Echtzeitanalyse Ethernet-basierter E/E-Architekturen im Automobil[1]

Felix Reimann[†], Andreas Kern[‡], Christian Haubelt[†], Thilo Streichert[‡] und Jürgen Teich[†]
[†]Hardware/Software Co-Design, Lehrstuhl für Informatik 12, Universität Erlangen-Nürnberg, Deutschland
[‡]Daimler AG

Kurzfassung

Ethernet ist aufgrund der hohen Bandbreite eine interessante Alternative zu proprietären Bussen, um ECUs im Automobil zu vernetzen. Es gibt bereits mehrere Ethernet-Derivate, welche Ethernet um Echtzeiteigenschaften erweitern. Diese Arbeit zeigt, dass auch mit Standard-Ethernet Aussagen zur Echtzeitfähigkeit von Systemen möglich sind. Hierzu wird eine Echtzeitanalyse von Standard-Ethernet vorgestellt und die Ergebnisse an Hand einer Fallstudie mit einer Simulation des Systems verglichen.

Abstract

As a result of the higher bandwidths, Ethernet is an interesting alternative to proprietary buses in networking ECUs in automobiles. In order to be able to provide real-time guarantees several Ethernet derivatives have been proposed. This work demonstrates that even with standard Ethernets real-time guarantees can be given. For this reason, a real-time analysis of standard Ethernet is presented and the results based on a case study are compared with a simulation of the system.

1 Einleitung

Elektrische und elektronische Architekturen (E/E-Architekturen) im Automobil bestehen aus einer Vielzahl unterschiedlicher Kommunikationssysteme mit verschiedenen Eigenschaften. Man unterscheidet derzeit zwischen Regel- und Steuerungsnetzwerken, wie beispielsweise LIN [6], CAN [5] oder FlexRay [2], und Multimedianetzwerken, wie MOST [8] oder LVDS, als reine Direktverbindung, die beispielsweise für die Anbindung von Bildschirmen oder Kameras genutzt werden.

Da zukünftig die Anzahl und Komplexität der Anwendungen im Fahrzeug weiter steigen wird, ergeben sich auch höhere Anforderungen an die zugrunde liegende Kommunikationsinfrastruktur. Neben der Forderung nach höheren Bandbreiten nehmen auch die Kommunikationsbeziehungen zwischen den verschiedenen Anwendungen deutlich zu.

Aufgrund dieser Forderungen werden neben den verfügbaren Lösungen aus der Automobilbranche auch alternative Kommunikationssysteme, wie beispielsweise Ethernet [4], als zukünftige Kommunikationstechnologien untersucht. Ethernet ist im Bereich der Consumerelektronik und dem Internet weltweit verbreitet. In diesen Domänen spielen Echtzeitanforderungen lediglich eine untergeordnete Rolle und die Kommunikation wird durch Best-Effort-Kriterien, also schnellstmöglich, ausgeführt. In der Industrieautomatisierung existieren bereits heute über ein Dutzend Ethernet-basierter Lösungen, welche teilweise erweitert wurden, um Echtzeitanforderungen zu unterstützen, siehe auch [15, 14]. AFDX [13] basiert ebenfalls auf Ethernet und ist ein in der Avionik eingesetztes Kommunikationssystem, welches beispielsweise im Airbus 380 oder der Boeing 787 eingesetzt wird. Auch in der Automobilbranche lässt sich ein Trend beobachten, welcher Ethernet als zukünftiges Kommunikationssystem im Automobil favorisiert [1]. Heutige und zukünftige Anwendungsszenarien sind beispielsweise in den Bereichen Rearseat-Infotainment, Onboard-Diagnose oder Laden von E-Fahrzeugen zu finden.

Dies führt bereits heute zu einer guten Verfügbarkeit Ethernet-tauglicher Hard- und Software, die bereits teilweise automobilspezifische Anforderungen erfüllt. Um die praktische Tauglichkeit von Ethernet zeigen zu können, gilt es nun aber vor allem die zeitlichen Anforderungen von fahrzeugkritischen Anwendungen bewerten zu können. Im Folgenden wird eine analytische Methodik vorgestellt, welche es erlaubt, bereits frühzeitig Abschätzungen für Ethernet-vernetzte Funktionen für Best und Worst Case Zielgrößen zu berechnen. Es wird gezeigt, dass selbst bei der Verwendung von Standard-Ethernet eine Aussage zum Echtzeitverhalten möglich ist. Das Echtzeitverhalten einer Fallstudie wird mit simulativ ermittelten Ergebnissen verglichen, um die Genauigkeit der Analyse bewerten zu können.

Das restliche Dokument gliedert sich wie folgt: Kapitel 2 beschreibt die verwendete analytische Methodik und darauf aufbauend die notwendigen Erweiterungen, um auf Ethernet basierende Systeme analysieren zu können. Im anschließenden Kapitel wird die umgesetzte Methodik anhand einer Fallstudie evaluiert. Das letzte Kapitel fasst die gewonnenen Erkenntnisse zusammen und gibt einen Ausblick für weitere Untersuchungen.

[1]Teilweise gefördert durch das BMBF-Projekt *Sicherheit in eingebetteten IP-basierten Systemen* (SEIS) [1].

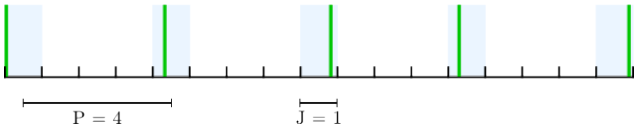


Bild 1: Mögliche Aktivierungen eines Tasks mit Periode $P = 4$ und Jitter $J = 1$.

2 Echtzeitanalyse von Ethernet

In E/E-Architekturen sind für viele Anwendungen Zeitschranken gegeben. Ziel der Echtzeitanalyse ist es, die Ende-zu-Ende-Latenzen im Worst Case zu bestimmen. Durch Simulation sind diese Zeiten nicht ermittelbar, da weder die entsprechenden Aufrufreihenfolgen der Prozesse noch die entsprechenden Eingangsdaten bekannt sind, die zu einer maximalen Verzögerung des verteilten Systems führen, beziehungsweise eine experimentelle Ermittlung der Größen nicht praktikabel ist.

Im Folgenden wird zunächst SymTA/S, ein Werkzeug zur analytischen Bestimmung des Zeitverhaltens verteilter Systeme, eingeführt. Auf dieser Basis wird im Anschluss die Echtzeitanalyse von mittels Ethernet kommunizierender Systeme vorgestellt.

2.1 SymTA/S

SymTA/S [3] ist ein Werkzeug zur analytischen Bestimmung der Ende-zu-Ende-Latenzen der Kommunikationspfade eines verteilten, heterogenen Systems. Im Folgenden werden die für die Echtzeitanalyse von Ethernet-basierten Netzwerken nötigen Teile von SymTA/S beschrieben.

Für jeden Prozess t ist eine maximale Rechenzeit l_t gegeben und jede Nachricht c hat eine Nachrichtenlänge l_c. Die Prozesse und Nachrichten, im Folgenden allgemein als *Tasks* benannt, sind durch sogenannte Ereignisströme verbunden, die festlegen, von welchen Vorgängern der aktuelle Task abhängt. Erst wenn sämtliche Vorgänger eines Tasks ein Datum generiert haben, kann der Task selbst ausgeführt werden. Das Aktivierungsmodell eines Ereignistroms wird durch das Tupel $(P, J) \in \mathbb{R}^+ \times \mathbb{R}^+$ beschrieben. P gibt die Periode an, in der das Ereignis auftritt. Der Jitter J ist die Größe desjenigen Intervalls, um das das Auftreten des Ereignisses um eine periodische Aktivierung schwanken kann, vgl. Abbildung 1.

Sind mehrere Tasks auf eine Ressource gebunden, kann der Task allerdings verdrängt werden. Ob und wie lange ein Task verdrängt wird, hängt von der Schedulingstrategie der Ressource ab. Zur Echtzeitanalyse unter Berücksichtigung des Schedulings stehen unter anderem folgende einfache Strategien zur Verfügung:

- *Prioritätsbasierte präemptive Ablaufplanung*: Jedem Task auf einer Ressource ist genau eine Priorität zugeordnet. Höherpriore Tasks werden vor niederprioren ausgeführt. Läuft ein niederpriorer Task und ein höherpriorer wird laufbereit, wird der niederpriore unterbrochen.

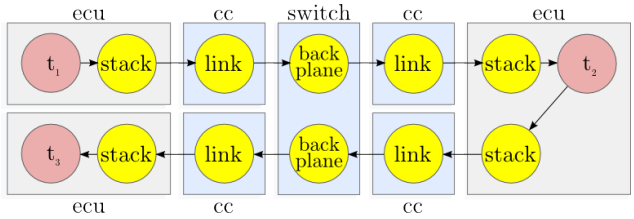


Bild 2: Drei Tasks t_1, t_2, t_3 kommunizieren über Ethernet. Zur Echtzeitanalyse muss neben den Tasks der Softwarestack, die Ethernet-Controller *cc* sowie der Switch modelliert werden. Die Ende-zu-Ende-Latenz wird vom Beginn der Berechnung von t_1 bis zum Ende der Berechnung von t_3 gemessen.

- *Zeitmultiplex*: Jedem Task wird ein Zeitschlitz mit einer bestimmten Länge zugeteilt. Im schlechtesten Fall wird ein Task genau dann laufbereit, wenn sein Zeitschlitz gerade vorbei ist.

- *Round Robin*: Wie beim Zeitmultiplex wird jedem Task ein Zeitschlitz mit einer bestimmten Länge zugeordnet. Statt ungenutzte Zeiten verstreichen zu lassen, wird die Ressource sofort dem nächsten lauffähigen Task zugeteilt.

In SymTA/S wird zunächst jede Ressource einzeln analysiert. Die Aktivierungsmodelle der ausgehenden Ereignisströme berechnen sich aus den eingehenden Ereignisströmen, der Rechenzeit sowie der Ablaufplanung. Die Periode bleibt gleich, während sich der Jitter erhöht. Die maximale Verzögerung wird erzielt, wenn sämtliche höherpriore Tasks gleichzeitig laufbereit sind, vgl. [7].

In verteilten Systemen beeinflusst die Aktivierung der Tasks auf einer Ressource die Aktivierung aller datenabhängigen Tasks im System. Bei zyklischen Abhängigkeiten wie sie beispielsweise im Switch auftreten, muss die Analyse der Antwortzeiten auf jeder Ressource so lange iteriert werden, bis ein Fixpunkt gefunden wird.

2.2 Ethernet-Modellierung

Abbildung 2 zeigt die einzelnen Komponenten, die im Folgenden modelliert werden, um eine von Task t_1 zu Task t_2 über Ethernet versendete Nachricht zu analysieren. Die Nachricht wird zunächst durch den Softwarestack net_{tx} gereicht, der die UDP/IP- und die MAC-Header hinzufügt. Danach wird die Nachricht über einen FIFO auf den Ethernet-Controller *cc* übertragen, der die Pakete versendet. Im Switch werden sie empfangen und ebenfalls in einem FIFO gespeichert. Entsprechend der Ablaufplanung auf dem Switch werden sie in den FIFO des Ausgangsports übertragen und versendet, sobald die Leitung frei ist. Auf der Seite des Empfängers muss die Nachricht durch den Softwarestack entpackt und dann an den empfangenden Task weitergereicht werden. Außerdem zeigt Abbildung 2 eine Kommunikation von t_2 nach t_3 über den selben Switch.

Im Folgenden wird die Modellierung der Ethernet-spezifischen Ressourcen *cc* und *switch* beschrieben.

2.2.1 Modellierung des Switches

Zentrales Element bei Ethernet stellen die Switches dar. Deren Arbeitsweise und Paketscheduling sind im Standard nicht näher spezifiziert, allerdings muss für eine Echtzeitanalyse die konkrete Umsetzung bekannt sein.

Man unterscheidet im Wesentlichen Switches mit Crossbar, die dedizierte Punkt-zu-Punkt-Verbindungen zwischen allen Eingangs- und Ausgangs-FIFOs bieten und Switches mit Backplane-Bus, über den sämtliche Pakete von den Eingangs-FIFOs zu den entsprechenden Ausgangs-FIFOs geschickt werden. Im Folgenden sollen die busbasierten Switches betrachtet werden. Bei diesen kommt als einfache und schnelle Busarbitrierungsstrategie häufig Round Robin zum Einsatz.

Mit diesen Voraussetzungen kann der Backplane-Bus des Switches als zentrale Ressource modelliert werden, über die sämtliche Nachrichten laufen. Die lauffähigen Nachrichten werden mittels Round Robin geplant. Ein Zeitschlitz f_{Switch} entspricht dabei genau der Zeit, die ein komplettes Ethernetpaket benötigt, um übertragen zu werden. Die maximal erlaubte Paketlänge l_{mtu} ist 1538 Byte oder ein beliebiger, kleinerer Wert (festgelegt durch die *Maximum Transmission Unit*, MTU [12]). Für einen Backplane-Bus mit einer gegebenen Bandbreite b_{Switch} ergibt sich damit

$$f_{\text{Switch}} = \frac{l_{\text{mtu}}}{b_{\text{Switch}}}. \tag{1}$$

2.2.2 Modellierung des Ethernet-Controllers

An einem Switch können an jeden Eingangsport verschieden schnelle *cc*s angeschlossen werden. Im Wesentlichen gibt es nach IEEE 802.3 Ethernet-Controller mit $b_{\text{cc}} = 10\,\frac{\text{MBit}}{s}$ bis hin zu $b_{\text{cc}} = 10\,\frac{\text{GBit}}{s}$ Bandbreite. Durch die Vollduplexübertragung können Daten gleichzeitig empfangen und gesendet werden. Entsprechend wird je eine Ressource vom und zum Switch pro ECU modelliert. Werden von einer ECU mehrere unterschiedliche Nachrichten versendet bzw. empfangen, wird eine Nachricht solange verzögert, bis alle anderen Nachrichten versendet bzw. empfangen wurden. Daher wird auch hier Round Robin zur Berechnung des Worst Case als Scheduler verwendet und die Länge der Zeitschlitze, wie beim Switch, aus der maximalen Framelänge und der Bandbreite b_{cc} des gewählten Bausteins berechnet.

2.2.3 Modellierung der Nachrichten

Ein Ethernetpaket hat inklusive Präambel und *Inter Frame Gap* (IFG) eine maximale Länge von 1538 Byte, vgl. [4]. Aus dem Overhead der Kopfdaten von 38 Byte pro Paket und einer minimalen Nutzlast von 46 Byte ergibt sich beim MAC-Protokoll für n Byte Nutzdaten eine effektive Nachrichtenlänge l_{mac} von:

$$\begin{aligned} l_{\text{mac}}(n) = {} & 38 \left\lceil \tfrac{n}{1500} \right\rceil + n \\ & + \max\{0\ ;\ 46 - (n \bmod 1500)\} \end{aligned} \tag{2}$$

Zusätzlich zur Nutzlast n kommen also die Headerdaten pro Paket hinzu sowie, falls ein Paket weniger als 46 Byte Nutzdaten beeinhaltet, ein entsprechend langes Füllfeld. Werden größere Daten übertragen, wird die Nachricht durch die IP-Schicht in mehrere Ethernetpakete zerlegt. Für ein IPv4-Paket kommen zusätzlich noch 20 Byte IP-Kopfdaten pro Ethernetpaket und 8 Byte UDP-Kopfdaten pro IP-Paket hinzu, vgl. [11, 10]. Damit ergibt sich für $n \leq 65\,507$ Byte Nutzdaten eine effektive Nachrichtenlänge l_{udp} von:

$$\begin{aligned} l_{\text{udp}}(n) = {} & 58 \left\lceil \tfrac{n+8}{1472} \right\rceil + n + 8 \\ & + \max\{0\ ;\ 26 - ((n+8) \bmod 1472)\} \end{aligned} \tag{3}$$

Betrachtet man jedes Paket einzeln, erhält man eine realitätsnahe Modellierung. So können die ersten Pakete einer Nachricht bereits durch den Switch übertragen werden, während andere Teile noch durch den oftmals langsameren *cc* verzögert werden. Hierzu modelliert man für eine Nachricht der Länge n Byte $\left\lceil \frac{l_{\text{udp}}(n)}{1538} \right\rceil$ einzelne Nachrichtenströme, die vom Softwarestack des Sendetasks über die *cc*s und dem Switch zum Softwarestack des Zieltasks geroutet werden.

Zur Modellierung von großen Nachrichten, wie sie beispielsweise bei Multimedia-Anwendungen auftreten, ist aufgrund der großen Anzahl zu betrachtender Pakete die Analyse zu komplex. Deshalb ist es alternativ möglich, eine große Nachricht durch einen einzelnen Nachrichtenstrom zu modellieren, der auf die selben Ressourcen wie bei obigem Ansatz gebunden wird. Die Länge des Nachrichtenstroms berechnet sich aus Formel (3). Da die Länge eines Zeitschlitzes im Switch und den Controllern entsprechend der maximal möglichen Größe eines einzelnen Ethernetpakets gewählt wurde (vgl. Formel (1)), können nach der Übermittlung eines Pakets jeweils ein Paket der im Worst Case gleichzeitig am Switch einlaufenden Nachrichten übertragen werden. Durch diese Art der Modellierung wird die Überapproximation der Worst Case-Analyse zwar vergrößert, allerdings lassen sich nur so auch sehr große Nachrichten analysieren. Die beiden Arten der Modellierung können gleichzeitig für verschiedene Nachrichten genutzt werden, um für kleinere Nachrichten bessere Worst Case Werte zu erhalten.

3 Fallstudie

Für die Bewertung des Modells zur Echtzeitanalyse von Ethernet-basierten Systemen wird im Folgenden eine Fallstudie auf Basis eines mit Ethernet vernetzten Multikamerasystems beschrieben. Hierbei werden die in SymTA/S ermittelten analytischen Zielgrößen mit den Ergebnissen eines adaptierten diskreten ereignisgesteuerten Simulators [9] verglichen.

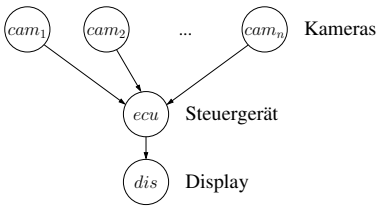


Bild 3: Der Datenabhängigkeitsgraph des Multikamerasystems. Die Kameras senden Daten an das Steuergerät, welches wiederum Daten an das Display schickt.

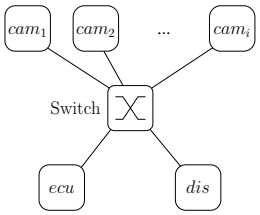
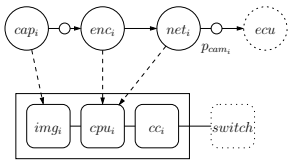
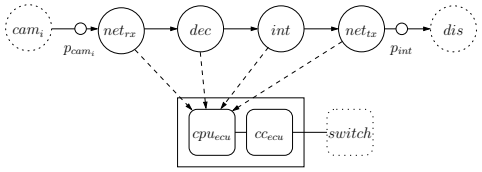


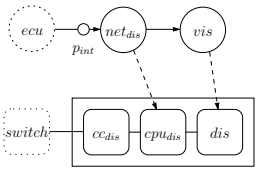
Bild 4: Die Architektur des Multikamerasystems. Die Kameras, das Steuergerät und das Display kommunizieren über einen zentralen Switch miteinander.



(a) Kamera



(b) Steuergerät



(c) Display

Bild 5: Die Komponenten Kamera, Steuergerät und Display.

3.1 Systemmodell

Das betrachtete Multikamerasystem besitzt folgendes Funktionsprinzip: Mehrere am Fahrzeug installierte Videokameras cam_i erfassen definierte Umgebungsbereiche des Fahrzeugs. Diese werden in einem zentralen Steuergerät aggregiert, zusammengefasst und über ein Display im Fahrzeug angezeigt. Ziel dieses bilddarstellenden Fahrerassistenzsystems ist die Unterstützung des Fahrers bei der Erfassung von Objekten, welche sich in der Umgebung des Autos befinden. Im Folgenden werden die Architektur, die benötigten Tasks, deren Datenabhängigkeiten und die verwendeten Parametersätze beschrieben.

Abbildung 3 zeigt den zugrunde liegenden Datenabhängigkeitsgraphen. Die Kameras erfassen hierbei Einzelbilder und leiten diese paketiert mittels Ethernet an das Steuergerät weiter. Aus den eingehenden Paketen der einzelnen Kameras werden im Steuergerät die Einzelbilder rekonstruiert. Liegen alle zusammengehörenden Einzelbilder der angeschlossenen Kameras vor, wird aus den Einzelbildern ein Gesamtbild generiert, dieses wiederum paketiert und an das Display gesendet. Das Display rekonstruiert das Bild und stellt es auf dem Bildschirm dar.

Die Architektur ist in Abbildung 4 dargestellt und basiert auf einer sternförmigen Vernetzung mittels eines zentral angeordneten Switches. Alle Komponenten können somit ausschließlich über den Switch miteinander kommunizieren. Der Switch selbst besitzt bis zu fünf Fast Ethernet Anschlüsse mit jeweils 100 $\frac{\text{MBit}}{s}$ Bandbreite und einen Gigabit Ethernet Anschluss, welcher eine maximale Datenrate von 1 $\frac{\text{GBit}}{s}$ besitzt. Der Hochgeschwindigkeitsbus im Switch besitzt eine Datenrate von 2 $\frac{\text{GBit}}{s}$. Im Folgenden wird die Funktionsweise und der Aufbau jeder Komponente beschrieben.

Das Kameramodul nimmt Einzelbilder auf, komprimiert diese und generiert im Anschluss daran UDP-Pakete, welche mittels Ethernet an das Steuergerät versendet werden. Abbildung 5a zeigt den Aufbau eines solchen Kameramoduls cam_i, welches aus den Ressourcen Imager img_i, Prozessor cpu_i und Ethernet-Controller cc_i besteht. Der Imager nimmt jeweils ein Einzelbild auf und sendet die generierten Daten mittels eines parallelen Interfaces direkt an den Prozessor. Nach jedem Einzelbild startet ein Hardware-Interrupt die Verarbeitung auf dem Prozessor. Dieser verarbeitet die Prozesse Encoder enc_i, welcher die Komprimierung eines Einzelbildes übernimmt, und den Softwarestack net_i, der die komprimierten Einzelbilder in Pakete aufteilt und die benötigten Paketköpfe generiert. Die Ausführungszeit des Softwarestacks unterscheidet sich je nach Implementierung und ausführender ECU. Anhand von Messungen an einer Referenzimplementierung, kann für den Lese- und Schreibstack folgende Ausführungszeit für Nachrichten der Länge n Byte verwendet werden:

$$l_{net_i}(n) = t_{fix} + t_{var}\,(n)\ \text{ns} \tag{4}$$

Die beiden Prozesse werden mittels präemptivem, prioritätsbasiertem Scheduling auf dem Prozessor ausgeführt. Nachdem net_i ein Paket generiert hat, wird dieses an den Ethernet-Controller übergeben, welcher das Paket an den Switch sendet.

Die Verarbeitung der Kamerabilder erfolgt, wie oben beschrieben, zentral im Steuergerät. Der Aufbau des Steuergeräts ist in Abbildung 5b dargestellt. Es besteht aus den beiden Ressourcen Prozessor cpu_{ecu} und dem Ethernet-Controller cc_{ecu}. Auf dem Prozessor werden

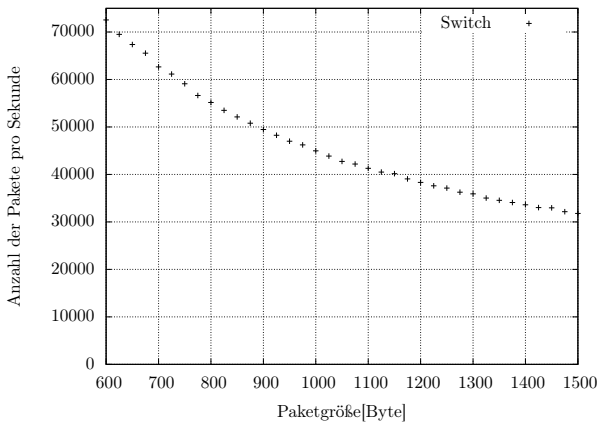


Bild 6: Die Anzahl der versendeten Pakete pro Sekunde für unterschiedliche Paketgrößen.

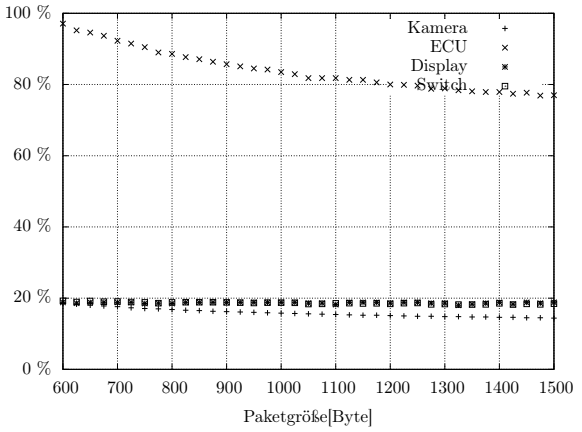


Bild 7: Die Ressourceauslastungen der einzelnen Komponenten für unterschiedliche Paketgrößen.

die Prozesse net_{rx}, *dec*, *int* und net_{tx} ebenfalls mittels präemptivem, prioritätsbasiertem Scheduling ausgeführt. Der Prozess net_{rx} entpackt empfange Pakete von Kameras und generiert daraus die aufgenommenen Einzelbilder. Liegt ein Einzelbild einer Kamera vor, dekomprimiert der Prozess *dec* das Einzelbild, um auf die Rohdaten zugreifen zu können. Liegen alle zusammengehörenden Einzelbilder der angeschlossenen Kameras decodiert vor, beginnt die Intergration der Bilder durch den Prozess *int*. Anschließend wird das generierte Bild an den Prozess net_{tx} übergeben, welcher das generierte Bild in Pakete zerlegt, die Paketköpfe erzeugt und an den Ethernet-Controller sendet, welcher diese an das Display schickt.

Das Display übernimmt die Darstellung der generierten Einzelbilder des Steuergerätes. Abbildung 5c zeigt den Aufbau des Displays, welches aus den Ressourcen Ethernet-Controller cc_{dis}, Prozessor cpu_{dis} und Display *dis* besteht. Der Prozess net_{dis} verarbeitet dabei eingehende Pakete vom Ethernet-Controller und erzeugt daraus wieder das Gesamtbild, welches dann auf dem Display angezeigt wird.

3.2 Ergebnisse

Bei der Evaluierung des Multikamerasystems wurden die Anzahl der versendeten Ethernet-Pakete pro Sekunde, die Ressourceauslastungen und die Ende-zu-Ende-Latenz des Systems für unterschiedliche Ethernet-Paketgrößen ermittelt. Abbildung 6 zeigt den simulativ ermittelten Verlauf der Anzahl der zu verarbeitenden Pakete pro Sekunde in Abhängigkeit zur maximalen Framelänge l_{mtu}. Je kleiner l_{mtu}, desto mehr Pakete werden benötigt um die Videodaten zu übermitteln. Da der Softwarestack für jedes Paket durchlaufen werden muss, hat l_{mtu} auch Einfluss auf die Ressourceauslastungen, vgl. Formel (4). Abbildung 7 zeigt die jeweiligen Auslastungen der Ressourcen cam_i, *switch*, *ecu* und *dis*. Außerdem ist zu sehen, dass das Steuergerät mit abnehmender Paketgröße in die Sättigung läuft, somit konnten nur Paketgrößen bis 600 Byte evaluiert

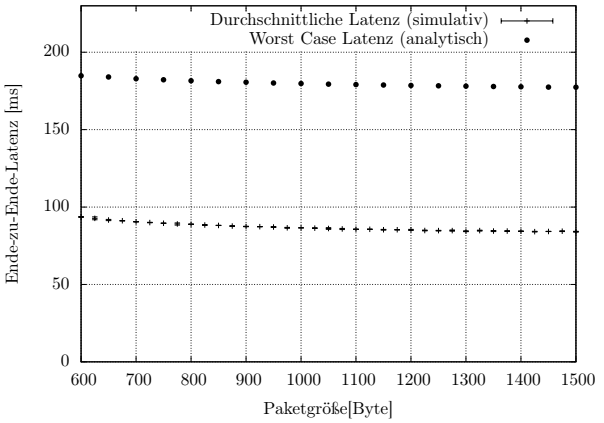


Bild 8: Die gemessenen und analysierten Ende-zu-Ende-Latenzen für unterschiedliche Paketgrößen.

werden.

Bild 8 zeigt die Evaluierung der Ende-zu-Ende-Latenzen, wobei zu erkennen ist, dass die analytisch ermittelten Zeiten um den Faktor 2 größer sind als die simulativ ermittelten Durchschnittszeiten. Dies liegt zum einen an der Tatsache, dass es durch Simulation kaum möglich ist, den schlimmstmöglichen Fall zu erreichen. Andererseits führt die hier vorgestellte Analyse zu einer Überapproximation, da eine Nachricht stets vollständig an einer Ressource empfangen werden muss, bevor der erste Frame gesendet wird. Dies ist aus Abbildung 9 ersichtlich: Die Nachricht m auf dem Switch wird korrekterweise solange verzögert, bis alle anderen Nachrichten übermittelt wurden. Allerdings könnte die Übertragung auf dem Ethernet-Controller bereits beginnen, sobald das erste Paket der Nachricht m am Switch übertragen wurde (schwarzer Pfeil).

Um dies zu analysieren, wäre es nötig, den genaueren paketbasierten Ansatz (vgl. Abschnitt 2.2.3) zu verwenden. Auf Grund der großen Anzahl an Paketen (über 200 pro Frame und Bildquelle) im Vergleich zur betrachtenden Periode (bestimmt durch die zu erzielenden 30 fps) ist dieser feingranulare Ansatz mit ei-

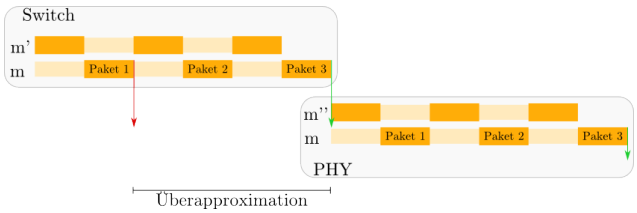


Bild 9: Überapproximation des Worst Case bei der Übertragung einer drei Pakete langen Nachricht vom *switch* zum *cc*.

nem Aktivierungsmodell, das ausschließlich auf Periode und Jitter zur Modellierung setzt, noch nicht effizient durchführbar.

4 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Modell zur Echtzeitanalyse von auf Standard-Ethernet basierten Systemen am Beispiel von SymTA/S entwickelt. Hierzu wurden die Ethernet-Komponenten *cc* und *switch* modelliert, so dass eine Analyse von Ethernet-Netzwerken auch bei großem Nachrichtenaufkommen möglich ist. Ein Fallbeispiel für ein Ethernet-basiertes Multikamerasystem wurde untersucht und mit Ergebnissen einer Simulation verglichen. Die resultierende Überapproximation entsteht durch die Annahme, dass bei der Analyse die Weiterverarbeitung des Frames erst beginnt, wenn alle Datenpakete eines Frames übertragen worden sind. Diese Vereinfachung war notwenig, um Ergebnisse in akzeptabler Zeit durch die Analyse berechnen zu können. Mittels Simulation kann dagegen nicht gewährleistet werden, dass der Worst Case im Simulationslauf auftritt.

Ein nächster Schritt ist die Integration mehrerer Systemfunktionen mit Echtzeitanforderungen über eine gemeinsame Ethernet-basierte Infrastruktur. Da hier die gegenseitige Beeinflussung der Anwendungen auf Netzwerkebene Auswirkungen auf die Echtzeitfähigkeit haben kann, sollen unterschiedliche Mechanismen zur Priorisierung von Ethernet-Paketen untersucht und verglichen werden.

Literatur

[1] E|ENOVA Innovationsallianz Automobilelektronik: *Sicherheit in IP-basierten Systemen*. `http://www.eenova.de/projekte/seis`

[2] FlexRay Consortium GbR (Hrsg.): *FlexRay Communications System – Protocol Specification*. 2.1 Rev. A. Industriestraße 6, 70565 Stuttgart, Germany: FlexRay Consortium GbR, Dezember 2005

[3] Henia, Rafik ; Hamann, Arne ; Jersak, Marek ; Racu, Razvan ; Richter, Kai ; Ernst, Rolf: System Level Performance Analysis – the SymTA/S Approach. In: *IEE proceedings. Computers and digital techniques* 152 (2005), S. 148–166. – ISSN 1350–2387

[4] The Institute of Electrical and Electronics Engineers, Inc. (Hrsg.): *Information technology – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*. IEEE Std 802.3, 2000. 3 Park Avenue, New York, USA: The Institute of Electrical and Electronics Engineers, Inc., Dezember 2000

[5] ISO und IEC: *ISO 11898: Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signallingrea network (CAN) for high-speed communication*. ISO 11898-1:2003. November 1993

[6] LIN Administration (Hrsg.): *LIN Specification Package*. 2.1. Bernhard-Wicki-Straße 3, 80636 Munich, Germany: LIN Administration, November 2006. `http://www.lin-subbus.org`

[7] Liu., C. L. ; Layland, J. W.: Scheduling algorithms for multiprogramming in a hard real-time environment. (1973), S. 46–61

[8] MOST Cooperation (Hrsg.): *Media Oriented System Transport – MOST Specification*. Rev. 3.0. 76185 Karlsruhe: MOST Cooperation, Mai 2008

[9] NS 2: *The Network Simulator 2*. `http://www.isi.edu/nsnam/ns/`

[10] Postel, J.: *User Datagram Protocol*. RFC 768. 4676 Admiralty Way, Marina del Rey, California 90291: Information Sciences Institute, University of Southern California, August 1980

[11] Postel, J.: *Internet Protocol – DARPA Internet Program – Protocol Specification*. RFC 791. 4676 Admiralty Way, Marina del Rey, California 90291: Information Sciences Institute, University of Southern California, September 1981

[12] Postel, J.: *The TCP Maximum Segment Size and Related Topics*. RFC 879. 4676 Admiralty Way, Marina del Rey, California 90291: Information Sciences Institute, University of Southern California, November 1983

[13] SYSGO AG: *Avionics Full DupleX Switched Ethernet*. `http://www.arnic.com`

[14] EtherCAT Technology Group: *EtherCAT*. `http://www.ethercat.org`

[15] TTTech Computertechnik AG: *TTEthernet*. `http://www.tttech.com/solutions/ttethernet/`. Version: 2008