Fakultät für Physik der Technischen Universität München
Teilinstitut E12

Das Datenaufnahmesystem für das Elektronenpaar-Spektrometer HADES

Mathias Münch

Vollständiger Abdruck der von der
Fakultät für Physik der Technischen Universität München
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. M. Lindner

Prüfer der Dissertation:

1. Univ.-Prof. Dr. H.-J. Körner

2. Univ.-Prof. Dr. St. Paul

Die Dissertation wurde am 8.5.2002 bei der

Technischen Universität München eingereicht und durch

die Fakultät für Physik am 4.7.2002 angenommen.

Inhaltsverzeichnis

Zusammenfassung

In der vorliegendenen Arbeit wird die Auslese, der Datentransport und das Event-Building für die Datenaufnahme des HADES-Experiments beschrieben.

HADES ist ein Dielektronenspektrometer am Schwerionensynchrotron (SIS) der Gesellschaft für Schwerionenforschung (GSI) in Darmstadt. Es soll Spektren in invarianten Massenbereich bis $1.1\,\mathrm{GeV/c^2}$ messen, sowohl in pion- und protoninduzierten Reaktionen als auch in Schwerionenkollisionen massiver Systeme. Bei Multiplizitäten bis zu 200 geladenen Teilchen werden Raten bis zu $10^5/\mathrm{s}$ erreicht werden. Die dabei entstehende Datenrate von mehreren Gigabyte pro Sekunde soll durch ein mehrstufiges Triggersystem auf $17\,\mathrm{MByte/s}$, in einer späteren Ausbaustufe auf $2\,\mathrm{MByte/s}$, reduziert werden. Die Auslese der Detektordaten erfolgt parallel durch mehrere CPUs, die Daten werden dann zu einem zentralen Rechner transportiert und dort gespeichert.

Die Verwirklichung des HADES-Datenaufnahmesystems umfasst die Festlegung der Leistungsdaten des Systems basierend auf den experimentellen Anforderungen, die Konzeption anhand der geforderten Leistungen, die Wahl und der Test der verwendeten Komponenten sowie Entwicklung der notwendigen Algorithmen und ihre Implementierung in ein konkretes System.

Die hochintegrierte Detektorelektronik wird von VMEbus-Steckkarten angesteuert, die auf insgesamt sieben Crates aufgeteilt sind. In jedem dieser Crates liest eine CPU die Steckkarten aus und versendet die Daten mit Raten bis $15\,\mathrm{MByte/s}$ über ein ATM-Netzwerk. Dieses übernimmt alle Aufgaben der Vermittlung und insbesondere die Aufteilung der zur Verfügung stehenden Transportkapazität. Ein handelsüblicher Rechner nimmt die Daten entgegen, setzt sie zu vollständigen Ereignisdaten zusammen und speichert sie in Dateien ab. Datentransport und Event-Building verwenden in hohem Maße handelsübliche Komponenten aus dem Rechner- und Telekommunikationsbereich. Hierfür notwendige Neuentwicklungen und Anpassungen in der Software werden in der Arbeit beschrieben.

Weiterhin werden vorbereitende Messungen dargestellt, die mit dem System im Labor durchgeführt wurden. Ein Prototyp wurde im Sommer 1997 bei einer Strahlzeit an der GSI erfolgreich eingesetzt, das System selbst ist seit Ende 1998 fast unverändert bei HADES-Messungen im Einsatz. Es wurden Datenraten bis $5\,\mathrm{MByte/s}$ auf Band erreicht. Auch auf diese Messungen geht die Arbeit kurz ein.

Die Nutzung industriell verfügbarer und weit verbreiteter Komponenten macht den Leistungszuwachs im Rechner und Telekommunikationssektor ohne Einschränkungen auch für die HADES Datenaufnahme sofort nutzbar. Konzepte zur weiteren Steigerung der Leistungsfähigkeit werden ausblickend diskutiert.

1. Einleitung – Motivation

Die starke Wechselwirkung bildet eine der grundlegenden Kräfte in der Natur und ist verantwortlich für die Stabilität von so unterschiedlichen Objekten wie z.B. Atomkernen und Neutronensternen. Ihre Beschreibung mit Hilfe der Quantenchromodynamik stützt sich auf Quarks als Bausteine der Materie sowie Gluonen als Übermittler der starken Wechselwirkung und ist damit formal der Quantenelektrodynamik (QED) ähnlich.

Entscheidende Unterschiede zur QED erschweren jedoch im Allgemeinen die Vorhersage von experimentell nachprüfbaren Phänomenen. Einer dieser Unterschiede ist die Größe der Kopplungskonstante $g(q^2)$, die nicht konstant ist, sondern vom Impulsübertrag bei der Wechselwirkung abhängt. Sie liegt für kleine q in der Größenordnung von $g/4\pi \approx 1$ und wird erst im Bereich hoher Energien klein. Dies erlaubt nur für hohe Impulsüberträge, im so genannten „asymptotisch freien Limit“, eine störungstheoretische Behandlung analog zur QED (Kopplungskonstante $e^2/4\pi \approx 1/137$) [1, 2].

Bei kleineren Impulsüberträgen ist die störungstheoretische Behandlung nicht möglich und die Berechnungen (starke QCD) werden zunehmend komplexer. Die starke Kopplung der Quarks bei großen Abständen führt zum Phänomen des „Confinement“, der Tatsache, dass keine freien Quarks und Gluonen beobachtet werden können, sondern nur aus diesen Teilchen zusammengesetzte Objekte, die Hadronen.

Im Zusammenhang mit den Hadronen stellen sich weitere Fragen, insbesondere was deren Masse angeht. So besteht z.B. das Proton aus zwei u-Quarks (Masse ca. 1 bis $5\,\mathrm{MeV}$) und einem d-Quark (Masse ca. 3 bis $9\,\mathrm{MeV}$)[1]. Das Proton selbst hat demgegenüber eine Masse von ca. $1\,\mathrm{GeV}$, also dem 50fachen seiner Konstituenten. Der überwiegende Teil der uns umgebenden Masse entsteht also aus dynamischen Effekten. Diese sind bis heute nicht vollständig verstanden.

Der große Unterschied der Massen der leichten Quarks und der in der QCD üblichen Energieskalen Λ_{QCD} macht allerdings auch im Bereich niedriger Energien einen Zugang zur starken Wechselwirkung möglich, und zwar über die Ausnutzung von Symmetrien.

1.1. Chirale Symmetrie

Betrachtet man die leichten Quarks, insbesondere das u- und das d-Quark, als völlig masselos, so wird die Lagrange-Dichte der QCD invariant gegenüber Isospin-Rotationen (chiral symmetrisch), die entsprechende Erhaltungsgröße ist der Vektor- bzw. Axialvektorstrom. Zwar werden dem u- und d-Quark die oben genannten, durch den Higgs-

[1] Quarkmassen aus [3]

Mechanismus erzeugten Massen zugeschrieben, und das Dazufügen dieser Masseterme zur Lagrange-Dichte bricht deren Symmetrie (*explizite* Brechung der Chiralen Symmetrie). Doch wegen des großen Masseunterschiedes zu den Hadronen ist die Annahme einer chiral symmetrischen Lagrange-Dichte dennoch erfolgreich z.B. zur Beschreibung von Pionenstreuung und -zerfall.

Allerdings zeigen die Existenz des (nahezu) masselosen Goldstone-Bosons π [4] und die Aufspaltung des Paritätsdubletts der Vektormesonen (ρ, a_1), dass der zu dieser symmetrischen Lagrange-Dichte gehörende Grundzustand nicht symmetrisch ist (*spontane* Brechung der chiralen Symmetrie). Eine solche Symmetriebrechung bedeutet aber, dass der Erwartungswert eines Mesonenfeldes im Grundzustand (im Vakuum) ungleich Null ist, oder, wenn man in die Beschreibung mit Quarks übergeht, $\langle \bar{q}q \rangle \neq 0$. Ein quantitatives Verständnis des Zusammenhangs zwischen der spontanen Brechung der chiralen Symmetrie, dem Erwartungswert des Mesonenfeldes und der Hadronenmasse fehlt bis heute.

Während man also an den beiden Extrema der Energieskala, im asymptotisch freien Limit einerseits und bei sehr geringen Dichten andererseits, erfolgreiche Beschreibungen der QCD hat, kann der dazwischenliegende Bereich mit keiner der beiden behandelt werden. Zum besseren Verständnis z.B. des Confinements ist aber genau dieser Bereich interessant. Aus numerischen QCD-Rechnungen erwartet man bei hohen Temperaturen und/oder Dichten einen Phasenübergang ins Quark-Gluon-Plasma [5], also eine Aufhebung des Confinements. Von diesem Übergang vom Confinement ins Deconfinement erwartet man mehr Information als von einem System, das stabil in einem der beiden Extremzustände ist.

Auch bei der Frage des Zusammenhangs zwischen spontan gebrochener chiraler Symmetrie und der Entstehung der Hadronenmasse [6] ist offensichtlich der Übergang von vollständig gebrochener Symmetrie bei verdünnten Systemen zur erwarteten ungebrochenen Symmetrie bei dichten Systemen von besonderer Wichtigkeit.

Verschiedene Modelle [7, 8] sagen eine Abnahme des $\langle \bar{q}q \rangle$-Erwartungswertes bei steigender Temperatur und Dichte voraus. Damit sinkt der Erwartungswert aller Mesonenfelder im Vakuum, die spontane Symmetriebrechung wird also, zumindest teilweise, aufgehoben. In der Folge müssen die Signaturen der spontanen Symmetriebrechung verschwinden, das Pion kein Goldstone-Boson mehr sein und die Entartung des Paritätsdubletts ρ, a_0 wiederhergestellt werden. In der Hauptsache erwartet man somit Änderungen der Mesonenmasse.

Andererseits legen aktuelle QCD-Gitterrechnungen nahe, dass bei geringen Dichten die Wiederherstellung der Chiralen Symmetrie und der Phasenübergang ins Quark-Gluon-Plasma im selben Temperaturbereich stattfinden [9]. Damit verschwinden die gebundenen Quarkzustände (Mesonen, Baryonen), so dass bei Phasenübergangstemperaturen die Aufhebung der spontanen Symmetriebrechung in den Mesoneneigenschaften nicht mehr messbar wäre. In wie weit beide Effekte auch bei hohen Dichten parallel ablaufen ist z.Z. unklar.

Im selben Temperatur- und Dichtebereich spielen sich auch noch weitere Effekte ab: Bei hinreichend hohen Energien werden die höheren Zustände der Nukleonen angeregt, die wiederum eng mit Mesonenzuständen verknüpft sind, wie z.B. das ρ-Meson mit

dem $[N^*(1\,520)N^{-1}]^{1-}$-Zustand. Eine Änderung der ρ-Masse könnte also auch durch die Kopplung des Mesons an den N^*-Lochzustand entstehen [10].

Insgesamt erwartet man also bei Erhöhung der Temperatur und Dichte von hadronischer Materie Änderungen der Mesoneneigenschaften. In welcher Form, bei welchen Bedingungen und mit welchen Auswirkungen scheint zur Zeit nur durch das Experiment zu klären zu sein.

Zur experimentellen Überprüfung der Aussagen müssen drei Voraussetzungen geschaffen werden:

1. Herstellen von Zuständen hadronischer Materie mit Dichten und Temperaturen oberhalb der Vakuumwerte, möglichst über einen ganzen Bereich von ρ- und T-Werten.

2. Ein Mechanismus, der Informationen aus dem Inneren dieser Materie bis zum Messgerät transportiert.

3. Das Messgerät zum Erfassen dieser Information.

1.2. Relativistische Schwerionenkollisionen

Die einfachste Möglichkeit, eine Umgebung von Kernmaterie mit einer Dichte $\rho > 0$ zu nutzen, sind Atomkerne bei normaler Nukleonendichte $\rho = \rho_0$. Gegenüber dem Vakuum sollten sich schon Auswirkungen auf die Mesonenmasse feststellen lassen [11].

Um allerdings in den Bereich der vollständigen Restauration der Chiralen Symmetrie bzw. des chiralen Phasenübergangs zu gelangen, muss die Kernmaterie komprimiert und aufgeheizt werden. Im Labor erfolgt das durch Kollisionsexperimente mit Schwerionen. Damit sich die Dichte der Kernmaterie erhöht, muss hierbei die Projektilgeschwindigkeit über der Schallgeschwindigkeit in Kernmaterie liegen. Nur dann können die Nukleonen nicht schnell genug aus der Reaktionszone entweichen und es bildet sich eine „Stoßfront“.

Dies führt zu *relativistischen* Schwerionenkollisionen, wie sie z.B. am Schwerionensynchrotron (SIS) der Gesellschaft für Schwerionenforschung (GSI) in Darmstadt erzeugt werden können. Bei Projektilenergien von $1\,\mathrm{AGeV}$ erwartet man eine Erhöhung der Baryonendichte auf $\rho \approx 2$ bis $3\rho_0$ und der Temperatur auf $T \approx 80\,\mathrm{MeV}$. Nach der Kollision kühlt das System über Expansion innerhalb von ca. $15\,\mathrm{fm/c}$ wieder ab, die dichte Phase bleibt also vergleichsweise lange erhalten. Die maximale Baryonendichte von $\rho \approx 10\rho_0$ wird bei Projektilenergien von 10 bis $20\,\mathrm{AGeV}$ erreicht [12].

Bei noch viel höheren Projektilenergien (*ultrarelativistische* Schwerionenkollisionen, $E_{\mathrm{Proj}} \gg 10\,\mathrm{AGeV}$) werden die Nukleonen im zentralen Stoßbereich nicht mehr vollständig gestoppt. Während die Temperatur im Feuerball weiter ansteigt, nimmt die Baryonendichte wieder ab [13].

1.3. Dileptonen als Sonden

Eine geeignete Sonde für Informationen aus der dichten Phase der Reaktion muss also die Eigenschaften der Kernmaterie zu einem frühen Zeitpunkt annehmen, darf aber während der späteren Expansions- und Abkühlungsphase nicht mehr beeinflusst werden. Geeignet erscheinen daher die dileptonischen Zerfälle der ρ-, ω- und ϕ-Vektormesonen, da die Leptonen nach dem Zerfall nicht mehr stark wechselwirken. Die aus den Teilchenimpulsen rekonstruierte Masse des Elektron-Positron-Paares enthält daher eine Aussage über die Masse des Vektormesons zum Zeitpunkt des Zerfalls. Voraussetzung ist damit auch, dass der Zerfall des Vektormesons noch in der komprimierten Kernmaterie stattfindet, was zumindest für das ρ-Meson, aber zu einem Gutteil auch für das ω der Fall ist (Tab. 1.1). Aus diesen Zusammenhängen ergibt sich, dass Kollisionen bei Projektilenergien bis $10\,\mathrm{AGeV}$, wegen der vergleichsweise langsamen Expansion des Feuerballs, besonders gut für die Untersuchung von Dichteeffekten geeignet sind.

	$m\,[\,\mathrm{MeV/c^2}]$	$c\tau\,[\,\mathrm{fm}]$	domin. Zerfall	e^+e^--Verzw.
ρ	768	1.3	$\pi\pi$	$4.4 \cdot 10^{-5}$
ω	782	23.4	$\pi^+\pi^-\pi^0$	$7.2 \cdot 10^{-5}$
ϕ	1 019	44.4	K^+K^-	$3.1 \cdot 10^{-4}$

Tabelle 1.1.: Wichtige Eigenschaften der leichten Vektormesonen ρ, ω und ϕ

Die Herausforderung an das Experiment ergibt sich aus dem geringen Verzweigungsverhältnis in den dileptonischen Kanal und dem hohen Untergrund durch andere Beiträge zum Dileptonenspektrum.

1.4. Experimente zur Dileptonenspektroskopie

Neben den Experimenten zur Dimyonenspektroskopie bei HELIOS und NA50, beide am CERN-SPS, wurden Dielektronenspektren von zwei Experimenten aufgenommen.

Die CERES/NA45 Kollaboration, ebenfalls CERN-SPS, maß mit den Reaktionen $\mathrm{S} + \mathrm{Au}$ bei $200\,\mathrm{AGeV}$ und $\mathrm{Pb} + \mathrm{Au}$ bei $158\,\mathrm{AGeV}$ Dileptonenspektren wie in Abb. 1.1 auf der nächsten Seite. Sie zeigen im Bereich von $0.2 < m_{\mathrm{ee}} < 0.7\,\mathrm{GeV/c^2}$ einen deutlichen Dileptonenüberschuss gegenüber den bekannten Komponenten des Spektrums. Modelle, die die Wiederherstellung der chiralen Symmetrie zur Grundlage haben, können die Ergebnisse zum Teil reproduzieren, aber die augenblickliche Energieauflösung und Statistik schließen auch andere Erklärungen nicht aus [14].

Im Bereich niedrigerer Projektilenergien bis zu $2.1\,\mathrm{AGeV}$ wurden Spektren vom Dileptonenspektrometer DLS am LBL-BEVALAC aufgenommen [16]. Insbesondere bei schweren Systemen ($\mathrm{Ca} + \mathrm{Ca}$) tritt auch hier ein Dileptonenüberschuss im Bereich $0.2 < m_{\mathrm{ee}} < 0.6\,\mathrm{GeV/c^2}$ auf, der von den bekannten Modellen nicht erklärt wird (Abb. 1.2 auf Seite 6). Eine Erklärung durch die Modifikation der Mesonenmassen ist allerdings auch im Fall von DLS umstritten, darüber hinaus ist eine Abweichung der

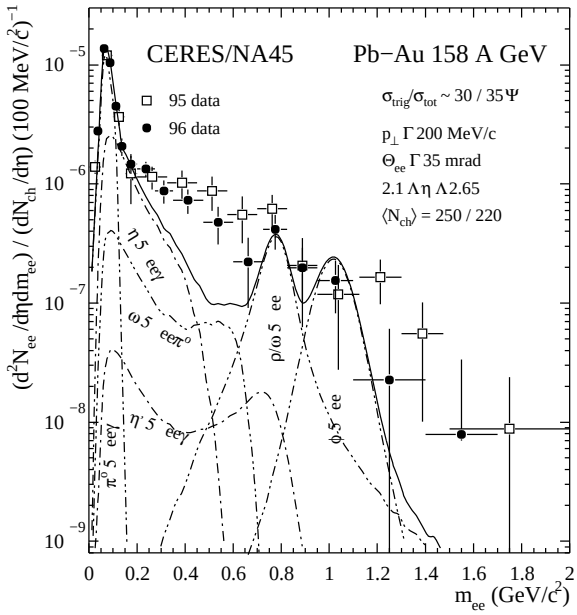


Abbildung 1.1.: Dileptonenspektrum des CERES-Spektrometers: Eingezeichnet ist auch das berechnete Spektrum aus den angegebenen Beiträgen. Es weicht von den Experimentdaten zum Teil um fast eine Größenordnung ab [15].

η-Produktion gegenüber den von TAPS am selben System gemessenen Werten bisher nicht erklärt.

1.5. Das HADES-Spektrometer

Wegen des kleinen Verzweigungsverhältnisses konnten die bisherigen Dileptonenexperimente nicht genügend Statistik zur klaren Differenzierung der Beiträge zum Spektrum der invarianten Massen sammeln. Zum Teil erlaubte auch die Energieauflösung keine Trennung.

Das HADES Experiment setzt an diesen beiden Punkten an und zeichnet sich durch folgende Eigenschaften aus:

Ausreichende Energieauflösung zur sicheren Trennung der Mesonenlinien. Das Spektrometer (Abb. 1.3 auf Seite 7) im engeren Sinne besteht aus einem supraleitenden Magneten mit sechs Spulen, der ein toroidales Magnetfeld bis zu $0.9\,\mathrm{T}$ erzeugt. Vor und hinter dem Magneten sind jeweils zwei Ebenen von Vieldraht-Driftkam-mern (MDC), die eine Bestimmung von Position und Winkel der Flugbahn geladener Teilchen erlaubt. Damit wird eine Impulsauflösung von 1.5% und eine invariante Massenauflösung von $\Delta m/m \approx 1\,\%$ erreicht. Die damit einhergehende, hohe Segmentierung der Kammern in sehr viele Driftzellen macht auch bei hohen Multiplizitäten die

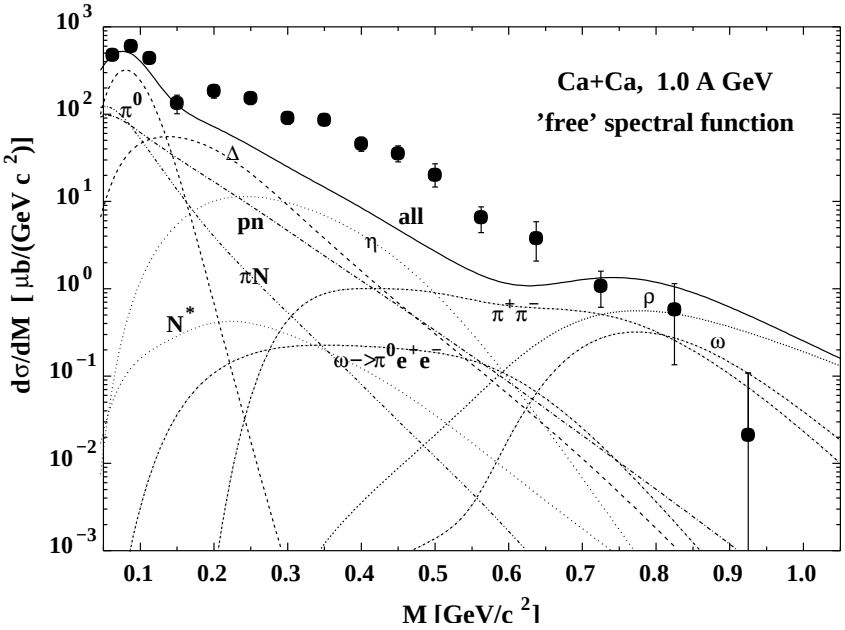


Abbildung 1.2.: Dileptonenspektrum des DLS-Spektrometers: Das berechnete Spektrum berücksichtigt die eingezeichneten Beiträge und die Akzeptanz des DLS [17].

Wahrscheinlichkeit von Zweifachtreffern gering. Die MDC sind näher beschrieben in [18].

Gute Unterdrückung des Untergrundes auch bei den hohen Multiplizitäten in Schwerionenstößen. Zur Unterdrückung des Hadronenuntergrundes in Schwerionenkollisionen, zur Selektion der Elektronen/Positronen und zur Erzeugung von Triggerinformation werden vor und hinter dem Spektrometer weitere Detektoren eingesetzt.

Direkt um das Target befindet sich ein RICH Detektor, dessen $\mathrm{C_4F_{10}}$-Gasradiator eine Cherenkovschwelle $\gamma_{\mathrm{thr}} = 18$ hat. Ausschließlich die Elektronen und Positronen aus der Kernreaktion sind schnell genug und erzeugen damit Cherenkovlicht. Die Cherenkovphotonen werden als Ringe annähernd gleichen Durchmessers auf eine Ebene abgebildet, die mit CsI als Photokonverter beschichtet ist. Die Photoelektronen werden dann in einer Vieldraht-Proportionalkammer detektiert und durch Kopplung auf eine zweidimensional segmentierte Kathode entsteht das Bild. Eine genauere Beschreibung des RICH-Detektors findet sich in [19].

Hinter dem Spektrometer ist im Polarwinkelbereich $\theta < 45^\circ$ ein Detektor zur Messung elektromagnetischer Schauer (PreShower) angebracht, der dort die Identifikation von Leptonen leistet. Im restlichen Polarwinkelbereich ist die Leptonenidentifikation über die Flugzeit möglich, so dass dort eine Flugzeitwand aus 648 Plastikszintillatoren mit einer Zeitauflösung von $100\,\mathrm{ps} \leq \sigma_{\mathrm{TOF}} \leq 160\,\mathrm{ps}$ verwendet wird [20].

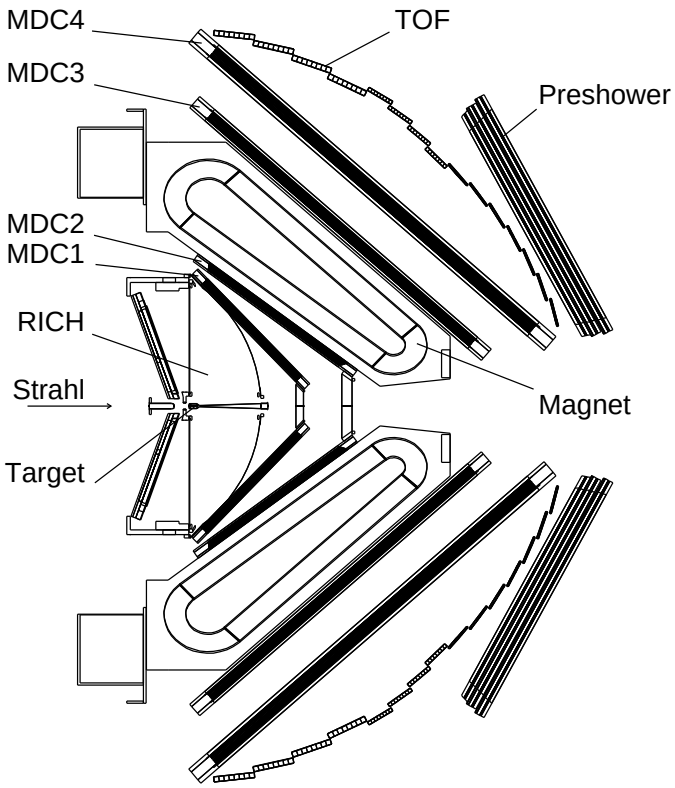


Abbildung 1.3.: Das HADES-Spektrometer besteht aus vier Detektorgruppen und einem supraleitenden Magneten.

Hohe Ratenfestigkeit und große Raumwinkelakzeptanz zum Erreichen von guter Statistik in kurzer Messzeit. Für ein Schwerionenexperiment (Gold-Gold) wird mit HADES die Messung vom 10^3 e^+e^--Zerfällen aus ω-Mesonen pro Tag angestrebt. Das erfordert eine Strahlintensität von $10^8\,\mathrm{Teilchen/s}$. In Verbindung mit einem „Target“ von 1% Wechselwirkungslänge ergeben sich $10^6\,\mathrm{Kollisionen/s}$. Davon sind 10 % zentrale Kollisionen, bleiben $10^5\,\mathrm{Kollisionen/s}$ als mittlere Ereignisrate. Einen Überblick über das gesamte Spektrometer gibt [21].

1.5.1. Anforderungen an das HADES Datenaufnahmesystem

Aus den oben beschriebenen Eigenschaften des HADES Spektrometers und den physikalischen Fragestellungen ergibt sich die besondere Rolle des Datenaufnahmesystems.

Die Auslegung auf hohe Multiplizitäten und die notwendige Unterdrückung des Untergrundes führen zu einer hohen Segmentierung der Zähler. In Verbindung mit der großen Raumwinkelakzeptanz und den damit einhergehenden großen Detektorflächen ergibt sich eine Zahl von über 72 000 auszulesenden Kanälen.

Nimmt man dazu die hohe Ratenfestigkeit, die auf 10^6 Kollisionen pro Sekunde

ausgelegt ist, so wird klar, dass das Datenaufnahmesystem für die Leistung des Gesamtsystems von entscheidender Bedeutung wird. Insbesondere erstrecken sich die Aufgaben der Datenaufnahme nicht mehr nur auf das Abspeichern der Daten zur späteren Analyse, sondern schon während des laufenden Experiments müssen die Informationen der Detektoren zur Teilchenidentifizierung (RICH, PreShower, TOF) in die Ereignisklassifikation und Datenreduktion eingebunden werden, um die Datenmenge überhaupt bewältigen zu können.

In den folgenden Abschnitten wird das Datenaufnahmesystem in seinen einzelnen Aspekten und anschließend Ergebnisse erster Testmessungen dargestellt.

2. Das HADES Datenaufnahmesystem

Ausgehend von den experimentellen Randbedingungen und den daraus folgenden Eigenschaften des HADES-Spektrometers werden in diesem Kapitel zunächst die Anforderungen an die HADES-Datenaufnahme formuliert. Aus diesen Anforderungen wird der Entwurf des Datenaufnahmesystems entwickelt. Hierbei genutzte Begriffe und Verfahren sind im Glossar kurz beschrieben und zum Teil im Anhang A weiter erläutert.

2.1. Triggerraten – Datenmengen

Der Entwurf des Datenaufnahmesystems geht von der in Abschn. 1.5.1 genannten, mittleren Ereignisrate von $10^5\,\mathrm{Kollisionen/s}$ aus. Um 50 % davon zu messen, muss das System nach Gleichung A.2 auf Seite 83 mit einer Totzeit von $\tau = 10\,\mu\mathrm{s}$ arbeiten.

Um zu keinem Zeitpunkt durch die Datenauslese oder den Datentransfer in der Leistung begrenzt zu werden, wurde der gesamte Datentransport für den Fall ausgelegt, dass 10^5 zentrale Schwerionenkollisionen pro Sekunde gemessen werden sollen (entsprechend 100% Totzeit).

Aus Simulationen [22] und Messungen [23] lassen sich daraus die Betriebsparameter des Datenaufnahmesystems bestimmen. Ein zentraler Gold-Gold-Stoß führt zu einer gemittelten Multiplizität von ca. 200 geladenen Teilchen. Davon sind 170 Protonen, 20 geladene Pionen, aber weniger als ein Lepton (Elektron oder Positron). Von den geladenen Teilchen werden ca. 50% in der Akzeptanz nachgewiesen.

Der RICH ist hadronenblind und „sieht“, wegen der entsprechend gewählten Anordnung zum Target, kaum geladene Teilchen. Photonen werden nur von Teilchen oberhalb der Cherenkov-Schwelle, also den Leptonen, erzeugt. Jedes Lepton generiert dabei einen Ring, in dem durchschnittlich 15 der insgesamt 27648 Pixel ansprechen [19].

Die Pulshöhenverteilung der nachzuweisenden, einzelnen Photoelektronen folgt einer Polyafunktion [24], es gibt also keine untere Schwelle für die Signalhöhe. Damit ist eine klare Trennung zwischen Signal und elektronischem Untergrund (Rauschen) nicht möglich. Jede Schwelle zur Rauschunterdrückung unterdrückt auch einen Teil der Photonensignale. Eine Schwelle von 3σ der Amplitudenverteilung des Rauschens würde im Mittel 0.27% aller Pixel selbst bei Abwesenheit von Photonen ansprechen lassen. Um weiteren Untergrund durch geladene Teilchen und Rundungsfehler bei der Berechnung des ganzzahligen Schwellwertes zu berücksichtigen, wurde beim RICH von ca. 1% An-

teil der Kanäle durch Rauschen ausgegangen, was mit $27\,648 \times 1\% = 276$ die Zahl der auszulesenden Kanäle dominiert.

Bei den anderen Detektoren gibt es eine klare Pulshöhentrennung zwischen Untergrund und Signal, so dass hier nur das Signal der geladenen Teilchen beachtet werden muss. Unter Berücksichtigung von Raumwinkel, Detektoreffizienzen und Ablenkung im Magnetfeld erbrachte die Simulation für den Gold-Gold-Stoß die Werte aus Tab. 2.1.

	pro zentraler Kollision			bei 10^5 Hz
	Kanäle > 0	Byte/Kanal	Byte/Koll.	MByte/s
RICH	276	4	1 104	105.29
MDC	480	24	11 520	1 098.62
TOF	60	16	960	91.55
PreShower	792	6	4 752	453.19
Gesamt			18 336	1 748.65

Tabelle 2.1.: Datenraten der einzelnen HADES-Detektorsysteme in der ersten Triggerstufe [25]

2.2. Weitere Anforderungen

Ein sicherer und effizienter Betrieb soll durch eine entsprechende Gestaltung der Benutzeroberfläche gewährleistet werden. Für die Nachvollziehbarkeit der Messungen ist eine Verbindung mit einer Experimentdatenbank vorzusehen, in der alle Informationen über vorgenommene Konfigurationen und Messungen eindeutig gespeichert werden [26].

Bereits während der Messung („online“) sollen Daten zur Analyse zur Verfügung gestellt werden, um die Qualität überprüfen zu können.

Das Datenaufnahmesystem soll an verschiedene Konfigurationen des HADES bzw. für Erweiterungen und die Zusammenarbeit mit anderen Experimenten anpassbar sein. Für Labortests oder auch unabhängige Messungen am Strahlplatz sollen mehrere Instanzen des Systems eigenständig und ohne notwendige Verbindung zu zentralen Ressourcen einsetzbar sein.

Um die ständig wachsende Leistungsfähigkeit der Rechner- und Kommunikationstechniken auch in Zukunft ausnutzen zu können und andererseits in der Test- und Vorbereitungsphase schon vorhandene Geräte nutzen zu können, soll das System unabhängig von der eingesetzten Hardware und Systemsoftware sein.

2.3. Entwurf des Datenaufnahmesystems

Um die beschriebenen Anforderungen erfüllen zu können, kommen beim HADES Datenaufnahmesystem sowohl die „Parallelisierung“ der Auslese als auch ein „mehrstufiges Triggersystem“ zum Einsatz.

Den resultierenden logischen Aufbau zeigt Abb. 2.1 auf der nächsten Seite.

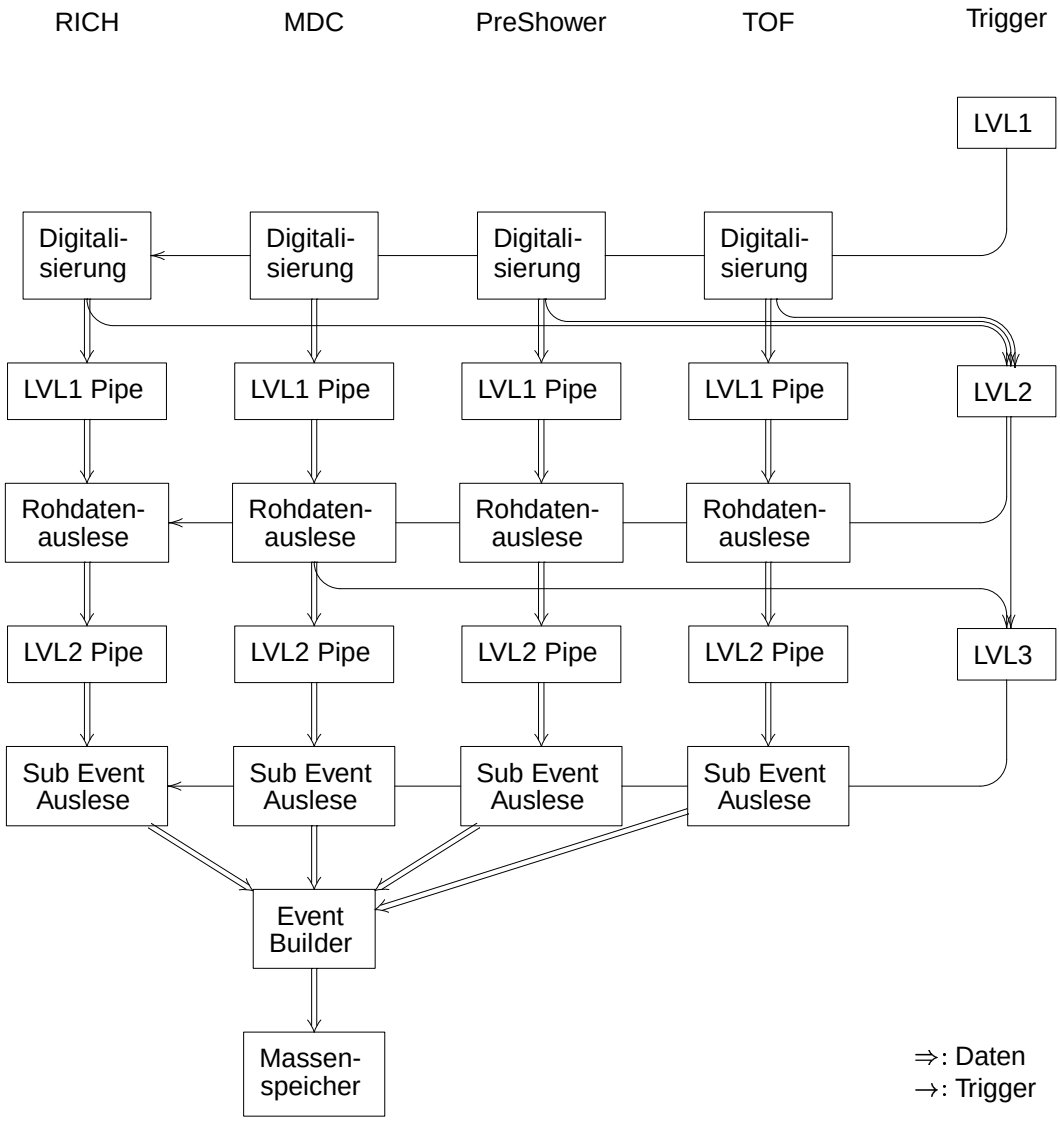


Abbildung 2.1.: Datenfluss durch das HADES Datenaufnahmesystem

2.3.1. Das Triggersystem

Das HADES-Triggersystem ist ausführlich beschrieben in [27, 28, 29, 30]. Hier sollen nur kurz die Eigenschaften dargestellt werden, die in direktem Zusammenhang zur Datenaufnahme stehen.

Das HADES-Triggersystem ist dreistufig:

LVL1-Trigger: Die erste Triggerstufe bildet einen Reaktionstrigger, der Informationen aus dem Start- und Vetozähler und der Flugzeitwand verwendet. Start- und Vetozähler sind beides strahlungsharte CVD-Diamantzähler, die vor und hinter dem Target direkt im Primärstrahl liegen. Mit der Bedingung $\mathrm{Start} \wedge \neg\mathrm{Veto}$ liefern sie einen einfachen Reaktionstrigger.

Aus der Flugzeitwand wird die Teilchenmultiplizität verwendet, die ein Maß für die Zentralität der Reaktion bildet. Insgesamt ergibt sich also:

$$\mathrm{Start} \wedge \neg\mathrm{Veto} \wedge (\mathrm{Mult}_{\mathrm{TOF}} > N) \Rightarrow \mathrm{Trigger}_{\mathrm{LVL1}}$$

Diese Bedingung wird ausschließlich mit analoger Elektronik und kombinatorischer Logik geprüft, die Triggerlatenz liegt bei ca. $10^2\,\mathrm{ns}$. Die Detektorinformation wird bis zur Triggerentscheidung in analogen Verzögerungsleitungen gepuffert oder durch Latenz der Elektronik selbst ausgeglichen. Gegenüber der gesamten Reaktionsrate wird für zentrale Schwerionenstöße eine Reduktion der Triggerrate um den Faktor $1:2$ (C+C) bzw. $1:10$ (Au+Au) erwartet.

LVL2-Trigger: Aus Informationen des RICH, des PreShower und der Flugzeitwand setzt sich der LVL2-Trigger zusammen. Die Leptonensignatur wird von elektronischen Digitalschaltungen auf der Basis von programmierbarer Logik oder Digitalen Signalprozessoren, den so genannten „Image-Processing-Units (IPUs)“ [27, 28, 29], gewonnen. Die Triggerbedingungen sind komplexer als im LVL1 und vom jeweiligen Experiment abhängig. Sie werden durch eine weitere elektronische Baugruppe, die „Matching-Unit (MU)“ geprüft. Die Latenz des LVL2-Triggers liegt bei ca. $10^2\,\mu\mathrm{s}$, die Detektordaten werden solange in digitalisierter Form in elektronischen Zwischenspeichern gehalten. Die LVL1-Triggerrate soll mit dieser Stufe maximal um den Faktor $1:100$ gesenkt werden.

LVL3-Trigger: Der LVL3-Trigger existiert bisher nur als Konzept, grundsätzlich sieht er die Einbeziehung der MDC-Information, also eine Spurrekonstruktion, zur Gewinnung eines Impulstriggers vor. Eine mögliche Lösung wird in Kapitel 7 diskutiert.

Eine Zusammenstellung der Trigger- und Datenraten in den einzelnen Triggerstufen liefert Tab. 2.2 auf der nächsten Seite.

Die Information über die Entscheidungen des Triggersystems wird über einen „Trigger-Bus“ an die einzelnen Teilsysteme verteilt.

Der LVL1-Triggerbus signalisiert der Ausleseelektronik, dass ein Ereignis stattgefunden hat und in die „LVL1-Pipe“, transportiert werden soll. Dabei wird jedes Ereignis

	Hz	MByte/s
LVL1	10^5	3 608.0
LVL2	10^3	17.5
LVL3	10^2	1.8

Tabelle 2.2.: Datenraten der verschiedenen Triggerstufen im HADES-Experiment: Die gegenüber Tab. 2.1 auf Seite 10 höhere LVL1-Rate ergibt sich aus zusätzlichen Daten, die für den LVL2-Trigger benötigt werden.

mit einem acht Bit langen „Trigger-Tag“ gekennzeichnet, das zentral erzeugt und verteilt wird. So kann beim Zusammenführen der Daten verschiedener Teilsysteme überprüft werden, ob die ausgelesenen Daten zum selben Ereignis gehören, oder ob ein Versatz entstanden ist. Darüber hinaus wird noch die Art des vom LVL1-Trigger gefundenen Ereignisses als „Trigger-Typ“ signalisiert, um z.B. zentrale von peripheren Kollisionen zu unterscheiden.

Der LVL2-Triggerbus gibt zu jedem LVL1-Trigger einen korrespondierenden LVL2-Trigger an alle Teilsysteme aus. In diesem LVL2-Trigger wird gekennzeichnet, ob die Online-Ereignisklassifikation eine positive oder negative Entscheidung getroffen hat. Damit werden die Daten von der LVL1-Pipe in die „LVL2-Pipe“ umkopiert (positiv) bzw. aus der LVL1-Pipe gelöscht (negativ).

2.3.2. Datenauslese

Die Datenauslese und -speicherung muss einerseits die genannten Datenmengen transportieren können, andererseits eng mit dem Triggersystem zusammenarbeiten. Die Daten müssen bis zur Triggerentscheidung zwischengespeichert werden, bevor ausschließlich die selektierten Daten weitertransportiert werden. Zusätzlich müssen die LVL2- und LVL3-Triggerstufen mit den notwendigen Daten versorgt werden. Daher ist die Datenauslese genau wie das Triggersystem dreistufig aufgebaut.

Die LVL1-Trigger (Multiplizität) werden von der „Frontend“-Elektronik bearbeitet. Diese führt bei jedem LVL1-Trigger die analoge Signalverarbeitung und Analog- zu Digitalwandlung durch. Nach der A/D-Wandlung werden die Messwerte u.U. weiter verarbeitet (z.B. nicht angesprochene Kanäle aus dem Datenstrom entfernt, „Nullenunterdrückung“) und in einen Zwischenspeicher, der LVL1-Pipe abgelegt. Unabhängig von der Datenaufnahme werden bei RICH, PreShower und TOF auch noch die relevanten Daten von der Frontend-Elektronik an die LVL2-Triggerprozessoren weitergegeben.

Das LVL2-Triggersystem klassifiziert das Ereignis und erzeugt den LVL2-Trigger. Dieser wird an die so genannten „Readout-Controller“ verteilt, die daraufhin, bei einem positiv klassifizierten Ereignis, die Daten aus der LVL1-Pipe in die „LVL2-Pipe“ kopieren. Im Falle des MDC-Readout-Controllers müssen auch noch die relevanten Daten an den LVL3-Trigger weitergegeben werden. Bei einer negativen LVL2-Triggerentscheidung werden die Daten in der LVL1-Pipe einfach gelöscht.

Die Readout-Controller sind als VMEbus-Einschub aufgebaut, die LVL2-Pipe ist

ein über das standardisierte VMEbus-Protokoll [31] zugreifbarer Speicherbereich. An der LVL2-Pipe findet also der Übergang statt zwischen speziell auf einen Detektor zugeschnittener Hardware und der gemeinsamen Software, die auf der Basis kommerziell verfügbarer Hardware die weitere Verarbeitung vornimmt.

2.4. Struktur der Software

Die gemeinsame Datenaufnahme-Software hat drei Hauptaufgaben:

Crate-Event-Building: Auslesen der Daten aus allen LVL2-Pipes in einem VMEbus-Crate, Zusammensetzen der Sub-Events zu einer gemeinsamen Datenstruktur, dem „Crate-Event“. Dieses liegt im Hauptspeicher der VMEbus-CPU. Im Vollausbau muss hier noch die Entscheidung des LVL3-Triggers beachtet werden (Kapitel 3).

Datentransport: Transport der Daten vom Crate-Event-Builder zum Event-Builder (Kapitel 4).

Event-Building: Empfangen der Daten von allen Crate-Event-Buildern, Zusammensetzen der Crate-Events zu gemeinsamen Events, Abspeichern dieser Events auf Massenspeicher (Kapitel 5).

All diese Vorgänge werden in den genannten Kapiteln genauer beschrieben, insbesondere werden dort auch die Konzepte, Messungen und Entwurfsentscheidungen dargestellt. In diesem Abschnitt soll zunächst ein Überblick über die Struktur der Software und die damit verbundenen Datenstrukturen gegeben werden, so wie sie in der endgültigen Implementation verwirklicht wurden.

2.4.1. Prozesse

Abbildung 2.2 auf der nächsten Seite gibt eine Übersicht über die Prozesse, die an der Datenaufnahme beteiligt sind.

Auf jeder VMEbus-CPU läuft ein „Readout“-Prozess, der das Crate-Event-Building durchführt und die fertigen Crate-Events in einem gemeinsamen Speicherbereich, einem so genannten „Shared-Memory“, ablegt. Ein Großteil dieses „Readout“-Prozesses ist auf allen CPUs gleich. Die Verfahren, die zur Auslese der LVL2-Pipe ablaufen müssen, sind jedoch je nach Readout-Controller-Typ verschieden. Deshalb ist ein spezieller Teil des Readout-Prozesses an die auszulesende Hardware angepasst. Standardisiert ist dagegen die Weitergabe der Daten über den gemeinsamen Speicherbereich, dies wird für alle Prozesse mit dem Programmteil „shmtrans“ (für „Shared-Memory-Transport“) erledigt.

Der „memnet“-Prozess entnimmt die Daten aus dem Shared-Memory und schickt sie über das Netzwerk weiter. Dieser Prozess kümmert sich um alle Eigenheiten des Netzwerktransports wie z.B. spezielle Größen für Netzwerkpakete, effiziente Ausnutzung der Bandbreite etc. Spiegelbildlich dazu läuft auf dem Event-Builder ein Prozess

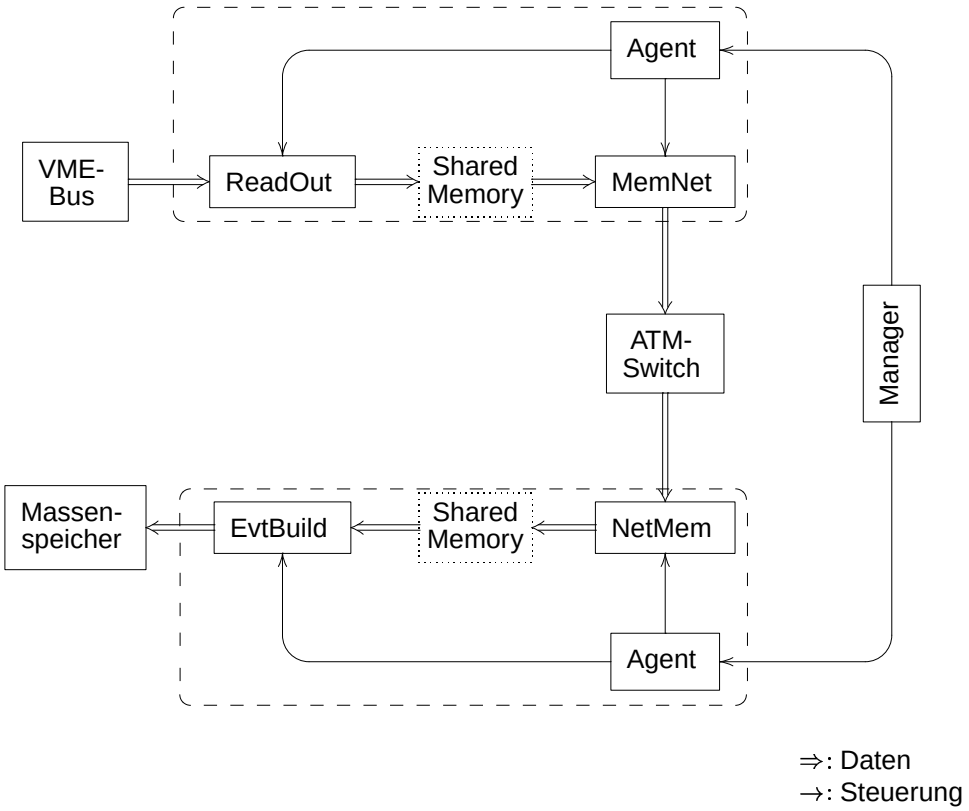


Abbildung 2.2.: Prozesse, die an der Datenaufnahme beteiligt sind. Daten werden vom „Readout“-Prozess aus der Hardware ausgelesen, laufen über die Netzwerk-Sende- und -Empfangsprozesse zum Event-Builder, der sie auf Massenspeicher ablegt. Das ganze System wird durch „Manager“- und „Agent“-Prozesse gesteuert und überwacht.

Abbildung 2.3.: Die einfachste Möglichkeit zur Nutzung der HADES-Datenaufnahme benötigt nur ein VMEbus-Crate.

„netmem“, der die Daten vom Netzwerk empfängt, alle netzwerkspezifischen Formatierungen wieder entfernt und die Crate-Events in einem gemeinsamen Speicherbereich ablegt. Im Gegensatz zum Sendeprozess muss der Empfangsprozess jedoch mit mehreren Gegenstationen kommunizieren können, da er ja mehrere Crate-Event-Builder bedienen muss. Im gemeinsamen Speicherbereich ist das Datenformat exakt so, wie es der Crate-Event-Builder erzeugt hat. Die beiden Prozesse „memnet“ und „netmem“ bilden zusammen den Datentransportteil der Datenaufnahmesoftware, die Funktionen sind im Programmteil „nettrans“ (für „Network-Transport“) realisiert.

Schließlich gibt es auf dem Event-Builder den Prozess „evtbuild“, der das eigentliche Event-Building durchführt. Hierzu entnimmt er den gemeinsamen Speicherbereichen des Empfangsprozesses jeweils ein Crate-Event, fasst diese zu Events zusammen und speichert die Events auf Massenspeicher ab. Zusätzlich stellt er fertige Ereignisse noch an einer Netzwerkschnittstelle zur „Online-Analyse“ zur Verfügung.

Ein System aus n Crates beansprucht also $2n + 2$ Prozesse. Zur Steuerung und Überwachung all dieser Programme dienen deshalb „Agents“, die auf jedem beteiligten Rechner laufen. Diese können wiederum von zentraler Stelle aus von einem „Manager“-Programm gesteuert werden, so dass der Bediener nur mit einem Prozess kommunizieren muss.

Ein interessanter Spezialfall ergibt sich, wenn nur ein einziges Crate ausgelesen werden soll. Da der „nettrans“-Teil transparent ist, also die Daten genau so beim Empfänger abliefert, wie sie der Sender gesendet hat, kann man ihn einfach weglassen (Abb. 2.3). In diesem Fall laufen also Crate-Event-Building und Event-Building auf ein und derselben VMEbus-CPU, ein Netzwerk ist nicht notwendig. Durch die Einfachheit dieser Konfiguration wird auch der ganze Agent/Manager-Überbau überflüssig, jeder Teil der Datenaufnahme kann direkt bedient werden. Diese Konfiguration („Standalone-Mode“) eignet sich besonders für Labor und Einzeltests und hat sich als unabdingbares Hilfsmittel zum unabhängigen Arbeiten und Fehlersuchen erwiesen.

Insgesamt wurde darauf geachtet, dass alle Prozesse ohne besondere Vorkehrungen oder Umgebungen lauffähig sind, so dass sie von Hand gestartet werden können. Dies ist wichtig, um mit den normalen Mitteln der Softwareentwicklung (Debugger, Profiler etc.) arbeiten zu können.

Weiterhin wurde sehr viel Sorgfalt darauf verwandt, bei der Programmierung vorgegebene Normen (ISO-C [32], ISO-POSIX 1003.1(ab) [33]) einzuhalten, oder, wo das nicht möglich war, Anpassungsbibliotheken zu schreiben. Dies erlaubt es, die Software auf unterschiedlichen Betriebssystemen einzusetzen, ohne Anpassungen vornehmen zu müssen. Dies wurde sehr oft ausgenutzt, insbesondere bei Tests auf Linux-VMEbus-

CPUs, wenn kein „LynxOS“ Betriebssystem vorhanden war. Auch der spätere Wechsel vom Betriebsystem „DEC-UNIX“ auf Linux als Plattform für das Event-Building wurde dadurch sehr einfach möglich. Auch in Zukunft sollte die Ausnutzung der jeweils leistungsfähigsten Systeme möglich sein.

2.4.2. Datenstrukturen

Abhängig von der aktuellen Aufgabe muss die Software die Experimentdaten in sehr unterschiedlichen Datenstrukturen bearbeiten und transportieren. Auch wenn die Motivation für einige der Datenstrukturen, insbesondere im Bereich des Datentransports, erst nach einer genaueren Darstellung (Abschn. 4) klar wird, seien sie hier schon alle kurz vorgestellt (Abb. 2.4 auf der nächsten Seite).

Die LVL2-Pipes liefern am VMEbus Sub-Events ab. Alle Sub-Events, die zu einem Trigger gehören, werden vom Crate-Event-Building ausgelesen und zu einer Datenstruktur pro Crate, dem Crate-Event zusammengefasst. Normalerweise werden mehrere Crate-Events in einer Nachricht („Message“) gepuffert und erst beim Erreichen einer vom Netzwerk vorgegebenen Segmentlänge in einem Paket verschickt. In seltenen Fällen kann es aber vorkommen, dass ein einzelnes Crate-Event, und damit die Nachricht, schon länger ist, als es der Netzwerktransport erlaubt. In diesem Fall muss die Nachricht unterteilt werden und in mehreren Paketen verschickt werden. Auf der Empfängerseite werden Paket- und Nachrichtenstruktur wieder rückgängig gemacht, die Crate-Events in ein Event zusammengefasst und zum Schluss mehrere Events in eine Datei geschrieben.

2.5. Datenaufnahmesysteme der Kern- und Teilchenphysik

Zu Beginn der Entwicklung der HADES Datenaufnahme wurden die meisten Experimente der Kernphysik noch mit einer geringen Zahl von auszulesenden Kanälen betrieben (10^2 bis 10^3). Die Betonung lag zumeist auf einer kurzen Totzeit ($100\,\mu\mathrm{s}$), die primären Datenmengen dagegen lagen im Bereich der Schreibraten auf Bandlaufwerke ($1\,\mathrm{MByte/s}$), so dass eine Datenreduktion überflüssig war. Diese Experimente arbeiteten (und arbeiten) mit allgemein verfügbarer, handelsüblicher Nuklearelektronik (NIM- und CAMAC-Standard) und Datenaufnahmesystemen wie in Abschnitt A.4 beschrieben.

Mit durchaus ähnlichen primären Datenraten wie HADES waren dagegen schon Anfang der achtziger Jahre die Experimente der Elementarteilchenphysik wie z.B. DELPHI am CERN-LEP konfrontiert. Bei ca. 250 000 auszulesenden Kanälen und einer „beam-cross-over (BCO)“-Rate von $45\,\mathrm{kHz}$ ergäben sich $10^2\,\mathrm{GByte/s}$ als LVL0-Datenrate. Bei DELPHI sind allerdings die Triggerbedingungen für ein interessantes Ereignis relativ einfach zu formulieren, so dass die ersten beiden Triggerstufen durch reinen Einsatz von kombinatorischer Logik festverdrahtet aufgebaut wurden (Latenz $37\,\mu\mathrm{s}$).

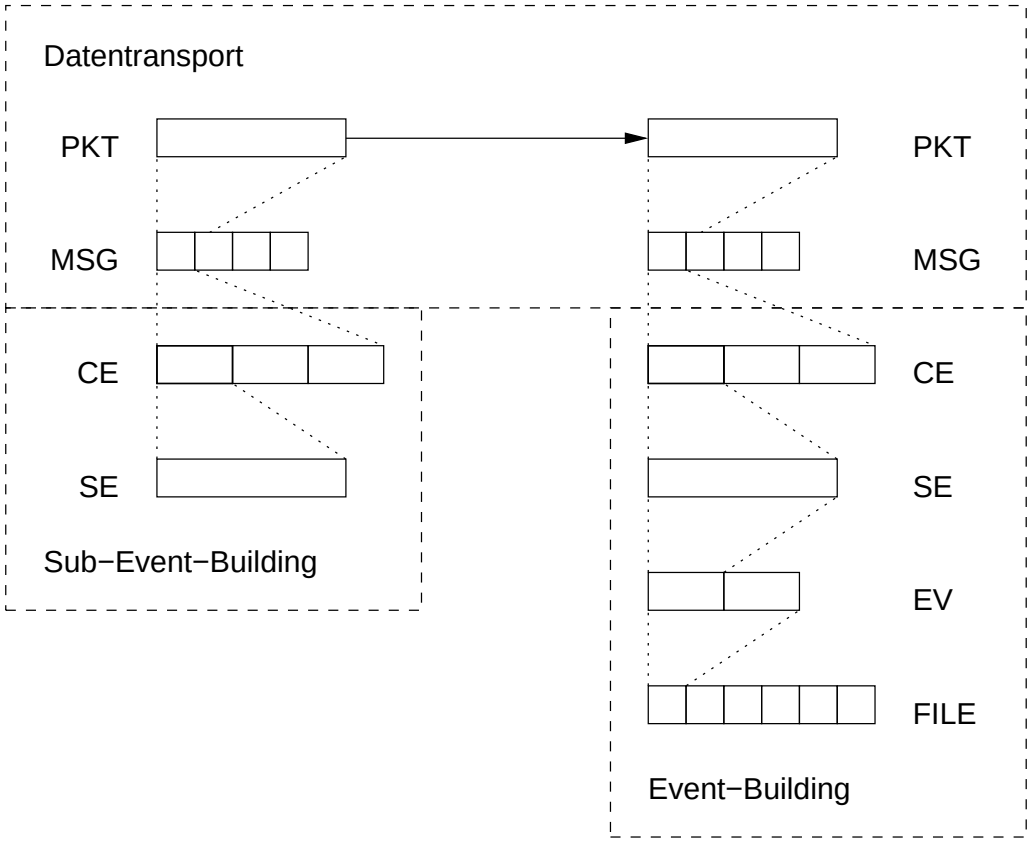


Abbildung 2.4.: Zusammenstellung der verschiedenen Datenstrukturen in der HADES-Datenaufnahme: In Crate-Event-Building werden Sub-Events (SE) aus den Elektronikmodulen ausgelesen und zu Crate-Events (CE) zusammengefasst. Diese werden in Nachrichten (MSG) zu größeren Einheiten gesammelt und diese unter Umständen wieder in kleinere Pakete (PKT) zerteilt. Pakete bilden die Einheit des eigentlichen Datentransports. Auf der Empfängerseite wiederholt sich der ganze Vorgang in umgekehrter Reihenfolge. Zuletzt werden die Sub-Events im Event-Building zu Events (EV) zusammengefasst und diese in Dateien (FILE) geschrieben.

Sie entsprechen daher eher dem HADES LVL1-Trigger. Die Rate, die von der Elektronik auszulesen und transportieren ist, liegt nach dem DELPHI LVL2-Trigger bei nur noch $4 \text{ bis } 8\,\mathrm{Hz}$ [34]. Trotzdem ergibt sich durch die erhebliche Größe der Einzelereignisse, ein typisches Z_0-Ereignis ist ca. $100\,\mathrm{KByte}$ groß, noch eine Datenrate von $1\,\mathrm{MByte/s}$. Der Auslese- und Datentransportteil von DELPHI erforderte durch diese speziellen Bedingungen schon die Entwicklung und den Einsatz von spezieller Ausleseelektronik auf Fastbus-Basis, auch der Datentransport wurde durch eigens von der DELPHI-Kollaboration entwickelte Hardware übernommen.

Entwicklungen, die parallel oder kurz nach Beginn des HADES Projekts in der Mitte der neunziger Jahre begannen, sind HADES in Anforderungen und Lösungen zum Teil ähnlich. So rechnet das COMPASS-Experiment am CERN-SPS (Proposal 1996 [35]), je nach Physikprogramm, mit Triggerraten im Spill von $1 \text{ bis } 100\,\mathrm{kHz}$ und Datenmengen von $30\,\mathrm{MByte} \text{ bis } 6\,\mathrm{GByte}$. Durch Ausnutzung des langen SPS-Beschleunigerzyklus von $14.4\,\mathrm{s}$ gehen die mittleren Datenraten auf $2 \text{ bis } 427\,\mathrm{MByte/s}$ zurück [36]. Die Zahlen, zumindest für den geplanten Vollausbau, gleichen also denen von HADES. Der Zeitplan für die Inbetriebnahme von COMPASS (kleine Datenraten im Jahr 2000/2001, Vollausbau im Laufe von 2002) gegenüber HADES (Vollausbau 1999) führt zu einer noch weitergehenden Ausnutzung von handelsüblichen Technologien und Rechenleistung. So nutzt der Datentransport zwar auch spezielle Übertragungsmedien (S-Link) für die Verbindung Frontend-Elektronik – Zwischenspeicher, und den Schritt Zwischenspeicher – Event Builder bildet dann ebenfalls ein Datennetzwerk (GigaBit-Ethernet). Im Gegensatz zu HADES sind allerdings keine speziellen Triggerstufen vorgesehen, sondern der Einsatz von mehreren, parallel arbeitenden Event-Buildern, ähnlich wie in Abschnitt 7.1 beschrieben. Die Datenreduktion soll durch Software-Filter-Algorithmen auf den Event-Buildern vorgenommen werden. Obwohl vom Datenfluss her unterschiedlich, wäre dies mit der dritten Triggerstufe von HADES (vgl. Abschn. 7.2) vergleichbar.

Dass der von COMPASS und HADES eingeschlagene Weg ein gut skalierbarer und zukunftssicherer ist, zeigt sich auch daran, dass die Planungen für die LHC-Experimente praktisch die selben Konzepte verwenden. Die Anforderungen liegen quantitativ um Größenordnungen höher, z.B. die primäre Datenrate $10^6\,\mathrm{GByte/s}$, die Datenrate am Event-Building $1\,\mathrm{GByte/s}$. Dennoch gleicht die Auslegung mit mehreren Triggerstufen, Computernetzwerken für den Datentransport und parallelem Event-Building und Event-Filtering dem hier vorgestellten [37].

3. Datenauslese über VMEbus

In dem dreistufigen Datenaufnahmesystem stellt die Auslese der primären Detektordaten den ersten Schritt dar und führt für jedes Detektormodul zur Bildung des Sub-Events. Die Auslese dieser Sub-Events über den VMEbus und das Zusammenfassen zum Crate-Event wird im Folgenden beschrieben.

Unter den Oberbegriff „Sub-Event-Auslese“ werden drei Schritte zusammengefasst:

Sub-Event-Building Das Bereitstellen der Detektordaten in einer Form, die sie von der Software aus zugänglich macht.

Datenauslese Das Auslesen dieser Daten über den VMEbus zur VMEbus-CPU.

Crate-Event-Building Das Erzeugen einer Datenstruktur, die einfach zu transportieren ist und später im Event-Builder schnell zu vollständigen Events zusammengesetzt werden kann.

3.1. Die LVL2-Pipe

Bis zur LVL2-Pipe (Abschn. 2.3.2) wird die gesamte Ablaufsteuerung und der Datentransport von spezieller Hardware durchgeführt (Frontend-Module, LVL1-Pipe, LVL2-Trigger etc.). Das letzte Glied in der Kette von Elektronik-Modulen bilden die Readout-Controller. Die Steuerung dieser VMEbus-Karten basiert auf frei programmierbaren Logikbausteinen (FPGA) oder Digitalen Signalprozessoren (DSP). Sie stellen die LVL2-Pipe als Speicher zur Verfügung, der über den VMEbus gelesen werden kann [38, 28, 29]. Damit bekommt man mit handelsüblichen VMEbus-CPUs Zugriff auf die Detektordaten.

Hier waren zwei Fragen zu klären:

1. Wie schnell können Daten von der LVL2-Pipe in die VMEbus-CPU übertragen werden? Welche speziellen Vorkehrungen müssen getroffen werden, um eine hohe Übertragungsgeschwindigkeit zu erreichen?

2. Wie kann der schreibende Zugriff vom Readout-Controller auf die LVL2-Pipe mit dem lesenden Zugriff der VMEbus-CPU so synchronisiert werden, dass keine Konflikte entstehen, die volle Übertragungsleistung aber erhalten bleibt.

3.1.1. Leistung des VMEbus

Datentransport über den VMEbus – „Block Transfer“

Im einfachsten Fall werden Daten über den VMEbus übertragen, indem der „Bus-Master“, hier die VMEbus-CPU, ein Datenwort nach dem anderen einzeln aus dem „Bus-Slave“, d.h. dem Readout-Controller, liest. Dabei wird also der Slave jedesmal neu adressiert, das Wort übertragen und der Slave für das nächste Wort erneut adressiert. Die erreichbaren Übertragungsraten hängen stark von der Leistungsfähigkeit der VMEbus-CPU und der angesprochenen VMEbus-Karte ab. Für die Übertragung von 32bit-Worten liegen sie typischerweise bei ca. $4\,\mathrm{MByte/s}$.

Alternativ dazu kann der VMEbus auch im so genannten „Block-Transfer-Mode (BLT)“ betrieben werden. Hierbei wird nicht zu jedem Zugriff eine Adresse angelegt, sondern es wird nur eine Anfangsadresse und die Zahl der zu lesenden Datenworte festgelegt und in den VMEbus-Steuerbaustein programmiert. Nach dem Start der Übertragung werden die Adressen automatisch um ein Datenwort erhöht. Ein zusammenhängender Block von Datenworten kann dann mit nur einem einzigen Adressierungsvorgang übertragen werden. Theoretisch sind im 32-Bit-Block-Transfer Übertragungsraten bis $40\,\mathrm{MByte/s}$ [31, 39] möglich.

Messungen zur VMEbus-Datenübertragungsrate

Um sicherzustellen, dass die bei HADES benötigten Leistungen auch in der Praxis erreicht werden können, wurden Messungen mit einer „CES RIO8062“ CPU [40] und einer VMEbus-Karte mit dem Chipsatz „CY7C 960/964“ der Firma Cypress durchgeführt [41].

Gemessen wurde die Zeit, die zur Übertragung von $256\,\mathrm{KByte}$ von der VMEbus-Karte zur CPU benötigt wurde. Wie oben geschildert, ergibt sich beim „Block-Transfer“ vor dem Start der Übertragung ein einmaliger Zeitaufwand zum Programmieren des VMEbus-Steuerbausteines. Um diesen Zeitaufwand zu messen, wurden die $256\,\mathrm{KByte}$ nicht in einer einzigen Operation transportiert, sondern in kleineren Einheiten, deren Größe von $128\,\mathrm{Byte}$ durch Verdopplung bis $8\,\mathrm{KByte}$ erhöht wurde, so dass zwischen 2 048 und 32 Übertragungsoperationen notwendig waren. Zu jedem Messpunkt wurde die Messung 100 mal wiederholt, so dass der statistische Fehler $1/10$ des Messwertes beträgt.

Wie Abb. 3.1 auf der nächsten Seite zeigt, liegt für die spezielle Hardware die Übertragungsrate bei $256\,\mathrm{KByte}/73\,\mathrm{ms} = 3.5\,\mathrm{MByte/s}$ für den normalen (D32/A32) VMEbus-Zugriff und bei maximal $256\,\mathrm{KByte}/15\,\mathrm{ms} = 16.5\,\mathrm{MByte/s}$ für den „Block-Transfer“ (BLT32).

Allerdings sieht man auch, dass unterhalb einer Blockgröße von $256\,\mathrm{Byte}$ der erhöhte Aufwand zum Starten des Block-Transfers die schnellere Übertragung überwiegt und der Block-Transfer damit langsamer als der D32/A32-Zugriff wird.

Trägt man, wie in Abb. 3.2 auf Seite 24, direkt die benötigte Zeit gegen die Zahl der Übertragungsoperationen auf, so ergeben sich Geraden, deren Achsenabschnitt die (konstante) Zeit für die eigentliche Datenübertragung angibt, während die Steigung die

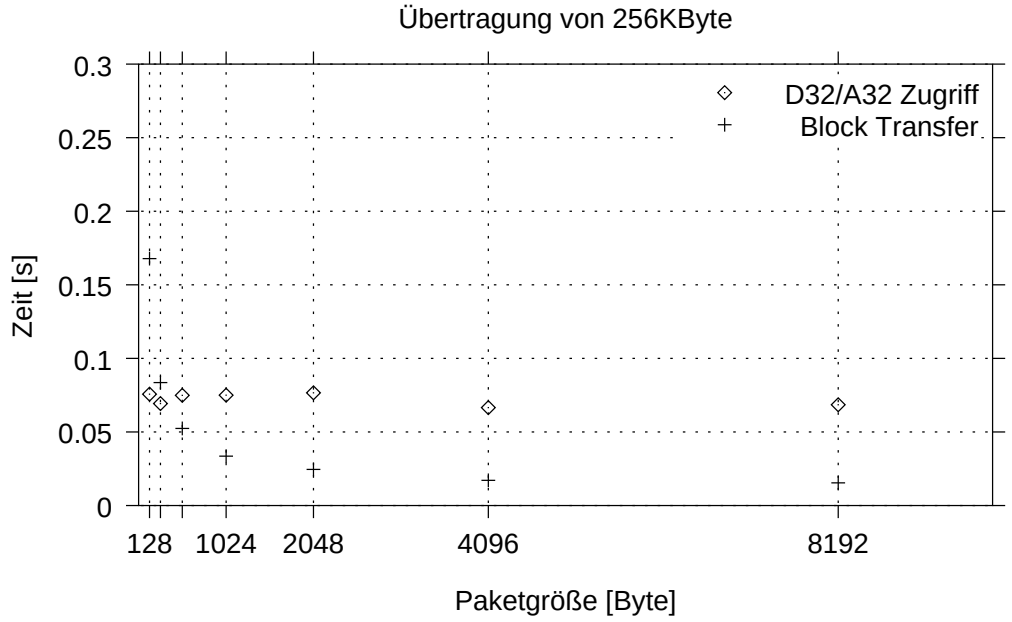


Abbildung 3.1.: Zeitbedarf zum Lesen von $256\,\mathrm{KByte}$ aus einer VMEbus-Karte: Beim Einzelzugriff (D32/A32) ist der Zeitbedarf konstant, während er beim Block-Transfer von der Paketgröße abhängt. Bei großen Paketen ist der Datentransport per Block-Transfer schneller.

zusätzliche Zeit pro Übertragungsoperation zeigt. Die reine Übertragungsgeschwindigkeit liegt demnach bei $256\,\mathrm{KByte}/13.25\,\mathrm{ms} = 19.3\,\mathrm{MByte/s}$. Die Programmierung des VMEbus-Steuerbausteines dauert $70\,\mu\mathrm{s}$.

Die Messergebnisse führen zu diesen Schlussfolgerungen:

- Zum Erreichen von Übertragungsraten von mehr als $4\,\mathrm{MByte/s}$ muss der VMEbus im Block-Transfer-Modus betrieben werden.
- Die Menge der in einer Operation zu übertragenden Daten muss über $256\,\mathrm{Byte}$ liegen.

Da im Block-Transfer-Modus die Daten sozusagen „an der CPU vorbei“ vom VMEbus in den CPU-Speicher übertragen werden, kann die CPU keinerlei Umformatierungen vornehmen und insbesondere keine Daten löschen oder einfügen. Also müssen die Sub-Event-Daten schon genau in der Form in der LVL2-Pipe stehen, in der sie später auch auf das Band geschrieben werden sollen.

3.1.2. Synchronisation

Die LVL2-Pipe ist ein Speicher, der sowohl vom Readout-Controller beschrieben als auch von der VMEbus-Seite gelesen werden soll. Um Wartezeiten zu vermeiden, sollen

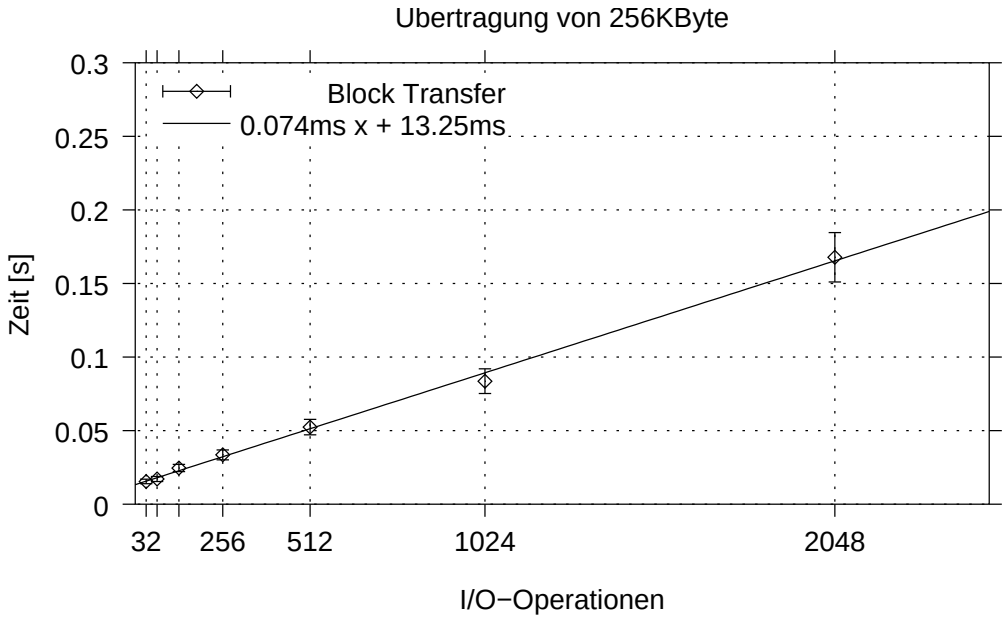


Abbildung 3.2.: Zeitbedarf zum Lesen von $256\,\mathrm{KByte}$ aus einer VMEbus-Karte im Block-Transfer: Der Zeitbedarf steigt linear mit der Zahl der Ein-/Ausgabeoperationen, es gibt einen konstanten Anteil für den Transport und einen für jede I/O-Operation.

beide Operationen gleichzeitig möglich sein. Dafür muss als erstes die Frage der Synchronisation zwischen dem Readout-Controller und der VMEbus-CPU gelöst werden. Das Problem wird klar, wenn man sich Abb. 3.3 anschaut.

Der Readout-Controller und die CPU müssen einen Speicher verwenden, der von beiden Seiten aus zugreifbar ist. Wenn diese Zugriffe allerdings gleichzeitig stattfinden, weil z.B. der Readout-Controller ein Wort schreibt und die CPU ein Wort liest, so ist schon auf Hardwareebene unklar, welche Adresse am Speicher anliegt. Dieses Problem lässt sich durch so genanntes „Dual-Ported-RAM“ lösen. Dies sind Speicherbausteine, die mit doppelten Adress- und Datenleitungen ausgerüstet sind und damit das gleichzeitige Schreiben und Lesen von verschiedenen Adressen erlauben. Diese Bausteine sind allerdings bei gleicher Kapazität teurer und verbrauchen insbesondere mehr Platz als

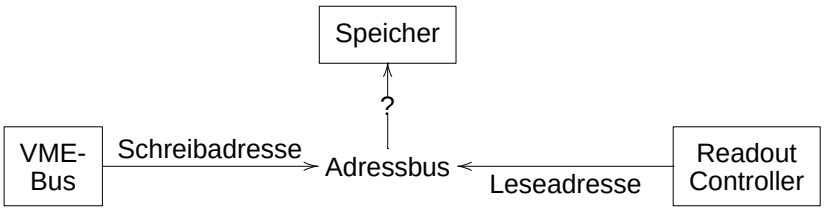


Abbildung 3.3.: Ohne Synchronisation entstehen Zugriffskonflikte in Hard- und Software.

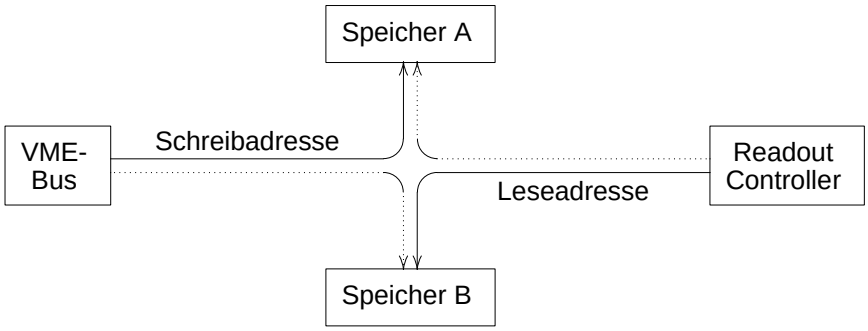


Abbildung 3.4.: Beim Doppelpuffer wird jedem Nutzer ein exklusiver Zugriff auf seinen Speicher garantiert. Ist eine Pufferhälfte voll bzw. leer, so müssen die Partner synchron umschalten.

normale Speicherbausteine.

Auch wenn man die Synchronisation in Hardware löst, so ist damit noch nicht die Synchronisation zwischen dem Readout-Controller und der Datenaufnahmesoftware gelöst. Was passiert, wenn z.B. der Readout-Controller gerade ein Sub-Event schreibt, aber noch nicht fertig geschrieben hat, und die CPU dieses Sub-Event gleichzeitig liest?

All diese Synchronisationsprobleme lassen sich mit einem Verfahren namens „Doppelpuffer“ lösen [42]. Dabei werden zwei getrennte Speicher benutzt, wobei der eine zu einem gegebenen Zeitpunkt nur vom Readout-Controller, der andere zur gleichen Zeit nur von der CPU benutzt werden kann (Abb. 3.4). Synchronisiert werden muss jetzt nur, wenn der Doppelpuffer umgeschaltet werden soll, also die beiden Speicher die Rollen tauschen sollen. Bezüglich der Hardware erlaubt diese Technik den Einsatz von normalen Speicherbausteinen, die Software kann Zugriffskonflikte einfach vermeiden, indem der jeweilige Partner das Umschalten des Doppelpuffers solange verzögert, bis sein Zugriff vollständig abgeschlossen ist.

Abbildung 3.5 auf der nächsten Seite zeigt die Algorithmen zum Lesen und zum Schreiben des Doppelpuffers. Zu beachten sind:

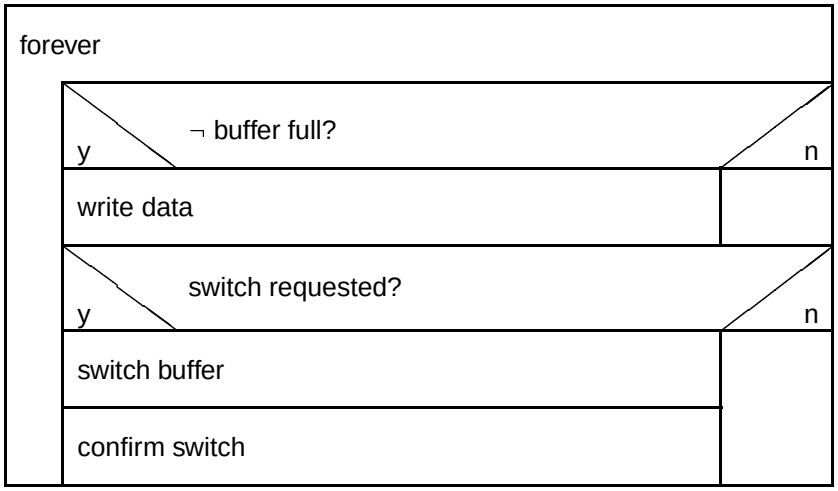
- Der lesende Teil wird beim ersten Zugriff einen leeren Puffer vorfinden, so dass die erste Operation ein „request switch“ ist.

- Der schreibende Teil überprüft nach jedem Schreiben einer Dateneinheit, ob ein „request switch“ gesetzt wurde und bearbeitet diesen gegebenenfalls. Dies ist wichtig, um Verklemmungen („deadlocks“) zu vermeiden.

3.2. Zusammensetzen des „Crate-Events“

Die LVL2-Pipe eines Readout-Controllers präsentiert sich also der VMEbus-CPU als Speicherbereich, der über VMEbus zugreifbar ist. In diesem Speicher liegen hintereinander die Sub-Events von aufeinanderfolgenden Kollisionsereignissen. Ist der Speicher

produce — Write to double buffer



consume — Read from double buffer

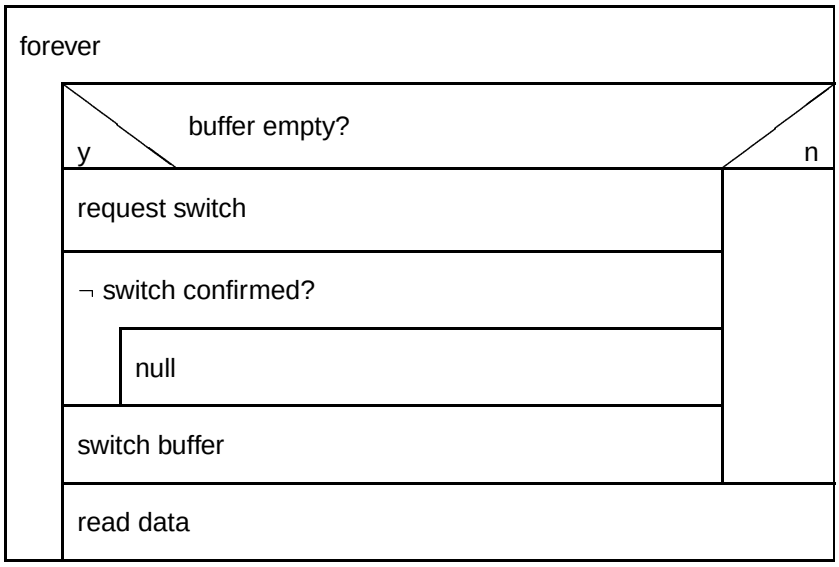


Abbildung 3.5.: Algorithmen zum Schreiben und Lesen des Doppelpuffers

vollständig gelesen, so muss über eine spezielle Operation („switch buffer“) der nächste Speicher voller Daten angefordert werden.

In einem VMEbus-Crate, das von einer CPU gesteuert und ausgelesen wird, werden aber normalerweise mehrere Readout-Controller stecken. Es gehören also zu einem Event typischerweise mehrere Sub-Events pro CPU. Damit der auf die Sub-Event-Auslese folgende Schritt des Event-Buildings nicht zu jedem Crate mehrere Sub-Events verarbeiten muss, sollen alle Daten eines Ereignisses aus einem Crate gemeinsam in einer Datenstruktur transportiert werden. Sie werden deshalb mit dem Algorithmus aus Abb. 3.6 auf der nächsten Seite zum so genannten Crate-Event zusammengesetzt. Da die nun entstandenen Crate-Events ausschließlich für den Transport der Daten und das Event-Building Bedeutung haben, aber keinerlei physikalische Information tragen, werden sie vor dem Schreiben auf Band wieder in einzelne Sub-Events zerlegt und die Crate-Event-Datenstruktur entfernt.

Zum Algorithmus des Crate-Event-Buildings sind die folgenden Anmerkungen von Bedeutung:

- Zuerst wird sichergestellt, dass alle Karten die Daten des aktuellen Ereignisses schon zur Verfügung stellen, also der entsprechende Speicher zugreifbar und das Trigger-Tag korrekt ist (siehe auch Abschn. 2.3.1).

- Dann kann man in einer Operation alle Sub-Events des aktuellen Triggers aus den VMEbus-Karten in den CPU-Speicher kopieren (readData). Der Speicherbereich, in dem die verschiedenen Sub-Events liegen, ist zwar nicht kontinuierlich, sondern auf mehrere Karten verteilt. Die verwendeten VMEbus-Steuerbausteine erlauben es jedoch, nicht nur einen Satz, sondern eine ganze Liste von Anfangsadressen und Datenlängen in einer Block-Transfer-Operation abzuarbeiten.

- Im Kopf jedes Sub-Events steht zwar das Trigger-Tag, der Triggertyp ist dem Crate-Event-Building dagegen nicht bekannt. Es können also keine verschiedenen Ausleseabläufe in Abhängigkeit vom Triggertyp genutzt werden. Dies impliziert, dass zu einem gegebenen Triggertyp entweder alle Karten eines Crates Daten abliefern müssen oder keine, da während der Auslese alle Karten gleich behandelt werden.

- Zu einer Verklemmung kann es kommen, wenn der Algorithmus auf den „confirm switch“ von einem Readout-Controller wartet, während der Puffer eines anderen Readout-Controllers voll ist. Dann verhindert die zweite Karte die Ankunft von neuen Ereignissen, während die erste Karte nicht umschaltet, weil sie auf ein Ereignis wartet. Das kann verhindert werden, wenn jede Karte einen „switch buffer“ erlaubt, sobald der Speicher nicht mehr leer ist.

buildCrateEvent — Assemble one crate event from the sub events of several VME cards

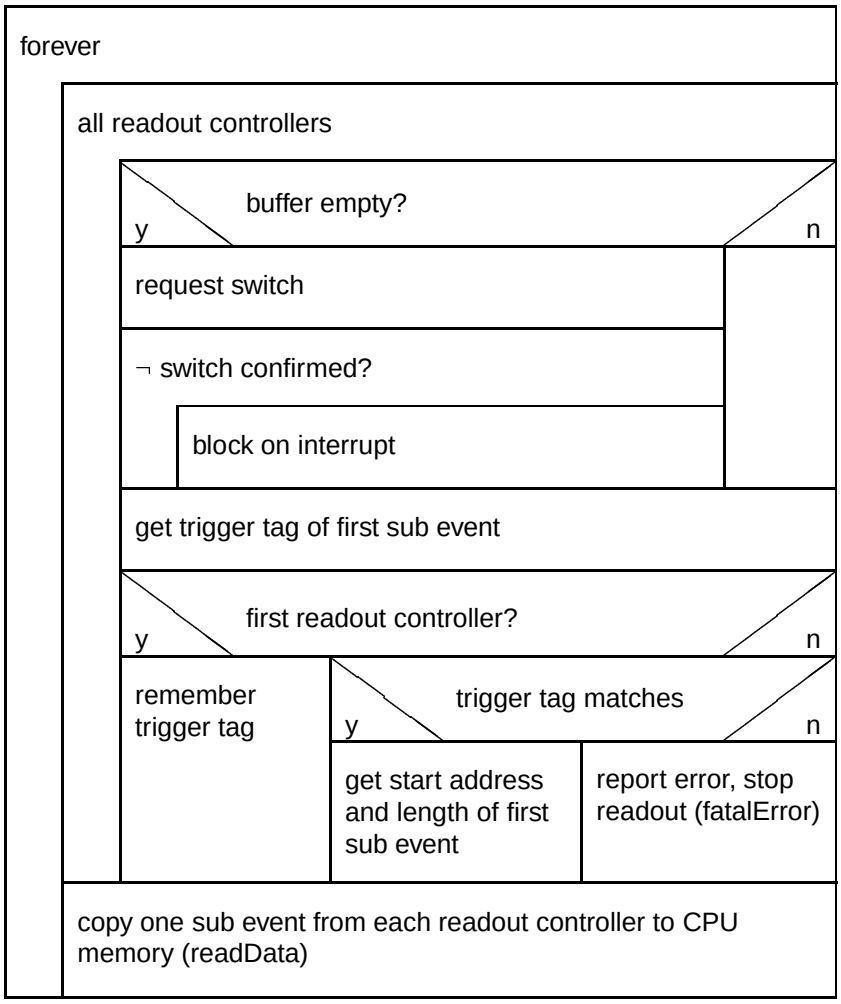


Abbildung 3.6.: Algorithmus zum Zusammensetzen des Crate-Events aus den Sub-Events mehrerer Readout-Controller

4. Datentransport über ATM

Das HADES Experiment soll Datenraten liefern, die mit den zum Zeitpunkt des Designs zur Verfügung stehenden Datentransportverfahren nurmehr schwer oder nicht zu beherrschen waren. Um ein leistungsfähiges und insbesondere auch in der Zukunft erweiterbares System zu schaffen, wurden systematische Voruntersuchungen durchgeführt und schließlich ein ATM-Netzwerk für den Datentransport ausgewählt.

4.1. Vorüberlegungen

Bei einfachen Datenaufnahmesystemen (Abb. A.2 auf Seite 82) ist das Problem des Datentransports meistens implizit gelöst. Vom Rechner, der Auslese und Datenverarbeitung durchführt, gibt es eine spezielle, direkte Verbindung zur auszulesenden Hardware (z.B. CAMAC-Crate-Controller, VMEbus-Interface). Damit sind alle Daten im Adressraum der jeweiligen CPU sichtbar. Datentransport findet durch Adressierung und Zugriff der CPU statt.

Bei Datenaufnahmesystemen, die eine Event-Building-Stufe erfordern (Abb. A.4 auf Seite 85) gilt dies meist noch für den Datentransport von der Digitalisierung zur Auslese (Kapitel 3). Danach ist aber offensichtlich ein weiterer Transport der Daten von der Auslese zum Event-Builder notwendig.

Der konservative Ansatz hierzu ist, die Bussysteme der jeweiligen Überrahmen über Kabel zu verlängern (FastBus, VSB, VIC) und damit alle Daten gleichzeitig im Adressraum der Event-Builder-CPU sichtbar zu machen. Diese Lösung hat den großen Vorteil, die Software sehr einfach zu machen. Das gesamte Experiment erscheint für den Event-Builder wie ein einziges, riesiges Crate, Datentransport findet weiterhin durch einfache Adressierung und Zugriff der CPU, also implizit statt.

Sie hat aber auch entscheidende Nachteile, insbesondere was die Leistungsfähigkeit angeht. Ein durch Hardware zur Verfügung gestelltes, transparentes Abbilden eines Bussystems in den Speicher einer CPU kann die speziellen Anforderungen einer Applikation nicht kennen, sondern muss mit allen erdenklichen Zugriffsmustern zurechtkommen. So erlauben diese Bussysteme Zugriff auf beliebige Datenworte in beliebiger Transportrichtung in beliebiger Reihenfolge, mit dem entsprechend hohen Aufwand an Steuerung und Synchronisation, obwohl z.B. für Datenaufnahme nur der Transport von zusammenhängenden Datenblöcken in einer Richtung benötigt wird.

Deshalb rücken für den Transport von großen Datenvolumen Lösungen in den Vordergrund, die die Daten explizit von einer Station zur anderen kopieren, statt die Ko-

pieroperation hinter vielen kleinen Speicherzugriffen zu verstecken. Man nennt solche Systeme dann nicht mehr „Bus-orientiert“, sondern „Link-orientiert“.

Für den benötigten Link kommen zum einen Speziallösungen (z.B. S-Link [43]) in Frage. Zum anderen können aber auch Technologien aus anderen Bereichen, insbesondere aus der Datenverarbeitung und Telekommunikation, übernommen werden. Das Interesse an solchen „softwarenahen“ Systemen hat aus drei Gründen stark zugenommen:

- Die Transportkapazitäten von Rechner- und Telekommunikationsnetzwerken haben stark zugenommen und unterscheiden sich nicht mehr von schnellen „hardwarenahen“ Lösungen.
- Die Event-Building-Stufe erfordert in der Hauptsache hohen Datendurchsatz und Ein-/Ausgabeleistungen, Eigenschaften, die weniger übliche Steuer- und Regelcomputer, als viel mehr die Rechner und Betriebssysteme aus der klassischen Datenverarbeitung auszeichnet.
- Nicht zuletzt führt das große Marktvolumen im Telekommunikationsmarkt zu guter Verfügbarkeit und einem günstigen Preis-/Leistungsverhältnis der Komponenten.

4.1.1. Netzwerke für Daten und Telekommunikation

Zum Zeitpunkt des Designs der HADES-Datenaufnahme (1996) waren typische Vertreter von schnellen Softwarenetzwerken FDDI (Fiber Distributed Data Interchange [44]) und ATM (Asynchronous Transfer Mode [45]). Fast-Ethernet war gerade in der Entwicklung und mit ersten Komponenten und Installationen verfügbar.

Alle diese Link-Netzwerke haben die gemeinsame Eigenschaft, dass sie die Daten in Form von Paketen verschicken. Es werden also nicht einzelne Datenworte zwischen den Rechnern ausgetauscht, sondern eine Dateneinheit aus mehreren Worten wird mit Information über Sende-, Empfangstation etc. versehen und dann übermittelt. Die Maximalgröße eines solchen Netzwerkpakets ist innerhalb einer Netzwerktechnologie festgelegt. Die Übertragung jedes einzelnen Pakets muss explizit von der CPU gestartet werden, ebenso, wie an der Empfangsstation das Paket durch die CPU entgegengenommen werden muss [46].

Diese Aufgaben überträgt man dem Betriebssystem, hauptsächlich aus Gründen der Einfachheit und Übertragbarkeit der Software von einer Hardware auf die andere. Die Nutzung von Systemresourcen mit Hilfe des Betriebssystems ist allerdings eine relativ aufwendige Angelegenheit, da hierzu ein Systemaufruf notwendig ist, der z.B. einen Wechsel des Prozessors vom Benutzer- in den Systemmodus und zurück erfordert [47].

Anteil des Betriebssystems an der Übertragungszeit

Zur Übertragung einer Nachricht von einem Rechner zum anderen sind also drei Schritte notwendig (Abb. 4.1 auf der nächsten Seite):

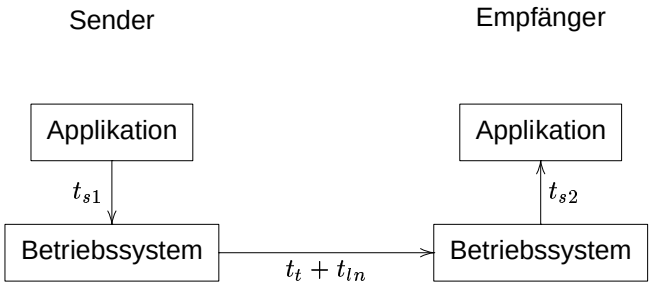


Abbildung 4.1.: Beiträge zur Übertragungszeit: Das Durchführen der Ein-/Ausgabeoperation benötigt die Zeit t_{s1} bzw. t_{s2}. Der Datentransport vom Sender zum Empfänger setzt sich zusammen aus t_t, der Zeitdauer bis alle Daten versendet sind, und t_{ln}, der Laufzeit der Signale.

- Die Sende-Applikation macht einen Systemaufruf, um das Betriebssystem zum Übermitteln der Nachricht zu veranlassen. Dieser Systemaufruf braucht die Zeit t_{s1}. Diese Zeit hat einen konstanten Anteil für den Aufruf selbst. Je nach Implementation werden bei diesem Aufruf aber auch die zu übertragenden Daten vom Benutzer- in den Betriebssystemspeicher und/oder die Speicher der Netzwerkschnittstelle kopiert, so dass in t_{s1} u.U. auch ein Anteil enthalten ist, der von der Paketgröße n abhängt.

- Die Netzwerkschnittstelle sendet die Daten. Bei gegebener Bandbreite hängt diese Zeit t_t ausschließlich von der Paketgröße ab. Bis alle Daten bei der empfangenden Netzwerkschnittstelle angekommen sind, vergeht zusätzlich die konstante Zeit t_{ln}, die Laufzeit einer Information durch das Netzwerk vom Sender zum Empfänger.

- Der Empfänger macht einen Systemaufruf, um die Daten von der Netzwerkschnittstelle abzuholen. Hierfür braucht er die Zeit t_{s2}, die sich wie t_{s1} zusammensetzt.

Wenn man also die Übertragung eines einzelnen Pakets betrachtet, vergeht von dem Zeitpunkt, ab dem die Sende-Applikation beginnt, das Paket zu übertragen, bis zu dem Zeitpunkt, an dem die Information der Empfangs-Applikation vollständig vorliegt, die Zeit

$$t_l = t_{s1} + t_t + t_{ln} + t_{s2}, \tag{4.1}$$

die so genannte Latenz.

Wenn man große Datenmengen, also viele Pakete hintereinander, überträgt, dann ändert sich die Überlegung. Hier zählt ja nur, wie schnell der Sender aufeinanderfolgende Systemaufrufe machen kann. Hierbei ist wichtig, dass das Betriebssystem die aufrufende Applikation nicht so lange aufhält, bis das Paket wirklich übertragen ist, sondern nur, bis die Daten in den Betriebssystem- bzw. Netzwerkschnittstellenspeicher kopiert sind. Es ergeben sich also zwei Fälle:

$t_{s1} > t_t$ Bei sehr kleinen Paketen ist die notwendige Zeit für den Systemaufruf größer als die für die Übertragung. Die Speicher der Netzwerkschnittstelle sind immer leer, jeder Systemaufruf braucht die Zeit t_{s1}. Die pro Sekunde übertragene Datenmenge ist $N = n\frac{1\,\mathrm{s}}{t_{s1}}$.

Dieser Modus ist offensichtlich sehr ineffizient, da die übertragene Datenmenge gar nicht durch die Netzwerkbandbreite, sondern durch die Leistungsfähigkeit des sendenden Rechners gegeben ist. Dazu kommt noch ein schwerwiegenderes Problem: Die Zeit t_{s2}, die der Empfänger für den Systemaufruf pro Paket braucht, ist in der selben Größenordnung wie t_{s1}, aber normalerweise ein paar Prozent größer. Dies liegt daran, dass die Sendeapplikation alle Parameter der zu sendenden Daten kennt, während die Empfangsapplikation diese erst aus den Daten extrahieren muss. Das ergibt einen kleinen Mehraufwand. Mit $t_{s2} > t_{s1}$ kann aber der Empfänger, unabhängig von der Datenmenge, die Daten gar nicht so schnell empfangen, wie der Sender sie sendet.

$t_{s1} < t_t$ Jetzt kann die sendende Netzwerkschnittstelle die Daten nicht so schnell versenden, wie die Sende-Applikation sie anliefert. Die Speicher der Schnittstelle laufen voll, der Systemaufruf muss warten, bis wieder Platz ist.

Dies geschieht offensichtlich bei großen Paketen. Die pro Sekunde übertragene Datenmenge entspricht jetzt wirklich der Netzwerkbandbreite. Ist darüber hinaus $t_{s2} < t_t$, so kann auch der Empfänger die Daten schneller verarbeiten, als sie angeliefert werden.

Übertragung mit Rückantwort – Latenz

Eine weitere gemeinsame Eigenschaft der paketübertragenden Protokolle ist der „Best-Effort“ Ansatz. Das Netzwerk versucht, ein Paket vom Sender zum Empfänger zu übermitteln, soweit möglich. Kommt es allerdings zu einem Fehler, so kann das Protokoll den Fehler nicht selbst beheben, das Paket geht verloren, ohne dass der Sender es bemerkt.

Um eine gesicherte Übertragung von Information zu erreichen, muss also der Empfänger die Ankunft eines Paketes bestätigen, indem er selbst ein Paket zurücksendet. Hier sind sehr viele Verfahren entwickelt worden, von denen hier nur das einfachste besprochen und untersucht werden soll.

Wir gehen davon aus, dass zu jedem Paket, das der Sender verschickt, der Empfänger genau ein Paket (von vernachlässigbarer Größe) zurückschickt. Der Sender wartet mit dem Versand des nächsten Pakets, bis er die Bestätigung für das vorhergehende erhalten hat.

Damit wird aber tatsächlich jedes Paket einzeln versendet, denn die Möglichkeit, den nächsten Systemaufruf zu machen, solange das vorhergehende Paket noch unterwegs ist, entfällt. Die Übertragung eines einzelnen Pakets dauert jetzt also die volle Zeit für den Hinweg plus die volle Zeit für den Rückweg, nämlich die Antwort, also

$$t_T = 2(t_{s1} + t_{s2} + t_{ln}) + t_t. \tag{4.2}$$

Komplexere Verfahren (z.B. „Sliding Windows“ [48]) reduzieren den Einfluss der Netzwerklatenz auf die Bandbreitenausnutzung, allerdings um den Preis eines relativ hohen Aufwands an Software und dementsprechend hoher CPU-Last.

4.1.2. Vorbereitende Messungen zum Datentransport

Um eine Entscheidung zu fällen, ob ein softwaregesteuertes, paketvermittelndes Netzwerk die Anforderungen erfüllen kann und welche der Technologien weiter untersucht werden sollte, mussten zuerst die Zeiten t_{s1}, t_{s2}, t_t und t_{ln} gemessen bzw. abgeschätzt werden.

t_{ln} kann aufgrund der kurzen Wege und wenigen Netzwerkkomponenten als vernachlässigbar erachtet werden.

t_t wurde ohne Messung als der Quotient von Paketgröße und Bandbreite angenommen: $t_t = n/B$.

t_{s1} wurde durch Messungen ermittelt. Diese werden unten erläutert.

t_{s2} wurde als $t_{s2} = t_{s1}$ genähert, da die Abweichung $t_{s2} > t_{s1}$ normalerweise klein (wenige Prozent) ist.

Messungen

Gemessen werden sollte die Zeit t_{s1}, die die Applikation braucht, um den Systemaufruf zum Senden von Daten durchzuführen.

Es wurde ein sehr einfacher Fall gewählt, um die Größenordnung von t_{s1} zu ermitteln. In einer Schleife wurde mit einem „sendto“-Systemaufruf Netzwerkpakete über das Internet-Protokoll „UDP“ versendet. Die Pakete waren jeweils $1\,\mathrm{Byte}$ lang. Verwendet wurde eine $100\,\mathrm{MBit/s}$-Fast-Ethernet Netzwerkschnittstelle. Auf einer Digital Alphastation 4/250 mit $266\,\mathrm{MHz}$ Taktfrequenz wurde bei 100 000 Schleifendurchläufen eine Zeit von $7.13\,\mathrm{s}$ gemessen, so dass die Abschätzung $t_{s1} \approx 70\,\mu\mathrm{s}$ ergab. Damit lag das übertragene Datenvolumen bei $100\,000\,\mathrm{Byte}/7\,\mathrm{s} \approx 15\,\mathrm{KByte/s}$, weit unterhalb der ca. $10\,\mathrm{MByte/s}$, die Fast-Ethernet erlauben würde[1]. Damit ist die gemessene Zeit also nicht durch die Bandbreite des Netzwerks bestimmt.

Ergebnisse

Mit diesen Informationen kann man die Effizienz der Datenübertragung bei verschiedenen Verfahren abschätzen. Allen Überlegungen gemein war eine angenommene Netzwerkbandbreite $B = 10\,\mathrm{MByte/s}$.

[1] Durch das Paketformat von Ethernet werden in Wirklichkeit ca. $1.5\,\mathrm{MByte/s}$ übermittelt, aber auch das liegt noch deutlich unter der Fast-Ethernet-Bandbreite

Eines der oben erläuterten Verfahren war die Übertragung mit Rückantwort. Bei $8\,\mathrm{KByte}$ großen Netzwerkpaketen, der im UDP maximalen Größe, ergibt sich

$$t_t = \frac{8\,\mathrm{KByte}}{10\,\mathrm{MByte/s}} = 781\,\mu\mathrm{s}$$

und

$$t_T = 2(t_{s1} + t_{s2} + t_{ln}) + t_t = 4 \times t_{s1} + t_t$$

die genutzte Bandbreite

$$B_{\text{eff}} = N/t_T = \frac{8\,\mathrm{KByte}}{4 \times 70\,\mu\mathrm{s} + 781\,\mu\mathrm{s}} = 7.54\,\mathrm{MByte/s}.$$

Lässt man die Rückantwort weg, so ergibt sich mit

$$t_T = t_{s1} + t_t$$

der schon nahe an B liegende Wert

$$B_{\text{eff}} = \frac{8\,\mathrm{KByte}}{1 \times 70\,\mu\mathrm{s} + 781\,\mu\mathrm{s}} = 9.54\,\mathrm{MByte/s}.$$

Die typische Größe eines Crate-Events im Experiment wurde mit ca. $1\,\mathrm{KByte}$ abgeschätzt. Wenn man das als Paketgröße einsetzt, ergibt sich

$$t_t = \frac{1\,\mathrm{KByte}}{10\,\mathrm{MByte/s}} = 98\,\mu\mathrm{s}$$

und für eine Übertragung ohne Rückantwort

$$B_{\text{eff}} = \frac{1\,\mathrm{KByte}}{1 \times 70\,\mu\mathrm{s} + 98\,\mu\mathrm{s}} = 5.95\,\mathrm{MByte/s}.$$

Dieser Wert liegt wiederum weit unter der theoretischen Bandbreite $B = 10\,\mathrm{MByte/s}$.

Folgerungen

Wie die Messungen zeigen, müssen zur Ausnutzung der zur Verfügung stehenden Bandbreiten (ca. $100\,\mathrm{Mbit/s}$) zwei Verfahren kombiniert werden.

Zum einen muss das Verhältnis von Übertragungszeit t_t zur Systemzeit t_s möglichst groß werden. Das erreicht man hauptsächlich durch möglichst große Pakete. Weiterhin verbietet dies auch den Einsatz von Softwareprotokollen wie z.B. TCP/IP, da diese komplexe Operationen bei Versenden und Empfang von Paketen durchführen, was wiederum t_s erhöht [49]. Die Datenaufnahmesoftware muss also die Netzwerkhardware möglichst unmittelbar bedienen.

Zum anderen kann auch bei dieser unmittelbaren Nutzung der Netzwerkhardware kein Übertragungsprotokoll mit Rückmeldung eingesetzt werden, da sich so die Netzwerkbandbreite nicht ausnutzen lässt. Ein abgesichertes Protokoll, das die Netzlatenz effizient behandelt, ist komplex und lässt wieder eine große Systemzeit erwarten. Die Datenübertragung muss also als reine Einwegverbindung vor sich gehen, wobei eine Fehlererkennung möglich sein muss, eine Fehlerkorrektur jedoch nicht.

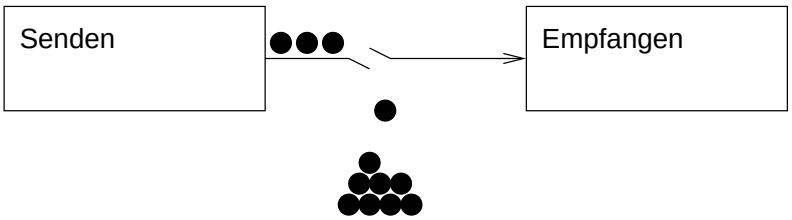


Abbildung 4.2.: Datenverlust durch Fehler während der Übertragung

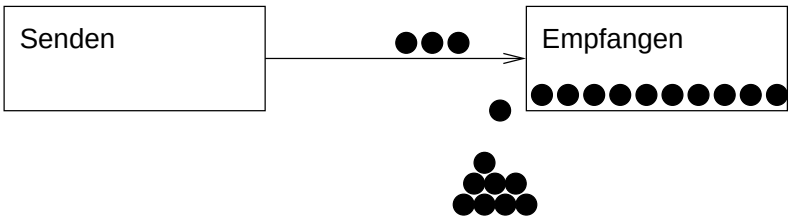


Abbildung 4.3.: Datenverlust durch Überlastung der Empfängers

4.1.3. Übertragungsfehler

Dies führt zur Forderung, dass Fehler bei der Datenübertragung möglichst selten vorkommen sollen. Zwei Fehlerquellen sind dabei zu beachten.

Fehler während der Übertragung

Übertragungsfehler können auftreten während der Übertragung der Daten, in der Hauptsache durch Störung des Signals im Übertragungsmedium (Abb. 4.2). Diese Art von Fehler wird durch ein fehlererkennendes und -behebendes Signalisierungsprotokoll auf der Hardwareebene behandelt. Bei aktuellen Glasfasernetzwerken sind die Signalisierungsprotokolle aus der „SONET“-Familie gebräuchlich [50].

Fehler durch Überlastung

Datenverlust kann allerdings auch an der Station auftreten, die die Daten empfangen soll, nämlich dann, wenn an dieser Station eine Überlastung auftritt („Congestion“) (Abb. 4.3). Dies ist der Fall, wenn das Netzwerk die Daten schneller anliefert, als die Station sie empfangen und verarbeiten kann. Wichtig ist zu bemerken, dass das Problem der Überlastung nicht nur die Endknoten des Netzwerks (Rechner) betrifft, sondern insbesondere auch die Vermittlungsknoten (Switches).

An den Vermittlungsknoten besteht das Problem immer dann, wenn die Menge der empfangenen Daten kontinuierlich die Bandbreite des Ausgabekanals übersteigt. Je nach Architektur des Vermittlungknotens kann das der einzige Grund für Datenverlust sein („Non Blocking Switch“), bei einfacheren Vermittlungsmethoden kann Über-

lastung aber auch auftreten, obwohl noch Ausgabekapazität frei ist („Blocking Switch“) [51].

An den Endknoten ist die Datenmenge, bei der Überlastung auftritt, sowohl abhängig von der Hardware, die den Rechner ausmacht, als auch von der Software, die die Daten weiterverarbeiten soll. Hier muss unter Software nicht nur das Anwendungsprogramm verstanden werden, sondern die Gesamtheit von Betriebssystem und allen Programmen auf dem Rechner.

Hierbei unterscheidet man zwei Bereiche, in denen jeweils wiederum mehrere Faktoren zusammenwirken.

Datendurchsatz Der empfangende Rechner muss im Mittel in der Lage sein, die Datenmenge zu verarbeiten und zu speichern, da er keinerlei Möglichkeit hat, den anfallenden Datenstrom zu verlangsamen. Diese Eigenschaft ergibt sich hauptsächlich aus der Leistungsfähigkeit der verwendeten Hardware und der Ausnutzung dieser Hardware durch das Betriebssystem. Generell kann man davon ausgehen, dass Betriebssysteme, die für den Einsatz in Servern im kommerziellen Bereich gedacht sind, auf Datendurchsatz optimiert sind.

Antwortzeit Der empfangende Rechner muss aber auch zu jedem einzelnen Paket in der Lage sein, dieses rechtzeitig von der Netzwerkschnittstelle abzuholen. Einfache Netzwerkschnittstellen haben Empfangspuffer von ca. $256\,\mathrm{KByte}$. Bei einer Datenmenge von $10\,\mathrm{MByte/s}$ wäre dieser Speicher also nach $25\,\mathrm{ms}$ vollgelaufen, so dass weitere Pakete verlorengingen. Innerhalb dieser Zeitspanne muss der Rechner also garantiert irgendeine andere Aufgabe (verarbeiten, speichern, steuern) unterbrechen und die Daten von der Netzwerkschnittstelle abholen. Diese Fähigkeit hängt hauptsächlich mit dem Betriebssystem zusammen, das einen so genannten „Echtzeit-Scheduler“ [52] haben muss.

4.1.4. Datenpufferung und Segmentierung

Im Normalfall ist die Größe eines Crate-Events viel kleiner als die Maximalgröße eines Netzwerkpakets. Die Forderung nach möglichst großen Paketen kann man nur erfüllen, indem man die Crate-Events von mehreren Ereignissen zusammenfasst und in einem Paket verschickt. Die sendende Station (der Crate-Event-Builder) muss also die Daten puffern, bis ein Netzwerkpaket voll ist, und dann erst verschicken. Andererseits muss die empfangende Station (der Event-Builder) die Netzwerkpakete zuerst wieder in Crate-Events zerlegen.

Eine weitere Vorkehrung muss für den Fall getroffen werden, dass die Größe eine Crate-Events die eines Netzwerkpakets übersteigt. Dies ist z.B. dann der Fall, wenn die Ausleseelektronik ohne Nullenunterdrückung betrieben wird, um Eichdaten zu gewinnen.

Hier muss die sendende Station das Crate-Event zuerst in Segmente zerlegen, die jeweils in einem Paket transportiert werden können. Der Empfänger muss aus diesen

Segmenten wieder das ursprüngliche Crate-Event zusammensetzen.

4.2. Wahl der Netzwerktechnologie

Als erstes musste also eine Netzwerktechnologie gefunden werden, die sowohl die großen Bandbreiten bereitstellt, nach Möglichkeit mit einem absehbaren Weg zu noch höheren Leistungen, als auch die unter Abschn. 4.1.3 erläuterten Schwierigkeiten möglichst vermindert.

FDDI bietet zwar ein deterministisches Verhalten, erlaubt den Einsatz von „Non-Blocking-Switches“ und hätte mit $100\,\mathrm{MBit/s}$ den Anforderungen der ersten Ausbaustufe genügt. Gegenargumente waren der relativ hohe Preis und vor allem die fehlende Ausbaubarkeit, da zum Zeitpunkt des Designs für FDDI keine Bandbreitenerhöhung vorgesehen war.

Ethernet war in der $100\,\mathrm{MBit/s}$-Version (Fast-Ethernet) verfügbar und die nächste Stufe, „GigaBit-Ethernet“ mit $1\,000\,\mathrm{MBit/s}$ in der Diskussion. Ein klares Argument für Ethernet war auch der Preis, der deutlich unter dem von FDDI oder ATM gelegen hätte. Bei Fast-Ethernet grundsätzlich nicht verfügbar und bei GigaBit-Ethernet noch völlig undefiniert waren dagegen Verfahren zur Vermeidung von Datenverlust durch Überlastung („Traffic-Management“). In einem reinen Rechnernetzwerk sind diese auch entbehrlich, für die Datenaufnahme ohne Bestätigungsprotokoll jedoch unbedingt notwendig. Sie hätten daher als Softwarelösung zusätzlich implementiert werden müssen, was einen höheren Personal- und Zeitaufwand für die Erstellung des Datenaufnahmesystems bedeutet hätte.

Die Untersuchungen konzentrierten sich deshalb auf ATM, das von Leistung und Eigenschaften her eine gute Eignung erwarten ließ.

4.2.1. Asynchronous Transfer Mode (ATM)

Asynchronous Transfer Mode (ATM) ist eine Netzwerktechnologie, die die Nutzung der selben Infrastruktur für unterschiedliche Anwendungen erlaubt. Von der CCITT[2] zuerst hauptsächlich als modernes Übertragungsmedium für den Telekommunikationsbereich geplant (Sprache, Video, Wide-Area-Computer-Netzwerke), wurde ATM vor allem vom ATM-Forum auch für Anwendungen im reinen Computer- und Local-Area-Netzwerk-Bereich populär gemacht. Dies ergibt eine einzigartige Eignung als Netzwerk in der Datenerfassung.

Wegen der Auslegung für allgemeine Telekommunikation hat ATM viele Eigenschaften, die in der Messtechnik unabdingbar sind (vorhersagbares Echtzeitverhalten, nicht blockierende Switching-Technologie, Steuerung der maximalen Übertragungsrate). Andererseits ist ATM für den Einsatz im LAN-Bereich mit Schnittstellen und Software für Standard-Computersysteme verfügbar.

Einige Eigenschaften von ATM sind dagegen für die Messtechnik bisher überflüssig, wie z.B. die Auslegung auf sehr große Netzwerke.

[2] Consultative Commitee International Telegraphy and Telephony

In folgenden sollen die wichtigsten Eigenschaften von ATM im Bezug auf Datenaufnahmesysteme, zum Teil auch im Vergleich mit konkurrierenden Technologien, aufgeführt werden.

Begriffsbestimmungen – Grundlegendes

Physikalisch bildet ein ATM-Netzwerk eine Stern-Topologie, wobei jede Station an einem Anschluss des Vermittlungsknotens (Switch) angeschlossen ist. Grundsätzlich geht ATM davon aus, dass ein Netzwerk aus mehreren Switches besteht, aber für das Datenaufnahmesystem eines Experiments wird im Normalfall ein Knoten ausreichen. Andererseits lassen sich auch zwei Stationen direkt miteinander verbinden, dann kann der Switch sogar ganz entfallen.

Die Verkabelung erfolgt normalerweise durch Glasfaser[3], wobei die z.Z. gebräuchlichen Bandbreiten $155\,\mathrm{MBit}$ (OC3[4]), $622\,\mathrm{MBit}$ (OC12) und in letzter Zeit auch $2.4\,\mathrm{GBit}$ (OC24) sind.

Der zentrale Begriff im Bezug auf die Vermittlung von Datenpaketen ist der virtuelle Kanal („Virtual-Channel“, VC). Jeder virtuelle Kanal ist eine von mehreren möglichen, logischen Verbindungen, die durch eine physikalische Verbindung gelegt werden. Zusätzlich existiert noch das Konzept des „Virtual-Path“ (VP), das aber bei kleinen Netzwerken nicht zum Tragen kommt.

Die Adressierung im ATM-Netzwerk erfolgt also durch die Angabe der Hardwareverbindung, die genutzt werden soll, und des virtuellen Kanals.

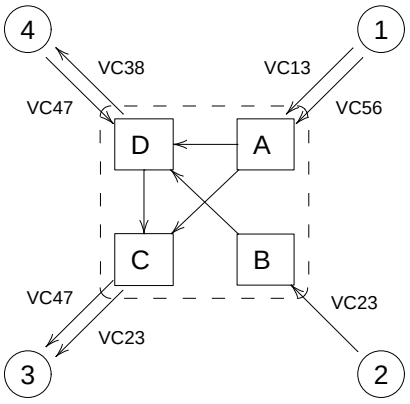
Die eigentliche Vermittlung, also das Ansprechen des tatsächlich gewünschten Knotens, übernimmt dann der ATM-Switch. Die Vermittlungsmatrix im Switch gibt an, welcher virtuelle Kanal von welcher Eingangsschnittstelle auf welchen virtuellen Kanal welcher Ausgangsschnittstelle geschaltet werden soll. Abbildung 4.4 auf der nächsten Seite zeigt ein Beispiel, wie die Verschaltung in einem ATM-Switch aussehen könnte. Während diese Vermittlungsmatrix in dynamischen Umgebungen wie Rechnernetzen, in denen jeder Rechner beliebig mit jedem anderen Rechner kommunizieren will, automatisch und u.U. jedesmal neu aufgebaut werden muss (dies ist bei ATM auch möglich), bietet ATM die Möglichkeit, die Vermittlungsmatrix festzulegen. Es tritt also keinerlei zusätzlicher oder gar undeterministischer Aufwand für das Aushandeln einer Verbindung auf.

4.2.2. Steuerung der Netzlast

Die vielleicht entscheidende Fähigkeit von ATM für die vorliegende Anwendung ist die Möglichkeit, einzelnen virtuellen Kanälen Anteile der insgesamt zur Verfügung stehenden Bandbreite zuzuweisen („Traffic-Management“). Damit lässt sich eine Überlastung sowohl des Switches als auch der empfangenden Station vermeiden.

Dass die sendenden Stationen im Mittel nicht mehr Daten produzieren dürfen als die empfangende Station verarbeiten und speichern kann, ist offensichtlich. Aber selbst

[3] Die $155\,\mathrm{MBit}$-Variante kann auch mit UTP5 Kupferkabeln betrieben werden.
[4] Optical Carrier Level 3



Port	VC	Port	VC
A	13	D	38
A	56	C	23
B	23	D	38
D	47	C	47

Abbildung 4.4.: Ein ATM-Switch (gestrichelt), der zwischen vier Netzwerkknoten vermittelt. Die Tabelle zeigt die Vorschrift, mit der die Ein- und Ausgänge (Port) und die Virtuellen Kanäle (VC) miteinander verschaltet werden.

unter diesen Bedingungen kann es bei einem Netzwerk ohne „Traffic-Management“ zu Überlastung kommen.

Als Beispiel betrachte man Abb. 4.5 auf der nächsten Seite. Die beiden Sender zusammen produzieren im Mittel nur $14\,\mathrm{MByte/s}$ an Daten. Da dieser Datenstrom aber nicht kontinuierlich ist, sondern in Spitzen erfolgt (z.B. wenn Datenpuffer vollgelaufen sind), können am Switch kurzzeitig bis zu $22\,\mathrm{MByte/s}$ auftreten. Diese können aber über die $15\,\mathrm{MByte/s}$ Ausgangsleitung nicht verschickt werden, so dass es im Switch zur Überlastung kommt. Da das ganze statistisch verteilt erfolgt, kann man auch nicht davon ausgehen, dass die internen Datenpuffer im Switch die Daten aufnehmen können.

Eine mögliche Lösung ist, den Switchausgang so zu bemessen, dass er immer alle Eingänge weiterleiten kann, also z.B. $10 \times 10\,\mathrm{MByte/s}$ Eingänge, $100\,\mathrm{MByte/s}$ Ausgang. In Spitzen würde dann allerdings eine Übertragung mit $100\,\mathrm{MByte/s}$ an den Empfänger erfolgen. Das können aber keine handelsüblichen Rechner mehr empfangen (z.B. gemessene Übertragungsrate GigaBit-Ethernet-Karte – 32bit PCI-Bus: ca. $50\,\mathrm{MByte/s}$ [53])

Bei ATM kann man im Voraus vorgeben, wieviel von der vorhandenen Bandbreite eines Anschlusses von welchen virtuellen Kanälen wirklich genutzt werden darf. Damit kann man die Übertragungsrate auf dem Switch-Ausgang an die Verarbeitungsgeschwindigkeit des Empfängers anpassen und diese dann wiederum auf die Eingänge aufteilen.

4.2.3. Echtzeitverhalten

Neben Festlegungen zur Steuerung der Netzlast kann man auch noch garantierte Übermittlungszeiten von einem Knoten im Netzwerk zu einem anderen vereinbaren. Bei einem einfachen Datentransport, wie bisher implementiert, ist diese Eigenschaft sicher verzichtbar, aber bei Einführung des LVL3 Triggers und damit auch eines LVL3 Trigger-

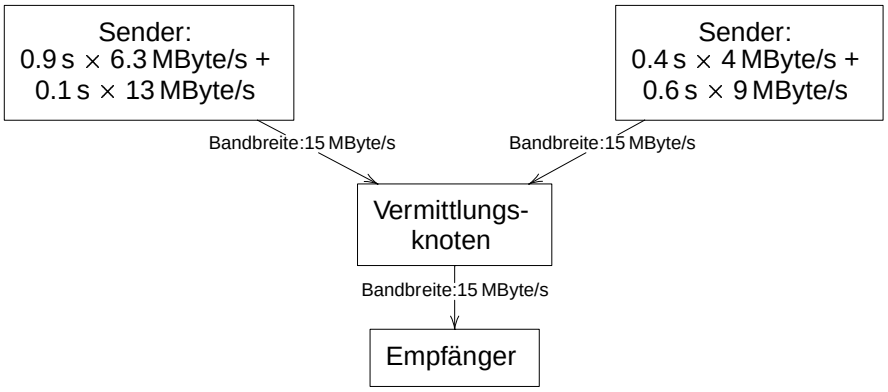


Abbildung 4.5.: Obwohl jeder Sender im Mittel $7\,\mathrm{MByte/s}$ produziert, kann in Spitzenzeiten die Ausgangsbandbreite des Vermittlungsknotens überschritten werden. Eine Steuerung der Netzlast würde die effiziente und gleichmäßige Nutzung der Bandbreiten erlauben.

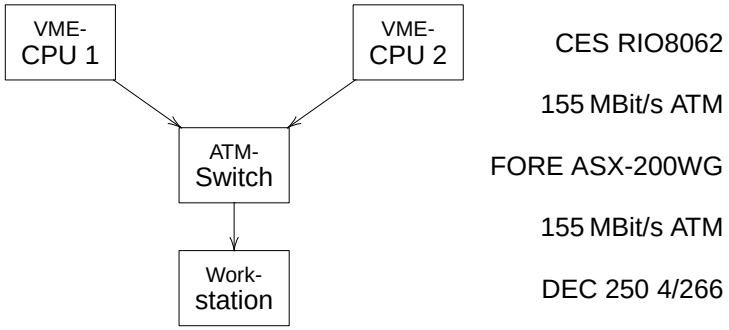


Abbildung 4.6.: Testumgebung zu Messung von Eigenschaften des ATM-Netzwerkes

busses über das Netzwerk ist eine garantierte Latenz sehr wichtig für die Konfiguration der LVL3 Zwischenpuffer.

4.2.4. Messungen im Labor

Testaufbau

Um die praktische Nutzbarkeit von ATM für ein Datenaufnahmesystem zu prüfen, wurde im Frühjahr 1998 eine Testumgebung geschaffen, die Messungen von Eigenschaften und Leistungsdaten von ATM erlaubte (Abb. 4.6). Diese bestand aus zwei VMEbus-CPUs vom Typ „CES RIO8062“ unter „LynxOS Version 2.4“, jeweils mit $155\,\mathrm{MBit/s}$-ATM-Schnittstelle, einem „Fore ASX-200WG“ ATM-Switch und einer Workstation vom Typ „DEC 250 4/266“ unter „DEC-UNIX 3.2“.

Mit dieser Anordnung sollten drei Fragen geklärt werden:

1. Stimmen die in Abschn. 4.1.2 gemachten Abschätzungen und Voraussetzungen mit der Praxis überein?

2. Können die VMEbus-Rechner die volle ATM-Bandbreite mit Daten versorgen, ohne durch die CPU-Leistung begrenzt zu sein? Und kann eine normale Workstation unter einem UNIX-Betriebssystem den Datenstrom empfangen, ohne dabei Pakete zu verlieren?

3. Sind die Softwareschnittstellen (Treiber, Bibliotheken) von Funktionsumfang und Qualität her ausreichend für das geplante Projekt. Kann insbesondere das „ATM-Traffic-Management“ von der Software aus genutzt werden?

Messungen

Die Abschätzung der nutzbaren Bandbreite in Abhängigkeit von der Paketgröße ließ sich durch die Messungen bestätigen. Dabei wurde von einer der VMEbus-CPUs ein Datenbereich von $256\,\mathrm{KByte}$ über ATM versendet, wobei der Datenbereich in Netzwerpakete zwischen $128\,\mathrm{Byte}$ und $8\,\mathrm{KByte}$ aufgeteilt wurde. Die Messungen wurden mit und ohne Bestätigungsprotokoll durchgeführt.

Die theoretisch nutzbare Bandbreite bei ATM über OC3 liegt bei $16.6\,\mathrm{MByte/s}$, also benötigt man zum Übertragen von $256\,\mathrm{KByte}$ $15\,\mathrm{ms}$. Dieser Wert wird, wie Abb. 4.7 auf der nächsten Seite zeigt, erst ab einer Paketgröße von $4\,\mathrm{KByte}$ erreicht, mit einem Bestätigungsprotokoll nie. Gleichzeitig ist damit gezeigt, dass die „RIO8062“ CPUs, entsprechende Paketgrößen vorausgesetzt, die ATM-Verbindung saturieren können.

Mit der selben Messung kann auch der Zeitbedarf für einen Systemaufruf berechnet werden, Abb. 4.8 auf Seite 43 zeigt dazu die Messwerte in einer anderen Darstellung. Auf der Abszisse ist hier die Zahl der notwendigen Ein- bzw. Ausgabeoperationen aufgetragen. Die Steigung der Geraden gibt also gerade den Zeitbedarf für eine Ausgabeoperation an und wurde mit einer Anpassung nach der Methode der kleinsten Quadrate zu $130\,\mu\mathrm{s}$ bestimmt. Auch das stimmt in der Größenordnung mit dem Wert aus den Abschätzungen (vgl. Abschn. 4.1.2) überein.

In einem weiteren Versuch wurde von beiden VMEbus-CPUs gleichzeitig mit maximaler Datenrate gesendet, die Workstation sollte den Datenstrom empfangen. Wie erwartet gingen hier Pakete verloren, da die Eingangsbandbreite die Ausgangsbandbreite um den Faktor zwei übertraf. Daraufhin wurde die Bandbreite für die VMEbus-Rechner auf jeweils $7\,\mathrm{MByte/s}$ begrenzt. Jetzt konnte die Workstation alle Pakete ohne Verlust empfangen. Damit war gezeigt, dass das ATM-Traffic-Management wie erwartet arbeitet. Zusätzlich ergab diese Messung, dass die eingesetzte Workstation einen Datenstrom von $14\,\mathrm{MByte/s}$ verlustfrei empfangen kann. Die dabei beobachtete CPU-Auslastung von 50 % und die Tatsache, dass die „DEC-UNIX-Echtzeiterweiterung“ dazu genutzt werden musste, zeigten jedoch, dass mit dieser Hardwarekonfiguration bereits die Grenze des machbaren erreicht war.

4.3. Implementation

Nachdem eine Rechner- und Netzwerktechnologie gefunden war, die während der Tests die Anforderungen erfüllen konnte, musste als nächstes die Software entwickelt werden,

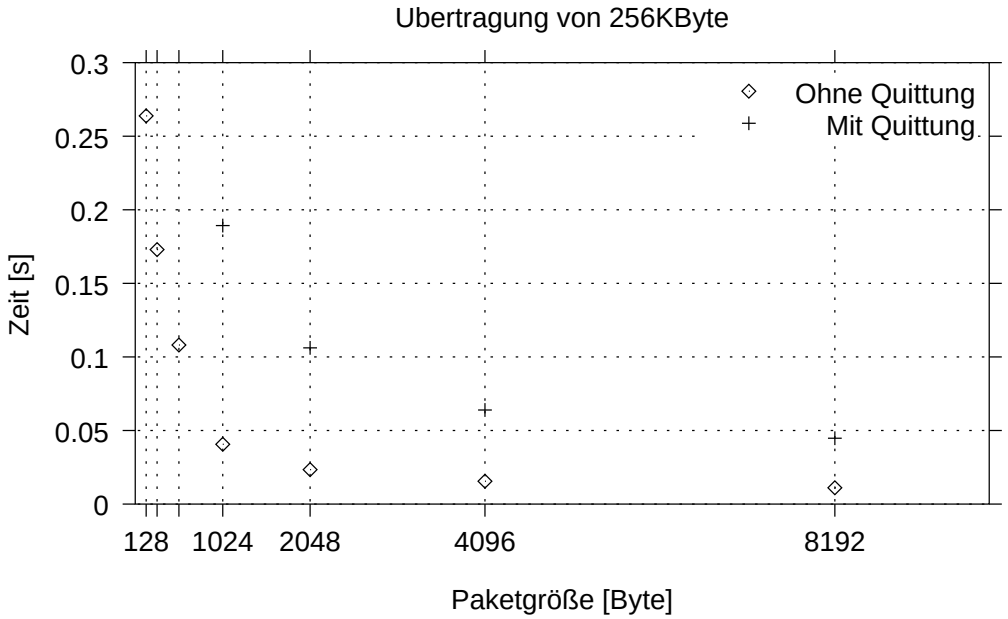


Abbildung 4.7.: Abhängigkeit der Datenübertragungszeit von der Netzwerkpaketgröße: Es wurde jeweils die Zeit gemessen, um $256\,\mathrm{KByte}$ zu übertragen. Es wurde zum einen ein Protokoll eingesetzt, das den Empfang von Paketen durch ein Quittungspaket bestätigt, zum anderen wurde ohne ein solches Protokoll gearbeitet.

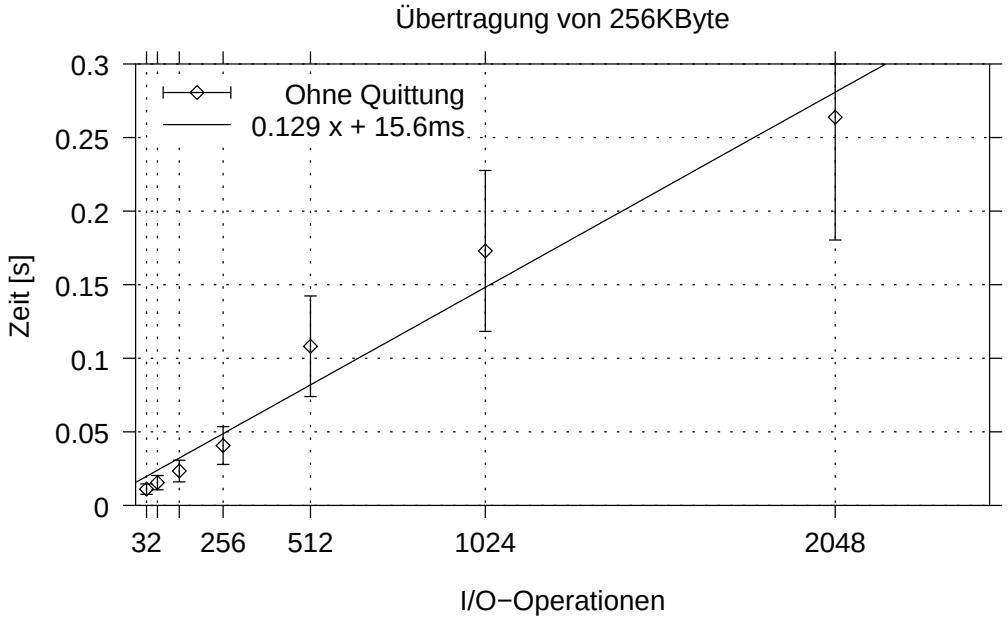


Abbildung 4.8.: Abhängigkeit der Datenübertragungszeit von der Zahl der Ein-/Ausgabeoperationen. Es wurde jeweils die Zeit gemessen, um $256\,\mathrm{KByte}$ zu übertragen, aufgeteilt auf 32 bis 2 048 Systemaufrufe. Eine Gerade durch die Datenpunkte gibt durch ihre Steigung die pro Systemaufruf benötigte Zeit.

die die volle Leistung von ATM für die Datenaufnahme zur Verfügung stellt.

Dieser Datentransportteil der Datenaufnahme findet im Speicher der VMEbus-CPU fertig formatierte Crate-Events vor. Diese müssen zum Event-Builder übertragen werden.

4.3.1. Datenpufferung

Wie im Abschn. 4.1.4 schon ausgeführt wurde, müssen für einen effizienten Transport die Daten mehrerer Ereignisse vor dem Versand zu größeren Einheiten ($> 8\,\mathrm{KByte}$) zusammengefasst werden. Die dazugehörige Datenstruktur, die also ihrerseits wieder mehrere Crate-Events enthält, ist die so genannte „Nachricht“ („Message“). Sie ist nur für den Versand der Daten von Bedeutung und wird auf der Empfangsseite wieder entfernt.

Die Größe dieser Nachrichten ist einer der empfindlichsten Einstell-Parameter des Systems.

Bei sehr kleinen Nachrichten übersteigt der Zeitaufwand für den Systemaufruf den der eigentlichen Datenübertragung, so dass der Empfänger unweigerlich zu langsam wird, um alle Daten zu empfangen (Abschn. 4.1.1).

Bei sehr großen Nachrichten wird die Nachricht größer als ein Netzwerkpaket, so dass sie vor dem Versenden unterteilt und beim Empfänger wieder zusammengesetz werden muss. Dieser Mechanismus bleibt zwar für sehr große Crate-Events notwendig, jedoch sind Nachrichten, die in ein Netzwerkpaket passen, aus mehreren Gründen vorteilhaft:

- Das Wiederzusammensetzen von kleinen Netzwerkpaketen zu großen Nachrichten erfordert Überprüfungs- und Vergleichsoperationen, bedeutet also zumindest einen kleinen Zusatzaufwand und damit Leistungsverlust.

- Beim Verlust eines Netzwerkpakets ist die gesamte Nachricht unbrauchbar, also gehen bei einer sehr großen Nachricht viele Crate-Ereignisse verloren. Da ein einziges fehlendes Crate-Ereignis wiederum das Gesamt-Ereignis unbrauchbar macht, ist der Verlust an Ereignisdaten bei sehr großen Nachrichten größer.

Die ideale Nachrichtengröße wäre also knapp unter der Netzwerkpaketgröße.

Allerdings kommt eine weitere Komplikation ins Spiel. Der Event-Builder muss ja alle schon empfangenen Crate-Events aufheben, bis auch das letzte Crate-Event des aktuellen Ereignisses verfügbar ist. Bei sehr unterschiedlichen Crate-Event-Größen haben in einer Nachricht aber sehr viele kleine Crate-Events Platz, so dass es lange dauert, bis sie verschickt werden. In dieser Zeit sind aber schon viele Nachrichten mit großen Crate-Events beim Event-Builder angekommen.

So sind Crate-Events des Trigger-Crates nur ca. $50\,\mathrm{Byte}$ groß, während Crate-Events vom RICH im Eichmodus ca. $40\,000\,\mathrm{Byte}$ groß sind.

Nimmt man eine Nachrichtengröße von $64\,\mathrm{KByte}$, der Paketgröße des AAL5, so empfängt der Event-Builder die erste Nachricht vom Trigger-Crate nach $65\,535/50 =$

buildMessage — Buffering of several crate events in one network message

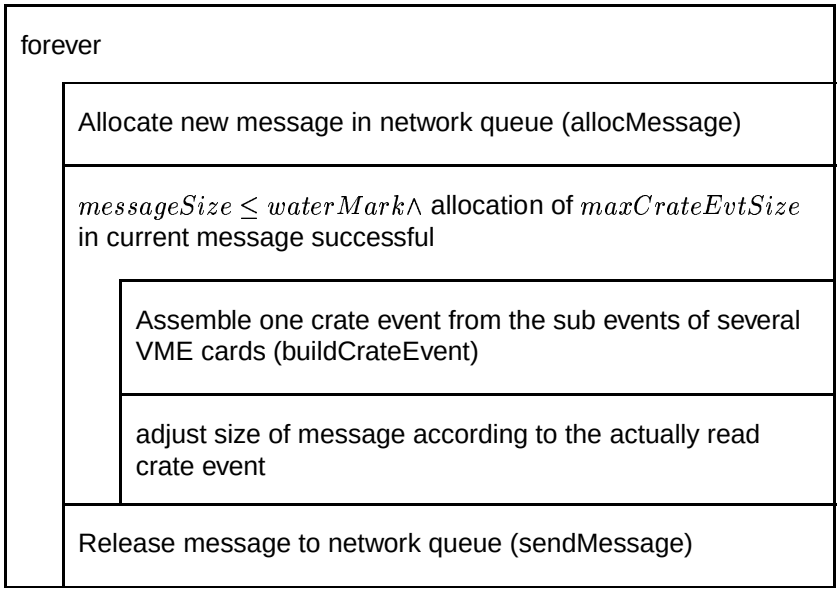


Abbildung 4.9.: Algorithmus zum Ansammeln von mehreren Crate-Events in einer Nachricht

$1\,310$ Ereignissen. Andererseits muss er aber für ein einziges RICH-Crate schon $1\,310 \times 40\,000\,\mathrm{Byte} \approx 50\,\mathrm{MByte}$ speichern. Da diese in einer Hälfte des Doppelpuffers Platz haben müssen, müsste der Event-Builder also für jedes einzelne Crate $100\,\mathrm{MByte}$ Platz reservieren, also bei 7 Crates $700\,\mathrm{MByte}$. Dies ist zwar durchaus machbar, wird aber bei weiter steigender Zahl der Crates bzw. noch kleineren oder größeren Crate-Events zunehmend problematisch.

Aus diesem Grund kann man über eine „Füllstandsmarke“ („Watermark“) die Nachrichtengröße für jedes einzelne Crate abhängig von der mittleren, erwarteten Crate-Event-Größe einstellen. Ist der angegebene Füllstand erreicht, so wird die Nachricht versendet, auch wenn sie das Netzwerkpaket noch nicht ausfüllt.

Abbildung 4.9 zeigt den letztendlich verwendeten Algorithmus. Zu bemerken ist, dass Kopieroperationen im Speicher vermieden werden, indem das Crate-Event-Building (Abb. 3.6 auf Seite 28) direkt im Speicher der Nachricht abläuft.

4.3.2. Segmentierung

Wie in Abschn. 4.1.4 dargestellt, können über Netzwerkverbindungen nur Pakete einer festgelegten, maximalen Größe transportiert werden. Diese SDU[5] (Service Data Unit) ist bei ATM unter Nutzung von AAL5 $64\,\mathrm{KByte}$ groß, in anderen Netzwerktechnologien $8\,\mathrm{KByte}$ oder auch nur $1\,500\,\mathrm{Byte}$.

[5] inklusive Paketkopf MTU (Maximum Transport Unit)

Um die Größe der Crate-Events nicht auf die SDU zu beschränken, führt man eine Protokollschicht namens SAR (Segmentation and Reassembly, Zerlegen und Wiederzusammensetzen) ein. Dieses Protokoll zerlegt eine Nachricht beim Absender in Segmente, von denen jedes kleiner als die SDU ist, und setzt diese auf der Empfängerseite wieder zur ursprünglichen Nachricht zusammen.

Die einfachste Implementation der SAR-Schicht besteht darin, auch hier wieder einen Datenkopf einzuführen. In diesem Kopf wird vermerkt, ob es sich um den Anfang, Mitte oder Ende einer Nachricht handelt. Unter Umständen wird im Segmentkopf auch jedes Segment durchnummeriert, um feststellen zu können, ob alle Segmente einer Nachricht angekommen sind.

Das Dazufügen des Kopfes ist in allen anderen Schritten der Datenaufnahme einfach, weil dort immer mehrere komplette Einheiten von Nutzdaten zu einer neuen Datenstruktur zusammengefasst werden, der Kopf also vor die neu entstehende Struktur gesetzt werden kann. Anders beim Segmentieren, hier wird eine schon existierende Nutzdatenstruktur in kleinere Einheiten unterteilt, die Köpfe müssen also eingefügt werden, was Kopieroperationen notwendig macht. Auf der Empfangsseite gilt die Überlegung symmetrisch, hier müssen die Köpfe aus den empfangenen Daten wieder entfernt werden.

Eine eindeutige Kennzeichnung jedes einzelnen Segments ist aber bei der HADES Datenaufnahme weder notwendig noch sinnvoll, da eine Neuübertragung nach dem Verlust eines Segments sowieso nicht vorgesehen ist. Unter diesen Umständen ist es ausreichend, das Ende einer Nachricht zuverlässig zu erkennen. Das darauffolgende Paket ist dann per Definition der Anfang der nächsten Nachricht. Die Vollständigkeit der Daten wird dadurch festgestellt, dass die Menge der empfangenen Daten mit der gesendeten übereinstimmt.

Wie kann das letzte Paket einer Nachricht aber zuverlässig erkannt werden? Die gängigste Programmierschnittstelle für Netzwerkkommunikation, genannt „socket“-Abstraktion, erlaubt es nicht, über die Nutzdaten hinaus irgendwelche Information vom Sender zum Empfänger weiterzugeben, so dass eine eindeutige Kennzeichnung des Nachrichtenende nicht explizit machbar ist. Wenn man allerdings betrachtet, zu welchen Paketlängen die Segmentierung einer großen Nachricht führt, dann erkennt man, dass alle Pakete genau die Größe der SDU haben, außer dem letzten. Eine Paketgröße kleiner als der SDU kann also als Kriterium für das Nachrichtenendes herangezogen werden.

In dem Spezialfall, dass die Länge der Nachricht genau ein Vielfaches der SDU ist, ist das letzte Paket allerdings exakt so lang wie die SDU, wird also nicht als Nachrichtenende erkannt. In diesem Fall wäre es das Einfachste, ein Paket der Länge 0 als Abschluss zu schicken. Dies ist allerdings in einigen Netzwerktechnologien, insbesondere im AAL5, nicht erlaubt.

Deshalb wird in diesem Fall die Wiederholung des Nachrichtenkopfes einfach um $4\,\mathrm{Byte}$ verlängert, so dass die Länge der gesamten Nachricht kein Vielfaches der SDU ist.

Zusätzlich muss noch der Fall betrachtet werden, dass genau ein Nachrichtenendepaket verlorengeht, so dass der Inhalt von zwei Nachrichten gemeinsam empfangen

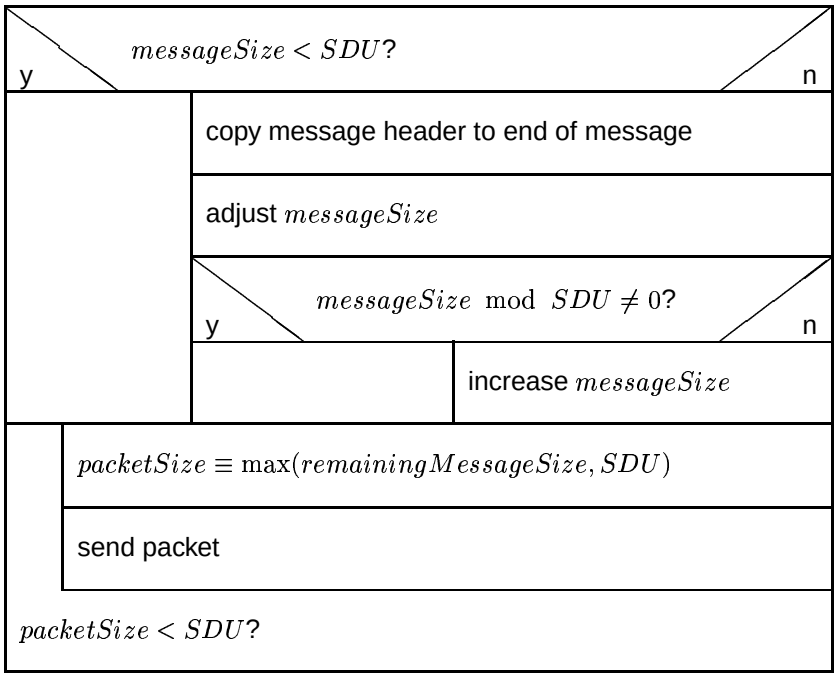
wird. Obwohl dies normalerweise an der nicht korrekten Nachrichtenlänge erkannt werden kann, ist ein Fall denkbar, bei dem von zwei aufeinanderfolgenden Nachrichten mehrere Pakte verlorengehen, so dass zum Schluss die Länge wieder stimmt. Es muss also überprüft werden, ob das Nachrichtenende zum Nachrichtenanfang passt.

Hierfür wird einfach der Nachrichtenkopf, der ja eine Sequenznummer enthält, am Ende der Nachricht nochmals versendet. Durch einfachen Vergleich der Sequenznummer am Ende der Nachricht mit der am Anfang kann der Empfänger feststellen, ob eine Vermischung von zwei Nachrichten stattgefunden hat.

Bei dieser Lösung müssen zum Empfangen einer Nachricht einfach nur die Pakete hintereinander im Speicher abgelegt werden und erst beim letzten Paket eine Aktion zum „Reassembly“ stattfinden. Diese Aktion besteht auch nur aus Überprüfen der Konsistenz der Nachricht, es müssen keinerlei Daten kopiert oder verändert werden.

Abbildung 4.10 auf der nächsten Seite zeigt den Algorithmus zur Aufteilung der Meldung in mehrere Pakete und das Wiederzusammensetzen beim Empfang. Der bei HADES angestrebte Fall, dass eine Nachricht vollständig in einem Netzwerkpaket Platz findet, wird vom Algorithmus zwar abgedeckt, aber zur Leistungsteigerung gesondert behandelt.

sendMessage — Send message to network, apply segmentation



receiveMessage — Receive message from network, apply reassembly

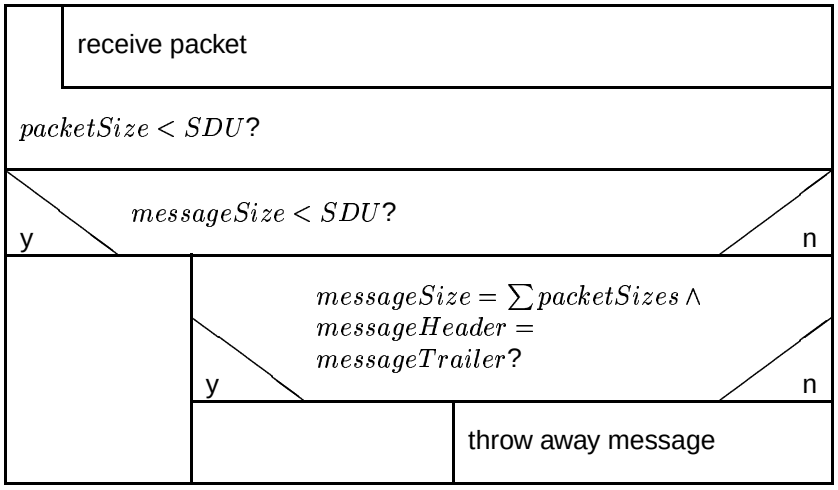


Abbildung 4.10.: Algorithmus zum Senden und Empfangen eine Netzwerknachricht. Je nach Größe wird die Nachricht segmentiert.

5. Event-Building

Die effiziente und fehlerfreie Zusammenführung des Datenstroms aus allen Teilsystemen und die Erzeugung sowie Speicherung von physikalisch eindeutigen Ereignissen ist die Aufgabe des Event-Building. Diese Prozesse finden auf einem handelsüblichen Rechner statt und sollen im Folgenden beschrieben werden.

5.1. Der Algorithmus

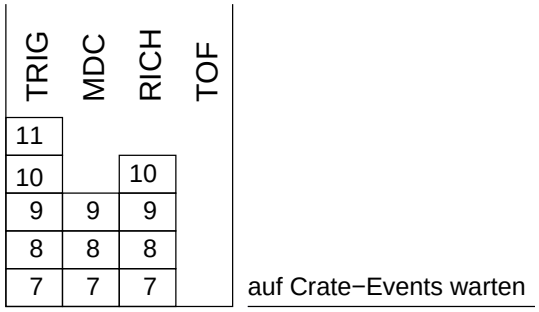
Auf der den Detektoren zugewandten Seite werden die Daten von der Sub-Event-Auslese zu Crate-Events formatiert und anschließend von der Datentransport-Software über das Netzwerk transportiert, wobei eventuelle Nebeneffekte des Netzwerktransports auf der Empfängerseite rückgängig gemacht werden.

Der Event-Builder findet also für jeden Crate-Event-Builder eine Warteschlange vor, in der die Crate-Events des jeweiligen Subsystems abgelegt sind. Die Crate-Events tragen dabei vom Triggersystem vergebene, eindeutige Nummern (Abschn. 2.3.1) und liegen in der Warteschlange chronologisch vor, sie dürfen sich also nicht „überholen“ (Abb. 5.1 auf der nächsten Seite).

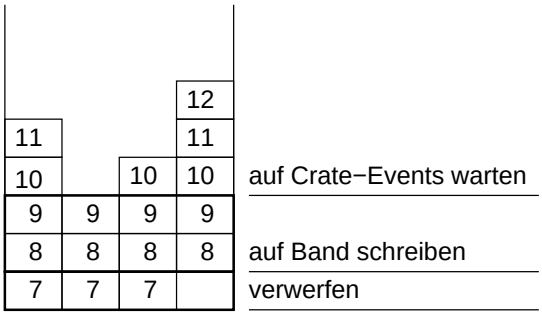
Grundsätzlich besteht das Event-Building darin, die gemäß der Triggernummer zusammengehörigen Crate-Events einzusammeln. Wenn alle gefunden wurden, wird daraus ein Event erzeugt und dieses gespeichert. Da Datenverlust während des Transports in der HADES-Datenaufnahme grundsätzlich erlaubt ist, muss der Algorithmus auch unvollständige Ereignisse erkennen und verwerfen können.

In den Abbildungen ist die Möglichkeit nicht beachtet, dass ein Subsystem für einen bestimmten Triggertyp gar keine Daten erzeugt. Damit wird die Entscheidung, wann ein Event vollständig ist oder nicht, vom Triggertyp abhängig (Abschn. 2.3.1). Besondere Bedeutung kommt dabei der ersten Warteschlange zu. Per Konvention beinhaltet das Crate-Event des ersten Crate-Event-Builders (in den Abbildungen mit „TRIG“ gekennzeichnet) ein Sub-Event, das den Triggertyp enthält. Im speziellen Fall von HADES bedeutet das, dass das erste Crate immer das Trigger-Crate mit der Matching-Unit sein muss.

Der Algorithmus selbst (Abb. 5.2 auf Seite 51) gestaltet sich dann einfach. Nachdem ein Crate-Event aus der ersten Warteschlange ausgelesen wurde ist das Trigger-Sub-Event und damit der aktuelle Triggertyp und die aktuelle Triggernummer bekannt. Damit ist festgelegt, von welchen Crate-Event-Buildern Daten zum aktuellen Ereignis ankommen müssen. Diese werden dann der Reihe nach ausgelesen. Sind in einer Warte-



(a) Solange Daten von einem Subsystem (TOF) noch ausstehen, werden die Daten der anderen Subsysteme zwischengespeichert und das Event-Building wartet.



(b) Haben zu einer Triggernummer (8 und 9) alle Subsysteme Daten geliefert, werden vollständige Events gebildet und abgespeichert. Daten mit niedrigerer Triggernummer (7) können nicht mehr ergänzt werden und werden verworfen.

Abbildung 5.1.: Die Crate-Events der einzelnen Subsysteme (TRIG, MDC, RICH, TOF) liegen am Event-Builder in getrennten Warteschlangen vor. Sie sind mit eindeutigen Nummern gekennzeichnet und stehen in chronologischer Reihenfolge in der Warteschlange.

buildEvent — Assemble one event from the crate events of several VME crates

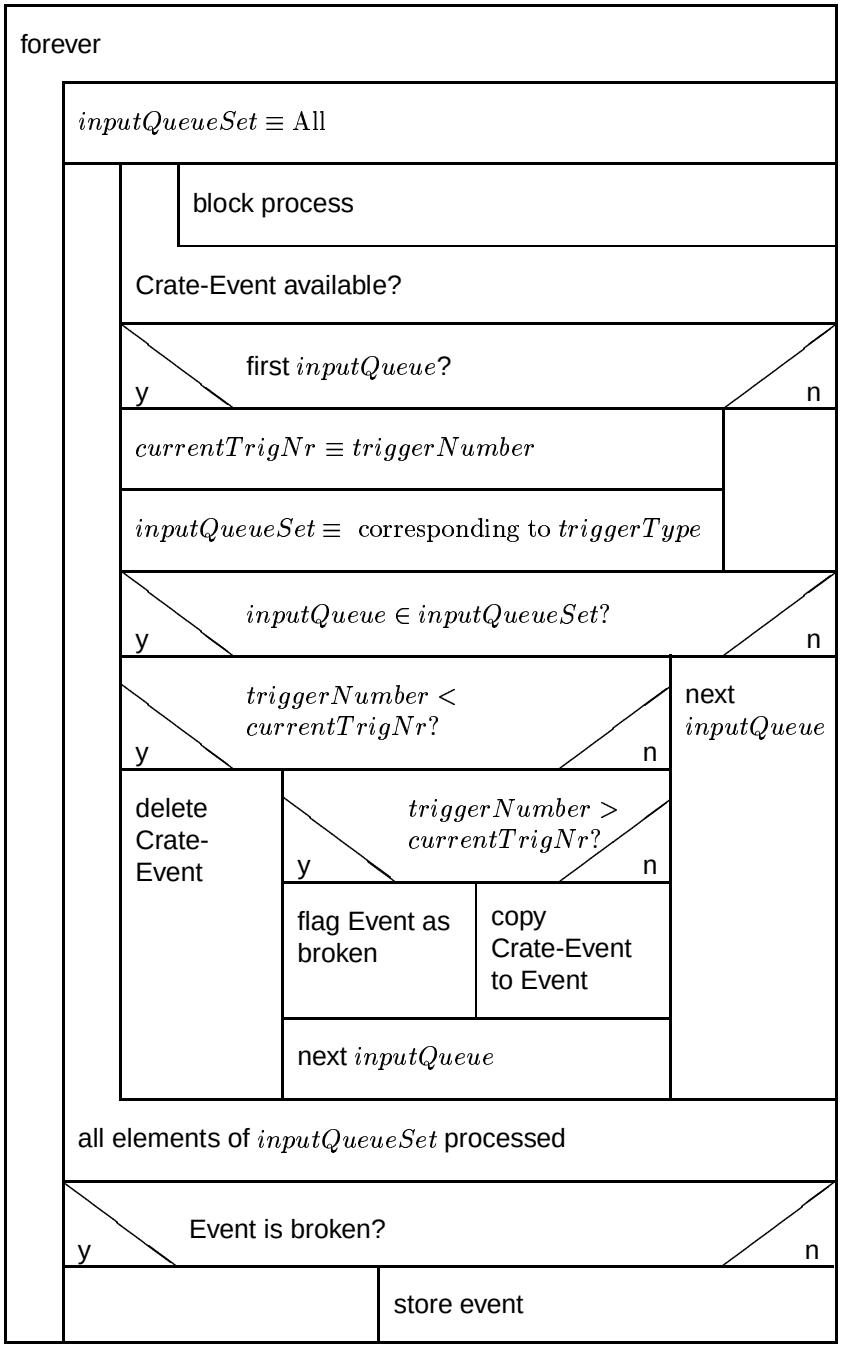


Abbildung 5.2.: Der Algorithmus zum Event-Building

schlange noch keine Daten vorhanden, so wird der Event-Builder bis zur Ankunft neuer Daten blockiert. Bei vorhandenen Daten wird überprüft, ob die Triggernummer mit der des Trigger-Sub-Events übereinstimmt. Wenn ja, wird das Crate-Event bis auf die Sub-Event-Ebene ausgepackt und die Sub-Events werden in das Event kopiert.

Wenn nicht, so gibt es zwei Möglichkeiten:

- Ist die Triggernummer größer als im Trigger-Sub-Event, so sind vom aktuellen Subsystem Daten verloren gegangen. Das Event kann nicht vollständig erzeugt werden, der Eventaufbau wird abgebrochen und das aktuelle Ereignis verworfen.

- Ist dagegen die Triggernummer des aktuellen Crate-Events kleiner als die Triggernummer des Trigger-Sub-Events, so sind vom Trigger-Subsystem Daten verloren gegangen oder von einem vorherigen Abbruch des Eventaufbaus wurden noch alte Crate-Events gefunden. In diesem Fall werden nur die Crate-Events in der aktuellen Warteschlange bis zur passenden Triggernummer verworfen, der Aufbau des Events allerdings fortgesetzt.

In der konkreten Implementation des Algorithmus sind folgende Besonderheiten zu beachten:

- Die Triggernummer ist z.Z. ein 32bit-Zähler, läuft also nach ca. 4 Milliarden Ereignissen über. Bei einer angenommenen LVL1 Triggerrate von $100\,\mathrm{kHz}$ wäre das nach ca. 12 Stunden ununterbrochener Datenaufnahme der Fall. Für das aktuelle Design des HADES-Experiments sind das noch akzeptable Werte, aber bei einer deutlichen Erhöhung der Triggerrate müsste dieser Zähler per Software auf 64bit erweitert werden.

- Die Sub-Events werden physikalisch im Speicher in das Event kopiert, wo sie dann als zusammenhängender Speicherbereich geschrieben werden können. Im Prinzip ließe sich diese Kopieroperation sparen, wenn man einen so genannten „Scattered Write“ einsetzt. Hierbei können im Speicher nicht kontinuierliche Daten mit einer Schreiboperation auf ein Gerät geschrieben werden.

  Die gemäß ISO-POSIX 1003.1b genormte Variante des „Scattered Write“ ist allerdings nur zusammen mit „Asynchronous IO“ einsetzbar, der wiederum nur auf wenigen Mikrocomputersystemen verfügbar ist. Die UNIX Variante writev/readv ist zwar weit verbreitet, aber nicht auf andere Systeme portabel. Zusätzlich hängt die wirkliche Leistungsfähigkeit des „Scattered Write“ von den Fähigkeiten der Hardware ab, eine solche Operation durchzuführen. Ansonsten muss das Verhalten vom Betriebssystem durch Kopieren nachgebildet werden, so dass nichts gewonnen ist. Darüber hinaus würde das Fehlen eines vollständig formatierten Events im Speicher den Online-Event-Server erheblich komplizieren.

5.2. Datenspeicherung

Selbst bei optimaler Wirkungsweise aller Triggerstufen erzeugt der Event-Builder eine Datenrate von $1.8\,\mathrm{MByte/s}$. Die Leistungsmessungen (Abschn. 4.2.4) zeigen, dass noch weitaus größere Datenmengen möglich sind. Unter der Annahme eines mittleren Wertes von $5\,\mathrm{MByte/s}$ führt dies zum Beispiel zur Akkumulation von $1\,\mathrm{GByte}$ Daten in $3\frac{1}{2}$ Minuten oder $18\,\mathrm{GByte}$ pro Stunde.

In einer Woche Strahlzeit fallen also, selbst wenn man nur 50 % Verfügbarkeit von Beschleuniger, Experiment etc. berechnet, immerhin noch $1.5\,\mathrm{TByte}$ an Daten an. Die Frage, wie und wohin diese Datenmengen gespeichert werden sollen, ist offensichtlich nicht mehr einfach zu beantworten.

5.2.1. Bänder

Beim Design der HADES-Datenaufnahme wurde vorgesehen, die Daten mit einem direkt am Event-Builder angeschlossenen Laufwerk auf Magnetband zu schreiben. Diese Methode ist auch die einzige bisher unterstützte neben dem Schreiben direkt auf Festplatte.

Mit einem DLT-8000 Laufwerk an einer LVDS-SCSI-Schnittstelle wurden dabei im Experiment Schreibraten von $5\,\mathrm{MByte/s}$ gemessen, bei entsprechender Optimierung sollten $10\,\mathrm{MByte/s}$ möglich sein. Die Leistungsdaten sprechen also durchaus für diese Lösung.

Problematisch ist also nicht das Schreiben der Daten, sondern vielmehr das spätere Zurverfügungstellen für die Analyse. Die im lokalen Bandlaufwerk geschriebenen Bänder sind zunächst nicht zugreifbar, sondern müssen erst auf Festplatte oder Roboter-Bandbibliotheken kopiert werden. Da die Datenaufnahme nahe der maximalen Schreibrate operiert, muss das Kopieren nahezu so lange dauern wie die Strahlzeit.

5.2.2. Online Speicherung

Mit dem Schreiben direkt auf Festplatte oder Roboter-Bandbibliothek könnte man also eine schnellere Verfügbarkeit der Daten erreichen.

Bei einem Datenvolumen von mehreren Duzend Terabyte pro Jahr scheint die ausschließliche Verwendung von Festplatten für die langfristige Speicherung auch heute noch unrealistisch.

Beim direkten Schreiben auf eine Roboter-Bandbibliothek sind in der Hauptsache Fragen der Bandbreite und der Zugriffszeiten zu klären. Aktuell gemessene Werte an der GSI über GigaBit-Ethernet liegen bei ca. $4\,\mathrm{MByte/s}$ wenn der Roboter exklusiv einem Nutzer zur Verfügung steht [54]. Das erlaubt es nicht mehr, die volle Leistung der HADES-Datenaufnahme auszunutzen. Eine Lösung könnte in einem Puffer-Konzept bestehen, das einige Terabyte Plattenplatz lokal am Event-Builder zur Verfügung stellt und diesen Puffer dann automatisch auf die Bandbibliothek entleert.

5.3. Online-Analyse

Die vollständigen „List-Mode-Daten“ werden auf Massenspeicher abgelegt, die zwar große Kapazität haben, aber erst nach einiger Verzögerung, z.B. durch das Kopieren der Bänder, zugreifbar werden.

Um schon während der Messung grobe Aussagen über die Qualität der aufgenommenen Daten und damit u.U. rechtzeitige Verbesserungen vornehmen zu können, müssen direkt aus dem aktuellen Strom Daten zur Verfügung gestellt werden, natürlich ohne die Datenaufnahme dabei zu verlangsamen. Dieser Dienst wird vom „Online-Event-Service“ geleistet, der das gerade aktuell fertigestellte Ereignis aus dem Event-Builder über das TCP/IP-Netzwerk zugreifbar macht. Das Format des Ereignisses wird dabei vollständig beibehalten, so dass die selben Programme zur Online- wie zur Offline-Analyse verwendet werden können. Der Dienst ist asynchron, der Event-Builder arbeitet also sofort weiter, ohne auf das Abholen des Ereignisses zu warten.

Der Einfachheit halber wurde dieser Dienst mit Hilfe der ONC-RPC Bibliothek [55] implementiert. Diese Bibliothek übernimmt alle Aufgaben der Netzwerk-Sitzungsverwaltung und Netzwerkkommunikation. Ein einfacher „Online-Event-Client“, der jeweils ein Event abholt und z.B. auf Platte speichert, ist in ca. 10 Programmzeilen geschrieben.

Die ONC-RPC-Bibliothek hat inhärent die „Multi-Client“-Fähigkeit, mehrere Analyse-Programme können sich also gleichzeitig mit dem Event-Builder verbinden und Daten abholen. Da der Event-Builder aber mit jedem einzelnen dieser Programme kommunizieren muss, bestand die Befürchtung, dass dies zu einer hohen CPU-Last im Event-Builder führen könnte.

Deshalb wurde an der GSI ein „Remote-Event-Server“ erstellt [56], der auf einem anderen Rechner als dem Event-Builder gestartet werden kann, eine Verbindung zum Online-Event-Service nutzt und selbst wieder mehrere Analyseprogramme bedienen kann. Dieser „Remote-Event-Server“ stellt die Daten nicht per RPC im Originalformat zur Verfügung, sondern per „socket“-Schnittstelle. Zur Nutzung dieser Schnittstelle durch das Analysepaket „ROOT“ wurde ebenfalls eine Bibliothek geschrieben [57].

6. Einsatz in Messungen

Mit der bisher beschriebenen Datenaufnahme wurden in den letzten Jahren mehrere Experimente am Schwerionensynchrotron der GSI in Darmstadt durchgeführt. Die Spanne reicht dabei von Messungen in Jahre 1997, die unter Verwendung von Detektor- und Ausleseprototypen durchgeführt wurden, bis zum Experiment Ende 2000, das aus Sicht der Datenaufnahme einen nahezu vollständigen HADES-Aufbau verwendete. In den folgenden Abschnitten sollen exemplarisch die Erfahrungen und einige ausgewählte Ergebnisse dieser Experimente vorgestellt werden.

6.1. Einsatz des Datenaufnahme-Prototyps

Wesentliche Komponenten der HADES Datenaufnahme wurden in einer Strahlzeit im Sommer 1997 eingesetzt und das grundsätzliche Konzept auf seine Brauchbarkeit überprüft.

Aufbau und Ergebnisse dieses Experiments wurden ausführlich in [23] beschrieben, hier seien nur die Aspekte, die speziell die Datenaufnahme betreffen, herausgegriffen.

6.1.1. Zielsetzung

Die Ziele des Experiments im Bezug auf die Datenaufnahme waren:

- Überprüfung der Brauchbarkeit der Softwarestruktur (Abschn. 2.4) unter realen Bedingungen im Experiment am Strahl. Inwieweit ist die Software vom Experimentator nutz- und bedienbar, welchen Aufwand bedeutet es, Änderungen an der Auslese und Integration von Spezialfällen durchzuführen? Können mehrere Leute parallel mit Teilsystemen testen und können diese Teilsysteme einfach wieder in ein Gesamtsystem überführt werden? Wie zuverlässig arbeitet das System?

- Nutzung von handelsüblichen Computernetzwerken für den Transport von Experimentdaten (Kapitel 4). Wie wirken sich die besonderen Eigenschaften (Paketorientierung, Latenzzeiten) aus, sind die vorgeschlagenen Algorithmen zum Datentransport brauchbar?

- Einsatz von Standard-Rechnern als Event-Builder (Kapitel 5). Sind Hard- und Systemsoftware für den Zweck geeignet? Wie verhält sich das System unter realen

Anforderungen bei gleichzeitigem Datenempfang, Event-Building, Datenschreiben und Benutzerinteraktion. Ist der vorgeschlagene Algorithmus zum asynchronen Event-Building unter Beachtung von möglichem Datenverlust geeignet? Wie stabil ist dieser Algorithmus im Fehlerfall?

6.1.2. Aufbau

Von allen HADES Detektoren waren Prototypen zum Experiment vorhanden, die Draufsicht auf die Anordnung zeigt Abb. 6.1 auf der nächsten Seite. Je nach Stand der Elektronik-Entwicklung konnten von den einzelnen Detektoren unterschiedlich viele Kanäle ausgelesen werden, so z.B. der RICH-Prototyp und die Flugzeitwand vollständig, von der MDC und dem PreShower jeweils ein schmaler Streifen in zwei Ebenen. Eine Übersicht gibt Tab. 6.1.

Detektor	Kanäle			
	X-Richtung	Y-Richtung	Ebenen	Gesamt
RICH	64	64	1	4096
MDC	1	16	2	32
TOF	2	32	1	64
PreShower	32	5	2	320

Tabelle 6.1.: Zahl der ausgelesenen Kanäle an den einzelnen Detektoren bei der Strahlzeit im Sommer 1997

Zur Erzeugung des LVL1-Triggers diente ein Multiplizitätszähler aus Szintillatoren, die über Photomultiplier ausgelesen wurden.

Ein $^{238}\mathrm{U}$-Strahl ($1\,\mathrm{AGeV}$, bis zu $10^6\,\mathrm{Ionen/s}$) traf auf ein festes $^{208}\mathrm{Pb}$-Target mit einer Wechselwirkungsdicke von 10%. Untersucht wurde das Verhalten der Zähler bei hohem Untergrund an geladenen Teilchen (Multiplizität ≈ 200) sowie das Detektorsignal für Leptonen, die hauptsächlich durch Paarkonversion im Target erzeugt wurden [58].

Zum Zeitpunkt der Strahlzeit war für die HADES Datenaufnahme noch keine Hardware angeschafft worden. Um trotzdem schon Antworten auf die obigen Fragen sammeln zu können, wurde ein Software-Prototyp erstellt, der schon über alle entscheidenden Eigenschaften der endgültigen Datenaufnahme verfügte. Dieses Prototypsystem wurde mit vorhandener Hardware und Systemsoftware implementiert und in Betrieb genommen. (Abb. 6.2 auf der nächsten Seite).

Der Prototyp des Datenaufnahmesystems bestand aus den folgenden Komponenten:

Trigger-Verteilung Die Trigger-Verteilung wurde mit den ersten HADES-DTU-Karten realisiert. Diese wurden in einem sehr einfachen Modus betrieben, der keine Steuerung der Detektorelektronik beinhaltete, sondern nur mit den Ausleseprogrammen kommunizierte. Diese übernahmen dann die gesamte Ablaufsteuerung.

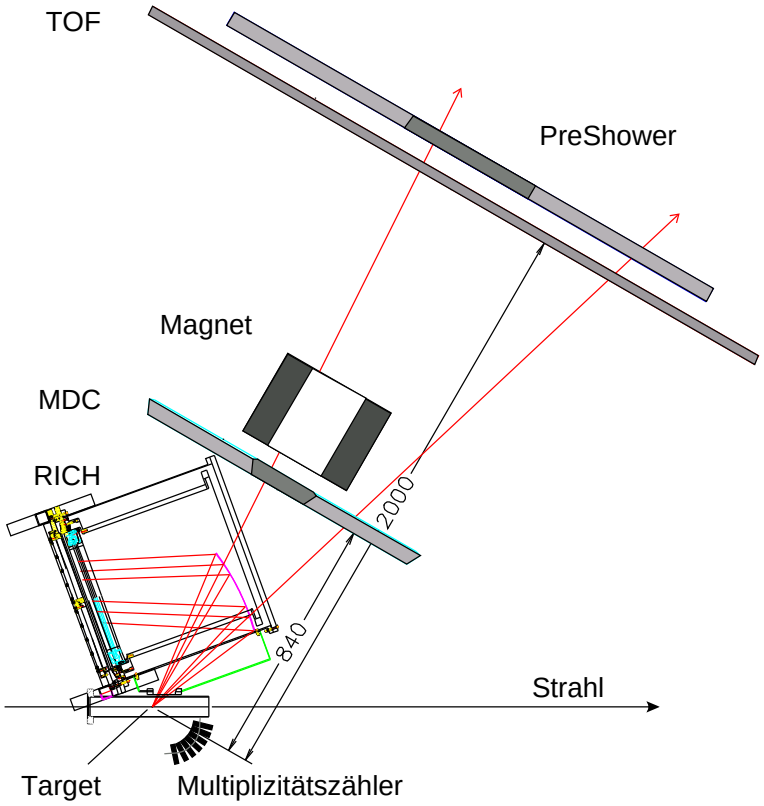


Abbildung 6.1.: Aufbau der an der Teststrahlzeit beteiligten Detektoren (Draufsicht)

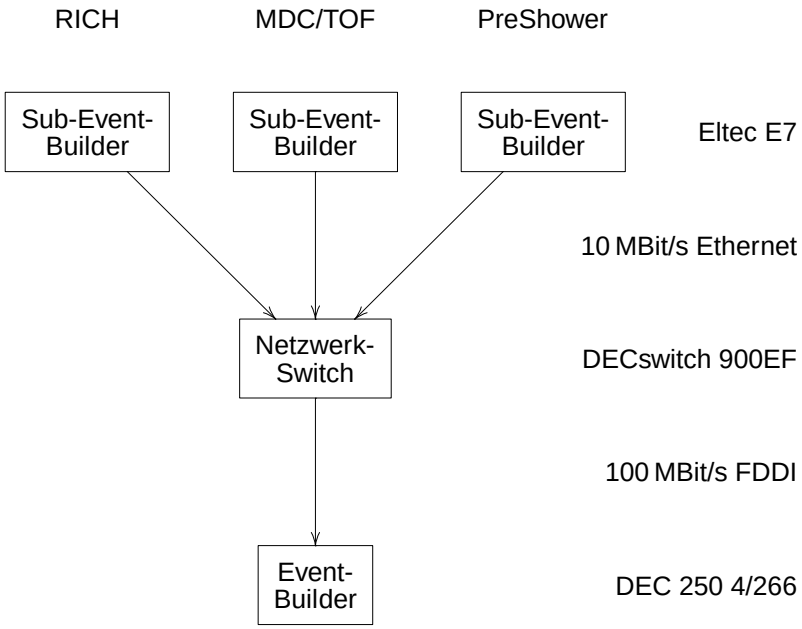


Abbildung 6.2.: Hardwarekomponenten, die in der Teststrahlzeit im Sommer 1997 an der GSI verwendet wurden.

Frontend-Elektronik Als Frontend-Elektronik kam zum Großteil fertige CAMAC-Elektronik zum Einsatz. Bei PreShower und MDC wurden auch schon frühe Prototypen der sich in der Entwicklung befindenden Detektor-Elektronik verwendet. Wegen der geringen Anzahl dieser Prototypen konnte von diesen Zählern nur ein Teil der Detektorfläche ausgelesen werden (vgl. Tab. 6.1 auf Seite 56).

Crate-Event-Builder Hier wurden 3 VMEbus-Rechner vom Typ „Eltec-E7“ mit Motorola 68040 CPU unter dem Betriebssystem „LynxOS Version 2.3“ verwendet. TOF und MDC mussten sich dabei ein Crate teilen, da nicht genügend DTU-Karten verfügbar waren.

Datentransport Die Crate-Event-Builder verfügten über einer 10BASE2 Ethernet Schnittstelle (Bandbreite $10\,\mathrm{MBit/s}$). Als Vermittlungsknoten kam ein „DECSwitch 9000“ zum Einsatz, der die Ethernet-Stränge mit einer FDDI Glasfaserleitung ($100\,\mathrm{MBit/s}$) verschaltete, die direkt mit dem Event-Builder verbunden war.

Event-Builder Eine „DEC Alphastation 250/4 266MHz“ mit „DLT1000“ Bandlaufwerk unter „DEC-UNIX 3.2“ bildete den Event-Builder.

Die wichtigsten Unterschiede zum geplanten HADES System waren:

Einfaches Triggersystem Die Elektronik erlaubte keine Nutzung mehrer Triggerstufen oder der Derandomisierung. Die Auslese wurde also vollständig synchron durchgeführt.

Geringere Datenraten Die 10BASE2-Schnittstellen der Eltec E7 Computer in Verbindung mit dem IP-Protokollstapel von „LynxOS 2.3“ erlaubte maximale Datenraten von $420\,\mathrm{KByte/s}$ pro Crate-Event-Builder. Bei HADES sind pro Crate-Event-Builder bis zu $15\,\mathrm{MByte/s}$ möglich.

Langsames Bandlaufwerk Das DLT1000-Bandlaufwerk, das am SCSI-II-Bus der Alphastation betrieben wurde, konnte maximal $1.7\,\mathrm{MByte/s}$ auf Band schreiben.

6.1.3. Ergebnisse

Eignung der Softwarestruktur

Während der Strahlzeit wurden ca. $7\,\mathrm{GByte}$ Experimentdaten auf Band geschrieben. Bei einer typischen Eventgröße von $1.2\,\mathrm{KByte/s}$ waren das über $7\,000\,000$ Ereignisse. Die physikalischen Fragestellungen zum Teilchenuntergrund, zur Detektorstabilität, zur Leptonensignatur in den Detektoren und zum Startzählerkonzept konnten mit diesen Daten beantwortet werden.

Alle Komponenten des Experimentaufbaus waren in der Prototypphase, entsprechend häufig mussten Tests durchgeführt, Spezialmessungen vorgenommen oder der Ausfall einer Komponente hingenommen werden. Das Datenaufnahmesystem erwies sich dabei als sehr flexibel. Als besonders wertvoll erwies sich die Möglichkeit, eine

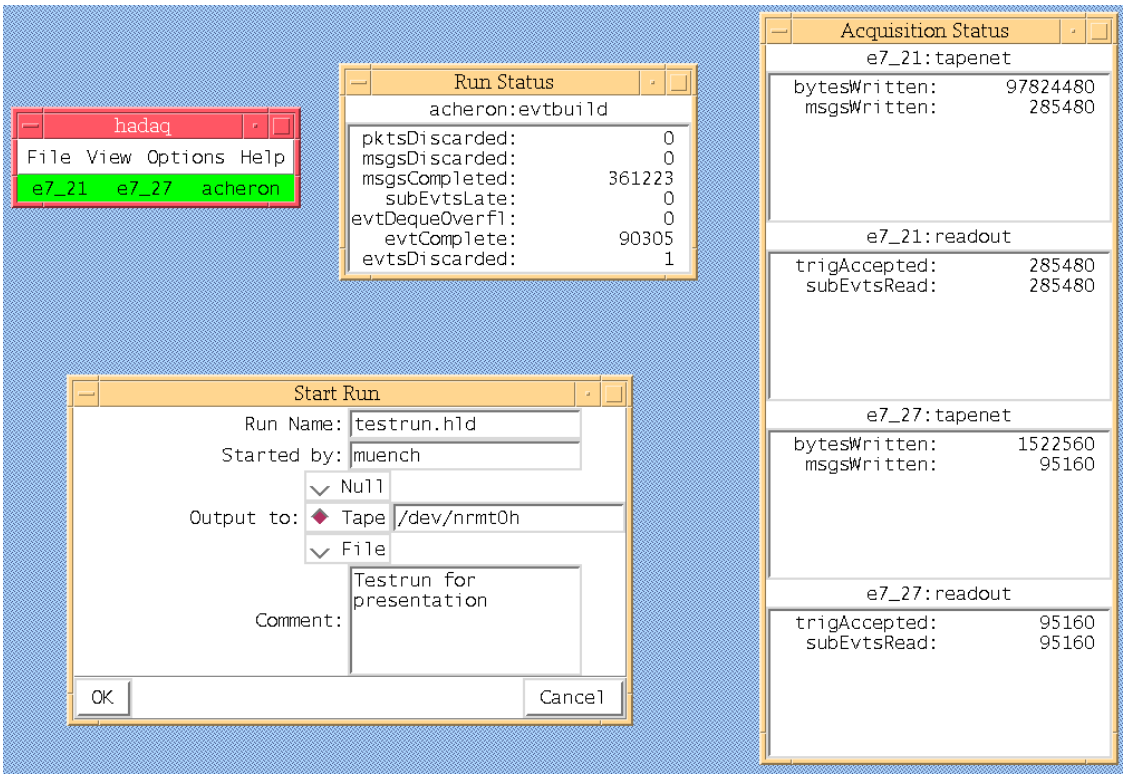


Abbildung 6.3.: Für den Prototyp der Datenaufnahme entwickelte Benutzeroberfläche: Sie erlaubte Steuerung und Überwachung der wichtigsten Operationen von einem zentralen Bildschirm aus. Im Fenster „hadaq“ wird der Status der beteiligten Rechner (e7_21, e7_27, acheron) farblich angezeigt. „Run Status“ und „Acquisition Status“ zeigen Statistiken zur geschriebenen Datei bzw. zur Frontend-Auslese. Das „Start Run“-Fenster zeigt das Formular zum Öffnen einer neuen Datei.

vollständige Datenaufnahme auf einer VMEbus-CPU laufen zu lassen. Damit waren völlig unabhängige Tests aller Detektorgruppen möglich. Es wurde gleichzeitig offensichtlich, dass die Verkabelung von Trigger-Bus, Startdetektorsignalen etc. einen ungleich höheren Aufwand beim Wechsel vom Einzelplatz- zum Gesamtsystem verursachte.

Die Programmierung und Änderung von Ausleseroutinen wurde von den Detektorgruppen selbst durchgeführt. Hierzu war nur die Kenntnis der Programmiersprache „C“ und natürlich der auszulesenden Hardware notwendig. Dies erlaubte den Gruppen in der Folge auch, die Datenaufnahme für lokale Tests z.B. in Krakau, München und der GSI zu nutzen.

Zur Steuerung der Datenaufnahme wurde noch während der Strahlzeit eine einfache, graphische Benutzeroberfläche auf der Basis von „Tcl/Tk“ [59] erstellt (Abb. 6.3). Diese wurde, wider alle Erwartung, in der Folge immer weiter verwendet und erst im

Jahr 2000 ein Nachfolger in Angriff genommen. Dies unterstreicht die Eignung der Softwarestruktur.

Datentransport über Rechnernetz

Hier zeigte sich die Korrektheit der Algorithmen zum Versenden der Experimentdaten über ein paketorientiertes Netzwerk. Wegen der geringen Datenproduktionsrate der E7 Computer wurde allerdings keine der beteiligten Komponenten an der Grenze der Leistungsfähigkeit betrieben, so dass keine Aussage über die Zuverlässigkeit in Extremfällen gemacht werden konnte. Diese wurden dann durch Messungen im Labor (Abschn. 4.2.4) nachgeholt.

Leistung und Stabilität des Event-Buildings

Das vermutlich wichtigste Ergebnis für den Fortgang des Projekts war die große Zuverlässigkeit des Event-Buildings. Der Datenverlust aufgrund verlorener Pakete war während der gesamten Strahlzeit $< 1\%$. Auch nach provoziertem, massivem Datenverlust bei einem Crate-Event-Builder (Abziehen und Wiederanstecken des Netzwerkkabels) synchronisierte der Algorithmus wieder schnell und zuverlässig.

Insbesondere wurde in keiner der aufgenommenen Dateien eine Vermischung von Sub-Events (Event-Mixing) beobachtet, sondern es wurden immer die erwarteten Korrelationen zwischen den einzelnen Detektorsignalen gefunden. Dieses Ergebnis wurde unter vollständig realistischen Bedingungen erzielt, da der Datentransport und das Event-Building schon asynchron abliefen.

Als Beispiel einer Korrelation zwischen zwei Detektorsystemen sei hier der Vergleich der gemessenen Flugzeiten (TOF) mit den vom RICH aufgenommenen Cherenkov-Ringen gezeigt (Abb. 6.4 auf der nächsten Seite). Die Ereignisse, in denen der RICH einen Ring gesehen hat, liegen im schnellen Teil des Flugzeitspektrums ($t < 13\,\mathrm{ns}$). Die Ereignisse mit größeren Flugzeiten stellen fehlidentifizierte Ringe im RICH oder nicht aufgelöste Mehrfachtreffer im TOF dar.

Zusammenfassung

Schon der Prototyp der Datenaufnahme erwies sich als vollständig nutzbares Datenaufnahmesystem. Der Transport von Experimentdaten über ein Rechnernetzwerk verhielt sich wie erwartet. Der Algorithmus zum Event-Building funktionierte effizient und zuverlässig. Insgesamt konnte das Ergebnis als Beweis für die Tragfähigkeit des Konzepts der geplanten HADES Datenaufnahme gewertet werden.

6.2. Messungen mit dem HADES Gesamtsystem

Nachdem die Messungen mit dem Datenaufnahmeprototypen erfolgreich waren, wurde im Laufe der Jahre 1997 und 1998 der Datentransport und das Event-Building für das HADES Gesamtsystem implementiert und im Herbst 1998 abgeschlossen. Seit dem

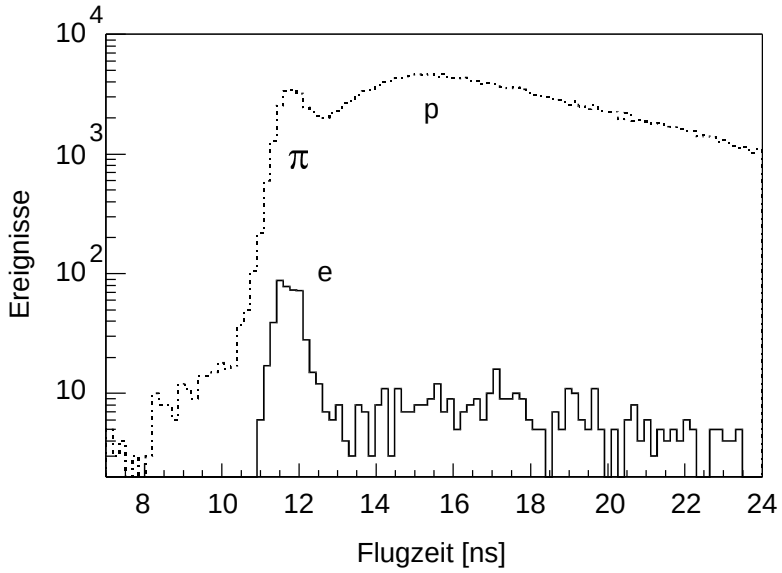


Abbildung 6.4.: Reaktion $^{238}\mathrm{U} + ^{208}\,\mathrm{Pb}, 1\,\mathrm{AGeV}$: Flugzeitspektrum aller gemessenen Teilchen (gestrichelt) im Vergleich zum Spektrum der Teilchen, zu denen im RICH ein Cherenkovring erkannt wurde. Es werden sowohl die schweren Teilchen unterdrückt (Flugzeit $> 13\,\mathrm{ns}$), als auch die leichten, aber immer noch langsamen Pionen. Nur die schnellsten Teilchen, Elektronen und Positronen, werden vom RICH identifiziert.

wurden immer wieder Experimente am Strahl durchgeführt, wobei nach und nach der Detektoraufbau und die Ausleseelektronik komplettiert wurden [60]. Dabei waren keine konzeptionellen Änderungen mehr notwendig. Aus praktischen Gründen ergaben sich einige Änderungen, wie z.B. der Wechsel von „DEC-UNIX“ auf Linux beim Event-Builder. Wie erwartet konnten diese jedoch ohne große Schwierigkeiten durchgeführt werden.

Im November 2000 war dann ein, aus Sicht der Datenaufnahme, vollständiges HADES zum ersten Mal im Einsatz und bis zum gegenwärtigen Zeitpunkt wurden noch drei weitere Strahlzeiten mit jeweils erweitertem Detektorsystem und verbesserter Triggerelektronik durchgeführt. Im Folgenden sollen einige Ergebnisse der Strahlzeit vom November 2000 beispielhaft dargestellt werden.

6.2.1. Zielsetzung

Im Experiment im November 2000 sollten nicht mehr einzelne Detektoreigenschaften oder Fragen zum grundsätzlichen Betrieb geklärt werden, sondern unter Einsatz aller zur Verfügung stehenden Systeme Daten genommen werden, um erste Schritte in Richtung des geplanten Physik-Programms zu unternehmen.

Dazu wurden folgende Ziele verfolgt:

- Korrelierte Messungen aller Detektoren ohne und mit Magnetfeld, um einerseits Eichmethoden zu testen, andererseits aber auch die für die Analyse der Daten notwendigen Algorithmen zur Ringerkennung im RICH und zur Spurrekonstruktion der Driftkammern durchzuführen.

- Erreichen einer möglichst hohen Statistik, um auch schon die Erstellung und Auswertung von Spektren zur Teilchenmasse zu erlauben.

- Messung der Effizienz für Photonennachweis im RICH mit einer Methode, die es erlaubt, die Gesamteffizienz des Detektors im endgültigen Aufbau, sozusagen „in vivo“ zu messen.

- Beginn der Integration des LVL2-Triggers in das Gesamtsystem. Zur Strahlzeit standen erste Elektronikmodule des LVL2-Triggers zur Verfügung. Diese sollten in die Steuerung und den Datenstrom des Experiments integriert werden, um Aussagen über Funktion und Effizienz machen zu können.

6.2.2. Experimentaufbau

Zum Experiment waren praktisch alle Detektoren installiert, abgesehen von der dritten (hier war nur eine Kammer vorhanden) und vierten Driftkammerebene. Auch der Magnet war betriebsbereit.

Es stand jedoch nicht die gesamte, notwendige Ausleseelektronik zur Verfügung, so dass insgesamt folgende Detektoren ausgelesen wurden:

- Start- und Veto-Diamant-Zähler
- 5 Sektoren des RICH
- jeweils 5 Sektoren der ersten und zweiten Driftkammerebene
- 6 Sektoren der TOF-Wand und 4 Sektoren des Tofino-Detektors
- 4 Sektoren des PreShower-Detektors
- die Matching-Unit des LVL2-Triggers

Diese Detektoren wurden mit insgesamt sieben Crate-Event-Buildern ausgelesen, betrieben von den schon im Vorfeld benutzten „CES RIO8062“ VMEbus-CPUs. Als ATM-Switch kam der auch schon vorher verwendete „Fore ASX200WG“ zum Einsatz, der Event-Builder war ein $2 \times 400\,\mathrm{MHz}$ Pentium II Doppelprozessorsystem unter Linux. Diese Maschine war erst kurz vorher als Ersatz für die Digital-Alphastation eingebaut worden.

Aus Sicht der Datenaufnahme war besonders interessant, dass zum ersten Mal der volle Ausbau an Crate-Event-Buildern erreicht wurde, so dass der Datentransport und das Event-Building mit der endgültigen Komplexität betrieben wurden.

Zum LVL2-Trigger waren Bildverarbeitungseinheiten für einen RICH-Sektor und die Matching-Unit vorhanden. Diese bildeten ein eigenes Sub-Event, so dass die Information des LVL2-Triggers als Teil des List-Mode-Datenstroms ausgelesen und gespeichert wurde. Andererseits wurde der LVL2-Trigger nicht zur Datenreduktion genutzt. Damit waren, trotz relativ kleiner LVL1-Triggerrate, die Datenvolumen im Bereich der Designwerte für den Endausbau.

6.2.3. Ergebnisse

Im Bezug auf die Datenaufnahme zeigten sich die wichtigsten Ergebnisse schon während der Strahlzeit. Die experimentellen Daten befinden sich noch in der detaillierten Auswertung, so dass hier nur ein paar qualitative Beispiele gegeben werden sollen.

Datenaufnahme

Wie bereits erwähnt, entsprach das Experiment bei Datentransport und Event-Building dem Vollausbau. Wegen Einschränkungen der Ausleseelektronik musste die LVL1-Triggerrate auf maximal $2\,\mathrm{kHz}$ durch Einführen einer konstanten LVL1-Totzeit von $500\,\mu\mathrm{s}$ begrenzt werden. Da aber keinerlei Ratenreduktion stattfand, war die Rate an der LVL2-Pipe gleich dieser LVL1-Rate und damit noch über dem Designwert von $10^3\,\mathrm{Hz}$ ohne bzw. $10^2\,\mathrm{Hz}$ mit LVL3 Trigger (Abschn. 1.5.1).

Einen Überblick über typischen Größen der Dateneinheiten während eines Messlaufs gibt Tabelle 6.2 auf der nächsten Seite. Im Gegensatz zu diesen Längen, bei denen die nicht angesprochenen Kanäle unterdrückt waren, lagen die Crate-Event-Längen für

	Länge [Byte]	
Crate	Crate-Event	Event
RICH0	333.30	3 043.22
RICH1	303.92	
RICH2	335.33	
MDC	1056.74	
SHOWER	496.84	
TOF	407.42	
TRIG	77.63	

Tabelle 6.2.: Mittlere Länge der Crate-Events und Events in einem normalen Messlauf für Kohlenstoff - Kohlenstoff - Kollisionen bei $1.5\,\mathrm{AGeV}$.

einen Kalibrationslauf ohne Schwellwerte für den RICH bei $38\,405\,\mathrm{Byte}$ und für den PreShower bei $24\,874\,\mathrm{Byte}$.

Es wurden Daten sowohl auf Festplatte als auch auf Bänder geschrieben. Die Schreibrate auf Platte war dabei auf ca. $2\,\mathrm{MByte/s}$ begrenzt. Oberhalb dieser Rate führte das Linux-Speicher- und -Filesystemmanagement dazu, dass ein Großteil der CPU-Leistung für das Betriebssystem verwendet wurde. Hier sind noch weitere Untersuchungen notwendig, inwieweit bei Linux, so wie bei UNIX Systemen, ein Schreiben auf Festplatte unter Umgehung des Speichermanagements möglich ist.

Beim Schreiben auf das DLT8000 Bandlaufwerk, das über LVDS-SCSI angeschlossen war, wurden ohne weitere Optimierung $5\,\mathrm{MByte/s}$ erreicht. Auch hier bleibt zu untersuchen, inwieweit diese Rate noch erhöht werden kann. Besonders interessant war die CPU-Last, die bei diesem $5\,\mathrm{MByte/s}$ Datenempfang, Event-Building und Schreiben erzeugt wurde. Auf dem verwendeten System lag die Auslastung bei ca. 2.5% pro CPU, so dass von der CPU-Leistung her noch viel Spielraum für eine Ratenerhöhung bleibt.

Beim Vollausbau zeigte sich aber auch, dass die seit Sommer 1997 immer weiter verwendete Steuerung und Benutzeroberfläche an ihre Grenzen stößt, sowohl was die Geschwindigkeit als auch was die Übersichtlichkeit der Bildschirmdarstellung angeht. Eine neue Lösung hierzu befindet sich in der Entwicklung [26].

Effizienzkalibration am RICH

Neben den gemeinsamen Messungen aller Detektoren war ein Teil des Strahlzeitprogramms im November 2000 auch einer neuen Methode zur Effizienzkalibration des RICH-Detektors gewidmet. Hierbei wird durch Strahlteilchen in zwei Festkörperradiatoren Cherenkovlicht erzeugt. Die optischen Eigenschaften (Brechungsindex, Transmission, Dispersion) dieser Radiatoren und die Geschwindigkeit der Strahlionen sind gut bekannt, die erzeugte Lichtmenge damit berechenbar. Die Radiatoren sind nahe der HADES Targetposition eingebaut, so dass der Lichtweg ähnlich dem der durch Leptonen im Gasradiator erzeugten Photonen bei Kollisionsexperimenten ist. Dadurch lassen sich von bekannten Ausgangsbedingungen aus die Leistungswerte aller Komponenten im

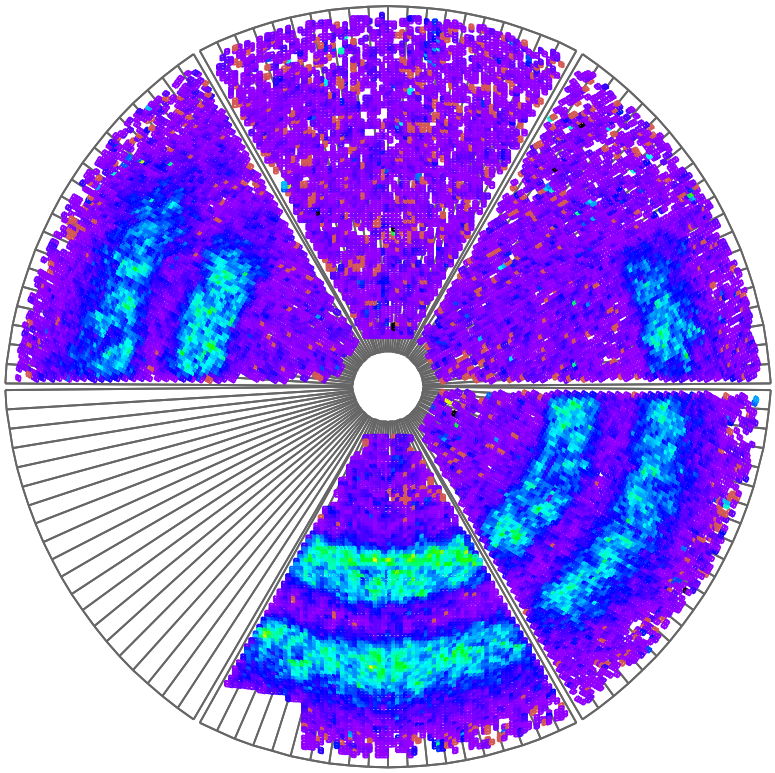


Abbildung 6.5.: Ca. 600 überlagerte Cherenkov-Ringe von Kohlenstoff-Ionen ($E = 600\,\mathrm{AMeV}, \beta = 0.794$) die einen MgF_2 bzw. SiO_2 Radiator durchdringen: Ein RICH-Sektor war nicht mit einem Spiegel versehen, die oberen Sektoren durch das Strahlrohr abgeschattet.

RICH differentiell (aufgelöst nach Wellenlänge) und integral bestimmen. Eine detaillierte Beschreibung findet sich in [61].

Abbildung 6.5 zeigt die Photonenverteilung für 500 akkumulierte Ereignisse mit einem $600\,\mathrm{AMeV}$ C-Strahl, wobei jeweils einer der Ringe mit einem der Radiatoren korrespondiert. In den Einzelereignissen waren die Photonen gut voneinander getrennt und damit abzählbar.

Die vorläufige Analyse der Daten ergab für zwei Sektoren des HADES-RICH im normalen Einsatz eine Kennzahl $N_0 = 90$ bzw. $N_0 = 105$. Diese Werte liegen nur leicht unterhalb der $N_0 = 110$, die für optimale Bedingungen aus einer Simulation mit HGEANT erwartet wurden.

Für die Datenaufnahme war dieses Eichexperiment interessant, weil nur der RICH und damit nur ein Teil (3 von 7) der Crate-Event-Builder betrieben wurde. Diese lieferten allerdings, wegen der hohen Photonenmultiplizität, sehr viele Daten. Die durchschnittliche Crate-Event-Größe in einem solchen Messlauf lag bei $1\,816\,\mathrm{Byte}$ im Gegen-

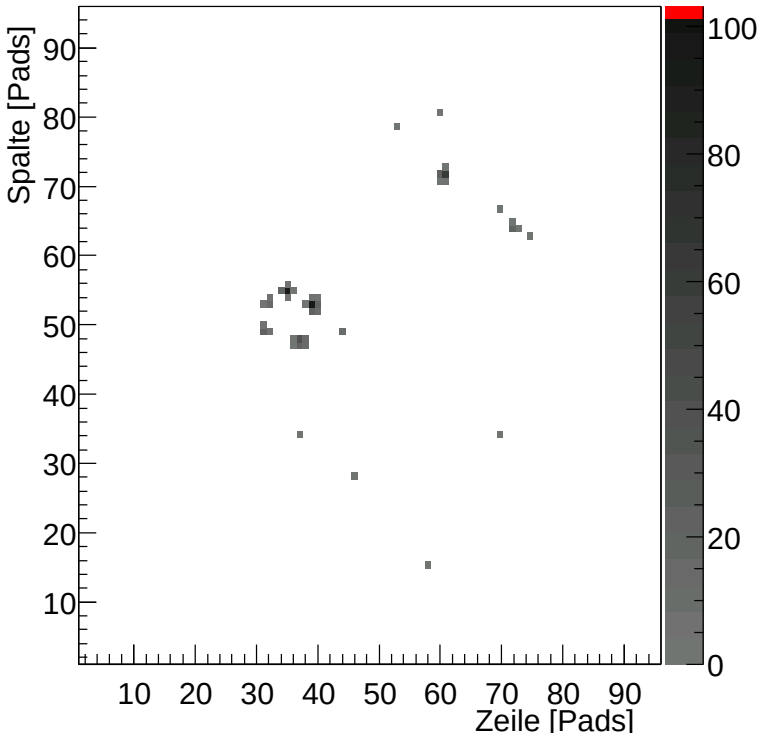


Abbildung 6.6.: Beispiel für ein Leptonenereignis in HADES-RICH

satz zu ca. $300\,\mathrm{Byte}$ bei normalen Kollisionsereignissen. Es zeigte sich, dass trotzdem die volle Schreibrate von $5\,\mathrm{MByte/s}$ auf Band erreicht wurde. Hierbei bewährte sich die großzügige Auslegung und einfache Bandbreitensteuerung der Datenpfade von jedem einzelnen Crate-Event-Builder zum Event-Builder.

Online-Ringerkennung

Ein weiterer Einzelaspekt der Strahlzeit war die erstmalige erfolgreiche Inbetriebnahme von Teilen des LVL2-Triggers und ihre Integration in die Datenaufnahme. Dabei wurde ein Sektor des RICH-Detektors mit einem Elektronikmodul zur Ringerkennung [27] ausgerüstet. Dieses gab die Daten (Mittelpunkte) der erkannten Ringe weiter an die Matching-Unit. Die Matching-Unit steuerte wiederum den LVL2-Trigger-Bus. Für diesen ersten Versuch wurden allerdings, unabhängig vom Ergebnis der Ringsuche, alle LVL2-Triggerentscheidungen positiv gefällt, so dass keine Datenreduktion stattfand.

Die Matching-Unit wurde durch einen eigenen Crate-Event-Builder ausgelesen, so dass die Entscheidungsgrundlage des LVL2-Triggers als Sub-Event in den Daten erschien. Da alle Ereignisse akzeptiert wurden, konnte so nachträglich die Arbeit des LVL2-Triggers überprüft werden.

Dazu wurde in einer Offline-Analyse in den RICH-Rohdaten nach Ringen gesucht. Ein Beispiele für einen solchen Cherenkovring ist in Abb. 6.6 gezeigt. Die gefundenen Ringmittelpunkte („RICH hits“) können dann mit den Mittelpunkten aus den Matching-

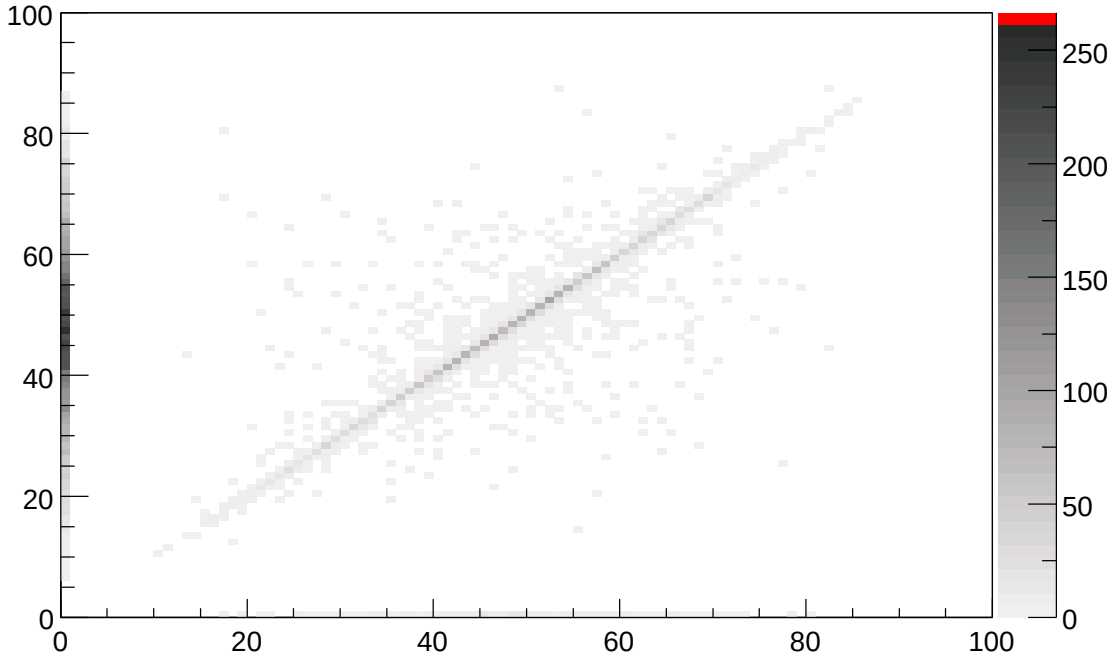


Abbildung 6.7.: Vergleich der online durch die Hardware identifizierten Ringmittelpunkte mit den durch die spätere offline Analyse gefundenen. Beide Algorithmen (LVL2-Trigger-Hardware und Offline-Analyse-Software) haben die selben Daten nach RICH-Ringen durchsucht. Das Bild zeigt die Korrelation der gefundenen X-Koordinaten.

Unit-Rohdaten verglichen werden. Das Ergebnis zeigt Abb. 6.7 [62].

Die Korrelation der Ringmittelpunkte ist offensichtlich, aufgrund verschiedener Algorithmen und Kriterien in der On- und Offlineringsuche wurden neben den von beiden Suchmethoden erkannten Ringen auch einige Ringe nur von jeweils einer Methode erkannt. Dies wird deutlich an den Streifen entlang der X- bzw. Y-Achse. Die Image-Processing-Unit des RICH ist also grundsätzlich in der Lage, Ringe zu finden und diese Information an den LVL2-Trigger weiterzugeben. Die Integration dieses Datenpfades in den Trigger und in die Datenaufnahme war erfolgreich und ist inzwischen für alle sechs Sektoren implementiert.

6.2.4. Spurrekonstruktion und Impulsmessung

Nachdem bis jetzt Tests zu speziellen Fragestellungen der Einzelsysteme im Vordergrund standen, soll zum Abschluss noch ein Einblick in das Zusammenspiel der Teilsysteme im HADES gegeben werden. Nicht zuletzt ist die Aufgabe des Datenaufnahmesystems ja die Speicherung korrelierter Daten aller Teilsysteme. Eine wirklich umfassende Analyse, die die gesamte Information auswertet um zu physikalischen Größen der Reaktionsprodukte zu gelangen, steht noch aus, so dass hier als Beispiel zwei Untergruppen der HADES Detektoren betrachtet werden.

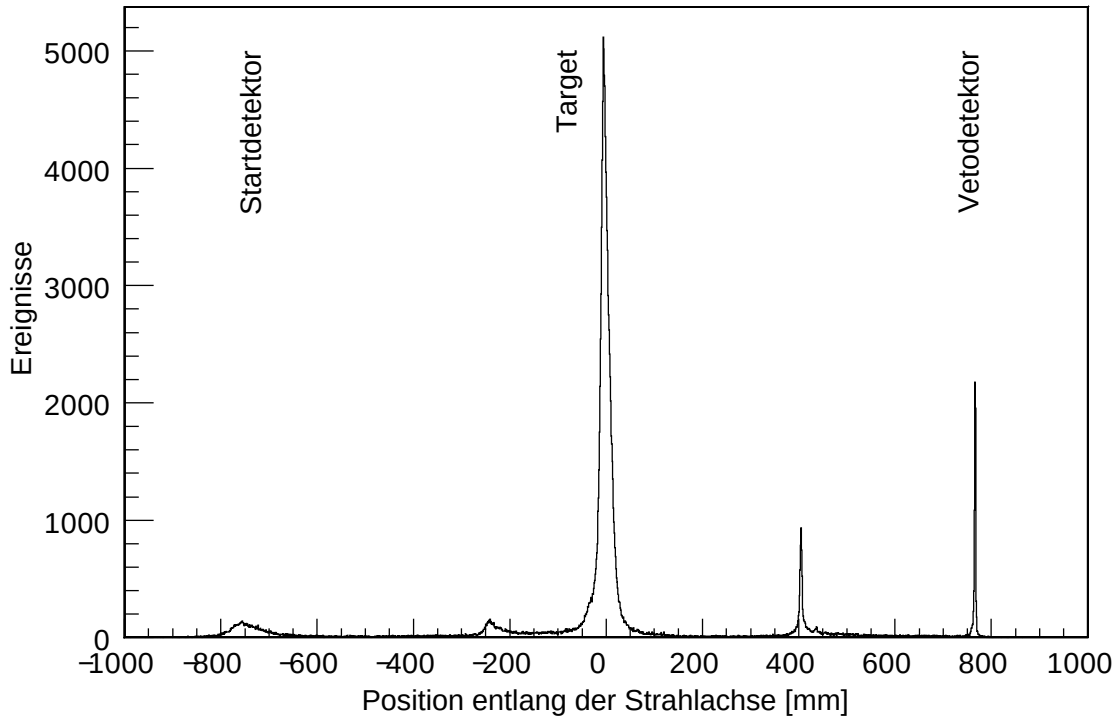


Abbildung 6.8.: Vertexrekonstruktion durch Spurmessung aller geladener Teilchen (Hadronen und Leptonen) in den Driftkammern. Aus den Spurstücken in den Driftkammern des MDC Ebene 2 und 3 werden durch Extrapolation die Schnittpunkte mit der Strahlachse bestimmt. Neben dem Target selbst sind Start- und Vetodetektor und dünne Fensterfolien im Strahlrohr zu sehen. Die Daten entstammen einer Messung von $\mathrm{C} + \mathrm{C}, 1.5\,\mathrm{AGeV}$ ohne Magnetfeld.

Im November 2000 war erstmalig ein Modul der Driftkammerebene 3, also der ersten Ebene hinter dem Magneten, in den Aufbau integriert. Damit standen in einem Sektor drei Ebenen zu Verfügung. Anhand der Messwerte aus der Ebene 2 und 3 wurden damit Algorithmen zur Positionsmessung in den Driftkammern und zur Rekonstruktion von Teilchenbahnen aus diesen Positionen getestet.

Zunächst wurden dabei die Driftzeiten noch gar nicht beachtet, sondern nur die geometrische Position der Drähte genutzt, die ein Signal gezeigt hatten. Aus den gemessenen Orten der Teilchen in den einzelnen Drahtebenen jeder Driftkammer lassen sich jetzt die Spurstücke der Teilchen rekonstruieren.

Um die Ergebnisse einfach überprüfbar zu machen, wurden dabei Messungen ohne Magnetfeld verwendet, so dass die Teilchenbahnen unabhängig vom Impuls Geraden bilden. Das erlaubt es, aus den Schnittpunkten dieser Geraden mit der Strahlachse den Ursprungsort des Teilchens zu bestimmen (Vertexrekonstruktion).

Ein Ergebnis zeigt Abb. 6.8. Neben den Reaktionen im Target selbst (Position $0\,\mathrm{mm}$) finden sich sowohl vor, als auch hinter dem Target noch weitere Maxima. Diese können

dem Start- und Vetozähler (Position $\pm 800\,\mathrm{mm}$) zugeordnet werden, sowie den Ein- und Austrittsfenstern zu dem gasgefüllten Strahlrohrabschnitt im RICH ($+400\,\mathrm{mm}$ und $-200\,\mathrm{mm}$).

Obwohl diese Messung noch nicht die volle Ortsauflösung der Driftkammern ausnutzt, zeigt sie doch, dass die mit dem Datenaufnahmesystem gewonnen Daten in der erwarteten Korrelation stehen und zur Rekonstruktion von Teilchenbahnen und -orte genutzt werden können.

Die Flugzeitwand war während des Experiments im November 2000 voll bestückt, damit konnten die Teilchengeschwindigkeiten in einem großen Raumwinkel und insbesondere in Korrelation mit anderen Detektoren bestimmt werden. Das obere Spektrum in Abb. 6.9 auf der nächsten Seite ist ein Flugzeitspektrum für alle Teilchenarten. Neben dem Maximum bei $6\,\mathrm{ns}$, das leichten Teilchen mit $\beta \approx 1$, also Leptonen und schnellen Pionen entspricht, gibt es eine breite Verteilung langsamerer Teilchen, hauptsächlich thermische Pionen und Protonen aus dem Feuerball der Kollision.

Fordert man für den selben Datensatz, dass auch der RICH in der entsprechenden Richtung vom Target ein Signal gesehen hat, so werden damit nur Teilchen mit $\gamma > 18$ selektiert. Das Ergebnis zeigt das untere Spektrum in Abb. 6.9 auf der nächsten Seite. Die langsamen Teilchen werden unterdrückt und auch die Zahl der schnellen Teilchen ist deutlich geringer, da auch die schnellen Pionen vom RICH herausgefiltert werden. Der verbleibende Untergrund entsteht vermutlich durch zufällige Koinzidenzen der sehr häufigen Hadronen mit fehlidentifizierten Ringmustern im RICH.

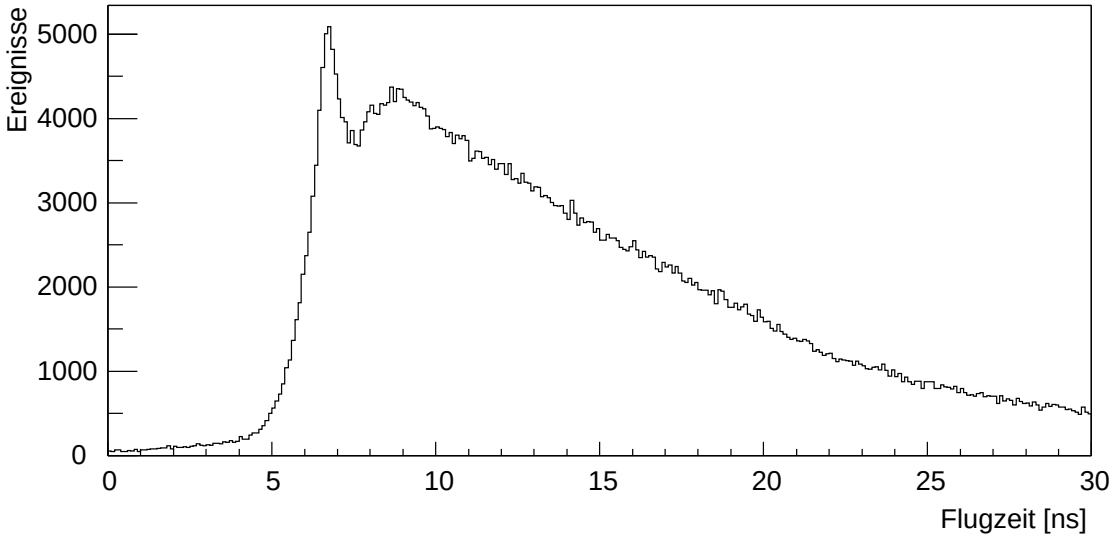
Da der HADES Magnet während der Strahlzeit in Betrieb war, kann über die Teilchenposition vor und hinter dem Magnetfeld die Ablenkung der Teilchen von der geraden Trajektorie gemessen werden. Diese Ablenkung entspricht einer Impulsänderung senkrecht zur Bewegungsrichtung des Teilchens und ist ein direktes Maß für den Teilchenimpuls vor Eintritt in das Feld [63].

Da nur ein einziger Sektor mit einer Driftkammer hinter dem Magnetfeld ausgerüstet war, wurde zur Positionsmessung hinter dem Magnetfeld der TOF-Detektor genutzt. Die Messung in polarer Richtung, also in der Richtung der Ablenkung im Magnetfeld, erfolgte in diesem Fall über die Koordinate der angesprochenen Szintillatorstreifen. Somit ist die Auflösung durch die Breite dieser Streifen ($2\,\mathrm{cm}$ bzw. $3\,\mathrm{cm}$) gegeben.

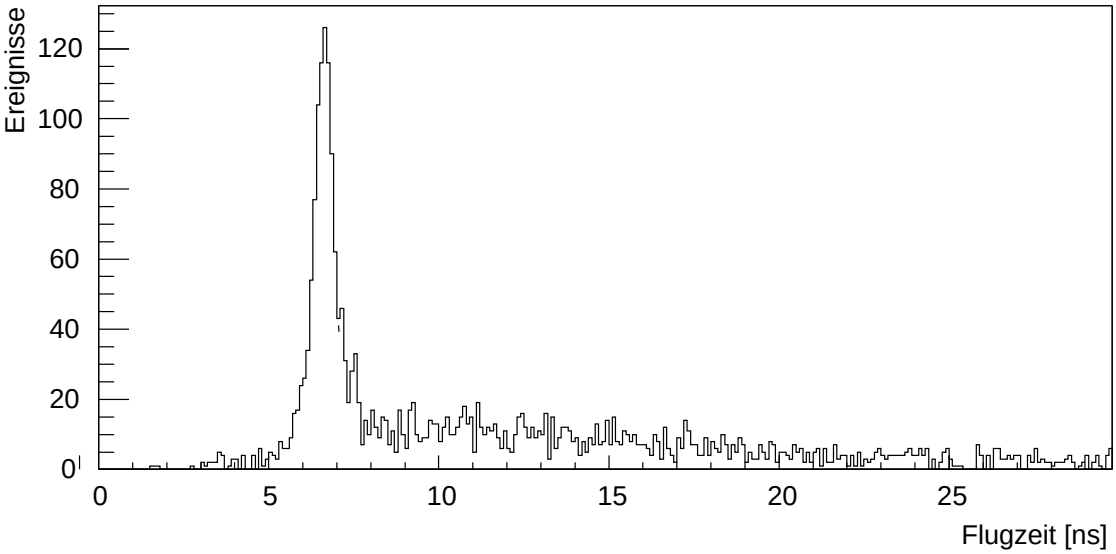
Ein so erstelltes Impulsspektrum zeigt Abb. 6.10 auf Seite 71. Aufgetragen sind die Teilchengeschwindigkeiten in Einheiten der Lichtgeschwindigkeit gegen die Teilchenimpulse in $\mathrm{MeV/c}$. Im oberen Spektrum sind die Messwerte aller geladenen Teilchen enthalten. Man erkennt die kinematische Kurve der schweren Teilchen (Protonen und sehr schwach Deuteronen), das Maximum der leichten Teilchen (hohe Geschwindigkeit, kleiner Impuls) liegt bei $\beta = 0.5$.

Im unteren Spektrum sind dagegen wieder nur die Ereignisse akkumuliert, in denen der RICH ein Lepton erkannt hat. Wie man sieht, werden außer den leichten Teilchen mit $\beta \approx 1$ tatsächlich alle anderen Teilchensorten, insbesondere auch die Pionen, deutlich unterdrückt.

Aus der Impuls- (Abb. 6.10) und Flugzeitinformation (Abb. 6.9) lässt sich zum Schluss noch ein Massenspektrum generieren, wie es in Abbildung 6.11 gezeigt ist. Aufgetragen ist die Zahl der Ereignisse gegen das Produkt aus der Ladung (in Einheiten

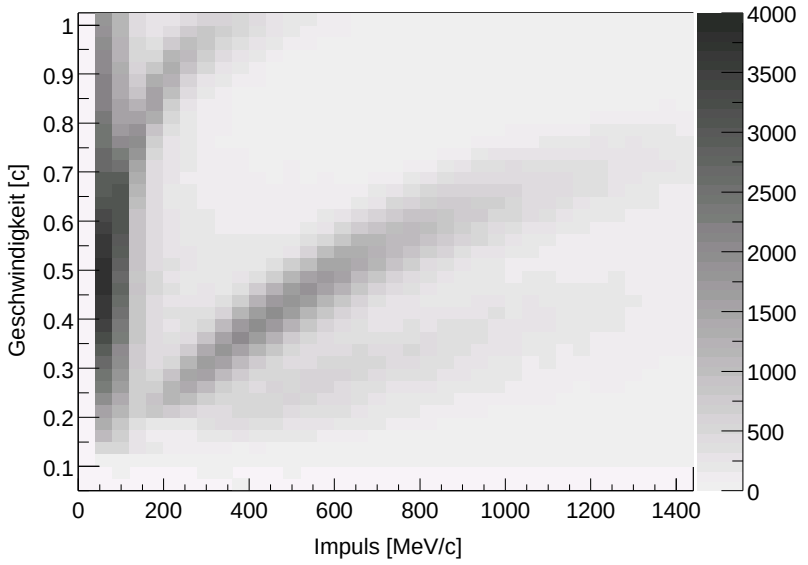


(a) Flugzeitspektrum aller geladenen Teilchen

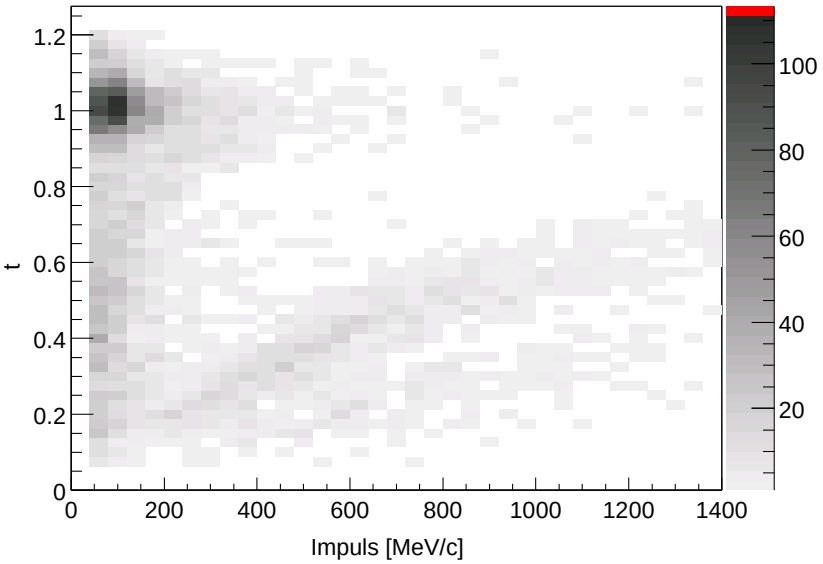


(b) Flugzeitspektrum der geladenen Teilchen, denen ein Cherenkovring im RICH zugeordnet werden konnte

Abbildung 6.9.: Flugzeitspektren geladener Teilchen, Reaktion $\mathrm{C} + \mathrm{C}$ bei $1.5\,\mathrm{AGeV}$. Aus der breiten Verteilung aller geladenen Teilchen (Elektronen, Pionen, Protonen etc) selektiert der RICH-Detektor die schnellsten (Elektronen).



(a) Ohne Leptonenidentifizierung durch den RICH wurden 1 537 008 geladene Teilchen akkumuliert. Im Spektrum erkennt man die unterschiedlichen Teilchenmassen auch bei $v/c \approx 1$.



(b) Mit der Information über die Teilchengeschwindigkeit aus dem RICH bleiben nur noch 6912 leichte Teilchen (Leptonen) im Spektrum übrig

Abbildung 6.10.: Geschwindigkeit der Reaktionsprodukte aus Kohlenstoff-Kohlenstoff-Kollisionen bei $1.5\,\mathrm{AGeV}$ aufgetragen gegen den Impuls

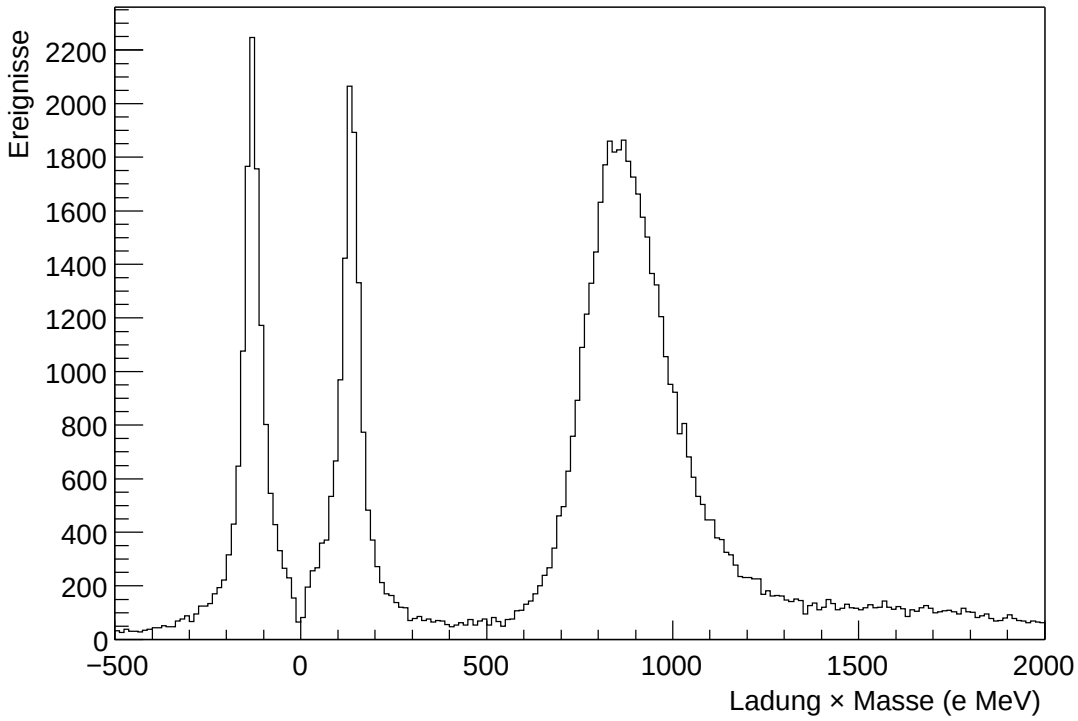


Abbildung 6.11.: Massenspektrum der Produkte aus der Reaktion $\mathrm{C} + \mathrm{C}$ bei $2\,\mathrm{AGeV}$. Zu sehen sind die Linien von π^-, π^+ und von Protonen

von e) und der Masse (in MeV). Es ergeben sich die drei Linien bei der Masse von π^-, π^+ und von Protonen. Die Asymmetrie in den Pionenlinien liegt in der unterschiedlichen Akzeptanz des HADES Detektors begründet, der für die Flugzeitmessung unter kleinen Vorwärtswinkeln noch nicht den endgültigen Ausbaustand erreicht hat [64].

6.3. Zusammenfassung

Die in diesem Kapitel vorgestellten Messungen belegen, dass die VMEbus-Auslese, der Datentransport und das Event-Building der HADES-Datenaufnahme beim Einsatz in Messungen die gestellten Anforderungen voll erfüllt. Ausgehend von frühen Überprüfungen des Konzepts und den darin gewonnenen Erfahrungen wurde das System bis zum Vollausbau entwickelt. Die volle Leistungsfähigkeit des Datentransports und Event-Buildings wurde in einem Beschleunigerexperiment gezeigt, in dem ohne Datenreduktion durch Multilevel-Trigger mit den Designwerten der Event-Building-Stufe ($2\,\mathrm{kHz}$ Ereignisrate, $5\,\mathrm{MByte/s}$ Datenrate) gemessen wurde. Die Analyse der Daten und die Überprüfung der erwarteten Korrelationen zeigt ein fehlerfreies Arbeiten des Eventbuildings. Darüber hinaus lässt die gemessene Auslastung der Rechnerressourcen noch ein großes Potenzial für Leistungssteigerungen erwarten.

Neben dem Erfüllen der grundsätzlichen Anforderungen war besonders die Flexibilität von Datentransport und Event-Building ein bemerkenswerter Punkt, der die schnelle Durchführung von Spezialmessungen mit einem Teilsystem (RICH-Kalibration) oder auch das schnelle Ausgrenzen fehlerhaft arbeitender Subsysteme erlaubte. Die Robustheit und Fehlertoleranz, die ursprünglich nur für die Behandlung von Datenverlust auf dem Transportweg eingebaut wurde, erwies sich auch in der Testphase der Ausleseelektronik als äußerst wertvoll.

Auch die Auslese der LVL2-Triggerinformation wurde schon erfolgreich durchgeführt, was eine Überprüfung der getroffenen Triggerentscheidungen als Vorbereitung auf die volle Inbetriebnahme der zweiten Triggerstufe ermöglicht. Damit stünde dann dem Betrieb des Gesamtsystems in einem Experiment mit der vollen Trigger- und Datenrate nichts mehr entgegen.

7. Ausblick

Das HADES-Datenaufnahmesystem baut auf Konzepten zum Datentransport auf, wie sie gerade z.Z. für sehr große Experimente z.B. am LHC diskutiert und geplant werden. Die gesamte Datenaufnahmesoftware wurde systemunabhängig ausgeführt und schon in der vergangenen Entwicklung wurden sowohl Hardware als auch Systemsoftware ausgetauscht und dem aktuellen Stand angepasst. Damit wird auch in Zukunft das System den jeweils neuesten und leistungsfähigsten Techniken folgen können.

Ein einfacher nächster Schritt wäre zum Beispiel die bessere Ausnutzung des Event-Builders durch schnellere Bandlaufwerke und den Einsatz von ATM über OC12 mit einer Bandbreite von $622\,\mathrm{Mbit/s}$.

Entsprechende Voruntersuchungen und Labortests vorausgesetzt, sind auch größere Anpassungen an aktuelle Entwicklungen in der Rechner- und Telekommunikationstechnologie denkbar. Erste Erfahrungen mit Traffic-Management über GigaBit-Ethernet aus dem Jahr 2000 [53] lassen es möglich erscheinen, den ATM-Datentransport durch einen über GigaBit-Ethernet zu ersetzen, wenn sich daraus Verfügbarkeits-, Leistungs- oder Preisvorteile ergeben würden. Ebenso würden die nur „weichen“ Echtzeitanforderungen an die VMEbus-Auslese den Einsatz von Linux auf den VMEbus-CPUs ermöglichen, sobald leistungsfähige VMEbus-Hardware und -Treibersoftware zur Verfügung stünde.

Eine nicht nur technische, sondern konzeptionelle Erweiterung des Datenaufnahmesystems würde am ehesten im Bereich des Event-Buildings ansetzen. Eine dritte Triggerstufe ist ja schon im ursprünglichen Konzept vorgesehen, aber auch durch einfaches Vervielfachen des Event-Builders ließen sich evtl. notwendige Leistungssteigerungen erzielen.

7.1. Event-Building mit mehreren Prozessoren

Wie die Messungen aus Kapitel 3 und 4 zeigen, sind die praktisch erreichbaren Bandbreiten von VMEbus und ATM/OC3 ungefähr gleich bei ca. $15\,\mathrm{MByte/s}$. Andererseits teilen sich sieben Crate-Event-Builder à $15\,\mathrm{MByte/s}$ einen Event-Builder mit auch nur $15\,\mathrm{MByte/s}$, der dann ein noch langsameres Bandlaufwerk bedient. Nun steht nicht zu erwarten, dass alle Crate-Event-Builder exakt mit maximaler Bandbreite arbeiten, dafür sind die ausgelesenen Detektoren zu unterschiedlich im Datenvolumen, aber zumindest RICH und MDC könnten ihre Bandbreiten sättigen, u.U. wären für MDC sogar die Erweiterung auf zwei VMEbus-Crates nötig. Um diesen Datenstrom zu speichern, wären drei bis vier Event-Builder notwendig.

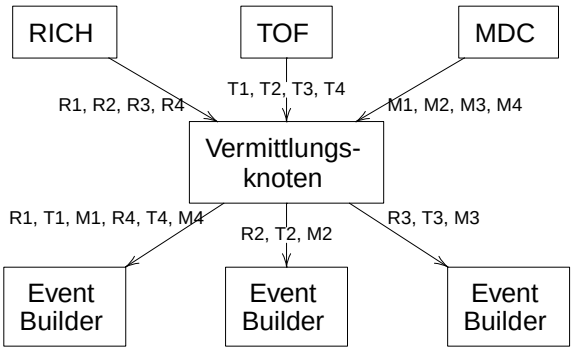


Abbildung 7.1.: Erweiterung der Parallelisierung auf mehrere Event-Builder: Ereignisdaten zu den Ereignissen 1, 2, 3 und 4 werden von den Subsystemen RICH (R), TOF (T) und MDC (M) erzeugt. Daten, die zu einem Ereignis gehören, werden nach einem vorgegebenen Verfahren auf mehrere Event-Builder verteilt.

Hier kommt jetzt die Flexibilität eines Telekommunikationsnetzwerkes zum Tragen. Für das Event-Building wird bisher ja nur das Konzentrieren des Datenstroms auf ein Gerät genutzt. Möglich ist aber die Übertragung zwischen beliebig vielen Quellen und Zielen. Zur Parallelisierung der Auslese (vgl. Abschn. A.4) käme dann also noch die Parallelisierung des Event-Buildings. Abbildung 2.1 auf Seite 11 würde also erweitert zu Abb. 7.1.

Die Crate-Event-Builder schicken ihre Crate-Events nicht mehr immer an den selben Event-Builder, sondern verteilen sie nach einen vorgebenden Algorithmus auf mehrere Ziele. Der Algorithmus muss sicherstellen, dass alle Crate-Events eines Ereignisses beim selben Event-Builder landen, die einfachste Form wäre, die Nummer des Event-Builders aus der Triggernummer modulo der Zahl der Event-Builder auszurechnen („Barrel Shifter“).

Die einzige Änderung, die dazu am existierenden System notwendig wäre, wäre im Programmteil „memnet“. Dieser müsste, statt nur einem, mehrere logische Netzwerkkanäle öffnen und die Crate-Events auf diese verteilen. Alles andere wäre durch eine entsprechende Konfiguration der Vermittlungsmatrix im ATM-Switch zu erledigen, der Rest des Systems, insbesondere der Event-Building-Algorithmus, bliebe unangetastet.

Schon mit dem einzelnen, vorhandenen ATM-Switch könnten so 8 Crate-Event-Builder mit 8 Event-Buildern verschaltet werden, was in einer Datenrate des Event-Buildings von $8 \times 15\,\mathrm{MByte/s} = 120\,\mathrm{MByte/s} \approx 0.5\,\mathrm{TByte/h}$ resultieren würde. Durch den Einsatz von schnelleren Schnittstellen und weiteren ATM-Switches lässt sich diese Zahl theoretisch beliebig steigern, aber schon bei einem halben Terabyte pro Stunde stellt sich die Frage, wie die schiere Datenmenge jemals weiterverarbeitet werden soll.

7.2. Dritte Triggerstufe

Die grundsätzliche Idee der dritten Triggerstufe besteht darin, außer den Informationen von RICH, TOF und PreShower, die schon in der zweiten Triggerstufe genutzt werden, auch noch die Messwerte der MDC in die Triggerentscheidung einzubeziehen. Die Hauptschwäche der zweiten Triggerstufe besteht darin, dass durch die Ablenkung im Magnetfeld die Teilchenpositionen im RICH einerseits und im TOF/PreShower andererseits nur noch in der azimutalen Koordinate zusammenhängen, in der polaren Richtung ist eine Zuordnung nicht möglich. Die dritte Triggerstufe könnte durch ein Verfolgen der Teilchenbahn durch die Driftkammern diese Zuordnung vornehmen, und damit die vorher falsch identifizierten Ereignisse weiter unterdrücken. Sie würde also insbesondere dann einen Vorteil erbringen, wenn die zweite Triggerstufe sich in der Praxis als weniger effizient erweist, als angenommen.

Grundlage der Triggerentscheidung ist also eine mehr oder weniger vollständige Rekonstruktion der Teilchenbahnen durch das Spektrometer, ein Algorithmus, der einen hohen Berechnungsaufwand erfordert. Die dritte Triggerstufe ist deshalb am ehesten auf leistungsfähigen CPUs zu implementieren, die kostengünstig mit „Allerwelts“-Rechnern zur Verfügung stehen.

Der logische Datenfluss für den LVL3-Trigger ist in Abb. 2.1 auf Seite 11 schon eingezeichnet, es werden Daten der zweiten Triggerstufe mit denen der Driftkammern zusammengeführt und das Ergebnis an die Crate-Event-Builder gemeldet. Physikalisch müssen also der Crate-Event-Builder des Trigger-Crates, dieser kennt die Daten der Matching-Unit, und die Crate-Event-Builder der MDC-Crates Daten an rechenleistungsstarke CPUs übermitteln und anschließend die Antworten an alle Crate-Event-Builder verteilt werden. Alle Crate-Event-Builder sind sowieso schon mit dem ATM-Switch verbunden, wenn man die LVL3-Trigger-CPUs auch an diesen anschließt, kann der Transport der Triggerdaten analog zum Transport der Ereignisdaten abgewickelt werden. Auch das Verteilen der Aufgaben auf mehrere LVL3-Trigger-CPUs wäre, analog zum vorigen Abschnitt, gelöst. Der Datentransport würde, ohne weitere Verkabelung, wie in Abb. 7.2 auf der nächsten Seite ablaufen.

Von der Matching-Unit werden dem LVL3-Trigger fertig geeichte Teilchenkoordinaten übermittelt, da diese für den LVL2-Triggeralgorithmus ohnedies errechnet werden müssen. Für die MDC-Daten wäre eine ähnliche Vorbereitung der Daten denkbar, da auf den endgültigen Readout-Controllern der MDC zwei Digitale Signalprozessoren zur Verfügung stehen werden, von denen einer die Koordinateneichung durchführen könnte. Trotzdem bleibt, auch bei geeichten Eingabedaten, ein Algorithmus, der mit der vorgegebenen Rate eine vollständige Spurrekonstruktion und den Vergleich mit den LVL2-Daten durchführen kann eine Herausforderung.

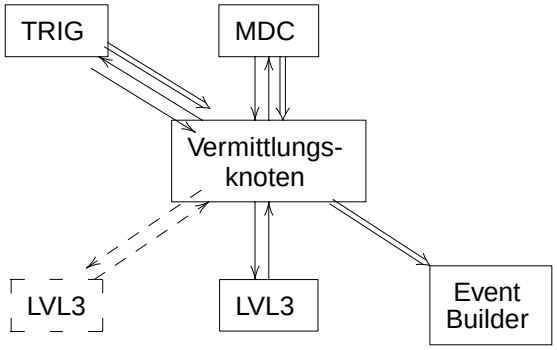


Abbildung 7.2.: Datenfluss unter Einbeziehung der dritten Triggerstufe: Von den Subsystemen werden zuerst nur die triggerrelevanten Daten versendet, die Daten von RICH, PreShower und TOF wurden schon im Trigger-Subsystem zusammengefasst. Erst wenn die LVL3-Triggerprozessoren die Entscheidung zurückgegeben haben, werden die Ereignisdaten an den Event-Builder geschickt. Das Netzwerk erlaubt die Nutzung beliebig vieler Triggerprozessoren (gestrichelt).

Anhang

A. Grundlagen der Datenaufnahme

Zur Einführung sollen die grundlegenden Eigenschaften und Begriffe von Datenaufnahmesystemen bei kernphysikalischen Experimenten zusammengefasst werden.

Die prinzipielle Position des Datenaufnahmesystems im Rahmen eines kernphysikalischen Experiments zeigt Abb. A.1 auf der nächsten Seite. Es hat die Aufgabe, die physikalischen Messwerte des Detektors, die meist in Form von analogen elektrischen Signalen vorliegen, auf permanenten Speichern abzulegen und einer späteren Analyse zugänglich zu machen [65].

Bei Beschleunigerexperimenten kommt eine besondere Bedeutung dem Triggersystem zu, da der Zeitpunkt einer Kern-Kern-Kollision nicht vorherbestimmbar ist. Das Triggersignal muss selbst zuerst aus den Messwerten extrahiert werden, um damit das eigentliche Auslesen und Abspeichern der Daten zu steuern.

A.1. Einfache Datenaufnahmesysteme

Abbildung A.2 auf der nächsten Seite zeigt ein einfaches Datenaufnahmesystem mit „echtem“, d.h. aus den Detektordaten abgeleitetem Trigger und der Busy-Logik.

Aus den Detektorinformationen wird über einen beliebig gearteten Diskriminator eine Signatur extrahiert. Bei Vorliegen der gewünschten Signatur startet das Triggersystem zwei Abläufe:

Zum einen wird die Digitalisierung der elektrischen Signale mit Hilfe von Analog-Digital-Umwandler (ADC) zur Messung von elektrischen Größen bzw. Zeit-Digital-Umwandler (TDC) zur Messung von Zeitabständen gestartet („Gate“ bzw. „Start“). Die Elektronik braucht zum Umwandeln der Signale in digitale Daten eine gewisse Zeit $\tau_{\mathrm{Conversion}}$, in der sie keine neue Umwandlung vornehmen kann.

Zum anderen signalisiert das Triggersystem der Auslesesoftware, dass Daten vorliegen und somit ausgelesen, verarbeitet und gespeichert werden können („Readout“). Dabei muss sichergestellt sein, dass die Auslese der Digitalwerte durch die Software erst erfolgt, wenn die Digitalisierung abgeschlossen ist, also nach Ablauf von $\tau_{\mathrm{Conversion}}$. Danach benötigt die Auslese eines Ereignisses die Zeit τ_{Readout}.

Die „Busy-Logik“ verhindert, dass ein neues Trigger- bzw. Readout-Signal erzeugt wird, solange noch eine Digitalisierung bzw. Auslese stattfindet. Während dieser Zeit (Totzeit: τ) können also keine neuen Ereignisse mehr gemessen werden, selbst wenn diese stattfinden.

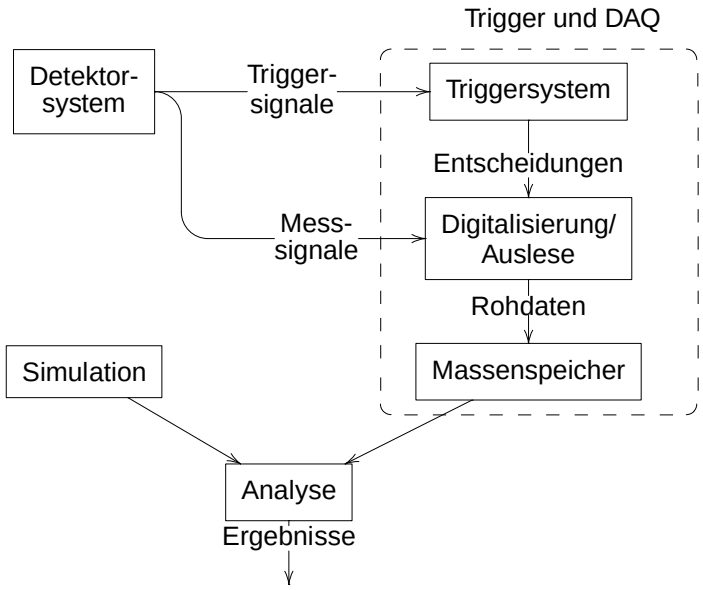


Abbildung A.1.: Position des Datenaufnahmesystems innerhalb eines kernphysikalischen Experiments

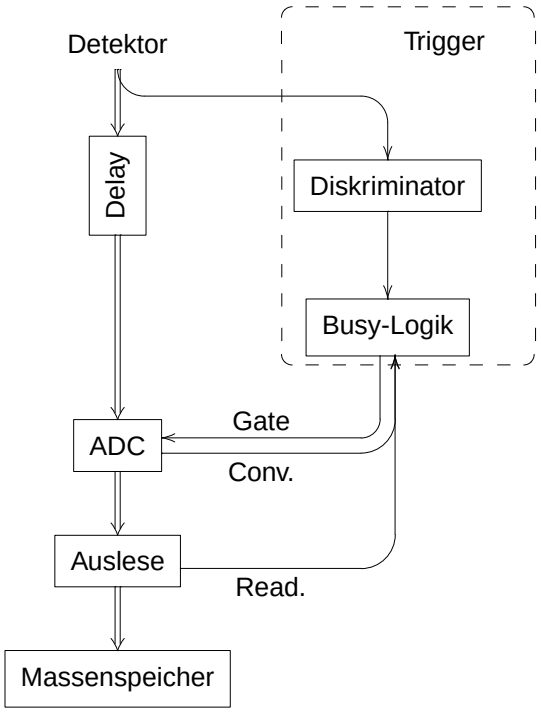


Abbildung A.2.: Prinzipschaltbild eines einfachen Datenaufnahmesystems: Das Triggersignal wird aus den Messwerten abgeleitet, die so lange verzögert werden, bis die Triggerentscheidung fest steht. Die Totzeit setzt sich aus dem Zeitbedarf der Triggerlogik, der Analog-Digital-Wandlung und der Auslese zusammen.

Datenaufnahmesysteme dieser Art bilden die „klassischen“ Systeme in kernphysikalischen Experimenten, häufig implementiert mit Elektronik nach dem CAMAC[1]-Standard und einfachen Steuerungen. Sie sind für Labormessungen und kleinere Experimente gut geeignet und nach wie vor in Gebrauch [66].

A.2. Totzeit

Als Totzeit τ wird der Zeitraum nach dem Beginn einer Messung bezeichnet, in dem keine weitere Messung durchgeführt werden kann. Diese Zeit ist im einfachsten Falle konstant, normalerweise aber um einen wahrscheinlichsten Wert verteilt.

Die Projektil-Target-Kollisionen in einem Beschleuniger-Experiment bilden einen Poisson-Prozess. Das heißt, die Wahrscheinlichkeit, dass eine Kollision stattfindet, ist in jedem Zeitabschnitt gleich groß. Betrachtet man allerdings die Zahl der Kollisionen in einem hinreichend langen Zeitintervall, so ist diese normalverteilt, es ist also sinnvoll, von einer mittleren Ereignisrate ($\mathrm{Ereignisse/Zeit\,[Hz]}$) zu sprechen.

Bei einer konstanten Totzeit τ und einer mittleren Ereignisrate R errechnet sich die gemessene Rate R' aus [67]

$$R' = R - RR'\tau = \frac{R}{1 + R\tau}. \tag{A.1}$$

Häufig wird auch das Produkt

$$R'\tau = \frac{1}{\frac{1}{R\tau} + 1} \tag{A.2}$$

als Totzeit bezeichnet und dann in Prozent angegeben.

Ein Spezialfall der Gleichung A.2 ist der Fall $\tau = 1/R$, wenn also der *mittlere* Zeitabstand zwischen zwei Ereignissen gleich der Totzeit ist. Daraus ergibt sich dann $R'\tau = 50\,\%$. Obwohl das Datenaufnahmesystem also *im Mittel* in der Lage wäre, alle Ereignisse zu verarbeiten, geht trotzdem die Hälfte der Ereignisse verloren. Somit ist eine Verkleinerung der Totzeit noch deutlich unter den mittleren Zeitabstand zweier Ereignisse immer noch vorteilhaft (z.B. $\tau = \frac{1/R}{2} \Rightarrow R'\tau = 33\,\%$).

A.3. Datenpuffer – Derandomisierung

Bei einem Datenaufnahmesystem wie in Abschn. A.1 beschrieben, setzt sich die Totzeit aus zwei Anteilen zusammen: der Zeit, die für die Digitalisierung von der Hardware benötigt wird, und der Zeit, die die Software zur Auslese, Verarbeitung und Speicherung braucht. ($\tau = \tau_{\mathrm{Conversion}} + \tau_{\mathrm{Readout}}$)

[1] Vom ESONE-Komitee (European Studies on Norms for Electronics) entwickelter Standard für Nuklearelektronik

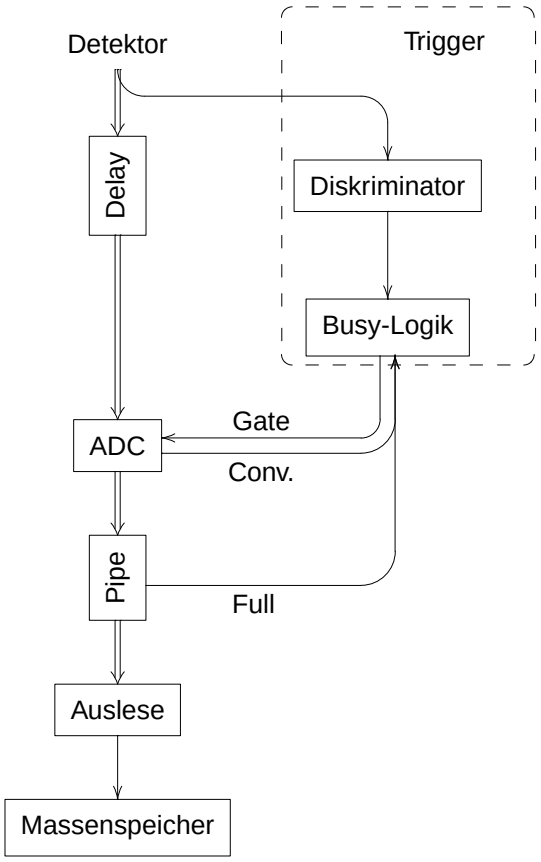


Abbildung A.3.: Datenaufnahmesystem mit Zwischenspeicher zur Derandomisierung: Solange der Zwischenspeicher (Pipe) nicht voll läuft, ist die Totzeit nur so lang wie die Dauer der A/D-Wandlung.

Das Einführen eines Zwischenspeichers zwischen der Digitalisierungsstufe und der Verarbeitungsstufe erlaubt es, ein neues Ereignis (oder mehrere) zu digitalisieren, während das vorhergehende noch verarbeitet wird. Damit kann die Totzeit auf $\tau = \tau_{\mathrm{Conversion}}$ gedrückt werden (Abb. A.3).

Voraussetzung ist dann nur noch, dass die mittlere Verarbeitungzeit unter dem mittleren Zeitabstand zwischen zwei Ereignissen liegt:

$$R < \frac{1}{\tau_{\mathrm{Readout}}} \Rightarrow \tau = \tau_{\mathrm{Conversion}}. \tag{A.3}$$

Ist das nicht der Fall, dann wird der Zwischenspeicher volllaufen und damit die Verarbeitungszeit wieder in die Totzeit eingehen:

$$R > \frac{1}{\tau_{\mathrm{Readout}}} \Rightarrow \tau = \tau_{\mathrm{Readout}}. \tag{A.4}$$

Da bei dieser Betrachtung die Zeiten jeweils durch ihre Mittelwerte ersetzt werden, nennt man dies Derandomisieren.

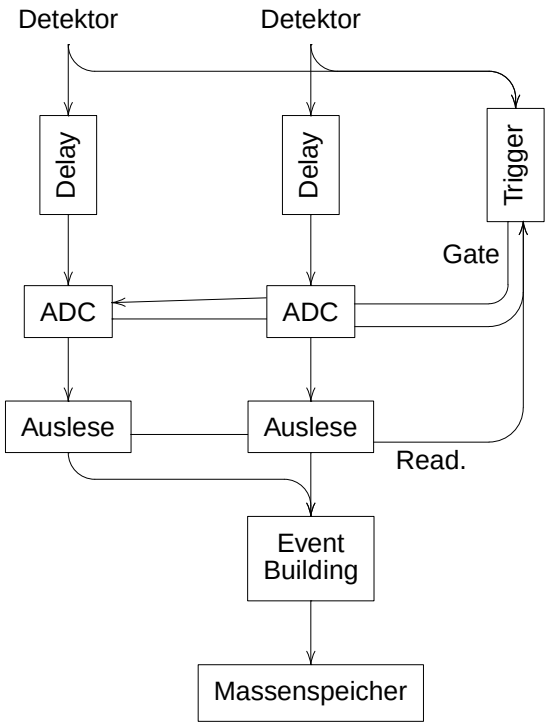


Abbildung A.4.: Parallele Auslese der Detektoren mit anschließendem Event-Building: Die Totzeit jedes einzelnen Ausleseastes ist kürzer als bei der Auslese mit nur einem Prozessor.

Nuklear-Elektronik-Module mit solchen Zwischenspeichern sind als Standardkomponenten auf dem Markt erhältlich, häufig als VMEbus[2]-Einschübe unter der Bezeichnung „Multi-Event“-fähig. Es muss allerdings beachtet werden, dass die Entkoppelung der Busy-Logik von der Datenaufnahmesoftware eine entsprechende Auslegung der Software erfordert, man die „Multi-Event“-Fähigkeit also mit einer einfachen Datenaufnahmesoftware nicht ausnutzen kann.

A.4. Parallelisierung

Wenn die mittlere Verarbeitungszeit länger ist als der mittlere Abstand zweier Ereignisse, führt Derandomisierung nicht zur Leistungssteigerung (Gleichung A.4 auf der vorherigen Seite). In diesem Fall ist die Parallelisierung der Auslese notwendig.

Dabei macht man sich zunutze, dass die Auslese und ein Teil der Verarbeitung der Daten für jede Digitalisierungseinheit unabhängig, also auch parallel, erfolgen kann. Durch die geringere Datenmenge, die pro Verarbeitungseinheit anfällt, sinkt die Verarbeitungszeit. Allerdings entsteht damit ein beträchtlicher Mehraufwand. Da man die

[2] Versa Module Eurocard, ein von den Firmen Motorola, Mostek und Signetics im Jahre 1981 vorgeschlagener Standard für ein erweiterungsfähiges BUS-System [39]

Daten eines einzelnen Ereignisses zur Analyse in einem gemeinsamen Datensatz haben will, müssen die unabhängig ausgelesenen Daten zusammengeführt werden, bevor sie gespeichert werden. Das macht einen weiteren Verarbeitungsschritt, das „Event-Building“ notwendig (Abb. A.4 auf der vorherigen Seite). Jedes Teilsystem wird dabei von einem eigenen Prozessor ausgelesen und die Daten zu „Sub-Events“ formatiert. Diese werden dann an einen zentralen Rechner übertragen, dort zu vollständigen „Events“ zusammengefasst und gespeichert.

Dabei muss absolut sicher gestellt sein, dass die Daten, die als zu einem Ereignis gehörig gespeichert werden, auch wirklich zu einem Ereignis gehören, also nicht z.B. ein Sub-Event des eintausendsten Ereignisses mit einem des eintausendersten Ereignis zusammengeführt wird. Tritt eine solche Vermischung der Daten („Event-Mixing“) ein, so sind die Daten in der Regel wertlos, weil keine Korrelationen zwischen den Messwerten gebildet werden können.

Systeme dieser Art sind in Experimenten verschiedenster Größenordnung im Einsatz und erlauben zum Teil sehr hohe Datenraten, als Beispiel sei nur das „Multi-Branch-System“ der GSI-Darmstadt ([68]) erwähnt.

Natürlich lassen sich Derandomisierung und Parallelisierung auch gemeinsam einsetzen.

A.5. Mehrstufige Trigger Systeme

Den wohl entscheidenden Unterschied zwischen allgemein eingesetzten Datenaufnahmesystemen und dedizierten Entwicklungen von sehr leistungsfähigen Systemen für ein spezielles Experiment stellt die Einführung von mehreren Triggerstufen dar. Diese erhöht nicht den gesamten Datendurchsatz des Datenaufnahmesystems, sondern erlaubt die Klassifizierung und Reduktion der Daten während der Messung.

Bei einem System wie in Abb. A.4 auf der vorherigen Seite werden nur Teile der Verarbeitung parallelisiert, einige Verarbeitungsschritte und insbesondere das Speichern muss mit dem gesamten Datenvolumen geschehen. Wenn die Datenrate die Schreibrate des Speichermediums übersteigt, dann gibt es wiederum zwei Möglichkeiten.

Grundsätzlich kann man vollständig parallelisieren, d.h. an die Ausleseeinheiten werden jeweils Speichergeräte angeschlossen, wobei in jedem Speichermedium jeweils ein Teil jedes Ereignisses abgelegt wird. Der Event-Building Schritt ist vor der Analyse notwendig, kann aber nicht mehr in Echtzeit stattfinden („Offline-Event-Building“). Dieses Verfahren ist zwar technisch einfach, aber wenig praxistauglich. Insbesondere hat man kaum die Möglichkeit, die Qualität der Daten während der Messung zu beurteilen und nicht zuletzt stellt sich das Problem, wie die schiere, parallel gesammelte Datenmenge später zusammengefügt und weiterverarbeitet werden soll.

Eine bessere, aber auch aufwendigere Methode stellt die Datenreduktion dar.

Wie oben beschrieben, hat die erste Triggerstufe (in einem einfachen Datenaufnahmesystem ist es die einzige) die Aufgabe, aus dem kontinuierlichen Strom von elektrischen Signalen, die der Detektor liefert, diejenigen herauszufiltern, die zu einem interessanten Ereignis gehören. So wird z.B. das „Gate“ des ADCs nur dann geöffnet, wenn

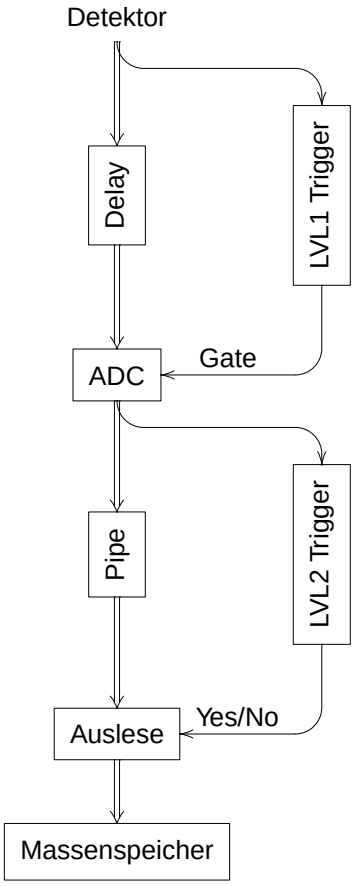


Abbildung A.5.: Mehrstufiges Trigger System: In jeder Stufe wird die Datenrate verringert. Der Zeitbedarf der langsamen, späten Stufen (Auslese, Massenspeicher) wird an den der schnellen frühen Stufe (A/D-Wandlung) angepasst.

das Signal wahrscheinlich von einem Teilchen im Detektor stammt.

In einem mehrstufigen Triggersystem haben die weiteren Stufen genauso die Aufgabe, aus dem Strom von Daten, den die erste Triggerstufe erzeugt, diejenigen zu selektieren, die physikalisch interessant sind. Jede Triggerstufe durchsucht also die Daten der vorhergehenden Stufe nach charakteristischen Signaturen und löst nur bei Vorhandensein dieser Signatur ein Triggersignal aus. Damit verringert sich die Triggerrate von Stufe zu Stufe.

Häufig werden dabei dreistufige Systeme gewählt, was sich aus der Aufteilung des Datenaufnahmesystems (Messen, Digitalisieren, Auslesen, Speichern) ergibt.

Eine zweite Triggerstufe würde also hinter der Digitalisierung eingesetzt, also noch mit den elektrischen, jetzt aber schon digitalen Daten arbeiten, während eine dritte Triggerstufe hinter der Auslese, und damit vor der Speicherung, eingeführt würde. Diese dritte Triggerstufe würde als Software mit Daten im engeren Sinne arbeiten.

Da jede Triggerentscheidung einerseits mit den selben Daten arbeitet, die sie filtern soll, andererseits aber jede Triggerentscheidung auch Zeit braucht, müssen die zu klassifizierenden Daten aufgehoben werden, bis die Entscheidung gefallen ist. In der ersten Triggerstufe (und damit natürlich auch bei einfachen, einstufigen Systemen) geschieht das mit einer analogen Verzögerungsleitung. Eine solche Verzögerung brauchen auch alle anderen Triggerstufen, wobei es sich hier um digitale Speicher handelt. Diese werden häufig „Pipeline“ oder kurz „Pipe“ genannt.

Wie man sich aus der Position der Pipe zur zweiten Triggerstufe in Abb. A.5 auf der vorherigen Seite leicht klarmacht, übernimmt diese gleichzeitig die Rolle des Zwischenspeichers der Derandomisierung, so dass diese ein implizites Verfahren in Datenaufnahmesystemen mit mehreren Triggerstufen darstellt.

B. Betrieb der Datenaufnahme

Die der Datenaufnahme zugrundeliegenden Techniken und Abläufe wurden in den vorangegangenen Kapiteln ausführlich erläutert. Wie dagegen die Vorgaben und Überlegungen im konkreten Betrieb des Systems aussehen, soll hier anhand einiger Beispiele verdeutlicht werden. Zur systematischen Beschreibung der einzelnen Programmfunktionen und -optionen sei auf die Handbuchseiten verwiesen.

Es sei noch bemerkt, dass ein großer Teil des Knowhows für die erfolgreiche Datenaufnahme darin besteht, korrekte und effiziente Ausleseprogramme für die spezielle Hardware zu schreiben. Dies erfordert einfache Kenntnisse der Programmiersprache „C“ und tiefgehende Kenntnis der speziellen Hardware, daher ist eine allgemeine Beschreibung im Rahmen dieser Arbeit nicht möglich. Da HADES aber ein vergleichsweise statisches System sein wird, sollten sich die Ausleseprogramme selten ändern. In den folgenden Beispielen wird davon ausgegangen, dass korrekte Ausleseprogramme existieren und alle Teile der Datenaufnahme bereits übersetzt wurden.

Meistens benötigt die verwendete Hardware vor dem Start der Auslese bestimmte Initialisierungsverfahren. Bei HADES ist vereinbart, dass diese Initialisierung nicht von der Datenaufnahme, sondern vom Steuersystem durchgeführt werden soll. Hier wird dieser Schritt deshalb einfach als „Initialisierung der Hardware“ bezeichnet, aber nicht näher beschrieben.

B.1. Auslese eines einzelnen VME-Crates

Hier existiert nur eine Auslese, ein Transportieren und Zusammenführen der Daten ist daher nicht notwendig. Das gesamte Datenaufnahmesystem besteht nur aus einem Auslese- und einem Event-Building-Prozess und wird auf der VMEbus-CPU betrieben. Diese Konfiguration ist die einfachste und wird als „Standalone-System“ bezeichnet.

Ein solch einfaches System ist nicht nur sinnvoll für kleine Labor- und Testmessungen, sondern insbesondere auch für Tests der Elektronik und der Software selbst. Wie bereits erwähnt, wurde sehr viel Wert darauf gelegt, dass die Software ohne spezielle „Laufzeitumgebung“ betrieben werden kann, so dass sie allen Fehleranalysetechniken vom einfachen „printf“-Befehl im Code bis zum Einzelschrittbetrieb im Debugger zugänglich ist.

Deshalb soll das „Standalone-System“ auch ganz konkret mit den Kommandos auf der Betriebssystemebene beschrieben werden.

Im einfachsten Fall genügt es, nach der Initialisierung der Hardware die beiden

Programme `daq_evtbuild` und `daq_readout` (in dieser Reihenfolge) zu starten. `daq_readout` erwartet, dass der gemeinsame Speicherbereich von `daq_evtbuild` bereits angelegt wurde, deshalb ist die Reihenfolge wichtig. Wurde `daq_readout` zwischenzeitlich abgebrochen, so muss auch `daq_evtbuild` abgebrochen und neu gestartet werden, da sich die beiden Programme sonst nicht mehr richtig synchronisieren.

Auf dem Bildschirm sieht das ganze dann so aus:

```
seb $ daq_evtbuild &
[1] 6654
seb $ daq_readout
```

Zum beenden werden die Programme mit einem „Interrupt“ (bei vielen Systemen mit `Ctrl-C` auslösbar) oder „Terminate“-Signal abgebrochen. Also:

```
^C
seb $ kill $!
```

Das Starten ganz ohne Optionen führt zwar zu einer laufenden Datenaufnahme, hat aber ein paar Einschränkungen.

Zum einen machen beide Programme Statistikausgaben auf den Bildschirm, wenn man alles in einem Terminal gestartet hat, schreiben beide durcheinander und das ganze wird etwas unübersichtlich. Mit der Option `-v` lässt sich diese Ausgabe unterdrücken. Mit der Eingabe

```
seb $ daq_evtbuild &
[1] 6654
seb $ daq_readout -v notice
```

werden von der Auslese nur noch wichtige Meldungen angezeigt, die Statistiken vom Event-Builder zeigen, wieviele Daten man schon aufgenommen hat.

Ein weiterer Punkt ist die Prioritätssteuerung, welcher der beiden Prozesse kann wann die CPU nutzen? Eine ausführliche Behandlung des Themas findet sich in [52]. Hier sei nur erwähnt, dass die Auslesefunktion oft Abfragen des Hardwarezustandes in engen Schleifen macht. Um ein Okkupieren der CPU durch den `daq_readout`-Prozess zu verhindern, sollte dessen Priorität mit der Option `-p` vermindert werden. Die Auslese würde dann mit

```
seb $ daq_readout -p -1 -v notice
```

gestartet.

Als nächstes führt ein `daq_evtbuild` ohne Optionen zwar ein vollständiges Event-Building aus, ohne Angabe eines Speichermediums werden die Daten zum Schluss allerdings weggeworfen. Das ist nicht so sinnlos, wie es sich anhört, z.B. kann auf diese Weise ein Online-Event-Client mit Daten versorgt werden. Will man allerdings ein List-Mode-Data-File schreiben, so kann die `-d` Option angeben, welches Ausgabegerät man nutzen möchte.

```
seb $ daq_evtbuild -d file
```

gibt die fertigen Events auf eine Datei im aktuellen Verzeichnis aus. Der Dateiname wird dabei nach den in [69] beschriebenen Regeln automatisch gebildet. Andere Ausgabegeräte, Verzeichnisse und Dateinamen lassen sich durch weitere Optionen steuern.

Die Befehlsfolge

```
seb $ daq_evtbuild -d file -v notice &
[1] 6654
seb $ daq_readout -p -1 -v notice &
```

würde also eine Datenaufnahme im Hintergrund starten, die normalerweise keine Ausgaben macht.

Eine stichprobenartige Kontrolle über den Inhalt der Ereignisse lässt sich, bei laufender Datenaufnahme, durch eine Verbindung zum Online-Event-Server durchführen. Mit

```
seb $ daq_sniff -h localhost | daq_anal -n 1
```

lässt sich ein Ereignis vom Event-Builder abholen und auf dem Bildschirm ausgeben.

Da alle Prozesse auf der selben CPU laufen und sich untereinander synchronisieren, gibt es beim „Standalone-System“ keine besonderen Einstellparameter, die beachtet werden müssten.

B.2. Ein System aus mehreren VMEbus-Crates

In einem System aus mehreren VMEbus-Crates müssen zum einen die Daten zum gemeinsamen Event-Builder transportiert werden, zum anderen muss der Event-Builder aus den Daten verschiedener Quellen erfolgreich Events zusammensetzen können. Bei einem großen System werden auch die Anforderungen an die Systemleistung höher werden. All dies erfordert nicht nur weitere Prozesse, sondern zum Erreichen der optimalen Leistung auch ein sorgfältiges Einstellen der Parameter.

Obwohl ein Starten direkt aus der Betriebssystemebene auch in einem komplexen System im Prinzip möglich ist, wird für den praktischen Einsatz ein Steuerungssystem notwendig sein. Hier werden deshalb die Werte der Einstallparameter diskutiert und u.U. auch mit kurzen Kommandofragmenten illustriert, aber nicht mehr vollständige Befehlsfolgen angegeben.

In der Hauptsache kommen zu den oben beschriebenen Prozessen die beiden Datentransportprozesse für das Senden (`daq_memnet`) und Empfangen (`daq_netmem`) dazu. Der Sendeprozess muss wissen, wohin (`-o`) er die Daten schicken soll, der Empfangsprozess, aus wievielen Quellen (`-m`) und woher (`-i`) er die Daten empfangen soll.

In der Grundform wären die Befehle also

```
seb $ daq_memnet -o ATM:0:50
```

und

```
eb $ daq_netmem -m 3 -i ATM:0:50 -i ATM:0:51 -i ATM:0:52
```

um z.B. Daten von drei VMEbus-Crates entgegenzunehmen. Die Angaben von Ziel und Quelle hängen vom verwendeten Transportnetzwerk ab und müssen auf die Verschaltungsmatrix im ATM-Switch angepasst sein. Möglich ist auch eine Nutzung von UDP/IP für kleine Datenmengen und Tests (`-i UDP:129.187.188.22:3000`).

Neben diesen grundsätzlichen Einstellungen, damit Daten überhaupt transportiert werden können, sind weitere Parameter zur Ausnutzung der Leistungsgrenzen wichtig.

Am offensichtlichsten ist dabei die Steuerung der Bandbreite. Sie wird beim `daq_memnet`-Prozess mit der Option `-w` in kBit/s angegeben. Die Bandbreite aller Sendeprozesse zusammengenommen darf zunächst die Ausgangsbandbreite des ATM-Switches (z.Z. $155\,000\,\mathrm{kBit/s}$) nicht übersteigen. Wirklich begrenzend ist aber normalerweise die Rate, mit der Daten auf's Speichermedium geschrieben werden können. Bei den gemessenen $5\,\mathrm{MByte/s}$ Bandschreibrate wären das ca. $50\,000\,\mathrm{kBit/s}$, die auf alle VMEbus-Crates verteilt werden müssten. Denkbar wären also z.B.

```
seb0 $ daq_memnet -w 30000 -o ATM:0:50
seb1 $ daq_memnet -w 15000 -o ATM:0:51
seb2 $ daq_memnet -w  5000 -o ATM:0:52
```

Da im Event-Builder zwischen Empfangsprozess und Event-Builder-Prozess noch Puffer existieren, kann auch, je nach Extraktionszeit im Synchrotron, aus der Spill-/Spillpause-Struktur ein Vorteil gewonnen werden. Hier ist aber sicherlich ein Herantasten an den optimalen Wert notwendig, denn beim Überschreiten der Bandschreibrate muss der Event-Builder Daten verwerfen und anschließend wieder neu synchronisieren, so dass die Event-Building-Effizienz relativ abrupt abnimmt.

Die Größe der schon angesprochenen Puffer lässt sich mit der Option `-q` steuern, sie muss für den Event-Builder und alle Empfangsprozesse auf den selben Wert gesetzt sein. Hier gilt natürlich in gewissem Maße „Je größer, desto besser!“, zumindest ein Spill sollte in einem Puffer Platz haben. Allerdings sollte man daran denken, dass der angegebene Wert die Größe einer Doppelpufferhälfte einer Empfangswarteschlange angibt. Das Betriebsystem muss also für jede einzelne Warteschlange das Doppelte dieses Wertes an Shared-Memory bereitstellen. Dies erfordert häufig Änderungen an Betriebssystemparametern. Für $8\,\mathrm{MByte}$ große Empfangswarteschlangen ($48\,\mathrm{MByte}$ Shared-Memory) könnten die Befehle auf dem Event-Builder also wie folgt lauten:

```
eb $ daq_netmem -q 8388608 -m 3 -i ATM:0:50 -i ATM:0:51 \
> -i ATM:0:52
eb $ daq_evtbuild -q 8388608 -m 3
```

Der abstrakteste Einstellparameter ist wohl die „Watermark“ des `daq_readout`-Prozesses. Dieser Wert stellt ein, bis zu welcher Länge Ereignisse in einer Netzwerk-Nachricht gepuffert werden, bevor diese verschickt wird. Dies ist eine echte Pufferung, der Event-Builder bekommt die Daten also tatsächlich erst zu sehen, wenn die „Watermark“ erreicht ist.

Bei sehr kleinen Crate-Event-Größen kann dies zu einer großen Verzögerung führen, bevor der Event-Builder das nächste Event zusammensetzen kann. Sendet ein anderes VMEbus-Crate gleichzeitig sehr große Crate-Events, dann können diese innerhalb der Verzögerung ihren Speicherplatz im Event-Builder überfüllen und verlorengehen. Die „sicherste“ Einstellung ist also eine „Watermark“ von 0, damit wird jedes Crate-Event einzeln und sofort verschickt und die Daten aller Subsysteme kommen beim Event-Builder praktisch synchron an.

Dies würde auf der Auslese-CPU also zu folgenden Befehlen führen:

```
seb $ daq_memnet -w 30000 -o ATM:0:50
seb $ daq_readout -w 0 -p -1 &
```

Allerdings erzeugen, wie in Kapitel 4 ausgeführt, sowohl zu kleine, als auch zu große Nachrichten Leistungseinbußen. Leistungsoptimal ist ein Wert, der die Nachrichtengröße knapp unter der Maximalgröße eines Netzwerkpakets einstellt. Dabei ist zu beachten, dass die Crate-Events während der Auslese am endgültigen Speicherort zusammengebaut werden, die Länge des nächsten Crate-Events aber erst mit der Auslese bekannt wird. Das letzte, ausgelesene Crate-Event wird also voraussichtlich über die „Watermark“ hinausgehen.

Der Wert sollte also so eingestellt werden, dass im aktuellen Netzwerkpaket (bei ATM-AAL5 Maximalgröße $64\,\mathrm{KByte}$) noch ein weiteres Crate-Event Platz hat. Da die Größe der Crate-Events selbst wieder von Trigger zu Trigger variiert, kann das nur über die Größenverteilung mit einer gewissen Wahrscheinlichkeit eingestellt werden. Die Leistungseinbuße beim Versenden von $56\,\mathrm{KByte}$ statt $64\,\mathrm{KByte}$ großen Pakten ist allerdings völlig vernachlässigbar, die durchschnittliche Crate-Event-Größe wiederum deutlich unter $8\,\mathrm{KByte}$ (vgl. Tab. 6.2 auf Seite 64). Ein Wert von $56\,\mathrm{KByte}$ ist also ein guter Startpunkt, um höhere Leistung zu erreichen.

```
seb $ daq_memnet -w 30000 -o ATM:0:50
seb $ daq_readout -w 57344 -p -1 &
```

Bei sehr unterschiedlichen Crate-Event-Größen ist es sinnvoll, die „Watermark“ ungefähr an diese Größen (und konsequenterweise an die zugewiesenen Bandbreiten) anzupassen. Damit kommen die Daten der einzelnen Teilsysteme zwar in großen Blöcken, aber ungefähr synchron beim Event-Builder an.

```
seb0 $ daq_memnet -w 30000 -o ATM:0:50
seb0 $ daq_readout -w 36000 -p -1 &
seb1 $ daq_memnet -w 15000 -o ATM:0:51
seb1 $ daq_readout -w 18000 -p -1 &
```

```
seb2 $ daq_memnet -w  5000 -o ATM:0:52
seb2 $ daq_readout -w 6000 -p -1 &
```

Dabei sollten aber auf jeden Fall (außer zu Testzwecken) „Watermarks“ von weniger als $2\,\mathrm{KByte}$ vermieden werden, weil ab diesem Zeitpunkt die Systemaufrufzeit die Datentransportzeit übersteigt und bei großen Datenmengen ein hinreichend schnelles Empfangen der Daten unmöglich wird.

Literaturverzeichnis

[1] KOCH, VOLKER: *Aspects of Chiral Symmetry*. Int.J.Mod.Phys., E6:203–250, 1997. nucl-th/9706075.

[2] HOLSTEIN, BARRY R.: *Chiral Perturbation Theory: a Primer*. lectures delivered at National Summer School in Nuclear Physics; hep-ph/9510344, Juni 1995.

[3] *Particle Physics Booklet*, Juli 2000.

[4] GOLDSTONE, J.: *What is a Goldstone Boson?* Web-Dokument. www-lns.mit.edu/˜eluc/communications/ask-physicist.html#1.

[5] STACHEL, J.: *Towards the Quark-Gluon Plasma*. Nucl.Phys., A654:119c–135c, 1999. nucl-ex/9903007.

[6] BROWN, G.E. und M. RHO: *Scaling effective Lagrangians in a dense Medium*. Phys.Rev.Lett., 66:2720–2723, 1991.

[7] COHEN, T. D., R.J. FURNSTAHL und D.K. GRIEGEL: *Quark and gluon condensates in nuclear matter*. Phys.Rev., C45:1881, 1992.

[8] LUTZ, M., S. KLIMT und W. WEISE: *Meson Properties at Finite Temperature and Density*. Nucl.Phys., A542:521, 1992.

[9] KARSCH, F.: *to be published in the proceedings of Quark Matter 2001*. Nucl. Phys. A (2001) in press.

[10] BROWN, G.E., G.Q. LI, R. RAPP, M. RHO und J. WAMBACH: *Medium Dependence of the Vector-Meson Mass: Dynamical and/or Brown-Rho Scaling?* Acta Phys.Polon., B29:2309–2321, 1998. nucl-th/9806026.

[11] SCHÖN, WALTER: ω *Meson Production in* $\pi^- + N$*ucleus Reactions*. In: IORI, I. (Herausgeber): *Proc. XXXVII International Winter Meeting on Nuclear Physics*, Bormio, März 1999.

[12] FRIMAN, B., W. NÖRENBERG und V.D. TONEEV: *The quark condensate in relativistic nucleus-nucleus collisions*. Eur.Phys.J., A3:165, 1998. nucl-th/9711065.

[13] BRAUN-MUNZINGER, PETER und JOHANNA STACHEL: *Dynamics of Ultra-Relativistic Nuclear Collisions with Heavy Beams: An Experimental Overview*. Nucl.Phys., A638:3–18, 1998. nucl-ex/9803015.

[14] KOCH, V., M. BLEICHER, A.K. DUTT-MAZUMDER, C. GALE und C.M. KO: *Dilepton Production and Chiral Symmetry*. In: *Proc. International Workshop XXVIII on Gross Properties of Nuclei and Nuclear Excitations*, Hirschegg, Januar 2000. nucl-th/0002044.

[15] AGAKICHIEV, G. und OTHERS: *Low-mass e+e- pair production in 158 A GeV Pb-Au collisions at the CERN SPS, its dependence on multiplicity and transverse momentum*. Phys.Lett., B422:405–412, 1998. nucl-ex/9712008.

[16] PORTER, R.J. und OTHERS: *Dielectron Cross Section Measurements in Nucleus-Nucleus Reactions at 1.0 A GeV*. Phys.Rev.Lett., 79:1229–1232, 1997. nucl-ex/9712008.

[17] BRATKOVSKAYA, E. L., W. CASSING, R. RAPP und J. WAMBACH: *Dilepton production and m_T-scaling at BEVALAC/SIS energies*. Nucl.Phys., A634:168–189, 1998. nucl-ex/9803015.

[18] BOKEMEYER, H. und OTHERS: *Development of low mass drift chambers for the HADES Spectrometer*. Nucl. Instrum. and Meth., 2000. to be published.

[19] ZEITELHACK, K. und OTHERS: *The HADES RICH Detector*. Nucl. Instrum. and Meth., A 433:201–206, 1999.

[20] AGODI, C. und OTHERS: *The Time of Flight Wall for the HADES Spectrometer*. IEEE Transactions on Nucl. Science, 45:665, 1998.

[21] FRIESE, J. und OTHERS: *HADES, the New Electron-Pair Spectrometer at GSI*. Nucl.Phys., A654:1017c, 1999.

[22] SCHÖN, HEIKE: *HADES – Ein Dielektronenspektrometer hoher Akzeptanz für relativistische Schwerionenkollisionen*. Doktorarbeit, Universtät Frankfurt, 1995.

[23] KASTENMÜLLER, ANTON JOHANN: *Nachweis von $\mathrm{e^+e^-}$-Paaren aus Schwerionenstößen mit einem RICH Detektor*. Doktorarbeit, Technische Universtät München, Physikdepartment E12, 2000.

[24] GENZ, H.: *Single electron detection in proportional gas counters*. Nucl. Instrum. and Meth., A 112:83–90, 1973.

[25] KOLB, B.W.: *HADES raw data event lengths*. WWW-Dokument. http://www-hades.gsi.de/daq/event_lengths.html.

[26] SAILER, BENJAMIN: *Die Datenaufnahmesteuerung für das HADES Detektorsystem*. Diplomarbeit, Technische Universtät München, Physikdepartment E12, Januar 2001.

[27] LEHNERT, JÖRG: *Echtzeit-Mustererkennung zum Elektronennachweis mit einem RICH-Detektor in relativistischen Schwerionenkollisionen*. Doktorarbeit, Justus-Liebig-Universität Gießen, Fachbereich 07, Juni 2000.

[28] LINS, ERIK: *Entwicklung eines Auslese- und Triggersystems zur Leptonenidentifizierung mit dem HADES-Flugzeitdetektor*. Doktorarbeit, Justus-Liebig-Universität Gießen, Fachbereich 07, März 2001.

[29] PETRI, MARKUS: *Entwicklung eines kombinierten Auslese- und Echtzeit-Triggersystems zum Nachweis von Elektronen/Positronen-Signaturen in einem elektromagnetischen Schauerdetektor*. Doktorarbeit, Justus-Liebig-Universität Gießen, Fachbereich 07, August 2000.

[30] TRAXLER, M. und OTHERS: *The Second Level Trigger System for the HADES Detector*. IEEE Transactions on Nuclear Science, 47/2:376, 2000.

[31] VMEBUS INTERNATIONAL TRADE ASSOCIATION, Scottsdale, AZ 85253 USA: *The VMEbus Specification*, 1987.

[32] ISO C: *Programming Languages — C*. Standard 9899, ISO/IEC, 1990.

[33] ISO POSIX-1: *Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface(API) [C Language]*. Standard 9945-1, ISO/IEC, 1990.

[34] JACOBSSON, R., PH. CHARPENTIER und CLARA GASPAR: *Upgrades of the DELPHI central DAS with the prospect of LEP's last two years at maximum energy*. DELPHI DAS Notes 189, CERN, Juli 2000. http://delphiwww.cern.ch/~pubxx/www/delsec/delnote/public/2000_150_das_189.ps.gz.

[35] COMPASS-COLLABORATION: *Common Myon and Proton Apparatus for Structure and Spectroscopy*. Technischer Bericht, CERN, 1996. http://wwwcompass.cern.ch/compass/proposal/ps/proposal.ps.

[36] SCHMITT, LARS und OTHERS: *Statusbericht zum Datennahmesystem des COMPASS Experiments am CERN*. Technischer Bericht, CERN, 2000. http://wwwcompass.cern.ch/compass/detector/daq/doc/bmbf_status-99-00.ps.gz.

[37] ALICE-COLLABORATION: *The ALICE data acquisiton*. Technischer Bericht, CERN, 1998. http://aldwww.cern.ch.

[38] BÖHMER, MICHAEL: *Das Auslesesystem für den Ringabbildenden Čerenkovdetektor im HADES Spektrometer*. Diplomarbeit, Technische Universtät München, Physikdepartment E12, Dezember 1999.

[39] PETERSON, WADE D.: *The VMEbus Handbook*. VEFA International Trade Association, Scottsdale, AZ 85253 USA, Zweite Auflage, 1991.

[40] CREATIVE ELECTRONIC SYSTEMS S.A., Geneva: *RIO2 8062 PowerPC based RISC I/O Board*. http://www.ces.ch.

[41] THEURER, CHRISTOPH: *Eine schnelle Auslesesteuerung für den Photonendetektor im HADES-RICH*. Diplomarbeit, Technische Universtät München, Physikdepartment E12, Februar 1998.

[42] KNUTH, DONALD E.: *Fundamental Algorithms*, Band 1 der Reihe *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, Zweite Auflage, 1973.

[43] *CERN S-LINK homepage*. WWW-Dokument. http://hsi.web.cern.ch/HSI/s-link/.

[44] SHAH, AMIT und G. RAMAKRISHNAN: *FDDI: a high speed network*. Prentice Hall International, London, 1994. ISBN 0-13-308388-8.

[45] MCDYSAN, DAVID E. und DARREN L. SPOHN: *ATM: theory and application*. McGraw-Hill, 1994. ISBN 0-07-060362-6.

[46] TANENBAUM, ANDREW S.: *Computer Networks*. Prentice Hall International, London, 1996. ISBN 0-13-349945-6.

[47] TANENBAUM, ANDREW S.: *Operating Systems, Design and Implementation*. Prentice Hall International, London, 1990.

[48] COMER, DOUGLAS E.: *Internetworking with TCP/IP*. Prentice Hall International, London, 1988. ISBN 0-13-470188-7.

[49] DUFEY, J-P und I. MANDJAVIDZE: *Event Building with Switching Networks*. In: *Proc. Workshop "Applications of ATM in High Energy Physics"*, Paris, September 1998. ESONE.

[50] PARTRIDGE, CRAIG: *Gigabit Networking*. Addison-Wesley, Reading, Massachusetts, 1993. ISBN 0-201-56333-9.

[51] DUFEY, J-P: *Switch Performance: A general view based on a simple model*. WWW-Dokument. http://lhcb-doc.web.cern.ch/lhcb-doc/presentations/conferencetalks/postscript/ 1999presentations/sitges.pdf.

[52] GALLMEISTER, BILL O.: *POSIX.4: Programming for the Real World*. O'Reilly & Associates, Sebastopol, CA, 1995. ISBN 1-56592-074-0.

[53] YASU, Y. und OTHERS: *Quality of Service on Gigabit Ethernet for Event Builder*. WWW-Dokument. http://jost.home.cern.ch/jost/Presentations/Yoshiji

[54] GÖRINGER, HORST: *Mass Storage at GSI, Vortrag im DV-Unterausschuss Massendatenhaltung*. http://www-aix.gsi.de/~goeri/talks/tsm2000/I0-contents.html, Februar 2001.

[55] BLOOMER, JOHN: *Power Programming with RPC*. O'Reilly & Associates, Sebastopol, CA, 1992. ISBN 0-937175-77-3.

[56] GÖRINGER, HORST: *The Hades Remote Event Server for Online Analysis*. GSI, Darmstadt, August 1999. http://www-aix.gsi.de/~goeri/hades/revserver.html.

[57] BERTINI, DENIS: *Hydra User Interface to the Remote Event Server*. GSI, Darmstadt, Oktober 2000. http://www-hades.gsi.de/~dbertini/rev.html.

[58] HADES COLLABORATION: *HADES: Status and Detector Performances*. Scientific Report, GSI, 1997.

[59] OUSTERHOUT, JOHN K.: *Tcl and the Tk toolkit*. Addison-Wesley, Reading, Massachusetts, 1994. ISBN 0-201-63337-X.

[60] HADES COLLABORATION: *Commissioning Results from the Dielectron Spectrometer HADES*. Scientific Report, GSI, 2000.

[61] SCHRÖDER, SÖNKE: *Entwicklung und Aufbau eines Systems zur Effizienzkalibration des HADES-RICH*. Diplomarbeit, Universtät Hamburg, Dezember 2000.

[62] LEHNERT, JÖRG. priv. Mitteilung.

[63] BRETZ, THOMAS: *Magnetfeldeigenschaften des Spektrometers HADES*. Diplomarbeit, Technische Universtät München, 1999.

[64] SANCHEZ, MANUEL. priv. Mitteilung.

[65] MATO, P.: *Trigger and Data Acquisition*. CERN Summer Student Lecture; http://cern.web.cern.ch/CERN/Divisions/PE/HRS/Recruitment/Sum_lecturenotes/T&DAQ.pdf, Juli 1999.

[66] STELZER, H.: *The Camac - PC Data Acquisition in Windows 95/98 and NT 4.0 Version 2.0*. Physikalische Nachweisgeräte Dr. H. Stelzer, 64409 Messel, 2000.

[67] RENK, BURKHARD: *Meßdatenerfassung in der Kern- und Teilchenphysik*. Teubner, Stuttgart, 1993.

[68] ESSEL, H. G. und OTHERS: *The New Data Acquisition System at GSI*. IEEE Trans. Nucl. Sci., 43(1):132–135, 1996.

[69] KOLB, B.W. und M. MÜNCH: *HADES raw data format*. WWW-Dokument. http://www-hades.gsi.de/daq/raw_data_format13.html.

Glossar

Crate-Event Datenstruktur, die aus mehreren Sub-Events besteht und in einer Operation zum Event-Builder transportiert werden kann.

Matching-Unit (MU) Elektronische Schaltung, die aus den Informationen der IPUs die Triggerentscheidung des LVL2-Triggers errechnet

Mehrstufiges Triggersystem Mehrere, hintereinander auf den gleichen Datensatz angewandte Ereignisklassifizierungen mit dem Ziel der Ratenreduktion, siehe Abschnitt A.5

Readout-Controller Elektronische Schaltung, die die Auslese der Frontend-Elektronik steuert und die gelesenen Messwerte am VMEbus bereitstellt

Sub-Event Messdaten, die zu einem Ereignis von einem Teilsystem gemessen wurden

Danksagung

Eine Danksagung am Ende einer Arbeit, die in einem so großen Lehrstuhl wie E12 und in einer so vielköpfigen und dynamischen Forschungskollaboration wie HADES durchgeführt wurde, droht, jedes Maß zu sprengen. Im Wissen, dass ich viele wichtige und hilfreiche Begleiter der Arbeit nicht erwähnen kann, sei mein besonderer Dank ausgesprochen an:

Herrn Professor Dr. Hans-Joachim Körner für seine kontinuierliche Motivation und insbesondere sein Verständnis für die vielen anderen Aktivitäten, die die Vervollständigung der Arbeit jedesmal wieder hinausschoben.

Herrn Professor Dr. Paul Kienle für sein mühevolles an sich halten, mir nicht jedesmal vorzuwerfen, wie man nur sein Leben „mit diesem Computerzeug vergeuden“ kann.

Herrn Dr. Jürgen Friese für soviel positive Energie und den politischen Rückhalt in den heißen Phasen des HADES-DAQ-Designs.

Herrn Dr. Wolfgang Koenig für lange, durchwachte Strahlzeitnächte, spannende Diskussionen zu fast jedem Thema (sogar Datenaufnahme) und die Unmengen Zigaretten, die er zum Festlegen einer angemessenen „Zeit bis zum Ende des Tests“ rauchen musste.

Herrn Dr. Josef Homolka für die Rückmeldungen in technischen und Designfragen und insbesondere die Entlastung in vielen, vielen organisatorischen Bereichen.

Herrn Dr. Roman Gernhäuser und Dr. Anton Kastenmüller, die das notwendige Selbstbewusstsein in unsere HADES Keimzelle einbrachten, ohne das man so ein ambitioniertes Projekt gar nicht anfangen kann („Wir sind nicht größenwahnsinnig, wir sind einfach nur die Besten“).

Herrn Benjamin Sailer, der doch lange gebraucht hat, bis er gelernt hat, meine „Das könnte man doch mal so probieren“-Ideen konsequent zu ignorieren.

Frau Laura Fabbietti für die moralische Unterstützung, wenn es um die Entscheidung „mit dem Fahrrad zur Uni oder doch zu faul“ ging.

Dem gesamten Lehrstuhl E12, dessen Mannschaft mir in diesen Jahren doch ein bisschen ans Herz gewachsen ist.

Und natürlich auch dem Rest der HADES Kollaboration, besonders der „Giessen-Crew“, die zwar für einige graue Haare, aber auch für einigen Spaß während der Aufbauphase des HADES-Experiments sorgte (und schließlich, irgendeiner muss ja schuld sein).

Herrn Dr. Andreas Stolz für viele gemeinsam geleerte Flaschen im Dienst (Adelholzener) und nach Dienst (edler Wein).

Nicht zuletzt meinen Kunden, insbesondere den Leuten der cTc GmbH, die „auf der anderen Seite des Terminkalenders“ viel zu erleiden hatten.