Modellierung des Zeitverhaltens netzbasierter Automatisierungssysteme

Jürgen Greifeneder und Georg Frey, Technische Universität Kaiserslautern

Die fortschreitende Verbreitung von Ethernet-basierten Strukturen mit dezentralen und verteilten Anwendungen in der Automatisierung führt zu den so genannten netzbasierten Automatisierungssystemen (NAS). Diese sind zwar in Anschaffung und Betrieb kostengünstiger, moderner und flexibler als herkömmliche Strukturen, weisen jedoch nicht-deterministische Verzögerungen auf. Die genaue Analyse der resultierenden Antwortzeiten ist somit nicht nur Voraussetzung für den verantwortungsbewussten Einsatz dieser Technologie sondern ermöglicht es auch, bereits im Vorfeld von Umstrukturierungen oder Erweiterungen, Fragen der Verlässlichkeit zu klären. In diesem ersten von zwei Beiträgen wird hierfür zunächst die für die speziellen Bedürfnisse der Strukturbeschreibung von netzbasierten Automatisierungssystemen entwickelte Modellierungssprache DesLaNAS vorgestellt und auf ein einführendes Beispiel angewendet. Im zweiten Beitrag wird darauf aufbauend gezeigt, welchen Einfluss die einzelnen Systemkomponenten (SPS, Netzwerk, I/O-Karten) sowie netzbedingte Verhaltensmodi wie Synchronisation und die gemeinsame Nutzung von Ressourcen auf die Antwortzeiten des Gesamtsystems haben. Zur Analyse selbst wird die wahrscheinlichkeitsbasierte Modellverifikation (PMC) angewendet.

Netzbasierte Automatisierungssysteme / Verteilte Steuerungen / Netzwerke / Antwortzeit / Wahrscheinlichkeitsbasierte Modellverifikation / Synchronisation zyklischer Prozesse

Modeling the timed behavior of Networked Automation Systems

The onward dispersion of Ethernet based structures with decentralized and distributed applications in automation leads towards Networked Automation Systems (NAS). The new modern structures are less expensive and at the same time more flexible than classical ones. However, they induce non-deterministic delays. Therefore, the detailed analysis of the resulting response times is not only prerequisite for the responsible use of this technology; it also enables to check dependability properties prior to changes or expansions of the system. In this first in the series of two papers, a new modeling language for NAS – DesLaNAS – is introduced and applied to an introductory example. The second paper discusses the influence of single components in an NAS (PLC, network, I/O-boards) as well as of network induced behavior like synchronization and resource sharing on the response time in detail. In the scope of the presented work Probabilistic Model Checking (PMC) is employed for the analysis.

Networked Automation Systems / Distributed control systems / Networks / Response time / Probabilistic model checking / Synchronization of cyclic processes

Einleitung

Moderne Automatisierungssysteme bestehen aus einem oder mehreren Controllern, einer (in der Regel größeren) Anzahl an Sensoren und Stellgliedern sowie einem diese einzelnen Komponenten miteinander verbindenden Netzwerk inklusive der zugehörigen Netzwerk-I/Os. Diese Anordnung – im Fachjargon als *netzbasierte AT-Systeme* (Networked Automation Systems, NAS) bekannt – hat vielfältige Vorteile. Hierzu gehören z.B. die gemeinsame Nutzung von Ressourcen (Sensoren, Verkabelung), die Möglichkeit des Querzugriffs aber auch die Möglichkeit der dezentralen örtlichen Trennung von Reglern untereinander einerseits als auch von Reglern und Prozess andererseits. Verwendet man für die Netzwerkstruktur Ethernet, kommen noch stetig sinkende Preise für die Hardware, eine nachhaltige Verbesserung von Qualität und Umfang der angebotenen Technologie und die Möglichkeit Kommunikationsschranken zu überwinden hinzu.

Die durch die Vernetzung einerseits und die dezentrale zyklusbasierte Ausführung andererseits hervorgerufene Verschmelzung von Regelungstechnik, Kommunikation und Berechnung lässt ein System entstehen, welches eine Superposition aus konstanten und zyklischen Verzögerungen aufweist. Stochastische Ausfälle von Komponenten oder Fehler bei der Kommunikation führen zu weiteren Verzögerungen. Der Zeit kommt hierbei doppelte Bedeutung zu. Auf der einen Seite ist sie als Inputvariable unerlässlich, auf der

anderen Seite stellt sie neben der reinen Funktionsanalyse die wichtigste Säule der Verlässlichkeitsanalyse dar. Um diese angehen zu können, ist daher eine Methodik erforderlich, die Zeiten, stochastische Verteilungen und deterministische Reihenfolgen aufweisende Modellstrukturen verarbeiten und zeit- sowie wahrscheinlichkeitsbasierte Antworten liefern kann. Dies wird detailliert in Abschnitt 2 beschrieben. Abschnitt 3 führt darauf aufbauend schrittweise die gewählte Beschreibungssprache DesLaNAS ein, welche Grundlage des weiteren (in Abschnitt 4 kurz umrissenen) Modellierungsprozesses ist. Aufbauend auf generischen Modulen erstreckt sich die Modellierung eines konkreten Systems auf die Instanziierung und Initialisierung notwendiger Modulblöcke sowie der Implementierung des sog. Signal-Trackings (Abschnitt 4) und sollte daher von einem Ingenieur aus der Automatisierungstechnik ausgeführt werden können.

Tabelle 1: Anforderungen an eine Modellierungssprache für NAS.

Darstellung	Modular	Einzelne Komponententypen kommen mehrfach vor. Hieraus folgt direkt dass die einzelnen Module parametrisierbar sein müssen
	Ingenieursintuitiv	Auch ohne Kenntnisse in PMC soll der Ingenieur Modelle erstellen können.
	Grafisch	Vereinfacht die Modellierung
Eigenschaften	Keine Nebenläufigkeit	Es ist immer nur ein Gesamtzustand aktiv. Allerdings hat jedes Modul seinen vom restlichen System mehr oder minder unabhängigen Teilprozess
	Determinismus	Automatisierungssysteme sollten stets deterministisch ausgelegt sein
	Zeitkontinuität	Modelle sind über kontinuierlicher Zeit beschrieben (es gibt nur eine globale Uhr)
	Diskretisierbarkeit	Für die Nutzung in PMC sind diskrete Modelle erforderlich
Modellierungs-Möglichkeiten	Abbildung von	➢ Zeitabläufen beliebiger Natur ➢ Abhängigkeiten zwischen den Modulen ➢ statistisch ermitteltem Verhalten ➢ mittels stochastischer Elemente abstrahierter Vorgänge, wie z.B. die Zeitverteilung eines bekannten (deterministischen) Prozesses, wie sie beispielsweise bei einer fehlerfreien Multi-User-Netzübertragung auftritt
	Formulierung von	➢ Eingangssignalen ➢ Abläufen ➢ Anfangsbedingungen ➢ zu verifizierendem Verhalten

Die formale Analyse derartiger Systeme eröffnet eine große Zahl von Anwendungsmöglichkeiten. Insbesondere ist es hiermit möglich, die Vor- und Nachteile unterschiedlicher Systemarchitekturen quantitativ zu bewerten, Qualitätsansprüche analytisch zu optimieren, System-Rekonfigurationen offline zu bewerten sowie die Folgen und Grenzen des gerade in modernen Ethernetstrukturen einfachen Hinzuklemmens weiterer Komponenten abzuschätzen, deren Auswirkungen ohne entsprechende Methoden nicht oder nur schwer absehbar sind.

Anforderungen an die Beschreibung von NAS und bisherige Ansätze

Die bekannten Verfahren zur Bestimmung von Verzögerungszeiten unterscheidet man in messtechnische, statische, simulative und verifikationsformale Ansätze. Messtechnische Ansätze lassen sich gut zur Validierung und insbesondere zur Parametrierung anderer Ansätze nutzen (Beispiele hierfür: [1, 2]). Statische Analysemethoden (z. B. [3, 4]) liefern Aussagen, die aufgrund der Konfiguration des Systems getroffen werden können. Das heißt, alle Prozesse im System werden quasi eingefroren und durch charakteristische Werte repräsentiert, wie z.. den Minimal-, den Maximal- oder den Mittelwert. Die simulativen Ansätze (z.B. [5, 6]) unterscheiden sich von den statischen insbesondere durch die erweiterten Abbildungsmöglichkeiten: Dynamik und zufällige Entscheidungen lassen sich mit Hilfe von Simulationen nachbilden. Grundsätzlich haben alle simulativen Ansätze jedoch das Problem, dass die so ermittelten Ergebnisse keine Vollständigkeitsgarantie aufweisen; das heißt das aus der Überlagerung einer großen Anzahl von Simulationsläufen gewonnene Bild kann nur für sehr lange Simulationsläufe als in Bezug auf die korrekte Lösung asymptotisch, nicht jedoch als exakt betrachtet werden. Hierbei ist die benötigte Anzahl der Simulationsläufe eine statistische Funktion der kleinsten theoretisch auftretenden Wahrscheinlichkeit. Tatsächlich treten jedoch gerade in NAS durch das dynamische Zusammenspiel vieler Komponenten und die Existenz von Ausfällen, Interferenzen und Störungen eine Vielzahl von Situationen mit kleinen Auftrittswahrscheinlichkeiten auf.

Verifikationsformale Ansätze (z. B. [7]) haben den Anspruch, alle praktisch möglichen Evolutionen des Systems abzudecken. Die Modellverifikation (Model-Checking, MC) ist ein formales Verfahren, bei dem ein formales Modell (in der Regel eine Automatendarstellung) und eine abstrahierte Eigenschaftsbeschreibungen einem Algorithmus übergeben werden, welcher durch Fusion dieser beiden (jeweils auf Kripke-Strukturen abgebildeten) Eingangsgrößen feststellen kann, ob die geforderten Eigenschaften vom Systemmodell erfüllt werden. Die wahrscheinlichkeitsbasierte Modellverifikation (Probabilistic Model-Checking, PMC, [8, 9, 10]) ist eine Fortentwicklung des MC, welche statt der Information erfüllt bzw. nicht erfüllt eine Erfüllungswahrscheinlichkeit zurückgibt. Hierzu ist neben probabilistischen Modellen (i. d. R. probabilistische zeitbewertete Automaten, PTA), auch eine spezielle Abfragelogik [11] notwendig, welche zur Formulierung der interessierenden Systemeigenschaften Verwendung findet. Näheres hierzu vgl. z. B. in [9].

Die korrekte Modellierung des mit physikalischen Vorgängen gekoppelten Automatisierungssystems erfordert die Berücksichtigung der Kontinuität der zugrunde liegenden Prozesse und somit die Verwendung einer kontinuierlichen Zeitbasis zur Beschreibung der Eintrittszeitpunkte von Zustandsänderungen. Zur Bestimmung einer Modellierungsform welche alle möglichen Verhaltensmodi eines NAS beschreiben kann, ist es wichtig, zunächst eine Beschreibungssprache zu definieren, welche die strukturbasierte

Denkweise des modellierenden Ingenieurs widerspiegelt (Description Language for Networked Automation Systems, DesLaNAS). Tabelle 1 zeigt zusammenfassend die Anforderungen, welchen eine Modellierungssprache für NAS genügen sollte.

Die Modellierungssprache DesLaNAS

DesLaNAS ist eine grafische Beschreibungsform für PTA, die aufgrund ihrer Strukturierungsformalien PTA modularisiert und infolge der Regeln für die Zustandsverknüpfung PTA auf eine Untermenge einschränkt, welche zum Einen zur Beschreibung von NAS ausreicht und zum Anderen für die Umsetzung in Analysetools (Diskretisierbarkeit, Determinismus) geeignet ist. DesLaNAS stellt also eine universelle Beschreibungssprache für NAS dar, mittels derer alle Verhaltensmodi eines NAS abgebildet werden können. Neben den reinen Strukturblöcken sind hierbei insbesondere die Verwendung von Signal-Tracking Modulen zur Erfassung von Abläufen und Eingangssignalen als auch die in textueller Form vorgenommene Parametrierbarkeit zu nennen.

Es wird eine modulare Modellierung gewählt, da einerseits in einem NAS viele Komponenten mehrfach auftreten (z. B. Switche, I/O-Karten, SPSen) und andererseits die direkte Erstellung eines alle möglichen Zustandsänderungen und Zustände des Gesamtsystems berücksichtigenden formalen Modells nicht nur sehr fehleranfällig, sondern vor allem auch äußerst zeitaufwändig wäre. Modular bedeutet hierbei, dass das Gesamtsystem aus zuvor definierten generischen Modulen zusammengesetzt wird, welche im jeweiligen Modell instanziiert und parametrisiert werden müssen. Das interne Verhalten der Module wird durch eine an die speziellen Bedürfnisse von NAS angepasste Form von probabilistischen zeitbewerteten Automaten (PTA) repräsentiert [12]. Von extern gestalten sich diese Modulblöcke als E/A-Automaten [13]. Den Einzelautomaten ist es einerseits gestattet, Zustandsübergänge asynchron auszuführen und andererseits von den anderen Einzelautomaten verschiedene Ein-/Ausgangsvariablen aufzuweisen (also Ereignisse aufzunehmen, bzw. zu generieren, die nicht in anderen Automaten vorkommen). Die zugrunde liegende Modellierung ist somit funktionsblockorientiert.

Wie Bild 1 verdeutlicht, bildet das so gewonnene grafische Modell die Grundlage für einen dreistufigen Transformationsprozess, bei dem zunächst eine Abbildung des in der Modellierungssprache verfassten grafischen Modells auf eine auf PTA basierende Automatenstruktur stattfindet. Dieses Modell wird anschließend in einen diskreten Automaten überführt, da die kontinuierliche Beschreibung aufgrund der durch Rechenleistung und Hauptspeicher gegebenen Einschränkungen nicht direkt im Rahmen der PMC verwendet werden kann. Aus dieser diskreten Beschreibung lässt sich der für die Durchführung des PMCs mittels des Verifikationswerkzeuges PRISM [14] erforderliche Programmcode dann direkt gewinnen. Während die beiden Automatenstrukturen (kontinuierlich und diskret) bereits in [12] diskutiert wurden, soll im Rahmen dieser Veröffentlichung genauer auf die Beschreibungssprache und damit einhergehend auf die in einem NAS auftretenden Verhaltensmodi eingegangen werden.

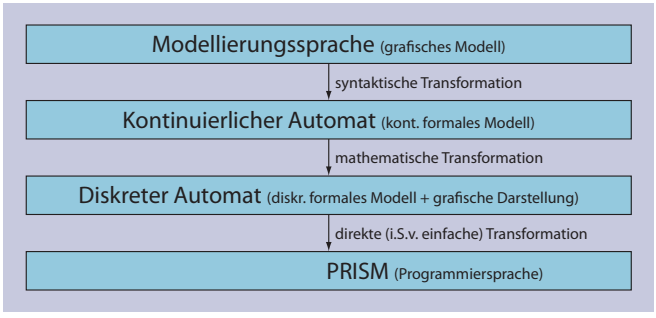


Bild 1: Entwurfsprozess.

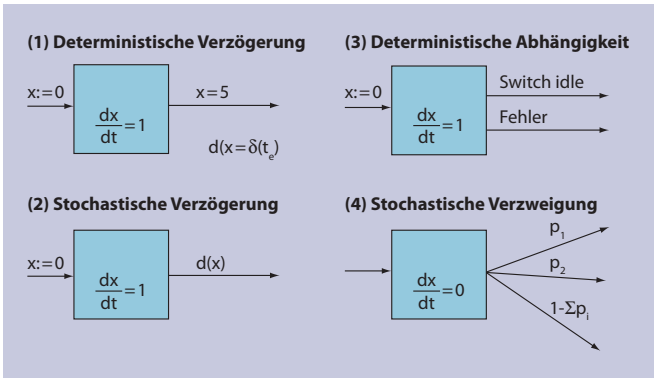


Bild 2: Grundtypen von Zustandsübergängen.

Selbstverständlich kann DesLaNAS nicht nur genutzt werden, um PRISM-Modelle zu erstellen, sondern bildet eine von der konkreten Softwareplattform und Analysemethodik unabhängige Beschreibungsform, welche dann z. B. auf ein PMC-Modell abgebildet werden kann [12].

Die in einem NAS auftretenden Zustandsübergänge lassen sich ganz allgemein auf vier Grundtypen zurück führen (Bild 2).

(1) *Deterministische Verzögerung:* Hierunter versteht man eine fest vorgegebene Zeitspanne, nach deren Ablauf der Zustandsübergang ermöglicht (und sofort durchgeführt) wird. Beispielhaft sei eine gleich bleibende Bearbeitungszeit oder die maximal zulässige Betriebszeit genannt. Ein Zustand kann (nur) genau einen solchen Übergang aufweisen, da bei mehreren stets nur der mit der kleinsten Zeitbedingung ermöglicht würde (das Auftreten zweier identischer Zeitbedingungen würde der Forderung nach deterministischem Verhalten widersprechen).

(2) Stochastische Verzögerungen werden mit Hilfe von Wahrscheinlichkeitsdichtefunktionen d(x) beschrieben. Das Integral über d(x) von $x = 0$ bis $x = t$ repräsentiert hierbei die Wahrscheinlichkeit, dass der Übergang innerhalb dieser Zeitspanne ermöglicht und durchgeführt wurde. Diese Form des Überganges findet sich überall dort, wo mehrere Nutzer eine Komponente teilen bzw. das genaue (zwar in der Regel deterministische, dafür aber komplexe) Verhalten einer Komponente mittels einer Übergangsfunktion abstrahiert wird. Beispielhaft sei hierbei die Übertragungszeit durch ein Netzwerk

genannt. Wie bereits im Fall deterministischer Verzögerungen ist auch für den stochastischen Fall lediglich genau ein Übergang je Zustand erlaubt. Anmerkung: Jede deterministische Verzögerung (Fall 1) kann als $d(x) = \delta(te)$ geschrieben werden, worin δ das Kroneckersymbol (Dirac-Impuls) und te den Aktivierungszeitpunkt repräsentieren. Da eine solche Schreibweise jedoch etwas ungewohnt ist, wird diese Umwandlung durch den Transformationsprozess (vgl. Bild 1) automatisch durchgeführt.

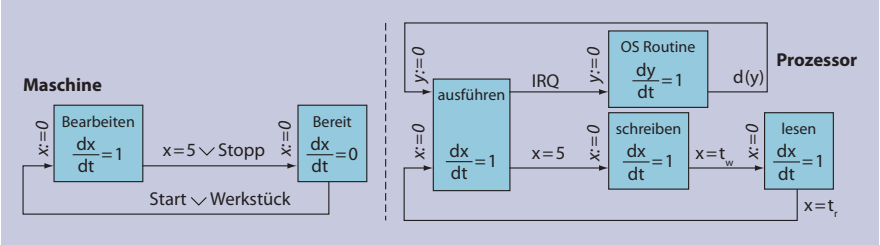


Bild 3: Beispiele für Zustände mit gemischten Übergangstypen.

(3) *Deterministische Abhängigkeit*: In diesem Fall ist die Aktivierung des Übergangs an den Eintritt einer vorgegebenen Situation gekoppelt. Wird eine Bedingung angegeben, so ist hiermit das beim Auslösen des zugehörigen Übergangs erzeugte Ereignis gemeint. Im Gegensatz zu den vorherigen beiden Grundtypen kann ein Zustand mehrere abgehende Übergänge dieses Typs aufweisen, wobei allerdings aufgrund der Ereignischarakteristik niemals zwei Bedingungen zur selben Zeit erfüllt sein können. Hieraus folgt direkt, dass UND-Verknüpfungen innerhalb von Bedingungen nicht möglich sind (siehe unten). Dieser Übergangstyp tritt in einem NAS z. B. beim Schreiben/Lesen der SPS oder dem Senden einer Anfrage durch die SPS-I/O auf.

(4) *Stochastische Verzweigung*: Im Gegensatz zu den vorherigen Übergängen erfolgt die Aktivierung einer stochastischen Verzweigung unverzüglich, wobei jedoch die Wahl des Pfades stochastischer Natur ist. Hieraus folgt erstens, dass eine stochastische Verzweigung nur mit mindestens zwei abgehenden Übergängen Sinn macht. Zweitens folgt, dass die Summe der Eintrittswahrscheinlichkeiten aller Übergänge aus einem Zustand eins ergeben muss. Schließlich folgt, dass der zugehörige Zustand nur transienter Natur ist, was bedeutet, dass während dem Aufenthalt in diesem Zustand keinerlei Zeit vergeht ($d/dt(x) = 0$). Zusammengenommen bedeutet dies, dass eine stochastische Verzweigung unter Weglassung des transienten Zustandes in Form eines gewichteten Verzweigungsvektors an jeden der anderen drei Grundtypen direkt (seriell) hinzugefügt werden kann. Stochastische Verzweigungen treten in NAS-Strukturen besonders im Zusammenhang mit Fehlern auf, wie z. B. dass die Wahrscheinlichkeit einer Fehlübertragung oder eines Messfehlers bei einem Prozent liegt. Ein praktisches Beispiel hierfür findet sich in Form der Modellierung von Fehlern [15].

Während selbstverständlich alle vier Grundtypen in beliebiger Reihenfolge und jeweils durch einen weiteren Zustand getrennt hintereinander angeordnet werden können, ist die Verwendung von Übergängen unterschiedlichen Typs am selben Zustand nur mit Einschränkungen erlaubt, bzw. im Falle einer Kombination aus (1) und (2) sogar verboten (da sich – wie bereits erläutert – (1) in (2) überführen lässt, folgt dies direkt aus vorgenannter Definition dieser Grundtypen).

Eine Verwendung von Grundtyp (4) mit einem der anderen Grundtypen ausgehend vom selben Zustand ist aufgrund der sofortigen Aktivierung dieses Übergangstyps nur in der zuvor erläuterten Form der Weglassung eines transienten Zustandes möglich und sinnvoll. Somit verbleibt die Diskussion einer Verknüpfung des Grundtyps (3) mit (2) bzw. (1). Hierbei lassen sich drei Varianten unterscheiden:

(A) Verknüpfung zweier Aktivierungsbedingungen innerhalb eines Übergangs, wie beispielhaft im linken Automaten von Bild 3 gezeigt. Hierbei wird das Verhalten einer Maschine modelliert, welche im Bereitschaftszustand (rechts) wartet, bis z.B. ein Werkstück eintrifft oder ein „Start"-Knopf gedrückt wird. Die Bearbeitung dauere dann 5 ms (deterministische Zeitbedingung) und kann durch die Aktivierung des „Stopp"-Knopfes abgebrochen werden (deterministische Bedingung). Die zugehörige Bedingung ist daher eine ODER-Verknüpfung der beiden Bedingungen *„x = 5"* und „Stopp". Da es sich bei den zu verknüpfenden Bedingungen um Ereignisse handelt, die definitionsgemäß nie zum gleichen Zeitpunkt aktiviert werden können, ist als Verknüpfungsoperator lediglich der ODER-Operator sinnvoll. Eine UND-Verknüpfung muss sequentiell unter Zuhilfenahme eines Zwischenzustandes (erst A und dann B sowie erst B und dann A) realisiert werden.

(B) Verwendung zweier unterschiedlicher Übergangstypen ausgehend von ein und demselben Zustand, wie beispielhaft im rechten Automaten von Bild 3 gezeigt. Hierbei ist das Verhalten eines Prozessors dargestellt, welcher als ersten Schritt sämtliche Inputs (Eingänge) einliest, diese anschließend verarbeitet und die zugehörigen Ergebnisse abschließend ausgibt. Der Einlesevorgang dauert $x = t_r$ Millisekunden, die Ausführung 5 Millisekunden und für die Ausgabe werden $x = t_w$ Millisekunden veranschlagt. Während der Bearbeitung (Zustand „ausführen") kann ein Interruptsignal (IRQ) eintreffen, welches die Bearbeitung unterbricht und eine Betriebssystemroutine (Zustand „OS Routine") startet. Solange diese abgearbeitet wird – die hierfür erforderliche Zeit wird durch eine Wahrscheinlichkeitsverteilung über der Routinenzeit *y* beschrieben – ist die Bearbeitung des Hauptprogramms gestoppt, das heißt die zugehörige Bearbeitungszeit *x* erfährt keine Veränderung, bis die Betriebssystemroutine abgearbeitet und der Prozessor folglich in den „ausführen"-Zustand zurückgekehrt ist. Anmerkung: In der Literatur werden Automaten bei denen lokale Uhren zeitweilig angehalten werden können als „Stopwatch"-Automat bezeichnet [16].

Allgemein lässt sich festhalten, dass es möglich ist, eine beliebige Anzahl Übergänge des Grundtyps *deterministische Bedingung* mit maximal einem mit einer Zeitbedingung ausgestatteten Übergang von einem gemeinsamem Zustand parallel ausgehend zu verwenden.

(C) Die Überlagerung von (A) und (B), wobei anzumerken bleibt, dass es mit der hier beschriebenen kontinuierlichen Automatenstruktur nicht möglich ist, eine Auswahl zwischen mehreren mit (zeitlichen!) Dichtefunktionen behafteten Übergängen aufgrund der Erfülltheit einer zugehörigen Zustandsbedingung zu treffen, da dies erstens einer UND-Verknüpfung entsprechen und zweitens gegen den Grundsatz der Nichtzulässigkeit mehrerer Zeitbedingungen verstoßen würde.

Modellierungsprozess

Für den Entwurf eines Modells zur Validierung mittels Probabilistic Model Checking (PMC) wird ein dreigliedriges Vorgehen vorgeschlagen.

1. Zunächst werden Einzelmodelle der Grundfunktionen (z.B. die Grundeigenschaften eines Sensormoduls seinen Signalwert auf Anfrage auszugeben, aber auch der periodische Durchlauf einer SPS) als unabhängige Automaten modelliert.
2. Im zweiten Schritt werden diese Module entsprechend ihrer Auftretenshäufigkeit im Modell instanziiert und um architekturabhängige Verknüpfungen ergänzt (die SPS sollte z.B. wissen, an welche I/O-Boards sie Anfragen schicken muss). Architekturabhängige Verknüpfungen sind rein deterministische, weder zeit- noch wahrscheinlichkeitsbewertete E/A-Automaten [17].
3. Im abschließenden dritten Schritt wird das Modell um die für PMC notwendige Signalverfolgung (Signal-Tracking) ergänzt und mit gültigen Anfangswerten initialisiert. Anders als bei traditionelleren Ansätzen muss dafür Sorge getragen werden, dass lediglich ein einziger kompletter Zyklus in der Analyse berücksichtigt wird. Unter einem Zyklus wird hierbei der vollständige Durchlauf verstanden, welcher sich von einer im Allgemeinen wahrscheinlichkeitsbewerteten Menge von Anfangszuständen ausgehend bis hin zur entweder erstmaligen Erfüllung einer vorgegebenen Eigenschaft oder dem Wiedereintritt in die Anfangsmenge (ohne dass die Eigenschaft in einem Zustand erfüllt gewesen wäre) ergibt. Wird dies nicht berücksichtigt, ergibt sich statt der gesuchten Eintrittswahrscheinlichkeit eine auf dieser Wahrscheinlichkeit beruhende geometrische Reihe, welche gegen eins konvergiert.

Die Forderung, die Berechnung nach einem Zyklus zu beenden, lässt sich auf zwei Bedingungen abbilden: Erstens muss die Anfangsmenge so gewählt sein, dass sämtliche möglichen Anfangszustände, entsprechend ihrer Auftretenswahrscheinlichkeit gewichtet, vertreten sind. Zweitens muss die Berechnung beendet werden, sobald der betrachtete Prozess einmal abgelaufen ist. Unter dem zu betrachtenden Prozess versteht man z. B. das Verhalten zwischen Auftreten eines Sensorsignals und der Reaktion des zugehörigen Aktuators (zur Analyse des Zeitverhaltens), oder den Signalverlauf zwischen Senden einer Anfrage und Empfangen der zugehörigen Antwort (z. B. zur Überprüfung einer ausreichenden Abtastrate). Der Automat zur Zustandsverfolgung beobachtet dabei den entsprechenden Verlauf anhand der von den Einzelautomaten erzeugten Ereignisse vom Anfangspunkt bis zu einem Endbereich. Der Endbereich kann aus einer beliebigen Mischung von sowohl temporären als auch konditionalen Kriterien bestehen. Eine Endbereichsdefinition könnte also z. B. lauten, dass der Beobachtungsprozess fortgesetzt wird, so lange eine bestimmte Eigenschaft (z. B. die Aktivierung eines Stellgliedes) nicht eingetreten ist und eine vorgegebene Maximalzeit nicht überschritten wurde.

Anwendungsbeispiel

Als einführendes Beispiel wird ein aus Sensor, SPS und Stellglied bestehendes Automatisierungssystem betrachtet. Die zu untersuchende Fragestellung betrifft die Analyse der Antwortzeit des Systems, wobei der Unterschied zwischen herkömmlicher direkter Verdrahtung und NAS-Strukturen thematisiert werden soll. Als Antwortzeit sei hierbei die Verzögerung definiert, die sich zwischen einem Signalwechsel an einem Eingang des Systems und der Reaktion am zugeordneten Ausgang ergibt. Hierbei wird angenommen, dass die SPS in den untersuchten Szenarien die Reaktion auf einen geänderten Eingangswert innerhalb eines Zyklus berechnet (z. B. O1: = I1). Weiter wird angenommen, dass der neue Signalwert am Eingang des Systems mindestens so lange ansteht, wie es zur Erfassung durch das System erforderlich ist.

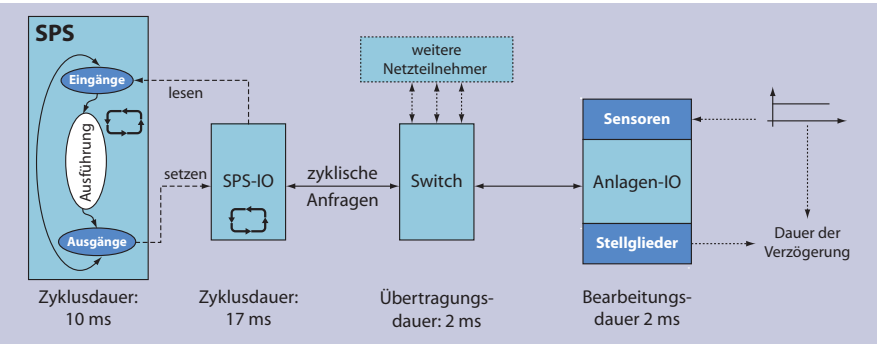


Bild 4: Struktur des Anwendungsbeispiels

Bild 4 zeigt das betrachtete System als NAS. Für die Netzübertragung wird eine Ethernet-UDP/IP-Übertragung angenommen, welche – wie Messungen am Lehrstuhl der Autoren gezeigt haben [18] – im Mittel 2 ms benötigt, wobei die Messwerte zwischen 1,5 und 2,5 ms streuen. Für die Analyse wird hier vereinfachend jede Netzwerkpassage mit 2 ms berücksichtigt, was aufgrund der Tatsache, dass in diesem Artikel nicht die Architektur des Netz-

werkes sondern die in NAS auftretenden Effekte untersucht werden sollen, sinnvoll ist.

Direkt verdrahtetes System

Der zu beobachtende Ablauf beginnt mit dem Eintreffen des externen Signals am Sensor und endet, sobald die SPS den entsprechenden Ausgang aktiviert hat. Da Filterung und andere Effekte vernachlässigt werden (bzw. lediglich durch eine am Ende hinzuaddierte Zeitverzögerung von 1 ms Berücksichtigung finden), lässt sich das Signal-Tracking Modul wie in Bild 5 gezeigt modellieren. Die eine im System vorhandene SPS wurde hierbei mit SPS1 benannt und das beim Verlassen des Zustandes „schreiben" erzeugte Ereignis konsequenterweise als SPS1.schreiben. Gleiches gilt für das Ereignis SPS1.lesen.

Das SPS-DesLaNAS-Modul (Bild 6) hat hierbei die drei Zustände schreiben, lesen sowie ausführen, die beiden Ausgänge schreiben und lesen und weist zyklisches Verhalten auf. Beim Eintritt in den mit „schreiben" gekennzeichneten Zustand wird die lokale Uhrvariable x zurückgesetzt. In diesem Zustand werden die Ausgänge gesetzt und nach Ablauf der zugehörigen Zeitdauer ($x = t_w$) in den Zustand „lesen" gewechselt. Nachdem das Einlesen der Eingänge begonnen wurde ($x = t_r + t_w$) wechselt der Automat in den mit „ausführen" gekennzeichneten Zustand, um dort zu verweilen, bis sämtliche Befehlszeilen sicher abgearbeitet wurden (modelliert durch eine obere zeitliche Schranke). Selbstverständlich verfügt eine SPS über mehr als diese drei Zustände, wie z. B. die Wartezeit zwischen Befehlsausführung und Ausgabe, welche notwendig zum Erreichen einer gleich bleibenden Zyklusdauer ist. Für die hier durchgeführte Analyse sind jedoch lediglich zwei Ereignisse von Interesse, nämlich: der Zeitpunkt, zu dem die Werte ausgegeben worden sind (Verlassen des Zustandes „schreiben") und der Zeitpunkt, ab dem die neuen Werte eingelesen werden, sich eine Änderung am Eingang also nicht mehr auf die im nächsten Berechnungsschritt verwendeten Variablen auswirkt. Um die Möglichkeit einer detaillierteren Modellierung dieses Moduls ohne Änderung der Schnittstellen offen zu halten, wird der Zeitpunkt des Lesebeginns als Verlassen des Zustands „lesen" modelliert. Diese Ausgangsereignisse werden zur Verdeutlichung in der grafischen Darstellung an die Zustandsübergänge (Kanten) geschrieben.

Die zugehörige Instanziierung und Parametrisierung dieser beiden Module (SPS und Signal-Tracking) ist in Bild 7 gezeigt. Jeder Modulblock führt hierbei in der Kopfzeile den Namen sowie die Klasse. Darunter finden sich auf der linken Seite die Liste der Parameter, in diesem Fall also t_w, t_r und $d(x)$, und auf der rechten Seite die Liste der Anfangswerte. P^0 steht hierbei für die Wahrscheinlichkeit, dass sich die SPS zum Zeitpunkt 0 im zugehörigen Zustand befindet. $t_{cyc} = 10$ ist die, in diesem Beispiel als konstant angenommene, Zykluszeit. X^0 gibt an, mit welcher Wahrscheinlichkeit die entsprechende Uhr zum Zeitpunkt 0 einen bestimmten Wert aufweist. GV(a,b) bedeutet gleichverteilt zwischen (a, b].

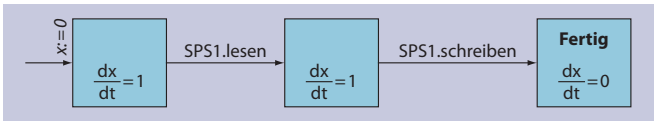


Bild 5: Signal-Tracking für das direkt verdrahtete System.

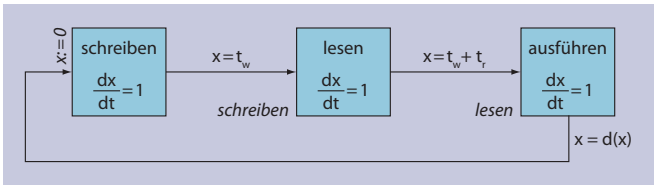


Bild 6: SPS-Modul.

Die Analyse, welche mit PRISM [14] einem Probabilistic Model Checker der University of Birmingham erstellt wurde, führt zu der in Bild 8 dargestellten diskreten Antwortzeitverteilung. Der Wert 10 % bei 15 ms bedeutet z. B., dass die Wahrscheinlichkeit einer Reaktionsverzögerung zwischen 14 und 15 ms 10 % beträgt (die Diskretisierung wurde mit einer Zeitschrittweite von 1 ms durchgeführt).

Networked Automation System

Im Falle eines NAS beginnt der zu beobachtende Ablauf zwar ebenfalls mit dem Eintreffen des externen Signals am

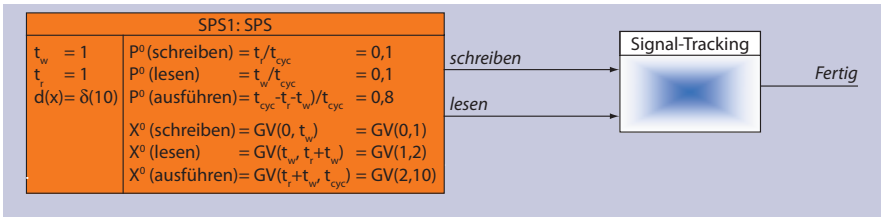


Bild 7: Instanziierung und Parametrisierung des direkt verdrahteten Systems.

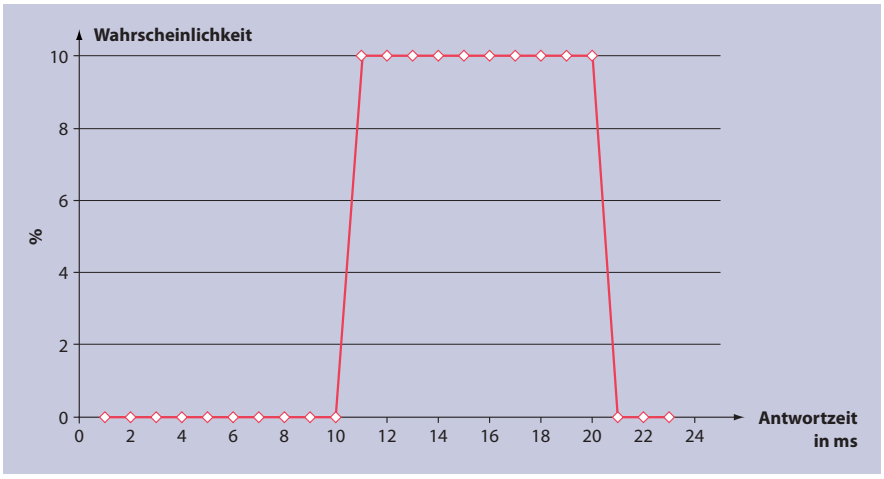


Bild 8: Antwortzeitverteilung für das direkt verdrahtete System

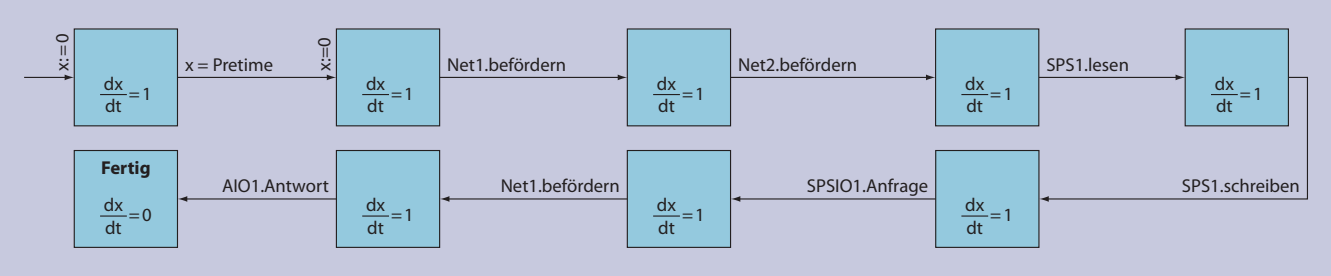


Bild 9: Signal-Tracking für das NAS des Beispiels.

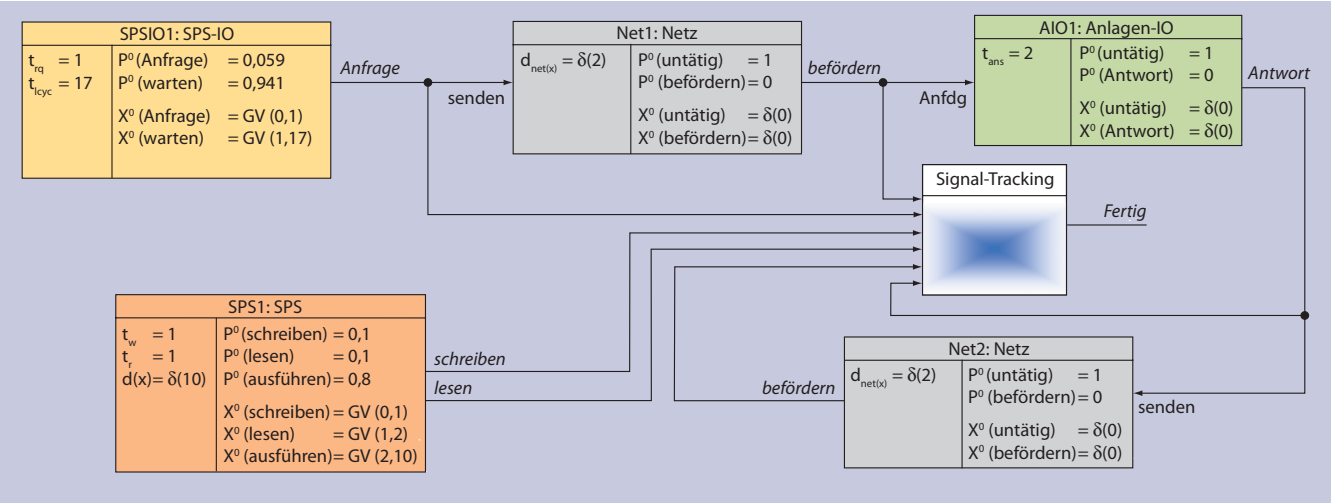


Bild 10: Instanziierung und Parametrierung des NAS.

Sensor, ist in der Folge jedoch etwas komplexer: Nachdem sich der Sensorwert ändert, wartet die Anlagen-I/O zunächst auf eine Anfrage der SPS-I/O, welche zyklisch alle 17 ms versandt wird. Trifft diese ein, wird ein Antworttelegramm vorbereitet. Die Antwort geht über das Netz zur SPS-I/O. Von dort wird sie ohne weitere Verzögerung über einen Backplane-Bus in den Eingangspuffer der SPS geschrieben. Nun wird auf den nächsten Lesevorgang der SPS gewartet. Nach dem Lesen wird der Wert in der SPS verarbeitet und eine Reaktion berechnet. Diese geht wieder zur SPS-I/O und wartet dort auf den nächsten Sendevorgang, das heißt im Sensorabfragetelegramm werden auch die neuen Stellwerte übermittelt. Danach erfolgt ein erneuter Netzdurchgang und eine Bearbeitung des Telegramms in der Anlagen-I/O.

Das Signal-Tracking Modul ergibt sich direkt aus der beschriebenen Systemstruktur (grafisch unter Verwendung von DesLaNAS erstellt) und der zu prüfenden Eigenschaft. Das die Eigenschaft „Verteilung der Antwortzeiten" repräsentierende Signal-Tracking Modul ist in Bild 9 gezeigt. Die Kanten wurden hierbei mit dem entsprechenden Ereignis beschriftet. Die Einschwingzeit (Pretime) ist notwendig, um dem durch Netzübertragung und SPS-IO gegebenen Seitenprozess ausreichend Zeit zu geben, seinen Initialzustand (Zustand zum Ereigniseintritt) zu bestimmen. Diese Bestimmung kann zwar auch analytisch vorgenommen werden, ist jedoch meist durch Zugabe einer ausreichend großen Einschwingzeit einfacher zu erreichen. Die zugehörige Instanziierungs- und Parametrisierungssicht des Systems ist in Bild 10 gezeigt.

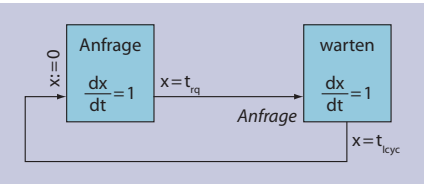


Bild 11: Automat der SPS-I/O.

Nicht alle entlang des Signalpfades erzeugten Ereignisse induzieren auch einen Zustandswechsel im Signal-Tracking-Modul. Selbstverständlich ist es möglich, sämtliche Ereignisse zu repräsentieren, allerdings würde dies den Rechenzeit- und Hauptspeicherbedarf unnötig erhöhen. Die Reduktion kann hierbei entweder analytisch oder durch entsprechendes Ingenieursgeschick vorgenommen werden. Grundsätzlich darf ein Ereignis B, welches im Signalpfad in einer Folge von Ereignissen A, B und C auftritt im Signal-Tracking genau dann weggelassen werden, wenn (1) das System keine Möglichkeit vorsieht von A nach C zu gelangen ohne B zu passieren, oder (2) auf eine Ereignisfolge A, B stets ein Ereignis C folgt.

Neben dem bereits eingeführten SPS-Modul werden für die netzbasierte Lösung noch eine Reihe weiterer Module benötigt. Im Folgenden werden daher die zusätzlich erforderlichen DesLaNAS-Einzelmodule vorgestellt. Das SPS-I/O-Modul (Bild 11) entspricht funktional gesehen weitestgehend demjenigen der SPS. Die lokale Uhrvariable *x* wird beim Eintritt in den Zustand „Anfrage" zurückgesetzt. Das Verlassen dieses Zustandes bezeichnet jenen Moment, zu dem die (oder die Menge der) Anfrage(n) abgesendet worden ist (sind). Obwohl diese Anfragen in serieller Reihenfolge versandt werden, wird angenommen, dass dies parallel

erfolgt. Diese Annahme ist so lange gültig, wie die Zeitschrittweite des für PMC notwendigen diskreten Modells deutlich größer als die zum Senden einer Anfrage erforderliche Zeit ist und die betrachteten Anfragen zeitlich eng hintereinander gesendet werden. Für das hier vorgestellte System ist es daher ausreichend, einen einzigen Anfrage-Zustand zu verwenden.

Sobald alle Anfragen versandt wurden, wechselt der SPS-I/O-Automat in den „warten" Zustand und verbleibt dort bis zum Ende des Zyklus. Der Eingang der Antworten wird nicht im SPS-I/O-Automaten abgebildet, da diese direkt in den Backplanebus geschrieben und somit der SPS unverzüglich zur Verfügung stehen.

Anmerkung: Weist ein Modul lediglich eine lokale Zeitvariable auf, wird diese mit *x* bezeichnet. Die lokalen Zeitvariablen der einzelnen Module sind jedoch unabhängig voneinander und nur über das Fortschreiten der globalen Zeit (*t*) gekoppelt.

Die Netzübertragung wird wie in Bild 12 dargestellt modelliert. Dieser Automat verharrt im mit „untätig" bezeichneten Zustand, ohne dass die lokale Uhr läuft. Sobald ein zu übermittelndes Paket eintrifft (Ereignis senden), wechselt der Automat in den mit „befördern" bezeichneten Zustand. Die Übertragungszeit selbst wird durch die Verteilungsfunktion dnet(x) – das heißt als Wahrscheinlichkeitsverteilung über der Zeitachse – gegeben. Für die hier durchgeführte Betrachtung wird die Netzübertragungszeit als jene Zeit definiert, die sich in Summe aus den Einzelprozessen einer Netzübertragung ergibt, wie z. B. Übertragung, Verpacken, Entpacken, Switching.

Die in der Automatisierungstechnik eingesetzten Netzwerkkomponenten sind in der Regel sehr schnell (Switching times von weniger als 0,08 ms) und selten überlastet [19]. Aus diesem Grunde ist es zulässig jede der zu modellierenden Datenübertragungen als von den anderen unabhängiges Modul zu instanziieren und Störeinflüsse wie Verlust, Stau und andere Übertragungsverzögerung als getrennte Prozesse zu modellieren. Folglich erfordert das zu modellierende System zwei Netzübertragungsmodule, namentlich eines für den Hinweg von der SPS-IO zur Anlagen-IO und eines für den Rückweg.

Bild 12: Die Netzübertragung beschreibender Automat.

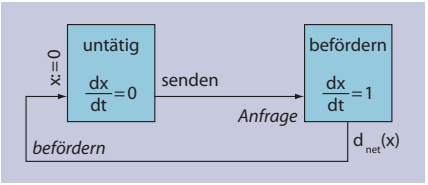
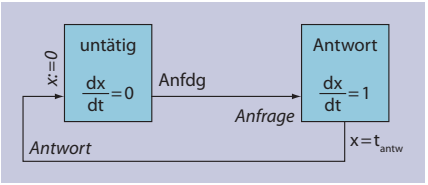


Bild 13: Automat der Anlagen-I/O.



Mit der Anlagen-I/O ist auch das vierte und letzte der benötigten Module benannt. Dieses ist wie in Bild 13 gezeigt aufgebaut und ähnelt von der Struktur demjenigen der Netzübertragung, weist jedoch im Unterschied zu letzterem eine konstante Zeitbedingung auf, da die Bearbeitungszeit der Anlagen-I/O als fix angenommen wurde.

Die Analyse ergibt die in Bild 14 rechts aufgetragene Trajektorie. Zum Vergleich sind nochmals die Ergebnisse für das direkt verdrahtete System (Bild 8) aufgetragen. Für einen Vergleich mit realen Messwerten und die Verwendung einer komplexeren Netzübertragungsfunktion siehe [12].

Bereits in diesem einfachen Beispiel ist zu erkennen, dass alle Vorteile der neuen Automatisierungsstruktur mit einer Erhöhung der Antwortzeiten bezahlt werden müssen. Auffällig ist hierbei, dass diese Erhöhung deutlich größer ist, als der durch die Netzwerkpassagen verursachte Zeitverlust von 4 ms.

Zusammenfassung und Ausblick

Netzbasierte Automatisierungssysteme (NAS) erlauben völlig neuartige, flexible Architekturen in der Automatisierungstechnik. Die Sicherstellung von Antwortzeiten stellt sich dabei allerdings als eine wichtige neue Entwurfs- und Analyseaufgabe heraus.

In diesem ersten von zwei Beiträgen wurde ein kurzer Überblick über Methoden zur Verzögerungsbestimmung gegeben. Hierauf aufbauend wurde die, für die besonderen Bedürfnisse bei der Modellierung von NAS entwickelte, Beschreibungssprache DesLaNAS schrittweise vorgestellt und der zur Anwendung gekommene Modellierungsprozess diskutiert. Anhand eines Beispiels wurde der Einsatz der Beschreibungssprache dokumentiert und die Frage diskutiert, welche Unterschiede sich im Zeitverhalten ergeben, wenn man statt einer direkt verdrahteten eine NAS-Struktur verwendet.

In einem zweiten Beitrag [15] wird gezeigt, welchen Einfluss einzelne Komponenten sowie netzbedingte Verhaltensmodi wie Synchronisation und

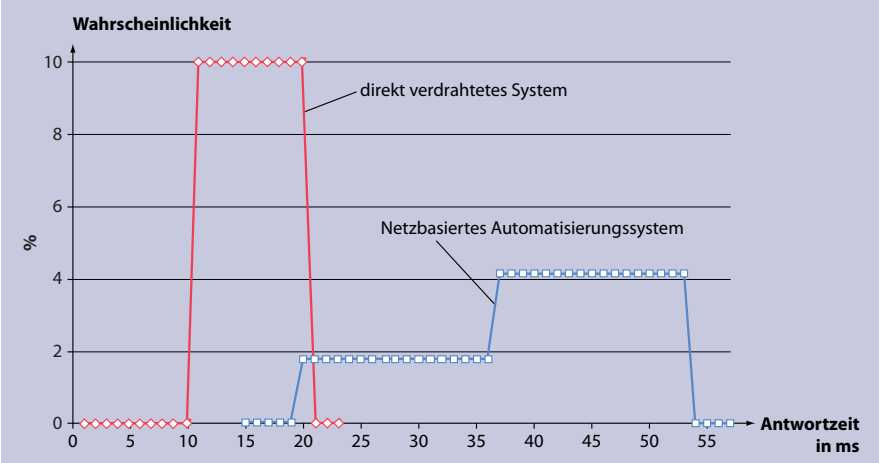


Bild 14: Antwortzeitverteilung für das direkt verdrahtete System und das NAS.

die gemeinsame Nutzung von Ressourcen auf die Antwortzeiten des Gesamtsystems haben. Zur Validierung der Ergebnisse wird außerdem ein Vergleich mit messtechnischen Ergebnissen vorgestellt.

Literatur

[1] *Parrott, J. T., Moyne, J. R., Tilbury, D. M.:* Experimental Determination of Network Quality of Service in Ethernet: UDP, OPC, and VPN, Proc. ACC, Minneapolis, USA (2006).

[2] *Irey, P., Chappell, B. L., Hott, R. W., Marlow, D. T., O'Donoghue, K., Plunkett, T. R.:* Metrics, Methodologies, and Tools for Analyzing Network Fault Recovery Performance in Real-Time Distributed Systems, Proc. IPDPS, Mexiko, pp. 1248–1257 (2000).

[3] *Miorandi, D., Vitturi, S.:* Performance Analysis of Producer/Consumer Protocols over IEEE802.11 Wireless Links, Proc. WFCS, Wien (2004).

[4] *Le Boudec, J.-Y., Thiran, P.:* Network Calculus. Springer Verlag, LNCS 2050 (2001).

[5] *Årzén, K.-E., Cervin, A.:* Control and Embedded Computing: Survey of Research Directions. Proc. 16th IFAC World Congress, Elsevier (2005).

[6] *Dingle, N. J., Harrison, P. G., Knottenbelt, W.J.:* Response Time Densities in Generalised Stochastic Petri Net Models, Proc. 3rd WOSP, Italien, S. 46–54 (2002).

[7] *Bérard, B., Bidiot, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, Schnoebelen, L.:* Systems and software verification, model-checking techniques and tools, Springer (2001).

[8] *Alur, R., Courcoubetis, C., Dill, D.:* Model-checking for probabilistic real-time systems. Proc. ICALP, Springer, LNCS, Band 510, S. 1–100 (1991).

[9] *Kwiatkowska, M., Norman, G., Parker, D.:* Modelling and Verification of Probabilistic Systems, in: Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems. CRM Monograph Series, Band 23, American Mathematical Society, S. 93–215 (2004).

[10] *Katoen, J.-P.:* Stochastic Model Checking, in: Stochastic Hybrid Systems (Cassandras, C. G., Lygeros, J., eds.), Control Engineering Series, Taylor & Francis, Boca Raton, USA, S. 79–107 (2006).

[11] *Hanson, H., Jonsson, D.:* A logic for reasoning about time and reliability, in Formal Aspects of Comp., Band 6, Nr. 5, S. 512–535 (1994).

[12] *Greifeneder, J., Frey, G.:* Probabilistic Timed Automata for Modeling Networked Automation Systems. Proc. IFAC Workshop on Dependable Control of Discrete Systems (DCDS), Paris, Frankreich, S. 143–148 (2007).

[13] *Lunze, J.:* Ereignisdiskrete Systeme, Oldenbourg Verlag, München (2006).

[14] *Kwiatkowska, M., Norman, G., Parker D.:* PRISM: Probabilistic symbolic model checker. Proc. TOOLS, Springer, LNCS Band 2324, S. 200–204 (2002).

[15] *Greifeneder, J., Frey, G.:* Analyse des Antwortzeitverhaltens netzbasierter Automatisierungssysteme, zur Veröffentlichung in: atp – Automatisierungstechnische Praxis 49 (2007), H. 10.

[16] *Cassez, F., Larsen, K.G.:* On the Impressive Power of Stopwatches. Proc. CONCUR, C. Palamidessi (Editor), Springer, LNCS Band 1877, S. 138–152 (2000).

[17] *Greifeneder, J., Frey, G.:* Analyse des Antwortverhaltens vernetzter Automatisierungssysteme mittels Probabilistic Model Checking. Proc. 9. Symp. zu Entwurf komplexer Automatisierungssysteme (EKA), Braunschweig, S. 131–151 (2006).

[18] *Gottheit, C.:* Experimentelle Bestimmung von Verzögerungen in verteilten Automatisierungssystemen, Technische Universität Kaiserslautern (2006).

[19] *Denis, B., Ruel, S., Faure, J.-M., Marsal, G., Frey, G.:* Measuring the Impact of Vertical Integration on Response Times in Ethernet Fieldbusses. Proc. Emerging Technologies and Factory Automation (ETFA), Patras, Griechenland (2007).

Entnommen aus: *Greifeneder, J.:* Formale Analyse des Zeitverhaltens netzbasierter Automatisierungssysteme, Dissertation, Technische Universität Kaiserslautern (2007).

Manuskripteingang: 23. April 2007.

Dipl.-Ing. *Jürgen Greifeneder* (32) arbeitet zurzeit unter der Betreuung von J.Prof. G. Frey an der TU Kaiserslautern an seiner Promotion. Sein Arbeitsgebiet umfasst Modellierung, Simulation und formale Analyse von Regelungs- und Steuerungssystemen, derzeitiger Schwerpunkt ist die Untersuchung Netzbasierter Automatisierungssysteme (NAS).

Adresse: TU Kaiserslautern, FB Elektro- und Informationstechnik, Erwin-Schrödinger-Str. 12, 67663 Kaiserslautern, Deutschland, Tel. +49 631 205-4549, Fax -4462, E-Mail: greifeneder@eit.uni-kl.de

J. Prof. Dr.-Ing. *Georg Frey* (37) ist seit 2002 Inhaber der Juniorprofessur Agentenbasierte Automatisierung an der Technischen Universität Kaiserslautern. Seine Hauptarbeitsgebiete sind Methoden zur Bereitstellung verlässlicher Automatisierungssysteme mit einem Fokus auf verteilten/vernetzten Systemen. Dies betrifft zum einen transparente, nachvollziehbare Entwicklungsmethoden (basierend auf UML und ereignisdiskreten Formalismen) sowie zum anderen Analyseverfahren (Model Checking und Simulation).

Adresse: siehe oben, Tel. +49 631 205-4455, E-Mail: frey@eit.uni-kl.de