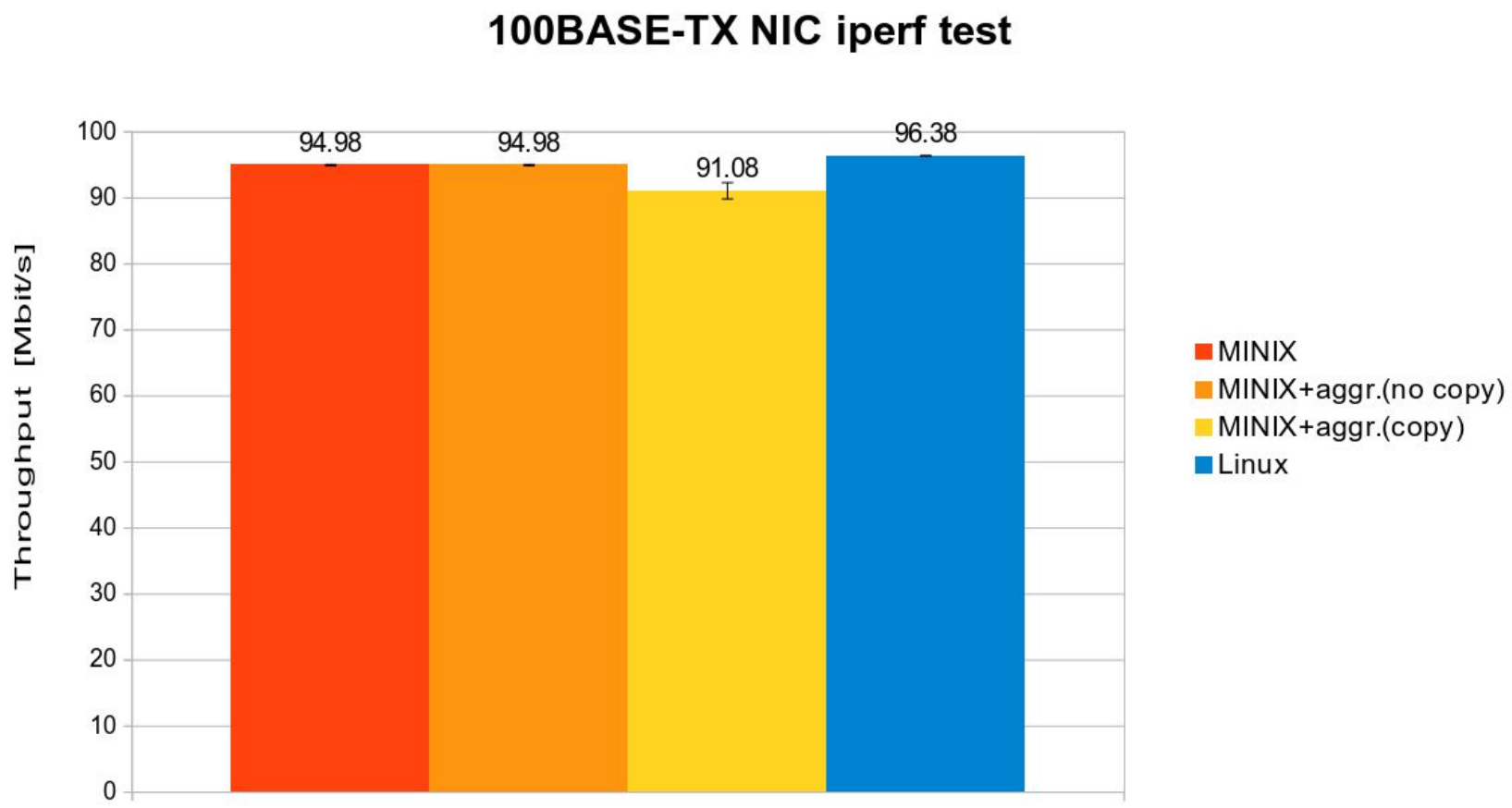
- TCP tests (iperf)
- Direct send to the driver (raw broadcast)

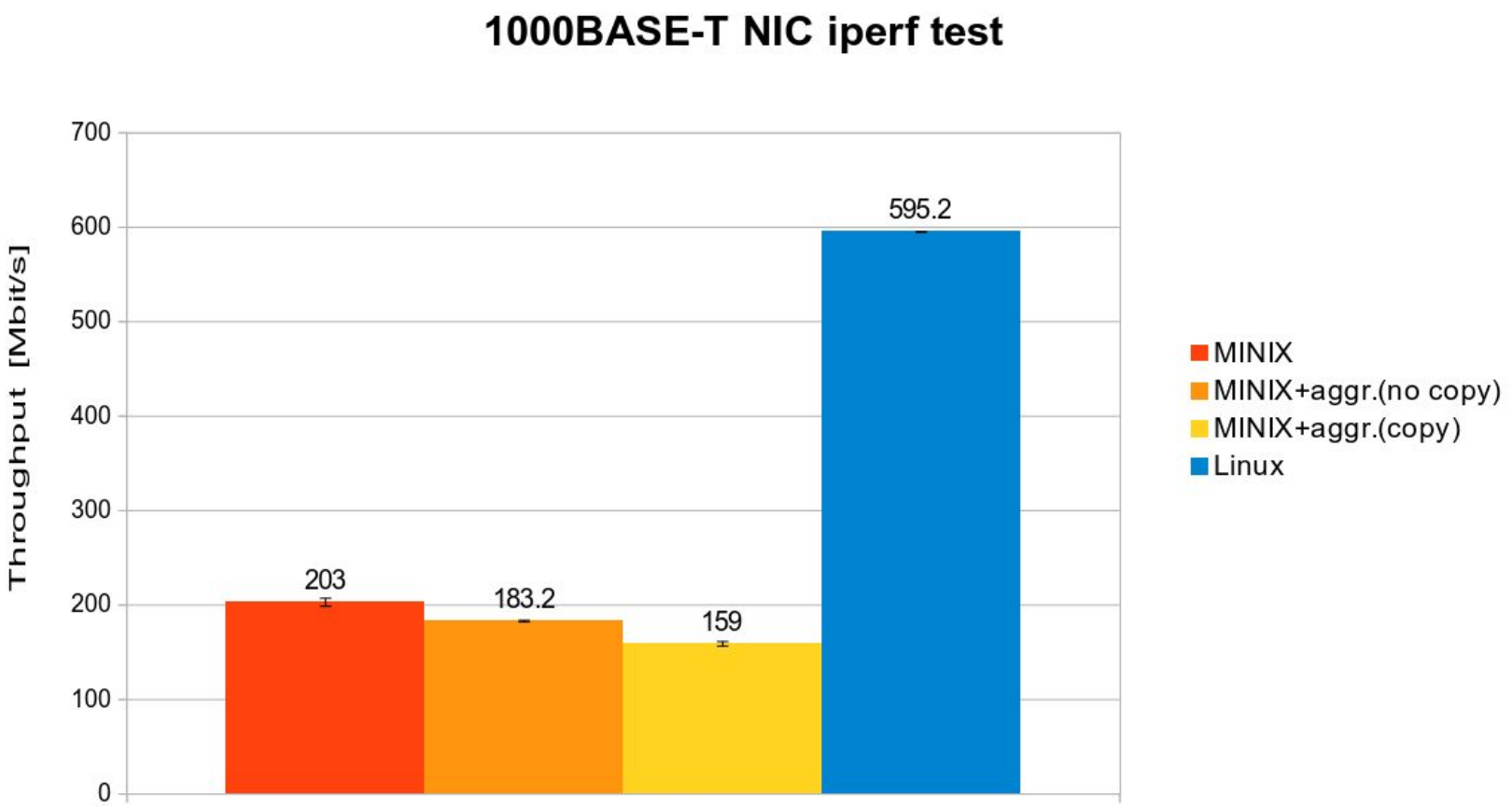
Type of OS setup:

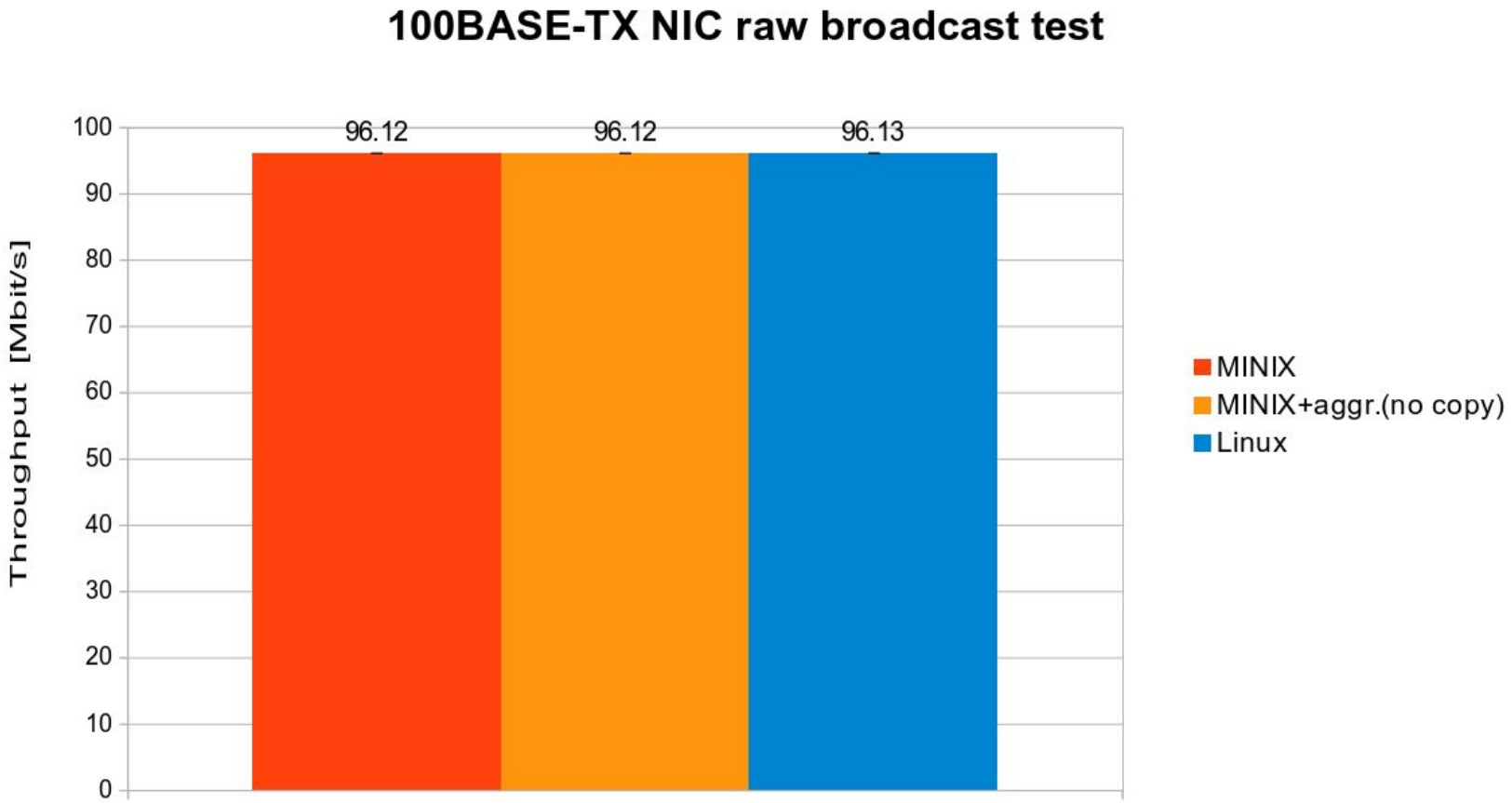
- MINIX3
- MINIX3 + aggregation (with and without extra copying)
- Linux

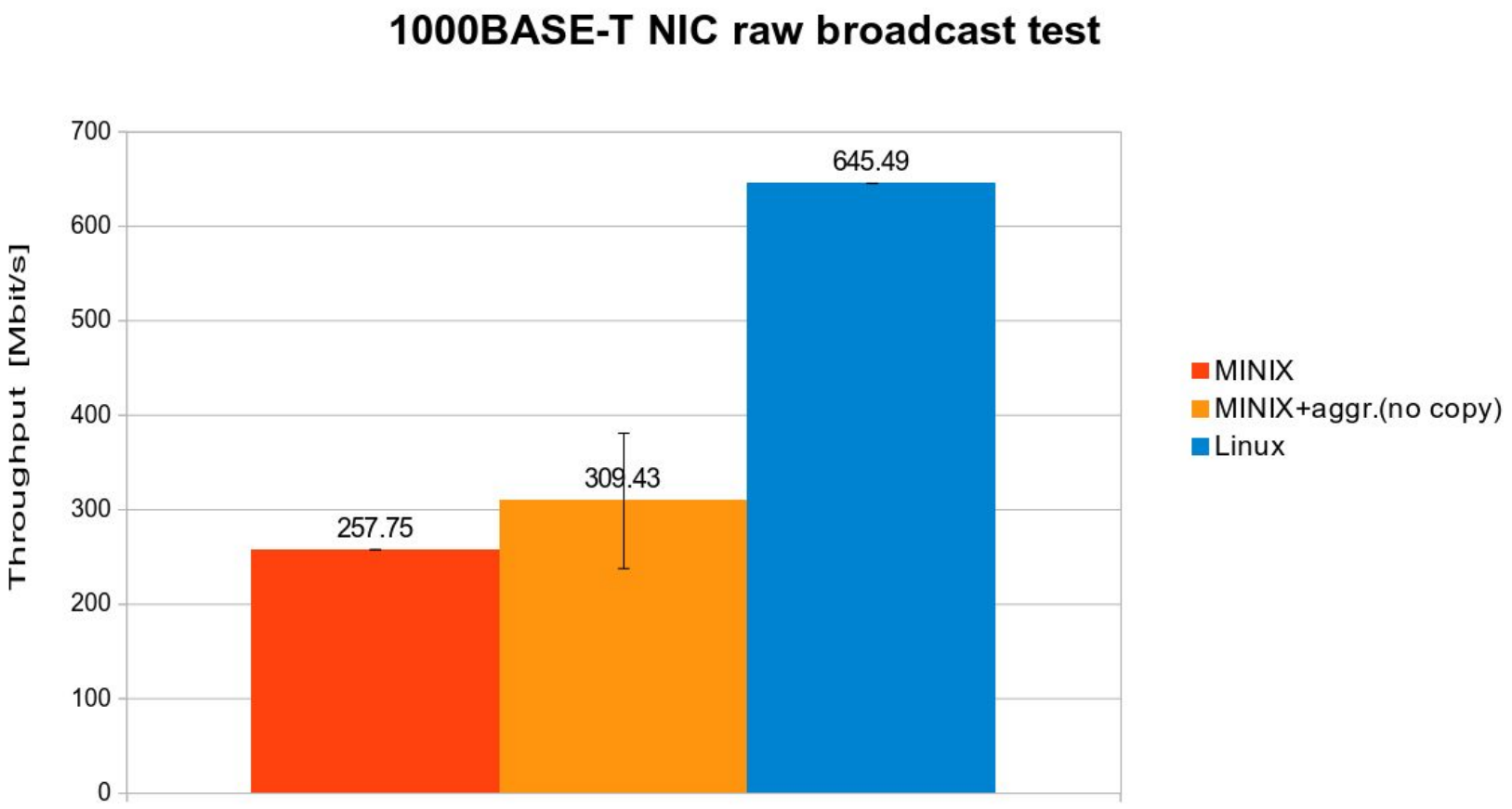
Type of NICs:

- 100BASE-TX
- 1000BASE-T


100BASE-TX NIC iperf test
Throughput [Mbit/s]
100
90
80
70
60
50
40
30
20
10
0
94.98
94.98
91.08
96.38
MINIX
MINIX+aggr.(no copy)
MINIX+aggr.(copy)
Linux


1000BASE-T NIC iperf test
Throughput [Mbit/s]
700
600
500
400
300
200
100
0
203
183.2
159
595.2
MINIX
MINIX+aggr.(no copy)
MINIX+aggr.(copy)
Linux


100BASE-TX NIC raw broadcast test
Throughput [Mbit/s]
100
90
80
70
60
50
40
30
20
10
0
96.12
96.12
96.13
MINIX
MINIX+aggr.(no copy)
Linux


1000BASE-T NIC raw broadcast test
Throughput [Mbit/s]
700
600
500
400
300
200
100
0
645.49
309.43
257.75
MINIX
MINIX+aggr.(no copy)
Linux

# Conclusion

MINIX3 can do link aggregation, especially for slower links.

Network stack could be improved and made more efficient.

# Future work

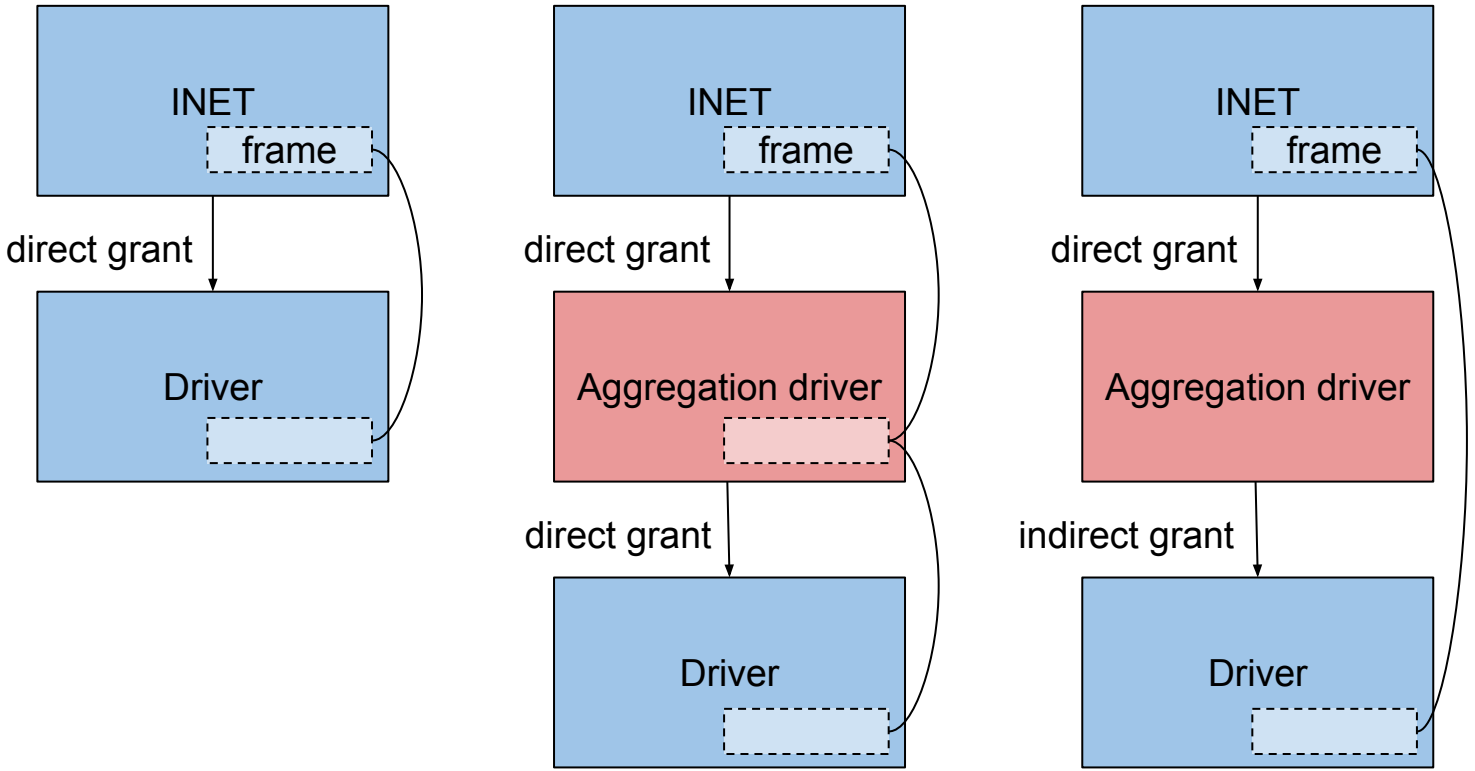
Add some of the other aggregation modes.

802.1q VLAN implementation?

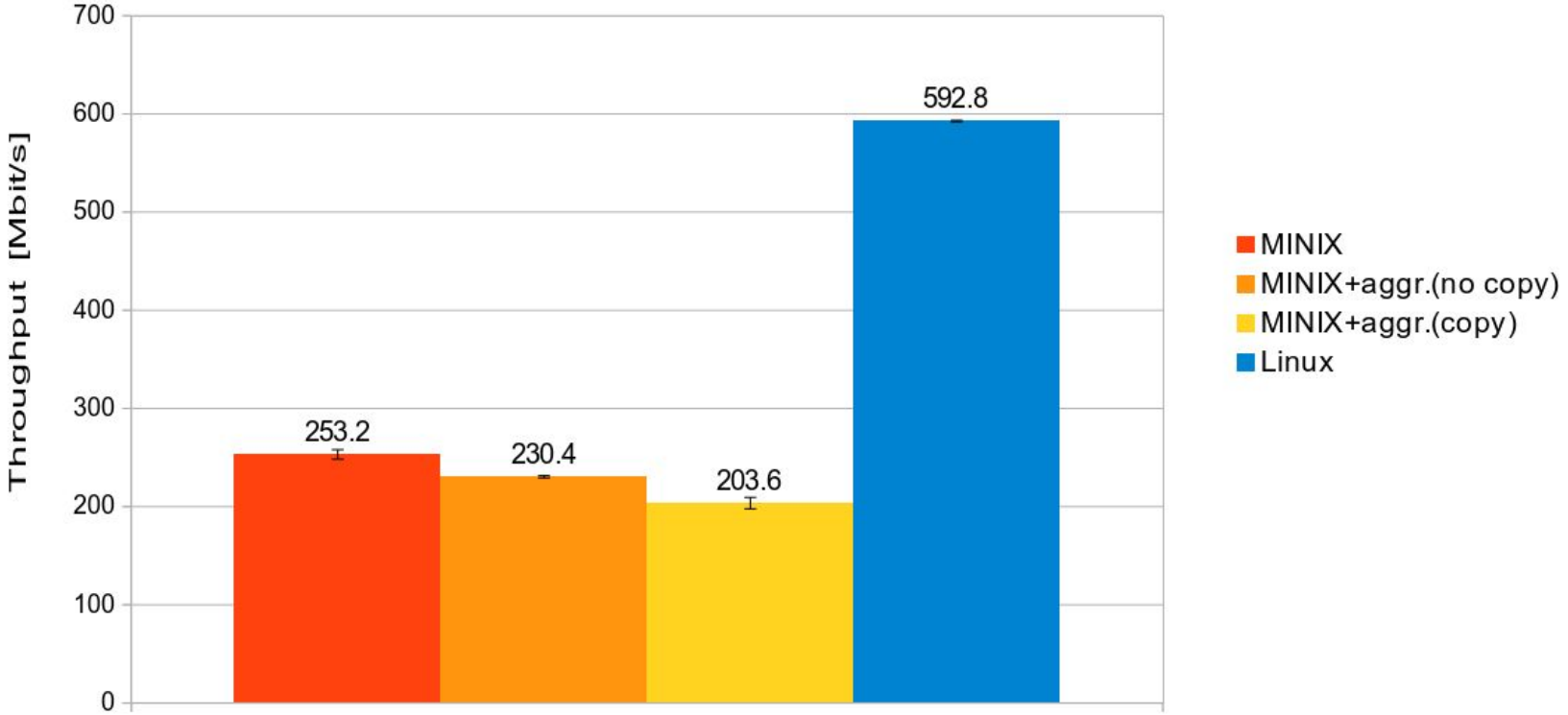
# Questions?

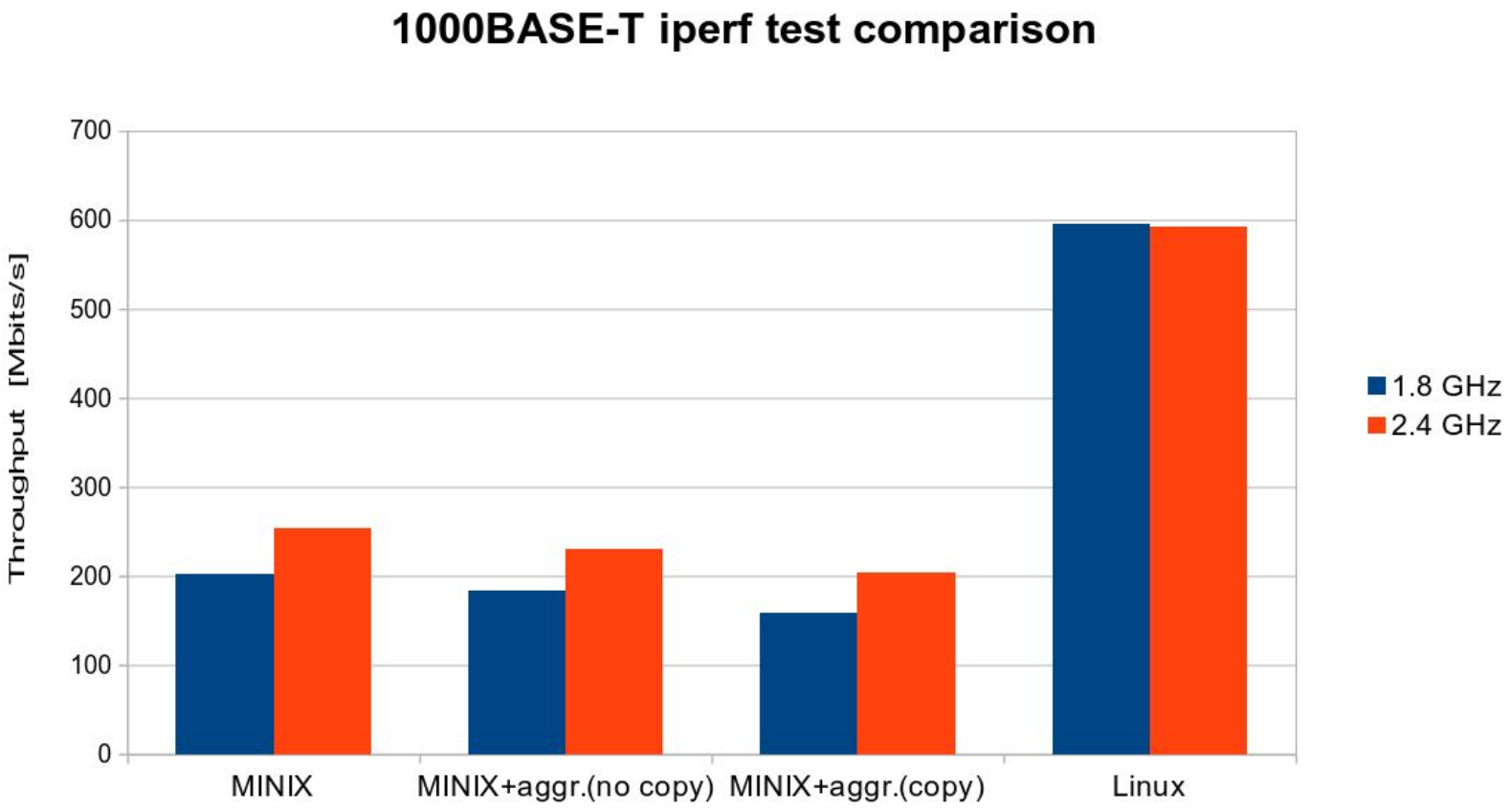
Hidden slides...

# MINIX3 LA Frame Forwarding

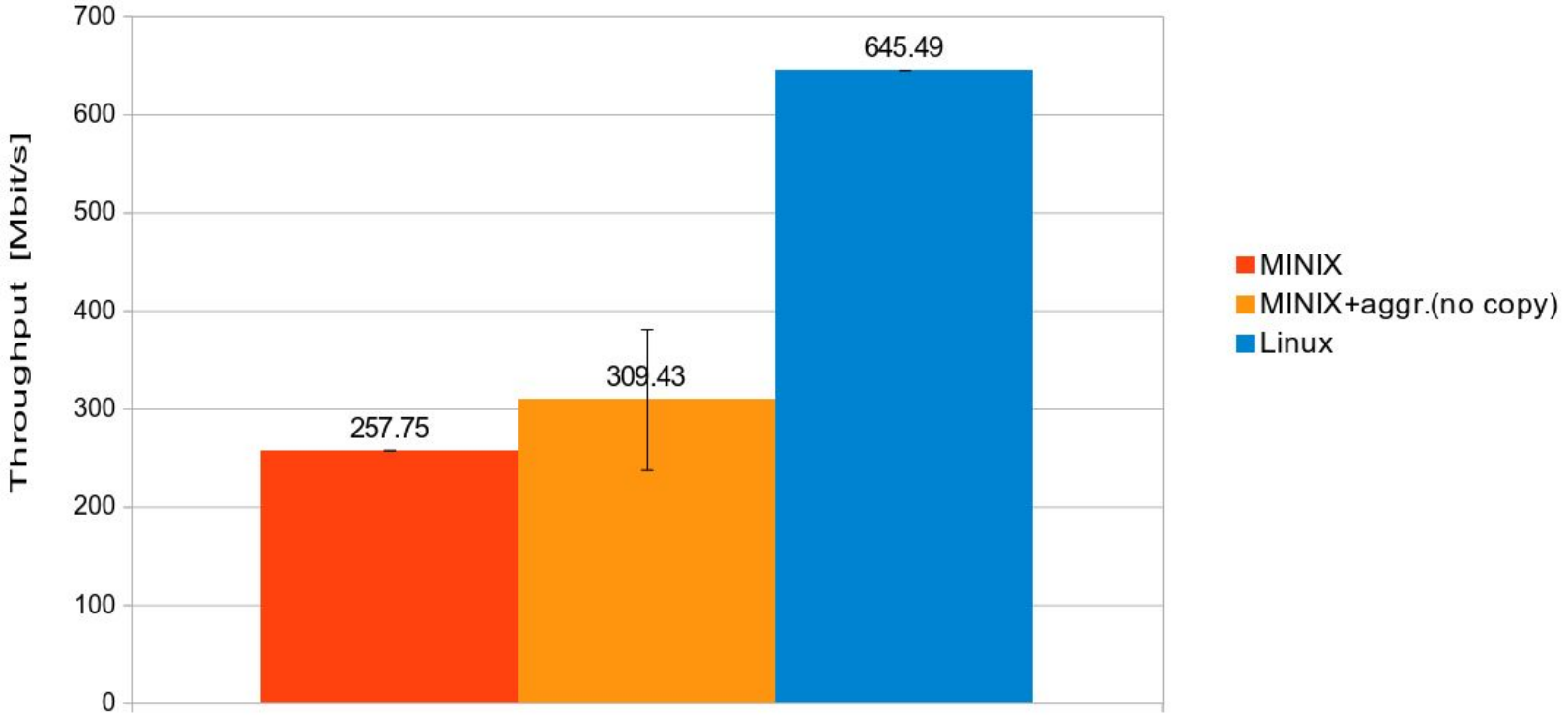


# 1000BASE-T NIC iperf test @2.4GHz

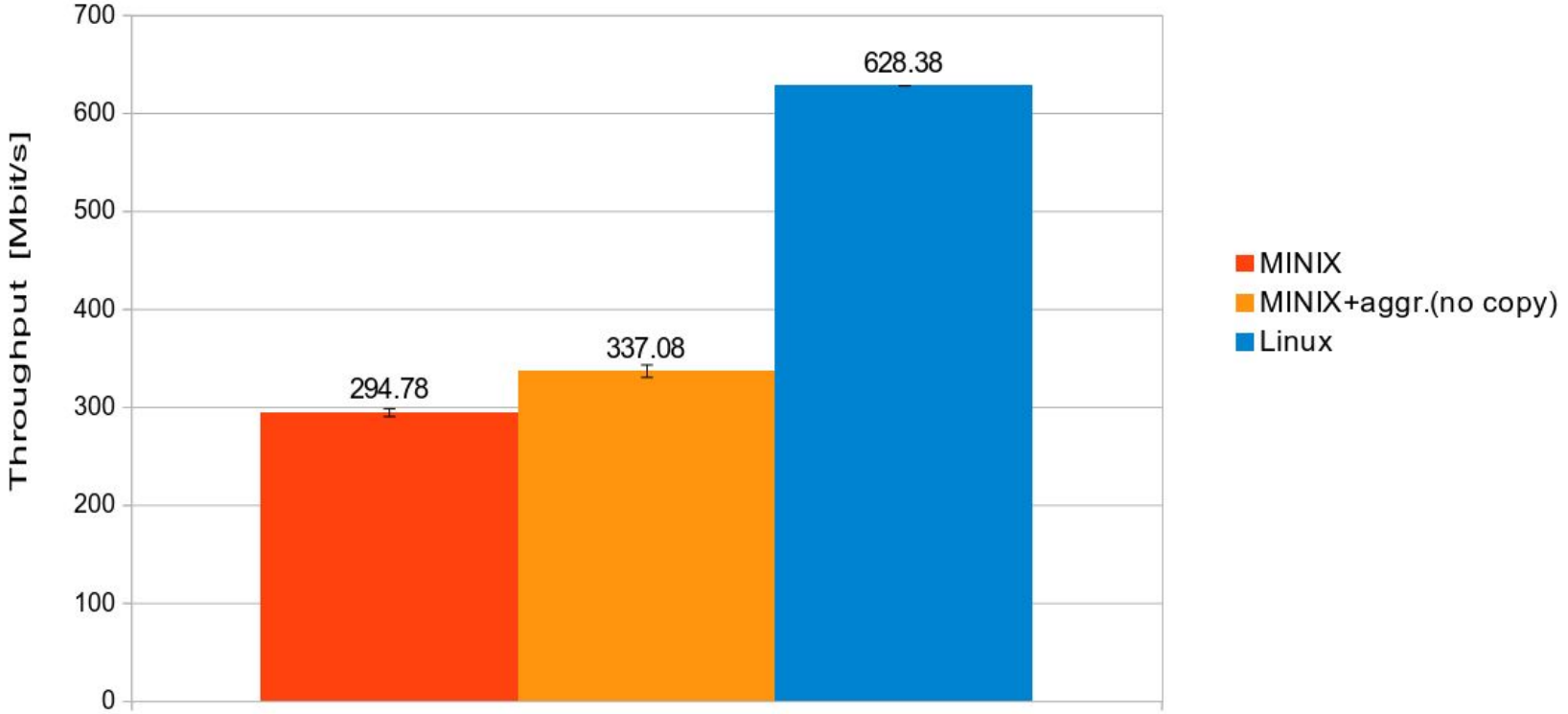



1000BASE-T iperf test comparison
Throughput [Mbits/s]
700
600
500
400
300
200
100
0
MINIX
MINIX+aggr.(no copy)
MINIX+aggr.(copy)
Linux
1.8 GHz
2.4 GHz

# 1000BASE-T NIC raw broadcast test 1.8GHz



# 1000BASE-T NIC raw broadcast test 2.4GHz



# Origins of Ethernet LA

## 1990s

Kalpana Inc. invented Ethernet switch and EtherChannel (acquired by Cisco in 1994)

Donald Becker wrote Beowulf patches for Linux

# LA implementation on MINIX3: Advantages & Flaws

+ Isolation of the LA module

- Unable to share data structures directly (no shared memory)