

# Pixelflut v6

## As fast as possible?

Sebastian Bernauer

November 14, 2019

# Inhalt

## Bestandsaufnahme

Pixelflut (v4)

Pixelflut v6

## Architektur

XDP

DPDK

## Netzarchitektur

Pixelflut v6

Sebastian  
Bernauer

Bestandsaufnahme

Pixelflut (v4)

Pixelflut v6

Architektur

XDP

DPDK

Netzarchitektur

# Pixelflut (v4)

- ▶ ASCII-Befehle über TCP, Zeile für Zeile

## Verwendung

```
echo "PX x y rrggbb[aa]" | nc 127.0.0.1 1234
```

- ▶ Server: shoreline von TobleMiner
  - ▶ 37G mit Dual AMD EPYC (!)
  - ▶ <https://github.com/TobleMiner/shoreline>

- ▶ Server: shoreline von TobleMiner
  - ▶ 37G mit Dual AMD EPYC (!)
  - ▶ <https://github.com/TobleMiner/shoreline>
- ▶ Client: sturmflut
  - ▶ 80G mit Laptop
  - ▶ <https://github.com/TobleMiner/sturmflut>

- ▶ Server: shoreline von TobleMiner
  - ▶ 37G mit Dual AMD EPYC (!)
  - ▶ <https://github.com/TobleMiner/shoreline>
- ▶ Client: sturmflut
  - ▶ 80G mit Laptop
  - ▶ <https://github.com/TobleMiner/sturmflut>

## Problem

Ein Client macht Server locker platt

Das Format der zu sendenden IPv6-Adresse:

- ▶ 64 bit festes Prefix
- ▶ 16 bit X-Koordinate
- ▶ 16 bit Y-Koordinate
- ▶ 8 bit R
- ▶ 8 bit G
- ▶ 8 bit B
- ▶ 8 bit Padding

Gesamt ergibt sich: Prefix:XXXX:YYYY:RRGG:BBPP

## Beispiel

ping 4000:42:0:0:0505:ffaa:ccff

- ▶ Netz: 4000:42::/64
- ▶ X = 5
- ▶ Y = 5
- ▶ Farbe: 0xffaacc



# Implikationen

- ▶ Ganz viele kleine Packete

# Implikationen

- ▶ Ganz viele kleine Packete
- ▶ Use moare bandwith as pixelflut v4 :)  
"PX 123 123 rrggbb\n" (18 byte)  
vs  
 $\geq$  64 byte Packet
- ▶ Linux kernel kommt an seine Grenzen

# Software

► Server: ?

Pixelflut v6

Sebastian  
Bernauer

Bestandsaufnahme

Pixelflut (v4)

**Pixelflut v6**

Architektur

XDP

DPDK

Netzarchitektur

# Software

- ▶ Server: ?
- ▶ Client: ?

Pixelflut v6

Sebastian  
Bernauer

Bestandsaufnahme

Pixelflut (v4)

**Pixelflut v6**

Architektur

XDP

DPDK

Netzarchitektur

- ▶ Server: ?
- ▶ Client: ?
- ▶ Selber machen!

- ▶ Üblicher Weg: Linux Kernel und C-Programm
  - Wie shoreline
  - Problem: Packete/s im Kernel

- ▶ Üblicher Weg: Linux Kernel und C-Programm
  - Wie shoreline
  - Problem: Packete/s im Kernel
- ▶ FPGA
  - Cool
  - Problem: Teuer

- ▶ Üblicher Weg: Linux Kernel und C-Programm
  - Wie shoreline
  - Problem: Packete/s im Kernel
- ▶ FPGA
  - Cool
  - Problem: Teuer
- ▶ XDP (Express Data Path)
  - Programm läuft direkt im Linux kernel
  - Kein Kopieren in userspace nötig
  - Keine Kontextwechsel nötig



- ▶ DPDK (Data Plane Development Kit)
  - Netzwerkkarte wird komplett im userspace angesprochen
  - Keine Interrupts, sondern Poll basiert

- ▶ Hat wunderbar funktioniert
- ▶ Pixelinformationen wurden in einer Datenstruktur im Kernel abgelegt
- ▶ Für das Erfragen eines Elementes der Datenstruktur
  - Alle Pixel aus Datenstruktur lesen: 2s
    - $1920 \times 1080 = 2.073.600$  System calls!
  - Nix da mit 60 fps
- ▶ Läuft auf jeder Netzwerkkarte

- ▶ Hat wunderbar funktioniert
- ▶ Pixelinformationen wurden in einer Datenstruktur im Kernel abgelegt
- ▶ Für das Erfragen eines Elementes der Datenstruktur
  - Alle Pixel aus Datenstruktur lesen: 2s
    - $1920 \times 1080 = 2.073.600$  System calls!
  - Nix da mit 60 fps
- ▶ Läuft auf jeder Netzwerkkarte

## Problem

Wir kriegen die Daten nicht aus der Datenstruktur im Kernel

## Möglichkeiten zur Lösung des Problems

- ▶ Kernel contribution: Komplette Datenstruktur mit einem Systemcall aus dem Kernel in den userspace kopieren

## Möglichkeiten zur Lösung des Problems

- ▶ Kernel contribution: Komplette Datenstruktur mit einem Systemcall aus dem Kernel in den userspace kopieren
- ▶ **Achtung abgespaced:**

## Möglichkeiten zur Lösung des Problems

- ▶ Kernel contribution: Komplette Datenstruktur mit einem Systemcall aus dem Kernel in den userspace kopieren
- ▶ **Achtung abgespaced:**
- ▶ Man kann in XDP Pakete manipulieren und an den Absender zurück senden.
  - Server empfängt alle Pixel-Pakete und aktualisiert Datenstruktur im Kernel
  - Client erzeugt einen konstanten Strom um 60fps rendern zu rendern, in einem Packet wird mittels Offset und Länge nach Pixelwerten gefragt
  - Server modifiziert Packet (welches richtige Länge hat ;) ) mit den gewünschten Pixeln und schickt es an den Absender zurück (In diesem Schritt kann auf Datenstruktur zugegriffen werden)
  - Keine Modifikation am Kernel notwendig

- ▶ Netzwerkkarte wird komplett im Userspace angesprochen
  - Keine Kontextwechsel
  - Keine Interrupts, sondern pollbasiert
  - Es gibt keine bequemen Kernel-Features wie Ping-Antwort oder Neighbor Discovery
  - Nur bestimmte Karten unterstützt (z.B. am Mellanox ConnectX-3)

# DPDK

Pixelflut v6

Sebastian  
Bernauer

Bestandsaufnahme

Pixelflut (v4)

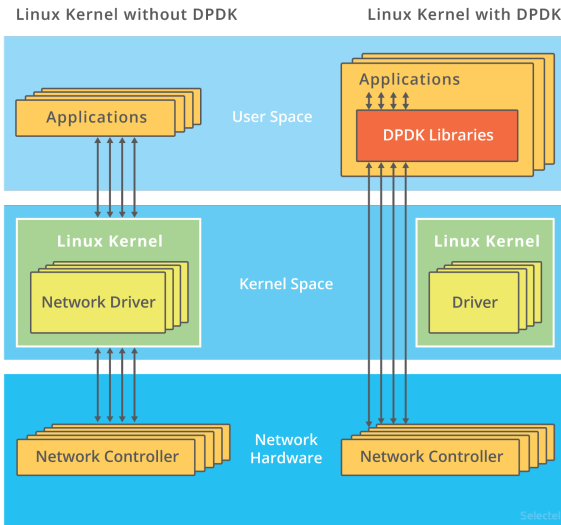
Pixelflut v6

Architektur

XDP

DPDK

Netzarchitektur

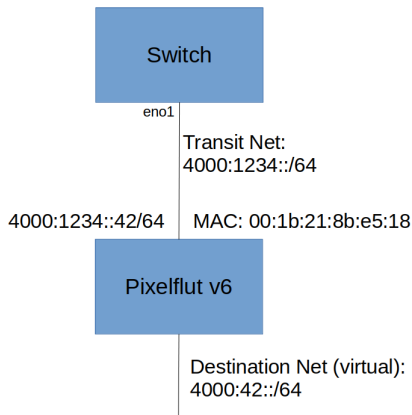




- ▶ Neighbor Discovery von  $> 2$  Millionen IPv6 Adressen (Full HD)?

- ▶ Neighbor Discovery von  $> 2$  Millionen IPv6 Adressen (Full HD)?  
→ Nein!

- ▶ Neighbor Discovery von  $> 2$  Millionen IPv6 Adressen (Full HD)?
  - Nein!
  - Transit-Netz mit einer einzigen next-hop IP-Adresse



Configuration for switch::

```
(optional) ip -6 addr add 4000:1234::1/64 dev eno1
```

```
ip -6 neigh add 4000:1234::42 lladdr 00:1b:21:8b:e5:18 dev eno1
```

```
ip -6 route add 4000:42::/64 dev eno1 nexthop via 4000:1234::42
```

The clients must send their packets to 4000:42::xxxx:yyyy:rrgg:bb00