Synchrone Audio- und Videowiedergabe auf mehreren Geräten

Chemnitzer Linux-Tage 2018

Michael – m.tretter@pengutronix.de

Problemstellung

- Sportübertragung
- Multi-Room Audio
- Video-Wand

Gliederung

- Real-time Transport Protocol
- Synchronisierung von Uhren
- RFC 7273 – RTP Clock Source Signalling
- Demo

GStreamer

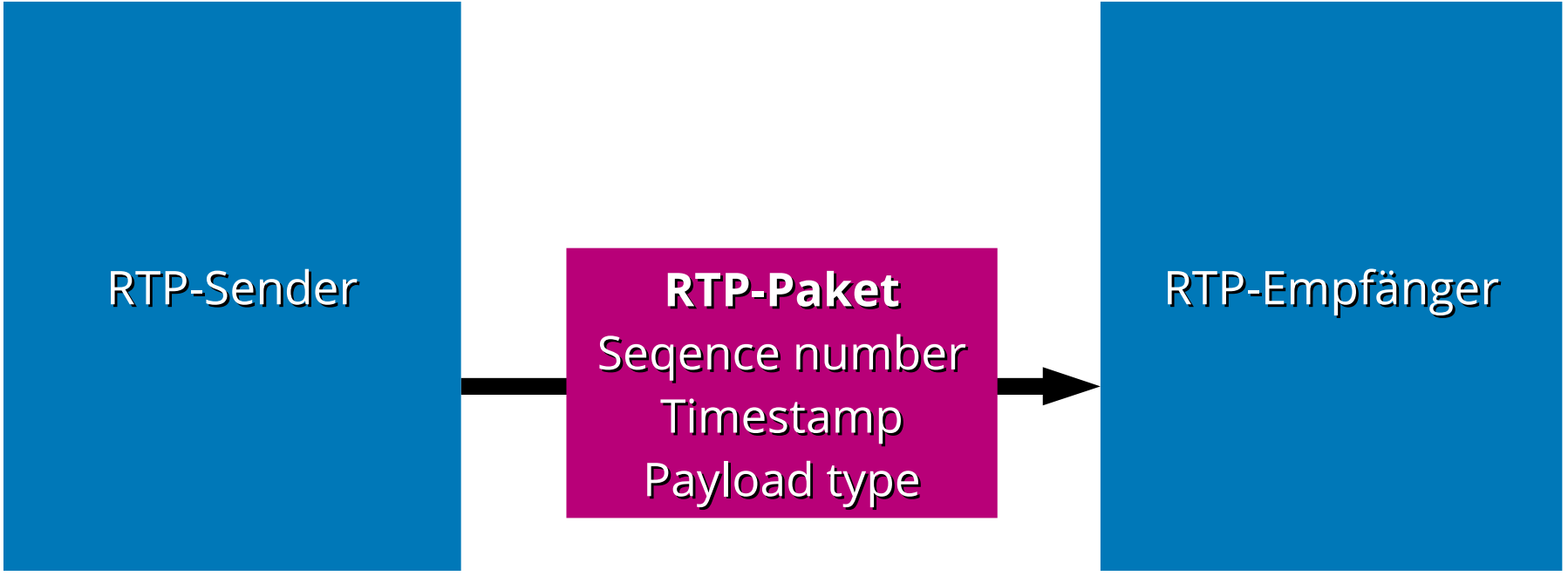
- Multimedia-Framework
- Audio und Video
- Pipelines aus Elementen


```
gst-launch-1.0 -v \
    filesrc location=./ehren-paper_lights-96.opus \
    oggdemux ! opusparse ! opusdec ! pulsesink
```

RTP – Real-time Transport Protocol

- Übertragung von Echtzeit-Daten
- Audio, Video, Simulationsdaten
- Üblicherweise über UDP
- Zum Beispiel für IP-Telefonie

RTP – RFC3550


RTP-Sender
RTP-Paket
Seqence number
Timestamp
Payload type
RTP-Empfänger

RTP – Sender

```
gst-launch-1.0 -v \
    filesrc location=ehren-paper_lights-96.opus ! oggdemux ! \
    rtpopuspay ! \
    udpsink host=localhost port=5004
```

- Payloader hängt von Payload ab
- $ gst-inspect-1.0 | grep “rtp.*pay”
- Empfänger-Adresse: Unicast, Multicast, Broadcast

RTP – Empfänger I

gst-launch-1.0 -v \

udpsrc address=**localhost** port=**5004** caps=”**application/x-rtp**” ! \

rtpopusdepay ! opusdec ! pulsesink

- Depayloader hängt wieder von Payload ab
- Der udpsrc muss mitgeteilt werden, dass sie RTP empfängt

RTP – Empfänger II

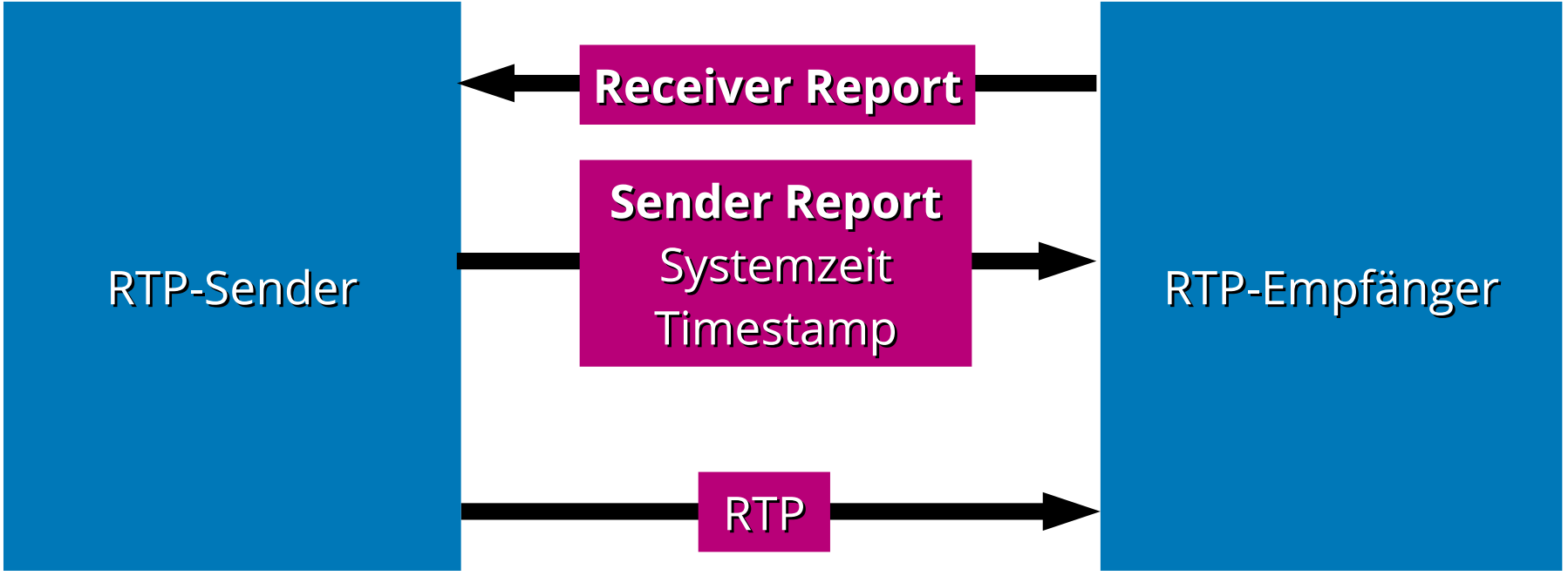
```
gst-launch-1.0 -v \
udpsrc address=localhost port=5004 caps="application/x-rtp" ! \
rtpjitterbuffer ! rtpopusdepay ! opusdec ! pulsesink
```

- Erkennt fehlende Pakete
- Korrigiert falsche Paket-Reihenfolge
- Rechnet RTP Timestamp in Gstreamer-Zeit um

RTP – Synchronisierung?

- Timestamp hat zufälligen Startwert
- Schrittgröße ist abhängig vom Payload
- Zeitliche Einordnung der Pakete untereinander
- Kein Bezug zu globaler Uhrzeit
- Nicht zur Synchronisation geeignet

RTCP – RTP Control Protocol



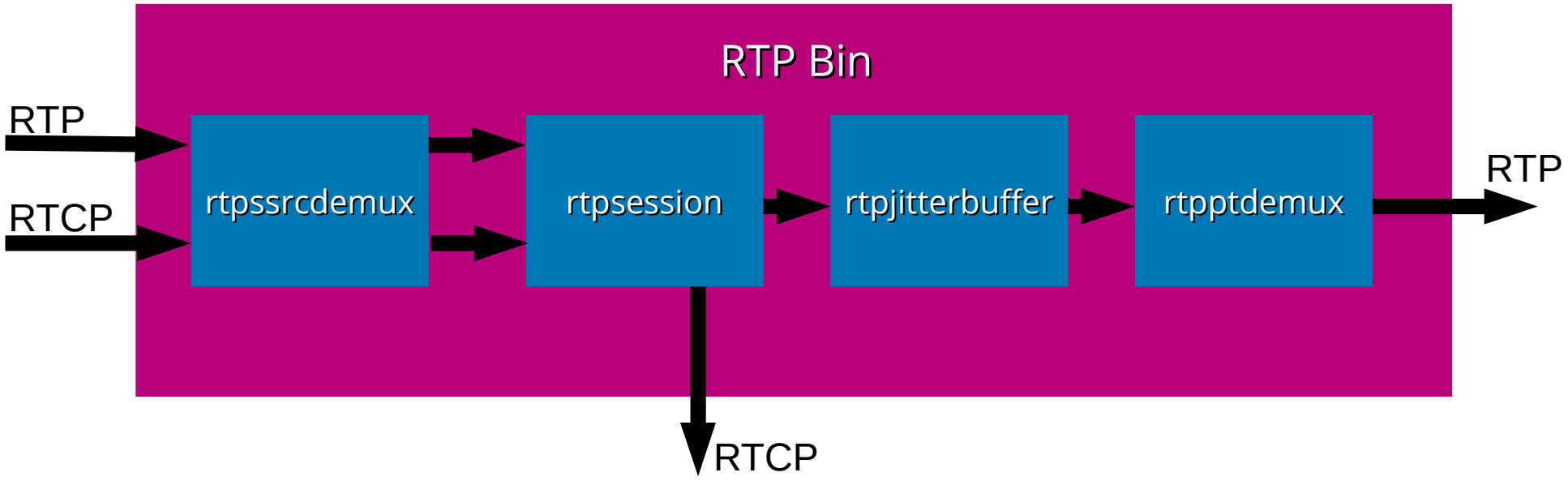
RTCP – Sender

```
gst-launch-1.0 -v \
filesrc location=./ehren-paper_lights-96.opus ! Oggdemux ! \
rtpopuspay ! \
.send_rtp_sink rtpsession name=session \
.send_rtp_src ! udpsink host=224.1.1.1 port=5004  \
session.send_rtcp_src ! \
udpsink host=224.1.1.1 port=5005 sync=false async=false
```

RTCP – Empfänger

```
gst-launch-1.0 -v \
udpsrc address=localhost port=5004 caps="application/x-rtp" ! \
rtpjitterbuffer name=buffer ! rtpopusdepay ! \
opusdec ! pulsesink \
udpsrc address=localhost port=5005 ! buffer.sink_rtcp
```

RTP Bin


RTP Bin
RTP
RTCP
rtpssrcdemux
rtpsession
rtpjitterbuffer
rtpptdemux
RTP
RTCP

RTP Bin – Sender

```
gst-launch-1.0 -v \
filesrc location=./ehren-paper_lights-96.opus ! oggdemux ! \
rtpopuspay ! \
.send_rtp_sink_0 rtpbin name=bin \
.send_rtp_src_0 ! udpsink host=localhost port=5004 \
bin.send_rtcp_src ! udpsink host=localhost port=5005 \
udpsrc address=localhost port=5005 ! bin.recv_rtcp_sink_0
```

RTP Bin – Empfänger

```
gst-launch-1.0 -v \
udpsrc address=localhost port=5004 caps="application/x-rtp" ! \
.recv_rtp_sink_0 rtpbin name=bin ! \
rtpopusdepay ! opusdec ! pulsesink \
udpsrc address=localhost port=5005 ! bin.recv_rtcp_sink_0 \
bin.send_rtcp_src_0 ! udpsink host=localhost port=5005
```

RTCP – Synchronisierung?

- Sender Reports: RTP Timestamp → Sender-Uhr
- Synchronisierung mehrerer Streams vom selben Sender
- Keine Synchronisierung mehrerer Empfänger

SDP – Session Description Protocol

v=0
o=mtr 1188340656180883 1 IN IP4 192.168.1.1
s=CLT Synchrone Audio-Wiedergabe
t=0 0
m=**audio 5004** RTP/AVP **96**
c=IN IP4 **224.1.1.1**
a=rtpmap:96 OPUS/48000

- Beschreibt RTP-Session
- Payload
- Ports
- Multicast-Gruppe
- Via HTTP, E-Mail, …

SDP – Lokale Datei

- gst-launch-1.0 -v \

  filesrc **location=./stream.sdp** ! **sdpdemux** ! \

  rtpopusdepay ! opusdec ! pulsesink

- Benutzt rtpbin für RTP
- Erstellt udpsrc Elemente
- Liefert RTP-Pakete auf dem Source-Pad

SDP – Remote Server

gst-launch-1.0 -v \

souphttpsrc location=http://some.host/stream.sdp ! \

sdpdemux ! rtpopusdepay ! opusdec ! pulsesink

gst-launch-1.0 -v \

rtspsrc location=http://some.host/stream ! \

rtpopusdepay ! opusdec ! pulsesink

Synchronisierung?

- RTP: Timestamp hat zufälligen Startpunkt
- RTCP: Sender Reports bei selbem Sender
- SDP: Beschreibt Start- und Endzeiten von Streams

Wir brauchen synchrone Uhren!

Synchronisierung von Uhren – NTP

- Network Time Protocol
- Referenz: NTP Server
- Client fragt Server nach aktueller Zeit

Synchronisierung von Uhren – GNSS

- Navigationssatellitensysteme (GPS, GLONASS, Galileo)
- Referenz: Uhr im Satellit
- GPS-Empfänger vorhanden?
- Signal verfügbar?

Synchronisierung von Uhren – PTP

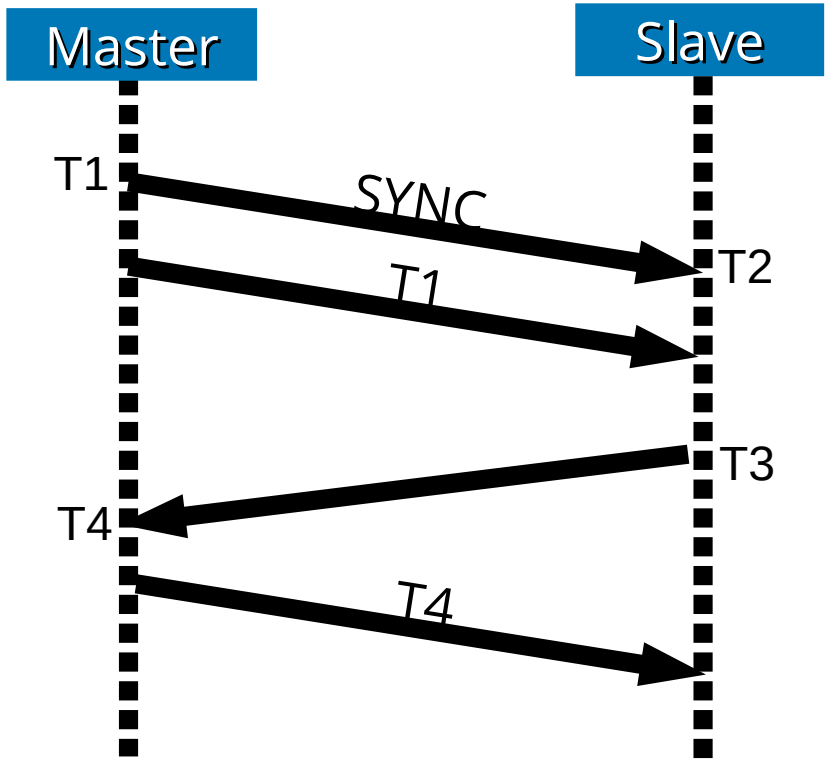
- Precision Time Protocol
- Referenz: Algorithmisch bestimmt
- Multicast im lokalen Netz

PTP – Precision Time Protocol

- IEEE 1588-2008
- Uhren sind prinzipiell gleichwertig
- Master, Slave, Grandmaster
- Best Master Clock Algorithm
- Clock Domain

PTP – Synchronisierung

- Offset durch Netzwerk
- d = ((T2 – T1) + (T4 – T3)) / 2
- Regelung nach T1 + d
- Hardwareunterstützung
  - Netzwerkcontroller
  - Switch



PTP – Linux PTP Project

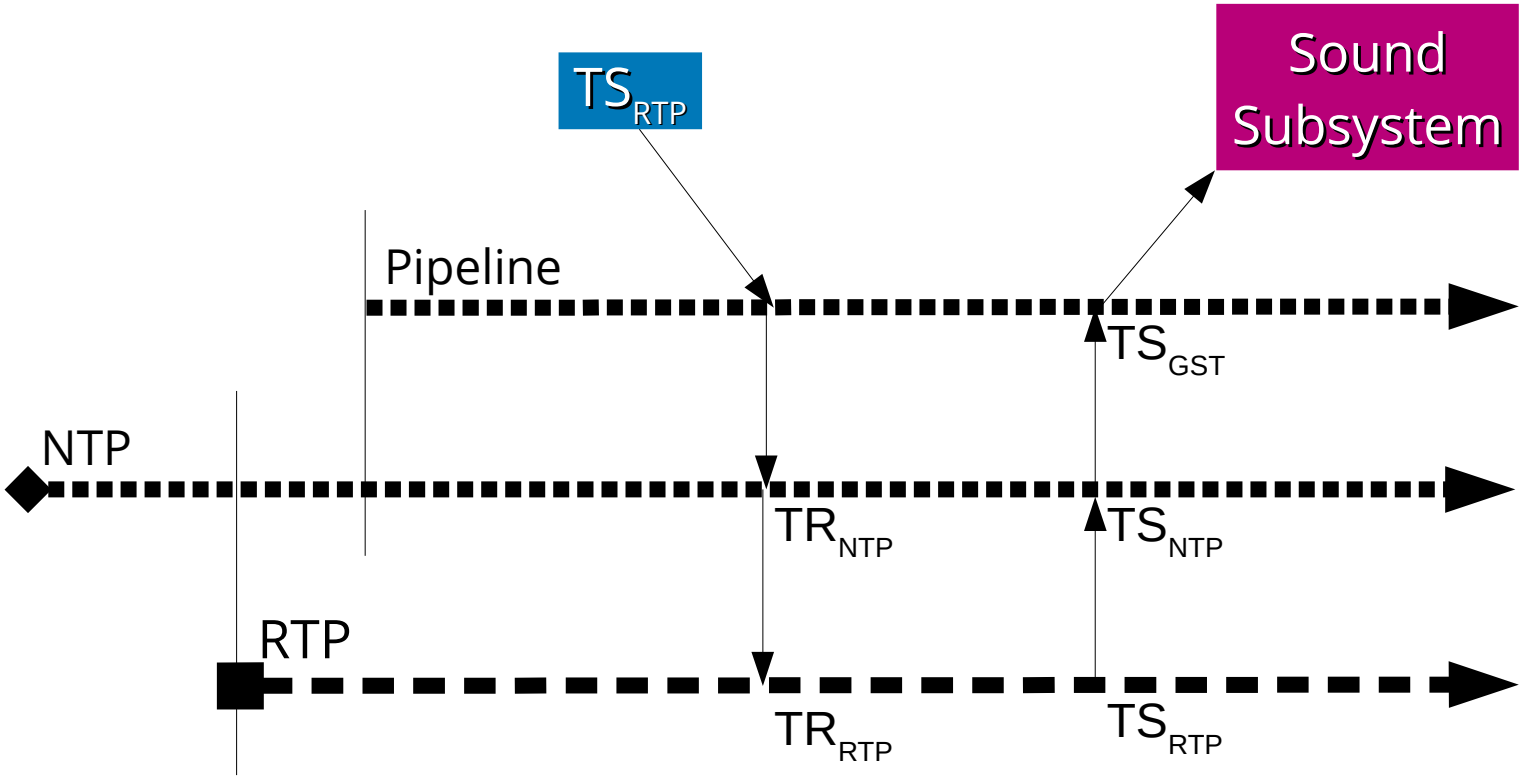
- Hardware prüfen: ethtool --show-time-stamping <devname>
- Paket installieren: linuxptp
- Daemon starten: ptp4l -i <devname> -m

RFC 7273 – RTP Clock Source Signalling

```
v=0
o=mtr 1188340656180883 1 IN IP4 192.168.1.1
s=CLT Synchrone Audio-Wiedergabe
t=0 0
a=ts-refclk:ntp=192.168.1.1
a=mediaclk:direct=3254
m=audio 5004 RTP/AVP 96
c=IN IP4 224.1.1.1
a=rtpmap:96 OPUS/48000
```

- Attribut in SDP
- Pro Stream oder Media
- Offset

RFC 7273 – Präsentationszeit



RFC 7273 – Sender

- GStreamer-Pipeline muss synchrone Uhr verwenden
- Mit gst-launch-1.0 nicht möglich

```
clock = GstNet.NtpClock.new(name, ntp_host, ntp_port, 0)
pipeline.use_clock(clock)
```

RFC 7273 – Empfänger

- Jitterbuffer: Property rfc7273-sync auf true und mode=4
- Bei sdpdemux nicht mit gst-launch-1.0 möglich
- Bei RTSP stream auf rtspsrc setzen

gst-launch-1.0 -v \

rtspsrc location=http://some.host/stream **rfc7273-sync=true mode=4** ! \

rtpopusdepay ! opusdec ! pulsesink

RFC 7273 – Achtung

- Latenz des Sound-Subsystem ist ungleich Null
- Gstreamer muss Sample zur richtigen Zeit übergeben
- Empfängerseite muss kalibriert werden
- Pulsesink: Property „render_delay“ anpassen

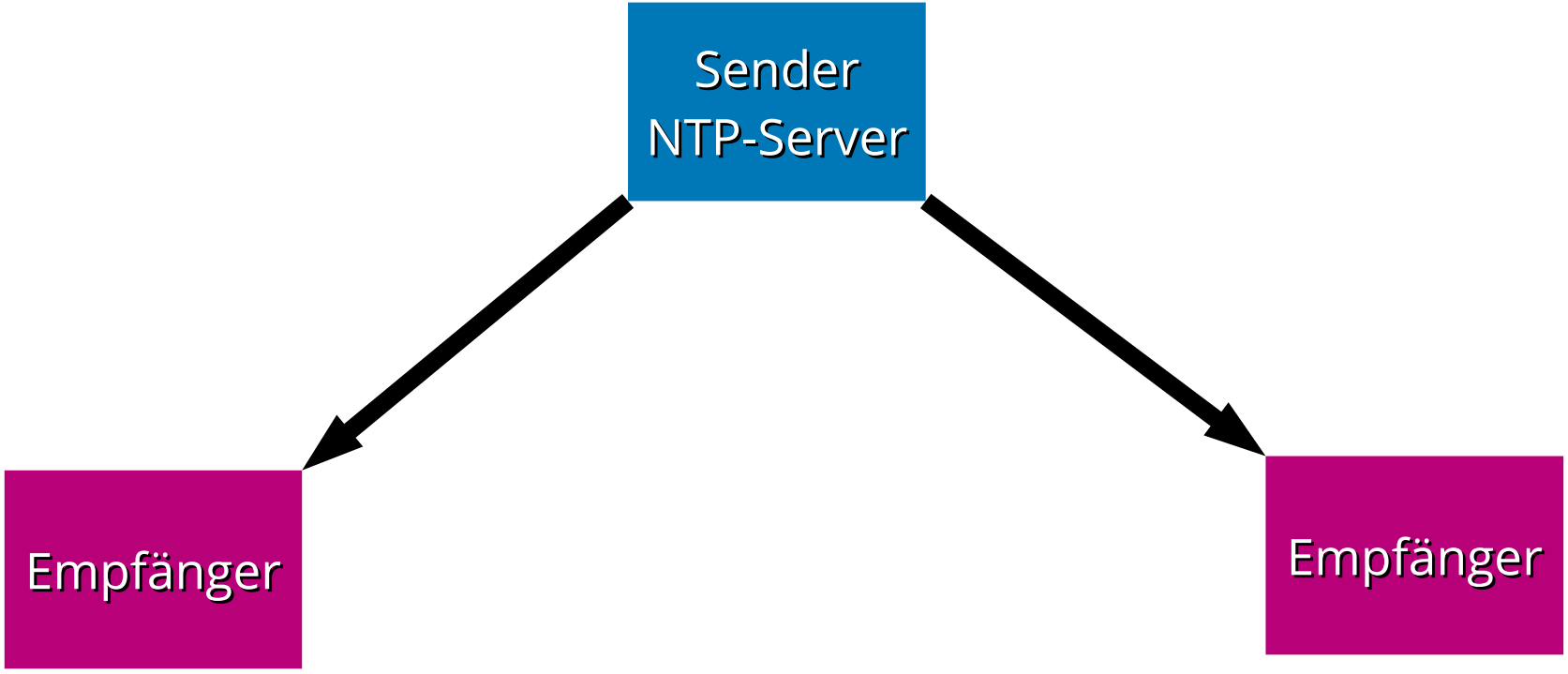
RFC 7273 – Test und Messung

- Vergleich mit dem Gehör
- Messen mit Line-In auf Mess-PC
- Visualisierung mit Oszilloskop

Video?

- Multiplex in Containerformat
- Separater RTP-Stream
- Payloader und Depayloader anpassen
- Richtigen Decoder und Video-Sink

Demo



Zusammenfassung

- Streaming mit RTP und RTCP
- Synchronisierung mit PTP
- RFC7273 zur Bekanntgabe der verwendeten Uhr
- Paper Lights by Ehren Starks (CC) BY-NC-SA

Synchrone Audio- und Videowiedergabe auf mehreren Geräten

Chemnitzer Linux-Tage 2018

Michael – m.tretter@pengutronix.de