Leistungsbewertung automatisch generierter Protokollimplementierungen [1]

J. Thees, R. Gotzhein

Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern

Email: {thees, gotzhein}@informatik.uni-kl.de

Zusammenfassung

Formale Beschreibungstechniken (FDTs) erlauben durch ihre formale Syntax und Semantik eine präzise Systembeschreibung und sind Grundlage für die formale Verifikation. Bei der Implementierung von Systemen wird jedoch nach wie vor von Hand implementiert, selbst wenn ausgereifte Werkzeuge zur automatischen Generierung von Kode direkt aus der formalen Spezifikation existieren. Die Ursache dafür liegt in dem Ruf dieser Werkzeuge, Kode mit extrem geringer Leistungsfähigkeit zu erzeugen. Es gibt jedoch kaum quantitative Leistungsvergleiche zwischen manuell und automatisch generierten Implementierungen, die dieses Vorurteil stützen oder widerlegen könnten.

In diesem Beitrag wird ein solcher Leistungsvergleich und die dazu entwickelten Werkzeuge anhand des Hochleistungsprotokolls XTP und der FDT Estelle vorgestellt. Er liefert eine *Bestandsaufnahme* des momentanen Entwicklungsstandes bei der automatischen Generierung von Kode aus Estelle-Spezifikationen im direkten Vergleich zu gut optimierten Handimplementierungen.

1 Einleitung

Die Hauptanforderungen an Protokollimplementierungen sind *Korrektheit* und *Effizienz*. Der Aspekt der Korrektheit legt den Einsatz von formalen Beschreibungstechniken (FDTs) nahe, wohingegen zur Erreichung hoher Effizienz nach wie vor handkodierte Implementierungen zum Einsatz kommen. Dieses Vorgehen stellt jedoch speziell bei großen Systemen die Übertragbarkeit der Eigenschaften der formalen Beschreibung auf die Implementierung in Frage.

Der Einsatz von Werkzeugen zur *automatischen Generierung von Protokollimplementierungen* direkt aus formalen Beschreibungen heraus könnte zur Lösung dieses Problems beitragen. Jedoch werden diese Werkzeuge in der Praxis wegen ihres Rufs, Kode mit extrem geringer Leistungsfähigkeit zu erzeugen, nur sehr selten eingesetzt. Dabei beruht dieser Ruf im wesentlichen auf dem Vorurteil, daß FDTs zu wenig implementierungsbezogen seien und die daraus automatisch generierten Implementierungen sich nicht mit handkodierten Implementierungen messen könnten. Jedoch gibt es *kaum direkte quantitative Leistungsvergleiche* zwischen handkodierten Protokollmaschinen und Implementierungen, die automatisch aus formalen Spezifikationen erstellt wurden.

In diesem Beitrag stellen wir einen solchen quantitativen Leistungsvergleich und die dazu entwickelten Werkzeuge anhand des Hochleistungsprotokolls "*Xpress Transport Protocol*" (XTP) und der FDT *Estelle* vor. Diese Untersuchung liefert erste quantitative Leistungsdaten über den aktuellen Stand der Entwicklung im Bereich der automatischen Protokollimplementierung im Vergleich mit einer gut optimierten Handimplementierung. Derartige Daten sind unerläßlich bei der Entscheidung, ob die automatisch generierte Implementierung eines Protokolls genügend Leistung für eine konkrete Anwendung erbringt, oder ob es notwendig und sinnvoll ist,

1. Diese Arbeit wurde von der Deutschen Forschungsgemeinschaft (DFG) im Rahmen des Projekts "Protokollimplementierung mit Estelle — Vom Prototypen zum effizientem Kode" (Go 503/4-1) gefördert.

eine Handimplementierung zu entwickeln. Diese Fragestellung hat besondere Bedeutung beim Einsatz von anwendungsspezifischen Protokollen, bei denen die Entwicklung einer gut optimierten Handimplementierung möglicherweise unrentabel ist.

2 Die FDT Estelle

Estelle [6, 4] ist eine formale Beschreibungstechnik ("formal description technique", FDT), die seit 1989 international genormt ist. Sie basiert auf ISO-Pascal und wurde zur Beschreibung von *Kommunikationsprotokollen* entwickelt. Estelle Spezifikationen beschreiben ein dynamisch modifizierbares System miteinander kommunizierender endlicher Automaten.

Estelle eignet sich durch seine operationale Verhaltensbeschreibung und seine einfache Typstruktur besonders für eine *automatische Implementierung*. Im Gegensatz zu anderen FDTs, die eigenschaftsorientierte Verhaltensbeschreibungen oder abstrakte, rekursive und axiomatische Typdefinitionen benutzen, kann Estelle relativ direkt implementiert werden: Estelle-Typen sind im wesentlichen Pascal-Typen und können direkt in eine Implementierungssprache wie C abgebildet werden. Gleiches gilt für die Verhaltensbeschreibung von Estelle-Spezifikationen, die im wesentlichen durch Transitionen mit sequentiellen, imperativen, Pascal-ähnlichen Anweisungsblöcken gegeben werden. Auch sie können direkt nach C abgebildet werden.

Für Estelle wurden bereits mehrere *Compiler* mit unterschiedlichen Zielsetzungen entwickelt. Allen gemeinsam ist, daß sie aus einer Estelle-Spezifikation Pascal-, C- oder C++-Quellkode generieren, der dann zu einem ablauffähigen Programm übersetzt werden kann. Im folgenden werden die beiden gebräuchlichsten Compiler kurz vorgestellt:

- Das Kodegenerierungswerkzeug **PET/DINGO** besteht aus dem Frontend PET (Portable Estelle Translator) und dem Backend DINGO (Distributed Implementaion Generator). Es wurde vom NIST [11] entwickelt und unterstützt die Erzeugung einer verteilt ausführbaren Implementierung einer Spezifikation. Das in Abschnitt 4.1.3 vorgestellte Performance-Evaluierung-Werkzeug PATO wurde für diesen Compiler entwickelt.
- Bei **EC** (Estelle Compiler) handelt es sich um einen *kommerziellen* Estelle-Compiler, der von Bull S.A. entwickelt [8] und später vom französischen Institut National des Télécommunications (INT) übernommen wurde. EC wurde mit dem Ziel entwickelt, eine *optimierte Implementierung* einer Estelle-Spezifikation zu erzeugen. Da es sich jedoch um ein kommerzielles Produkt handelt, sind die Quellen des Compilers und der Laufzeitbibliothek nicht zugänglich, was die Möglichkeiten zur genaueren Leistungsanalyse stark einschränkt.

3 Das 'Xpress Transport Protocol'

Als Grundlage für den quantitativen Leistungsvergleich zwischen automatisch generierten Implementierungen und Handimplementierungen von Kommunikationsprotokollen haben wir das 'Xpress Transport Protocol' (XTP, [13]) Version 4.0 gewählt: Zum einen sind für dieses Protokoll sowohl eine gut optimierte Handimplementierung (Abschnitt 3.1) als auch eine formale Estelle-Spezifikation (Abschnitt 3.2) verfügbar, zum anderen ist es als Vertreter der Klasse der Hochleistungsprotokolle für eine effiziente Implementierung besonders geeignet.

3.1 SandiaXTP

SandiaXTP 1.4 [9, 10] ist eine optimierte, handcodierte Implementierung von XTP 4.0, die von den Sandia National Laboratories entwickelt wurde. Die XTP Protokollmaschine ist hier als

UNIX-Daemon-Prozeß realisiert, der u.a. über UDP oder IP eine Kommunikation zwischen verschiedenen Hostrechnern erlaubt. Die Anwendungen von XTP arbeiten als getrennte Prozesse, die über Sockets (für Synchronisation und Austausch von Kontrollinformationen) und gemeinsamen Speicher ("shared memory" für die Übertragung der Nutzdaten) mit dem XTP-Daemon am selben Host kommunizieren. Die Ankopplung an den Basisdienst erfolgt im Falle von IP und UDP vollständig über Sockets.

SandiaXTP hat eine durchgehend objektorientierte Struktur und ist in C++ implementiert. Durch effiziente Algorithmen und den gezielten Einsatz von Inline-Kode konnte eine hohe Effizienz erreicht werden. So wird z.B. der Nutzdatenanteil eines Sendeauftrags nur zweimal kopiert: Beim Sendeaufruf eines Clients werden die Daten in einen Shared-Memory Puffer kopiert, vom XTP Daemon ohne weiteres Kopieren bearbeitet und schließlich direkt aus diesem Puffer als IP-Sendeauftrag an das Betriebssystem übergeben.

3.2 Eine Estelle-Spezifikation von XTP

Die der Untersuchung zugrundeliegende Estelle-Spezifikation wurde ursprünglich am Institut National des Télécommunications für XTP 3.6 erstellt [2] und später an der PUB Bukarest an XTP 4.0 angepaßt. Da Estelle nur geschlossene Systeme beschreiben kann, wurde der XTP Protokollautomat in ein vollständiges Umgebungsszenario eingebettet. Dieses enthält zusätzlich ein Netzwerkmodul und zwei Benutzermodule. Die Grobstruktur dies Kommunikationsszenarios ist in **Abbildung 1** dargestellt.

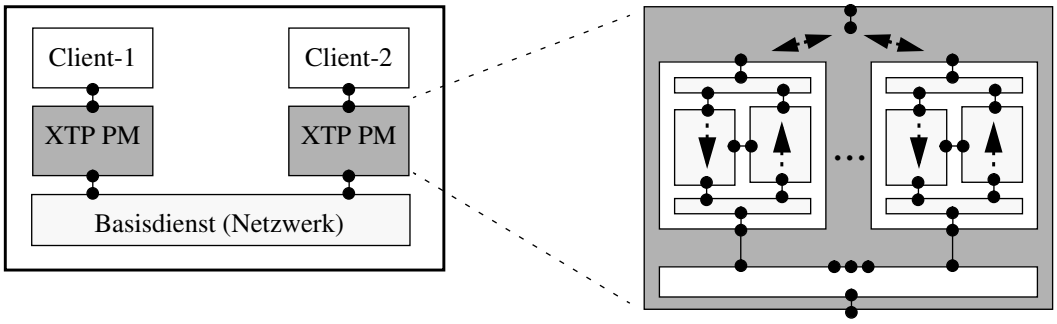

Abbildung 1: Die Modulinstanz- und Verbindungsstruktur der Estelle-Spezifikation

Ziel bei der Erstellung der Spezifikation waren im wesentlichen die Entwicklung einer geeigneten Dienstschnittstelle und die Erprobung der Protokollfunktionalität. Performanceaspekte wurden dabei nicht berücksichtigt. Dies macht sich besonders in der starken Strukturierung des Protokollautomaten in eine komplexe Modulhierarchie bemerkbar: Die sich daraus ergebende Notwendigkeit zur häufigen Übertragung der Nutzdaten zwischen Modulen führt zu hohem Kopieraufwand. So wird ein Datenpaket auf seinem Weg vom Benutzermodul zum Netzwerk insgesamt siebenmal (!) kopiert, ebenso auf dem Weg vom Netzwerk zum Benutzermodul. Außerdem macht die Modularisierung zusammen mit der eingeschränkten Möglichkeit zu gemeinsamem Speicher in Estelle zusätzliche interne nachrichtenbasierte Kommunikation zwischen den Komponenten einer Protokollmaschine nötig.

4 Leistungsmessung

In diesem Abschnitt werden die Methoden und Werkzeuge dargestellt, die beim quantitativen Leistungsvergleich zwischen den verschiedenen Protokollimplementierungen zum Einsatz gekommen sind. Der Vergleich der Estelle-Implementierungen mit der Handimplementierung

(Abschnitt 4.1) machte eine Aufspaltung der Estelle-Spezifikation in zwei (pragmatisch realisierte) kommunizierende offene Systeme notwendig. Eine zweite Untersuchung (Abschnitt 4.2) vergleicht die verschiedenen Estelle-Compiler anhand von Benchmark-Spezifikationen. Weitere Details zum Meßverfahren und zu den gewonnenen Ergebnisse sind in [3, 12] zu finden.

4.1 Vergleich zwischen SandiaXTP und automatisch generierten Estelle-Implementierungen

Zu einem fairen Leistungsvergleich zwischen den XTP Implementierungen waren neben einigen Anpassungen am realisierten Protokollumfang (z.B. Prüfsummenberechnung) auch einige strukturelle Änderungen notwendig. So kann das geschlossene Kommunikationsszenario der Estelle-Spezifikation vom Estelle-Compiler EC nur in einen einzelnen UNIX-Prozeß umgesetzt werden, wohingegen SandiaXTP ein vergleichbares Szenario nur zwischen mehreren Daemon-Prozessen auf unterschiedlichen Hostrechnern darstellen kann. Aber auch die Möglichkeiten zur Erzeugung einer verteilten Implementierung der Estelle-Spezifikation durch PET/DINGO oder UHH führen zu keiner mit SandiaXTP vergleichbaren Implementierung, da die Kommunikation zwischen den verteilten Estelle-Komponenten nicht auf die Protokollkommunikation von XTP beschränkt werden kann, sondern ganz überwiegend aus Estelle-spezifischen Synchronisationsnachrichten besteht.

Um dennoch ein realistisches Kommunikationsszenario nachbilden zu können, wurde die Estelle-Spezifikation in zwei Spezifikationen zerteilt, die jeweils ein Benutzermodul, eine Protokollmaschine und ein Netzwerkmodul enthalten. Mit Hilfe von primitiven Funktionen [5] wurden dann die Netzwerkmodule beider Spezifikationen so modifiziert, daß diese Module als Zugang zu realen Kommunikationsmedien (z.B. IP) dienen konnten, wobei durch geeignete Kodierungs- und Dekodierungsfunktionen die implementierungsspezifischen Estelle-Datenstrukturen in binärkompatible XTP Pakete umgewandelt wurden. Dadurch konnten mit den zur Verfügung stehenden Estelle-Compilern jeweils zwei Prozesse erzeugt werden, die — genau wie SandiaXTP — über ein reines XTP-Protokoll kommunizierten. Es war dadurch sogar möglich, eine direkte Kommunikation zwischen SandiaXTP und den aus Estelle generierten Prozessen durchzuführen.

4.1.1 Netzwerksimulator

Als Basisdienst für den Vergleich zwischen SandiaXTP und den Estelle-Implementierungen wurde ein eigens dazu entwickelter *Netzwerksimulator* eingesetzt. Dieser erlaubt es, die Eigenschaften des Basisdienstes (Durchsatz, Verzögerung, Zuverlässigkeit) exakt zu kontrollieren, und gewährleistet somit die Reproduzierbarkeit der Meßergebnisse. Durch den Einsatz von Shared-Memory als Datentransportmedium kann der Aufwand für die Übertragung eines Pakets vom Sender- zum Empfängerprozeß auf eine Kopieroperation beschränkt werden. Dies erlaubt höchste Datendurchsatzraten und ermöglicht so die realistische Simulation eines Hochgeschwindigkeitsmediums. Da mit Hilfe des Netzwerksimulators alle Komponenten des Kommunikationssystems auf demselben Hostrechner arbeiten, existiert auch eine ausreichend genaue globale Zeitbasis, welche die zeitliche Verfolgung der Verarbeitung eines Pakets über die Grenzen einer Protokollmaschine hinweg erlaubt.

4.1.2 Benutzerprofile

Aufgrund der große Flexibilität von XTP hat das Anwendungsprofil der Benutzermodule bzw. -prozesse großen Einfluß auf die Kommunikation zwischen den Protokollmaschinen und damit auch auf den Datendurchsatz des Gesamtsystems. Um trotz der unterschiedlichen Schnittstellen von SandiaXTP und der in Estelle spezifizierten Protokollautomaten mehrere vergleichbare Be-

nutzerprofile dynamisch einsetzen zu können, wurden für beide Protokollrealisierungen dynamisch parametrierbare, generische Benutzermodule bzw. -prozesse erstellt. Diese erlauben die Simulation verschiedener Übertragungscharakteristika, wie z.B. “Nachrichten Ping-Pong” oder die Übertragung großer Datenmengen (“Bursts”). Die Einstellung des Benutzerprofils ist interaktiv oder über Parameterdateien möglich, so daß leicht mit verschiedenen Einstellungen experimentiert werden kann.

Eine Ausnahme bei der dynamischen Auswahl von Benutzerprofilen ist die maximale Datenpaketgröße bei der Estelle-Spezifikation: Da Estelle keine Datenpakete (Arrays) variabler Größe beschreiben kann, muß bei der Erstellung einer Implementierung eine konkrete Obergrenze für die Nutzdatenmenge eine Paketes (sowohl an der Benutzerschnittstelle als auch an der Netzwerkschnittstelle) angegeben werden. Bei einer Änderung dieser Größe muß die Implementierung neu erzeugt werden. Wie sich zeigen wird, hängt die Leistung der aus der Estelle-Spezifikation erzeugten Implementierungen neben der zu übertragenden Datenmenge auch ganz wesentlich von dieser maximalen Nutzdatenmenge der Pakete ab, da bei der Übertragung solcher Pakete durch Interaktionspunkte immer die gesamte Datenstruktur des Pakets kopiert wird. Deshalb wurde bei den Leistungsmessungen sowohl die *aktuelle* als auch die *maximale* Nutzdatenmenge (“Limit”) variiert.

4.1.3 Das Performance-Monitoring-Werkzeug PATO

Zur detaillierten Leistungsmessung in Protokollimplementierungen wurde das Performance Monitoring- und Analysewerkzeug PATO (“Performance Analysing Tool”, [7]) eingesetzt, das im Rahmen einer Diplomarbeit u.a. zur Untersuchung von automatisch generierten Implementierungen von Estelle-Spezifikationen entwickelt wurde. PATO besteht aus einer Bibliothek von Funktionen, die in dem zu überwachenden Programm das Setzen von Zeitstempeln erlauben, und einem Auswertungswerkzeug. Die Aufrufe zur Erzeugung der Zeitstempel werden entweder manuell (bei SandiaXTP und EC) oder automatisch (durch eine entsprechend modifizierte Version von DINGO) eingefügt.

4.1.4 Meßergebnisse

Am Beispiel von SandiaXTP und den von den verschiedenen Estelle-Compilern erzeugten Implementierungen der Estelle-XTP-Spezifikation wurde eine systematische vergleichende Leistungsmessung durchgeführt [3, 12]. Dabei wurden unter Variation verschiedener Parameter, wie Paketgröße, Eigenschaften des Basisdienstes und Benutzerprofil, quantitative Messreihen[2] durchgeführt. In diesem Abschnitt werden beispielhaft die Ergebnisse des Leistungsvergleichs anhand des Benutzerprofils “*Nachrichten-Ping-Pong*” unter Variation der Paketgröße zusammenfassend dargestellt. Dabei überträgt ein XTP Anwender über eine bestehende Verbindung ein Datenpaket zu einem zweiten Anwender, und dieser überträgt es anschließend zurück (Quittungs- und Kontrollpakete werden dabei nicht übertragen). Dieses Benutzerprofil erlaubt eine quantitative Bestimmung der reinen Paketbearbeitungszeiten durch die XTP Protokollautomaten, da zu jedem Zeitpunkt nur in einer Komponente des Systems Aktivität besteht und so der Ablauf der Übertragung nicht-konkurrierend (und somit deterministisch) verläuft.

Als Basisdienst wurde der in Abschnitt 4.1.1 vorgestellte Netzwerksimulator (keine Ratenbegrenzung, kein Verlust oder Verfälschung) eingesetzt. Die im folgenden Messungen beziehen sich auf die einfache Übertragung eines Datenpakets von Ende zu Ende. Die angegebenen Zeiten sind jeweils die Summe der reinen Bearbeitungszeiten innerhalb der Protokollmaschinen des Senders und des Empfängers bei deaktivierter Prüfsummenberechnung. Die Bearbeitungs-

2. Die Meßbedingungen sind im Anhang zu finden.

zeiten in den Benutzermodulen, die Prozeßumschaltzeiten und die Übertragungszeiten im Netzwerksimulator sind darin nicht enthalten.

		übertragene Nutzdaten [Byte]					
		10		1,000		10,000	
Implementation	Limit [Byte]	absolut	relativ	absolut	relativ	absolut	relativ
SandiaXTP	—	0.80 ms	1	1.52 ms	1.9	2.68 ms	3.4
EC	10	3.06 ms	3.8	—	—	—	—
	1,000	3.66 ms	4.6	6.74 ms	8.4	—	—
	10,000	10.2 ms	12.8	13.3 ms	16.6	37.6 ms	47.0
PET/DINGO	10	33.9 ms	42.4	—	—	—	—
	1,000	36.4 ms	45.0	39.4 ms	49.2	—	—
	10,000	81.7 ms	102.1	84.7 ms	105.9	108.3 ms	135.4

Tabelle 1: Verarbeitungszeit bei Ende-zu-Ende Kommunikation

In **Tabelle 1** und **Abbildung 2** sind neben den Durchlaufzeiten der SandiaXTP Handimplementierung die entsprechenden Werte der mit EC und PET/DINGO erzeugten Implementierungen dargestellt, wobei für die Estelle Implementierungen jeweils drei verschiedene (bei der Übersetzung festgelegte) Obergrenzen für die Nutzdatenmenge einer Übertragung ("Limit" 10, 1000 und 10000 Bytes) unterschieden werden. Da in der Estelle-Spezifikation bei der Übertragung eines Pakets über einen Interaktionspunkt nicht nur die *genutzte* sondern diese *maximale Datenmenge* übertragen wird (siehe Abschnitt 4.1.2), hat die Obergrenze einen erhebliche Einfluß auf die Übertragungszeiten.

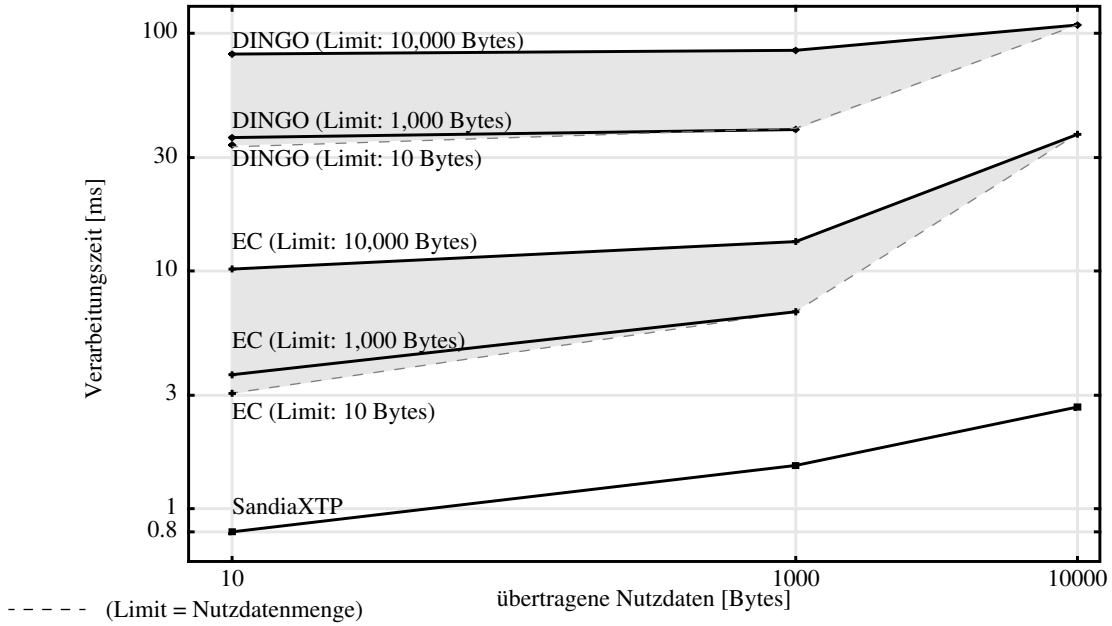

Abbildung 2: Doppelt-logarithmische Darstellung der Verarbeitungszeit bei Ende-zu-Ende Kommunikation (siehe Tabelle 1)

Die Werte für verschiedene Paketgrößen erlauben Rückschlüsse auf die Ursachen des Zeitverbrauchs bei der Paketverarbeitung. So dominiert bei Paketen mit 10 Byte Nutzdaten (Obergrenze ebenfalls 10 Byte) deutlich der Aufwand für die Verarbeitung der Paketheader gegenüber dem Kopieraufwand für die Nutzdaten. In diesem Bereich kann die von EC erzeugte Implementierung mit einem Zeitfaktor von 3.8 fast in den Leistungsbereich von SandiaXTP vorstoßen. Die von DINGO generierte Implementierung ist dagegen mehr als eine Zehnerpotenz langsa-

mer. Betrachtet man nun die Vergrößerung des Zeitverbrauchs beim Übergang zu 10,000 Byte Paketen, so erhält man ein Maß für den Kopieraufwand in den Protokollmaschinen. Hier zeigt sich bei SandiaXTP mit 1.88 ms Differenz zum 10 Byte Paket der Vorteil der zentralen Paketpufferung, durch die mit lediglich vier Kopieroperationen ein Paket von Ende zu Ende transportiert werden kann (siehe Abschnitt 3.1). Die Estelle-Spezifikation erfordert dagegen eine weitaus größere Anzahl von Kopieroperationen (siehe Abschnitt 3.2), die sich im Verhältnis zu SandiaXTP in der 18-fachen (34.5 ms bei EC) bzw. 40-fachen (74.4 ms bei DINGO) Steigerung der Übertragungsdauer bei großen Paketen widerspiegelt.

Die Ergebnisse machen deutlich, daß aus Estelle-Spezifikationen automatisch generierte Implementierungen bei kleinen Datenpaketen durchaus in die Größenordnung von gut optimierten Handimplementierungen vorstoßen können. Dies gilt besonders vor dem Hintergrund, daß die *zugrundegelegte Estelle-Spezifikation ohne Berücksichtigung von Effizienzaspekten erstellt wurde*. Gleichzeitig werden aber auch einige Schwächen von Estelle und den daraus abgeleiteten Implementierungen deutlich: Da Module das wesentliche Strukturierungsmittel von Estelle sind, werden gerade große Spezifikation sinnvollerweise in eine Vielzahl von Modulen zerteilt. Dies führt jedoch wegen der beschränkten Möglichkeiten für gemeinsamen Speicher zu vielen Nachrichtenübertragungen und damit zu vielen Kopieroperationen, was speziell bei großen Paketen die Effizienz merklich verschlechtert. Dies wird zusätzlich dadurch verschärft, daß Estelle keine Datenarrays variabler Größe kennt und daher immer Pakete maximaler Größe übertragen werden müssen (s.o.). Wir werden in Abschnitt 5 einige Entwicklungen aufzeigen, durch die diese Schwächen überwunden werden können.

4.2 Ein Estelle-Benchmark

Aufbauend auf die Untersuchungen der XTP Spezifikation wurde ein Estelle-Benchmark [3] entwickelt, der es gestattet, die automatische Implementierung derjenigen Estelle-Mechanismen isoliert und quantitativ zu bewerten, die sich als besonders laufzeitintensiv erwiesen haben. So können z.B. Modulinstanziierungen, Nachrichtenübertragungen mit unterschiedlich großen Argumenttypen, verschiedenen Auswahlsituationen usw. auch ohne Quellen für Compiler oder Laufzeitbibliothek (wie z.B. bei EC) quantitativ bewertet werden.

Bisher liefert dieser Benchmark nur Einzelwerte für die verschiedenen Estelle-Mechanismen. Wir erarbeiten zur Zeit jedoch ein Gewichtungsschema, das es erlauben soll, aus diesen Einzelwerten Prognosen für die Leistung der Compiler für verschiedene Anwendungsgebiete abzuleiten. Erste Ergebnisse für den Vergleich zwischen EC und DINGO bestätigen jedoch die relativen Ergebnisse aus dem XTP-Vergleich.

5 Ausblick

Anhand des Hochleistungsprotokolls XTP und der FDT Estelle wurde untersucht, inwieweit automatisch aus einer formalen Spezifikation generierte Implementierungen in ihrer Leistungsfähigkeit mit Handimplementierungen konkurrieren können. Ausgangspunkt für die Untersuchungen waren eine Estelle-Spezifikation von XTP 4.0 sowie die SandiaXTP-Handimplementierung dieses Protokolls. Es konnte gezeigt werden, daß automatisch generierter Kode fast in den Leistungsbereich effizienter Handimplementierungen vorstoßen kann. Dies ist insbesondere auch deshalb bemerkenswert, weil bei der automatischen Implementierung — anders als bei der bereits umfassend optimierten Handimplementierung — noch ein erhebliches ungenutztes Potential zur Leistungsverbesserung besteht.

Um dieses Potential zu quantifizieren und zu nutzen, soll in zukünftigen Arbeiten die Optimierung der automatischen Implementierung gezielt vorangetrieben werden. Dabei gibt es verschiedene Ansatzpunkte, die zusammen verfolgt werden sollen. Ein erster Ansatzpunkt ist die Optimierung der Estelle-Spezifikation von XTP. Wie bereits diskutiert, wurden Performanceaspekte bei der Entwicklung der für die Messungen verwendeten Estelle-Spezifikation bislang nicht berücksichtigt. So bedingt z.B. die komplexe Modulhierarchie des Protokollautomaten einen hohen internen Kopier- und Synchronisationsaufwand in der generierten Implementierung. Ferner soll der Einfluß einzelner Spezifikations-Stilelemente auf das Leistungsverhalten durch Estelle-Benchmarks und den Einsatz des Werkzeugs PATO quantifiziert werden, um so eine gezielte Optimierung zu ermöglichen. Ein zweiter Ansatzpunkt zur Optimierung besteht in der Verbesserung des generierten Kodes. Ein experimenteller Kodegenerator, der die Untersuchung der Auswirkungen verschiedener Implementierungsverfahren auf das Leistungsverhalten erlaubt, befindet sich in der Entwicklung. Erste Leistungsvergleiche mit den in dieser Arbeit genannten Estelle-Compilern belegen diesen Ansatz. Ein weiterer Ansatzpunkt ist die Weiterentwicklung der FDT Estelle. Beispielsweise kann der Kopier- und Synchronisationsaufwand durch gezielte Spracherweiterungen [1] erheblich reduziert werden.

6 Literatur

[1] Bredereke, J., Gotzhein, R.: *Increasing the Concurrency in Estelle*, in: R. L. Tenney, P. D. Amer, M. U. Uyar (Hrsg.), *Formal Description Techniques VI*, North-Holland, 1994

[2] Budkowski, S. (Hrsg.): *Formal Specification, Validation and Performance Evaluation of the Xpress Transfer Protocol (XTP)*, Research Report No. 931004, Institute Nationale des Télécommunications, Evry, Frankreich, 1993

[3] Dahl, S.: *Performancevergleich von Protokollimplementierungen*, Diplomarbeit, Universität Kaiserslautern, Oktober 1996

[4] Dembinski, P., Budkowski, S.: *Specification Language Estelle*, in: M. Diaz et al. (eds.), *The Formal Description Technique, Estelle*, North-Holland, 1989, pp. 35-75

[5] Gotzhein, R., Rößler, F., Thees, J.: *Towards "Open Estelle"*, in: 6. *GI/ITG Fachgespräch "Formale Beschreibungstechniken für verteilte Systeme"*, Erlangen-Nürnberg, May 1996

[6] ISO/TC97/SC21: *Estelle - A Formal Description Technique Based on an Extended State Transition Model*, ISO/TC97/SC21, IS 9074, 1989

[7] Penner, H.: *Erstellung eines Software-Monitors zur Analyse automatisch generierter Protokollimplementierungen*, Diplomarbeit, Universität Kaiserslautern, Januar 1996

[8] Richard, J. L., Claes, T.: *A Generator for C-Code for Estelle*, in: M. Diaz et al (eds.), *The Formal Description Technique Estelle*, North-Holland, 1989, pp. 397-420

[9] Sandia National Laborities: *SandiaXTP Reference Manual*, Release 1.4, Livermore, California, USA, September 1995

[10] Sandia National Laborities: *SandiaXTP — An Object-Oriented Implementation of XTP 4.0 Derived from the Meta-Transport Library: User Manual*, Release 1.4, Livermore, California, USA, October 1995

[11] Sijelmassi, R., Strausser, B.: *The PET and DINGO tools for deriving distributed implementations from Estelle*, Computer Networks and ISDN Systems, Vol. 25, No. 7, 1993, pp. 1115-1130

[12] Thees, J., Gotzhein, R.: *Leistungsbewertung automatisch generierter Protokollimplementierungen mit Estelle — eine Bestansaufnahme*, Interner Bericht 290/97, Fachbereich Informatik, Universität Kaiserslautern, 1997

[13] XTP Forum, *Xpress Transport Protocol Specification*, XTP Rev. 4.0, XTP Forum, Santa Barbara, USA, 1995

Anhang A: Randbedingungen der Leistungsmessungen

Hardware:	SUN SPARCstation-20, 192 MB, 2 Prozessoren (75 MHz)
Betriebssystem:	SUN Solaris 2.5
C/C++ Compiler:	GNU gcc 2.7.2 mit Option "-O2 -msupersparc"
Estelle-Compiler:	EC 3.1, PET/DINGO 1.0, UHH 1.4