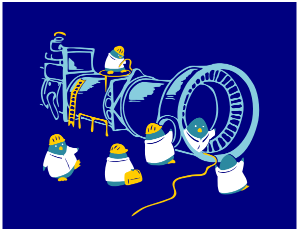
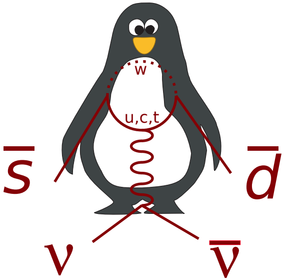
NA62 online pc farm design and implementation

Jonas Kunze

Universität Mainz

05.12.2011





$K^+ \to \pi^+ \nu\bar{\nu} \Leftrightarrow V_{td}$ of CKM matrix

SM branching ratio: $(8.5 \pm 0.7) \cdot 10^{-11}$

Measurement from BNL (Brookhaven)

The only experimental data so far (7 candidates) E787 and E949:
$BR(K^+ \to \pi^+ \nu\bar{\nu}) = (17.3^{+11.5}_{-10.5}) \cdot 10^{-11}$

NA62 aims $\sigma < 10\%$ with 100 events

$\approx 10^{13}$ K^+ decays required

High efficiency needed $\Rightarrow$ high data rate

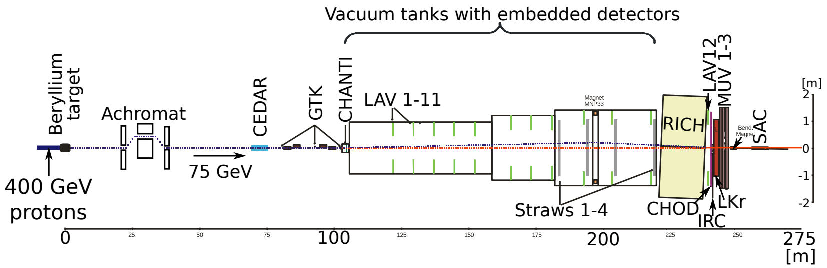
Signal-to-noise ratio of $1/10$ planned with an event rate of 10 MHz

Background channels to be suppressed by the detector

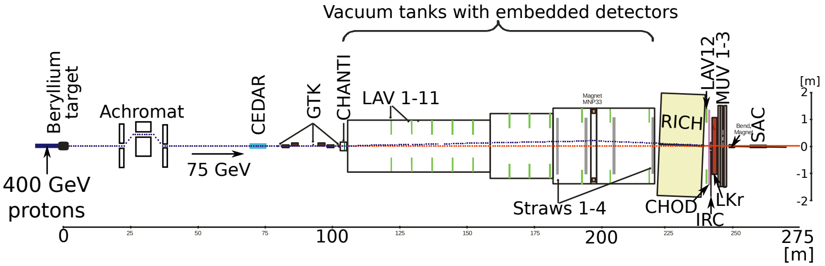
$K^+ \to \mu^+\nu$	$(63\%) \to \mu$ veto
$K^+ \to \pi^+\pi^0$	$(21\%) \to \gamma$ veto
$K^+ \to \pi^+\pi^+\pi^-$	$(6\%) \to$ charged particle veto
$K^+ \to \pi^0 e^+\nu$	$(5\%) \to \gamma$ veto
$K^+ \to \pi^0\mu^+\nu$	$(3\%) \to \gamma, \mu$ veto
$K^+ \to \pi^+\pi^0\pi^0$	$(2\%) \to \gamma$ veto

$\Rightarrow$ Detector dominated by vetos

Experiment overview



CEDAR

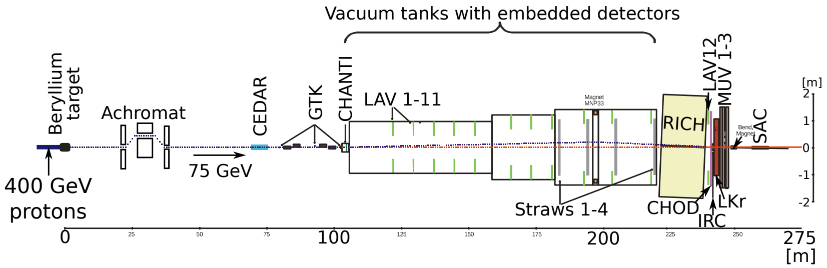


Differential Cerenkov counter

K^+ identification

Mirrors collect K^+-specific Cerenkov light

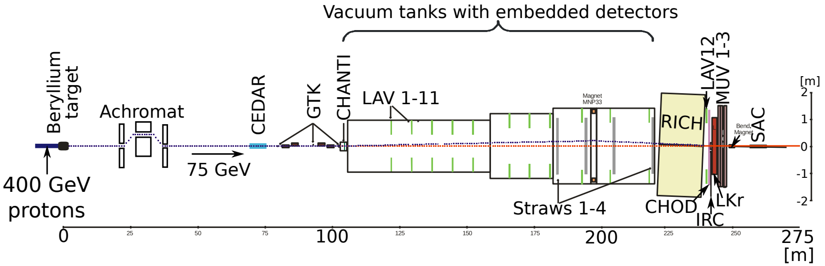
GTK



Gigatracker (second Achromat)

Timing, momentum and angle of all particles using silicon pixels
750 MHz event rate!!!

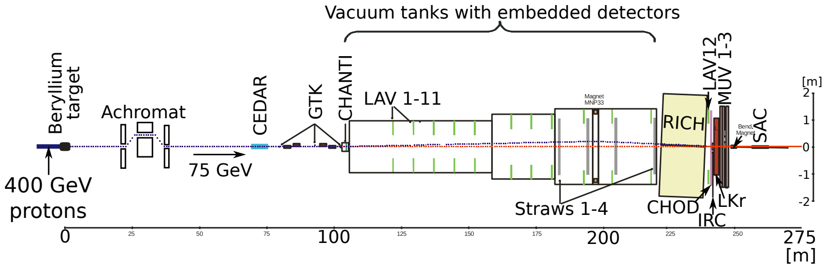
Straws



Spectrometer

Tracking of charged particles within a magnetic field

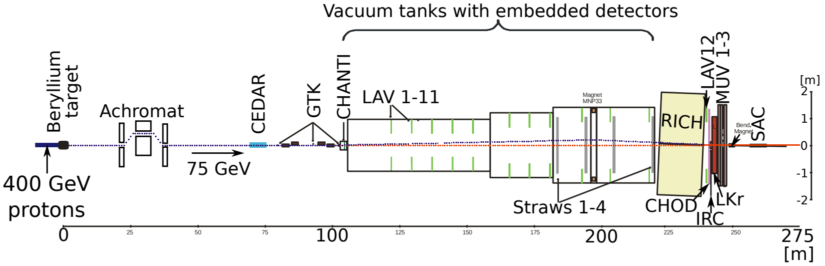
Photon vetos part 1: LAV, IRC, SAC



Photon vetos

Huge range of angle to be covered
Scintillators at different positions (lead-glass reused from OPAL)

Photon veto part 2: LKr



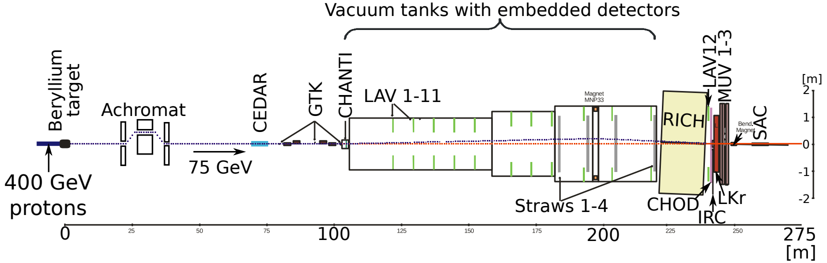
Liquid Krypton calorimeter

Photon veto for central angles range

Muon veto

Reused from NA48 $\Rightarrow$ can be used as calorimeter for analysis of additional rare decays

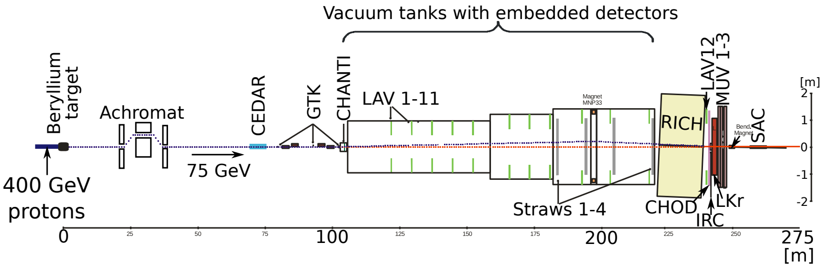
Rich



Ring Imaging Cherenkov

Separate π^+ from μ^+ between 15 and 35 GeV/c (L0 trigger)

MUV



Muon veto

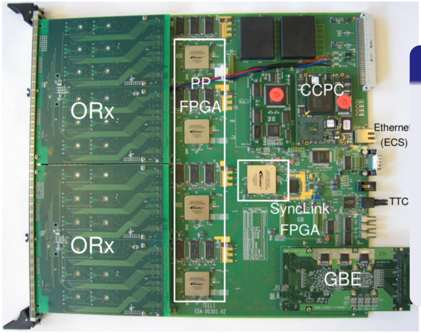
Scintillator-iron-sandwich mainly produced in Mainz

Data rates

10 MHz unsteady K^+ rate

Detector	Event size [B]	Data rate [GBps]
CEDAR	216	2.16
GTK	2250	22.50
CHANTI	192	1.92
LAV	160	1.60
STRAW	768	7.68
RICH	160	1.60
CHOD	$\ll$ 1000	$\ll$ 10
MUV	768	7.68
IRC & SAC	576	5.76
LKR	**222 k**	**2220**
Sum	**≈227 kB**	**≈2.3 TBps**

The TEL-62 board



Board with Altera Stratix III FPGAs

- Based on TELL-1 from LHCb
- 1 Credit card PC
- 4+1 Gigabit ethernet interfaces
- 2 or 4 GB DDR2 memory
- Up to 4 TDC daughter cards

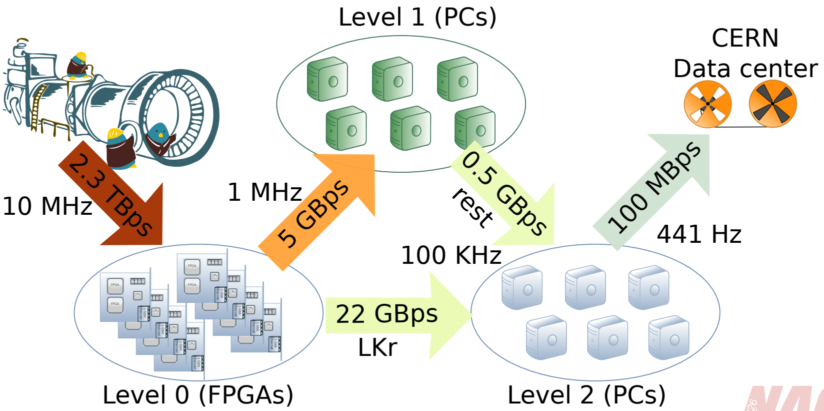
We will have about 500 Tel-62 and similar boards at NA62

These boards are used for DAQ and for L0 triggering

DAQ and Trigger system

Three levels to filter data

Data transmission via ordinary 10 gigabit ethernet:



The Triggers

Original concept:

Level 0 (hardware)

≈500 FPGA boards (mainly Tel-62) downstairs at the experiment

Level 1 (software, on subdetector level)

≈10 Subdetector specific PCs (8 cores with GPUs) upstairs (up to 100m fibre for 10G ethernet)

Level 2 (software, whole detector)

≈30 PCs (8 cores) that merge the subdetector data to one event (eventbuilding) and process a third trigger decision

The Triggers - assignments

Level 0

Hit in the Rich, no γ and no μ

Level 1

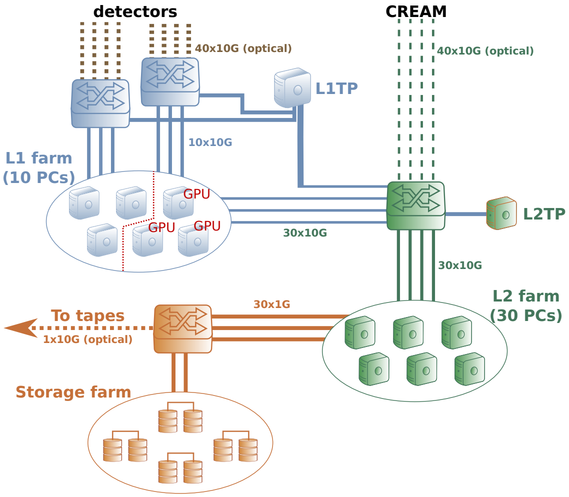
Every subdetector decides separately
Mainly data integrity checks
Ring fitting of the RICH via GPUs
Track fitting for straws using GPUs planned

Level 2

Eventbuilding (merging of all subdetector data) and reconstruction

First topology proposal

Original concept:



Special hardware purchased for performance tests

High performance 12-core PC Dell R710

2*6 Cores @ 2.93GHz (3.33GHz turbo)
24GB memory @ 1333MHz

Low power 12-core PC Dell R710

2*6 Cores @ 2.26GHz (2.8GHz turbo)
24GB memory @ 1333MHz

48 x 1G-port switch Dell PowerConnect 6248

Two SFP+ (10 Gbps) modules (4 ports total)

Test farm @ Mainz

2 big 12 core machines for farm simulation

Connected via 10Gbps SFP+ to the switch and to each other

11 L0 simulator PCs (1-4 cores)

Diskless
21 Cores total
21 * 1Gbps links connected to the switch (up to 4Gbps per PC)

Burst time and duty-cycle

Only 3-9 sec. burst and long break

Duty cycle: $T_{Burst} / T_{Break} \approx 0.3$

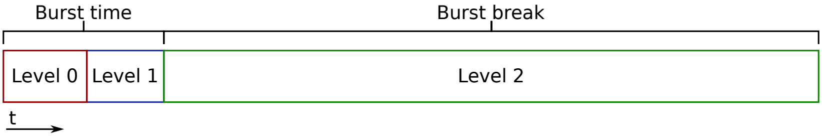


Abb.: NA48 approach

Reuse L1 PCs

My proposal to use resources more efficiently

Reuse L1 PCs during burst break for L2 computation by combining L1 and L2 to one farm

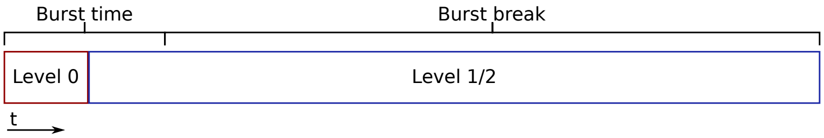
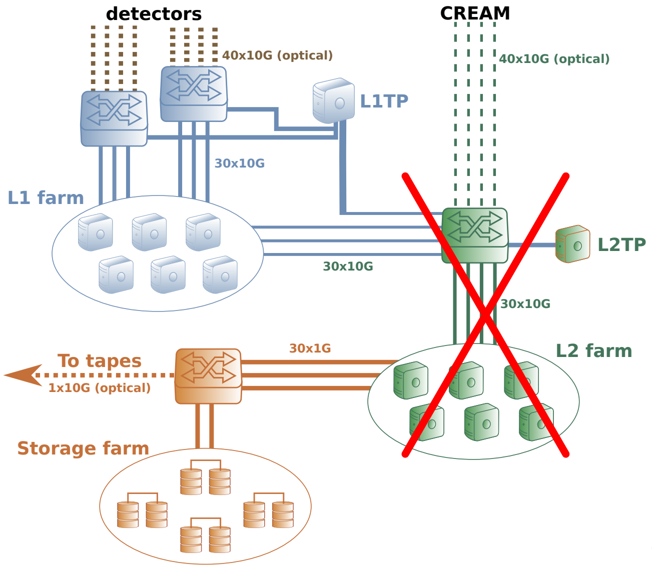
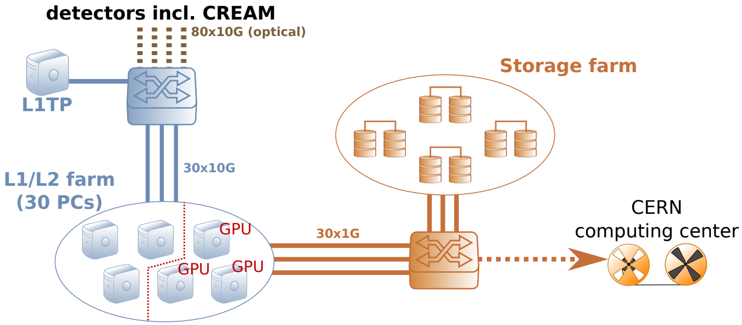


Abb.: New proposal

Don't separate L1 and L2!



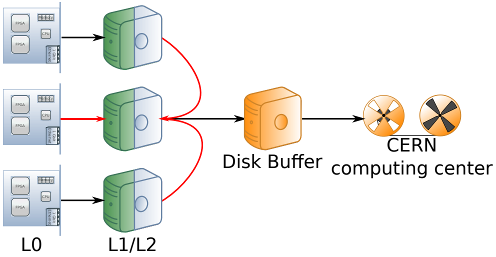
Combine L1 and L2 to one farm



We safe about 80k

- No L1 PCs anymore
- Less switches, less network cards

New bottleneck

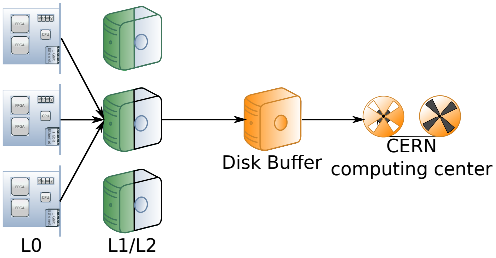


Risk of overloading the farm (L1 and L2 concurrently)

Possibilities to solve this problem:

- Implement a very intelligent load balancing algorithm
- Wait with L2 until end of Burst (L1 has finished)

New proposal
Event building @ L1

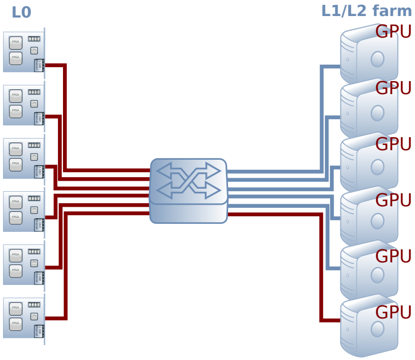


Every subdetector sends data of an event to one single PC

- ➕ No broadcast of a L1 decision needed anymore (no L1TP)
- ➕ Easier to implement load balancing (self-sustaining PCs)
- ➖ Every farm PC must serve every subdetector ⇒ needs GPUs

UDP Storm

Congestion at one single farm PC



Bundle events for better performance

Smallest Event: 160 B

Jumbo ethernet frame: 9 kB

55 Events in one bundle

$\Rightarrow 55 \cdot 5kB = 275kB$ total per bundle

Round Robin tables at L0

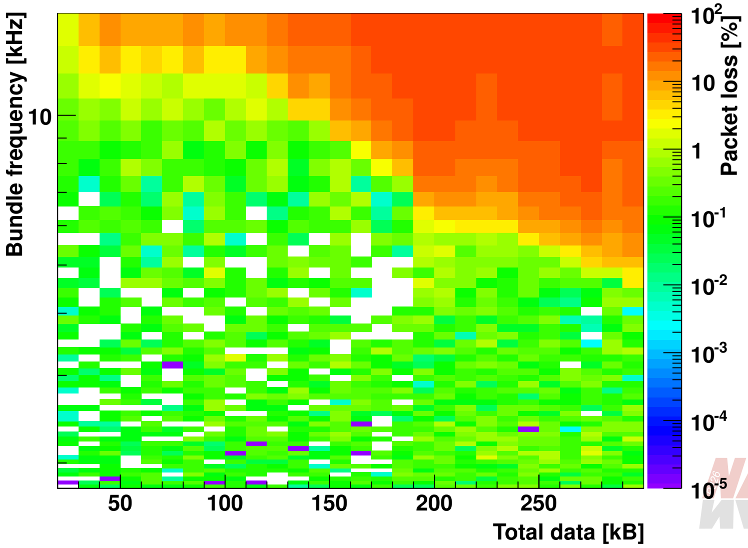
Send events 1-55 to PC A

Send events 56-110 to PC B

. . .

Packet loss

22 L0 boards simulated



Packet loss

Under construction

High packet loss

Packets are rejected by Kernel at receiver side (Not at switch or network card)!
I'm investigating if we can use standard drivers or have to implement own ones

Packets get lost in software, not hardware!

$\Rightarrow$ UDP storms don't cause problems
Events can be merged at L1 which makes load balancing much easier (no L1TP needed)

No expensive core switch needed

One expensive core switch was originally planned

- + Many ports (high density)
- + Fast backplane (any to any @ 10Gbps bidirectional)
- − High cost (about 180k)

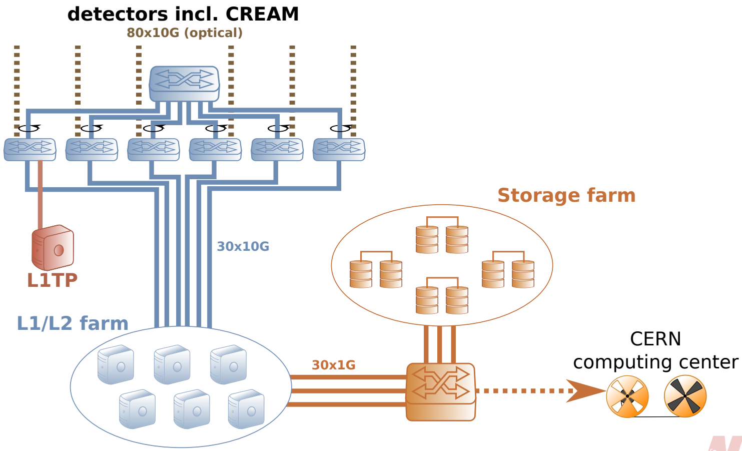
Tree topology with cheap 24 and 48 port switches

- − Slow backplanes (any to any @ 10Gbps only unidirectional)
- − Bottleneck at switch-to-switch connection
- + Low cost (about 80k)

Only one way transmission with about 8Gbps per link

It is feasible to use slower switches

Tree topology (Hexapus)



Current investigations and outlook

Implementation of a high performance L1/L2 farm software

- Data transmission L0 $\rightarrow$ farm and L2 $\rightarrow$ tapes
- Interface for the actual trigger algorithms
- Monitoring and administration system with a complex web interface

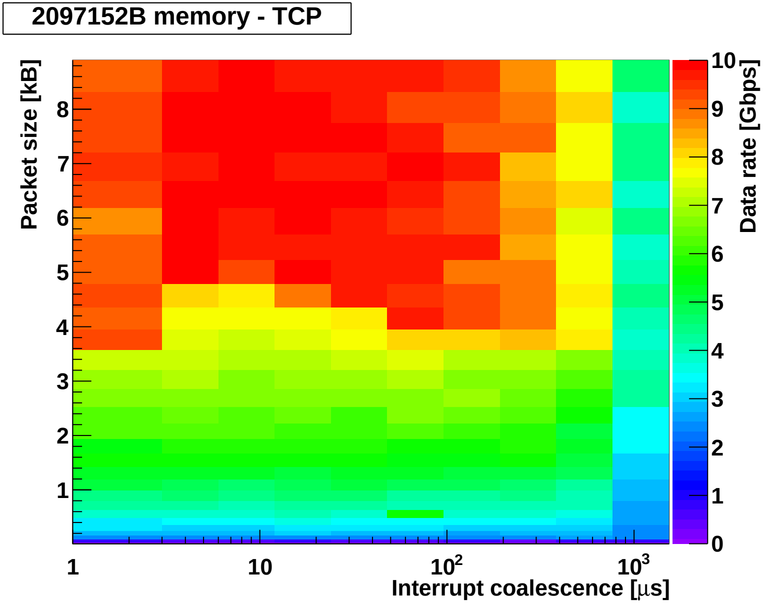
Further performance tests

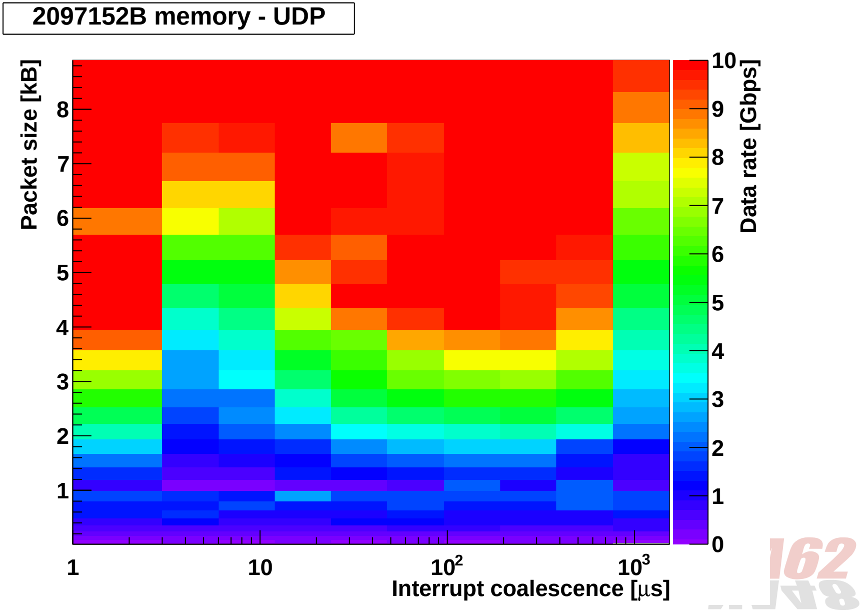
- Do we need special network drivers to reduce packet loss?
- How many PCs will be needed?

Conclusion

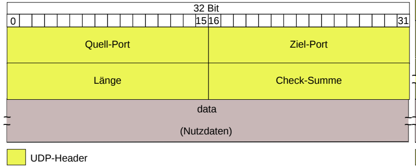
- High energy and high precision ⇒ a lot of data
- Using ordinary ethernet saves money and time and gives you the ability to quickly switch between different approaches
- Unsteady data production allows new approaches
  - Considering trigger levels as logical object, not as real farms saves a lot of money
- The new farm design allows us to have a central software architecture which is much easier to implement ⇒ Now Mainz will manage the whole chain from L0 to persistent memory and install one central farm instead of only L2

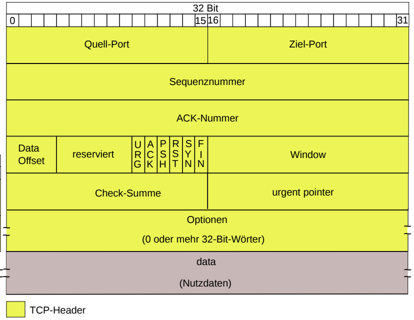
Thank you!


2097152B memory - TCP
Packet size [kB]
8
7
6
5
4
3
2
1
1
10
10^2
10^3
Interrupt coalescence [µs]
Data rate [Gbps]
10
9
8
7
6
5
4
3
2
1
0

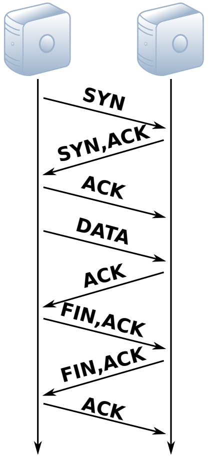

2097152B memory - UDP
Packet size [kB]
1
2
3
4
5
6
7
8
1
10
10^2
10^3
Interrupt coalescence [µs]
Data rate [Gbps]
0
1
2
3
4
5
6
7
8
9
10
NA62

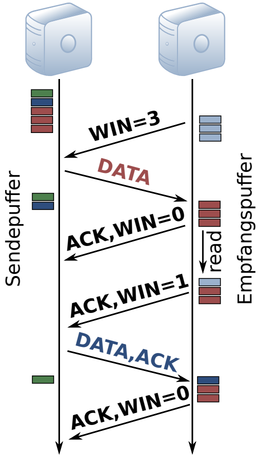
TCP vs. UDP


32 Bit
0
15
16
31
Quell-Port
Ziel-Port
Länge
Check-Summe
data
(Nutzdaten)
UDP-Header

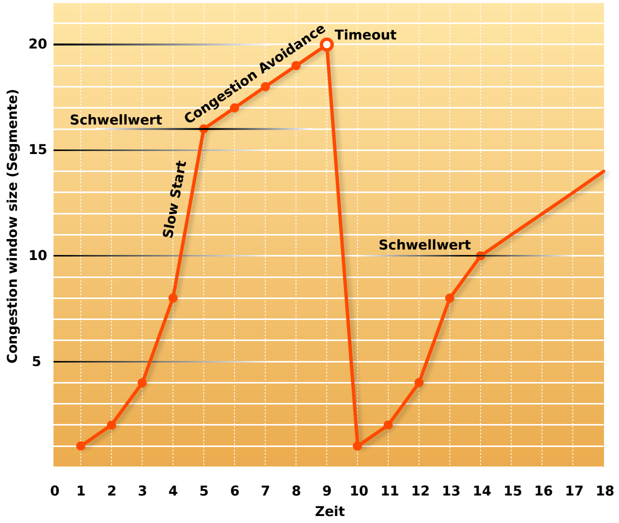

32 Bit
0
15
16
31
Quell-Port
Ziel-Port
Sequenznummer
ACK-Nummer
Data Offset
reserviert
URG
ACK
PSH
RST
SYN
FIN
Window
Check-Summe
urgent pointer
Optionen
(0 oder mehr 32-Bit-Wörter)
data
(Nutzdaten)
TCP-Header

TCP: Reliability and flow control





Congestion Avoidance



Performance tests

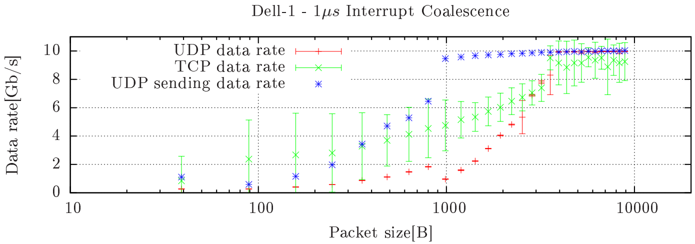
TCP optimizes the usage of network resources

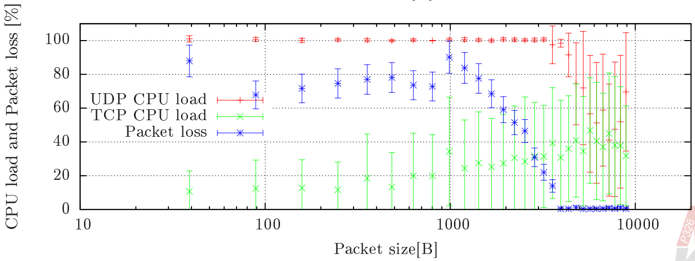
But what does this cost?

Intuitionally one would guess: Higher CPU usage and longer latencies.

but. . .

Data rate and CPU usage





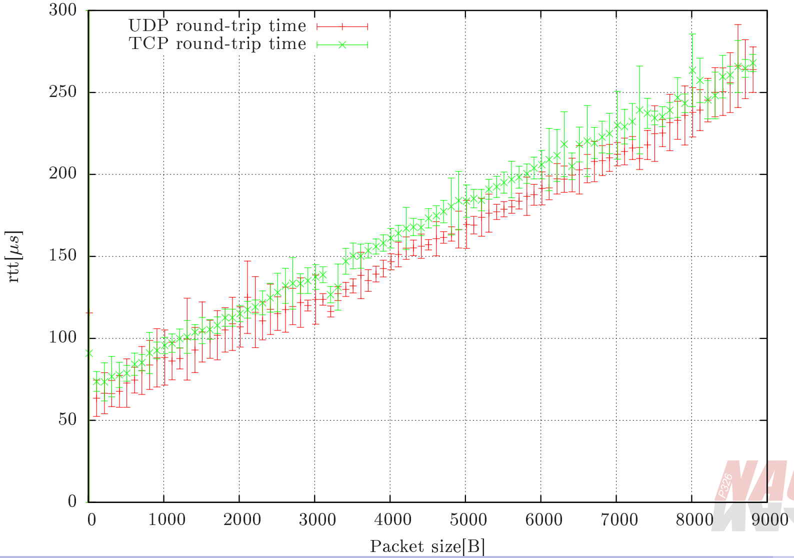
TCP Offload Engine

Network cards and drivers are optimized for TCP

- Checksumms and fragmentation calculated on hardware
- Some drivers ignore interrupt coalescence of $0\mu s$

Using TCP reduces CPU usage $\Rightarrow$ more space for computation

Timing


UDP round-trip time
TCP round-trip time
rtt[μs]
Packet size[B]
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000
6000
7000
8000
9000

Results

- TCP reduces CPU usage $\Rightarrow$ more space for computation
- TCP has flow control and congestion avoidance
- TCP is reliable

TCP in FPGAs

TCP means high payload in hardware!
$\Rightarrow$ TCP can only be used for PC to PC communication at NA62!

TCP/UDP vs. basic IP

Using standard interrupt/kernel based socket programming...

- is optimized for TCP (drivers)
- + is easy and many libraries can be used (e.g. boost::asio)
- − induces high latency ($\approx 30 - 150\mu s$)
- − induces high packet loss??!!

Programming own Kernel modules...

- − is hard stuff (only few small libraries)
- − is bound to hardware
- + highest performance possible