[Informatie, Vol. 25, No. 1, pp. 5-11, January 1983]

COMMUNICATIE PROTOCOLLEN
ontwerp, analyse en standaardisatie

dr.ir. G.J. Holzmann

Samenvatting

De computer communicatie heeft zich in relatief korte tijd tot een buitengewoon belangrijk nieuw vakgebied binnen de informatica ontwikkeld. Op informele wijze wordt in dit artikel getracht de lezer inzicht te bieden in de boeiende, soms amusante, problemen die ontstaan als men een werkelijk eenduidige informatie uitwisseling tussen ver van elkaar verwijderde computersystemen mogelijk wil maken. Een van de belangrijkste doeleinden van dit artikel is tevens om te wijzen op het gevaar van voortijdige standaardisatie op communicatie protocollen en ontwerpmethoden. Het belang, maar ook de complexiteit, van de ontwikkeling van formele protocol analyse methoden wordt toegelicht. Het artikel besluit met een voorstel voor de opzet van meer serieus onderzoek naar communicatie protocollen.

Keywords: communicatie protocol, data netten, computer netwerken, protocol ontwerp, protocol analyse.

Inleiding

In 1793 kreeg Claude Chappe, een Frans ingenieur, van zijn regering de opdracht om een systeem te ontwerpen waarmee berichten zo snel mogelijk van Lille naar Parijs en terug, over een afstand van meer dan 200 kilometer, konden worden doorgegeven. [1]

Telefoon was er uiteraard in die tijd nog niet; die kwam pas in 1876. Achteraf weten we dat het principe van de elektrische telegrafie al in 1753 beschreven was, in een ingezonden brief in de *Scots Magazine* van een mysterieuze "C.M." wiens identiteit nooit met zekerheid is vastgesteld [2]. Maar Chappe kende die brief vrijwel zeker niet. De eerste bruikbare variant van de elektrische telegrafie (de elektro-magnetische telegraaf van Cooke en Wheatstone) werd bovendien pas in 1840 voor het eerst gebruikt.

Maar Chappe had een hele originele oplossing voor het communicatie probleem. Hij liet 25 houten torens bouwen, met bovenop elke toren een soort seinpaal of "semaphore" (Fig. 1).

Figuur 1

Die semaphore kon in verschillende standen worden gezet en aan elke stand kende Chappe een betekenis toe. Elke toren werd vervolgens toegerust met een verrekijker, en met een beambte die tot taak had de stand van de semaphores op de naastliggende torens in de gaten te houden en zo nodig over te nemen op de semaphore van zijn toren.

Berichten konden zo met grote snelheid in de vorm van semaphore standen van toren naar toren worden doorgeseind. Chappe's "optische telegraaf" was dan ook onmiddellijk een succes, ook al bleek de elektro-magnetische telegrafie op den duur betrouwbaarder en sneller te zijn. Voordat deze systemen echter in gebruik kwamen was het communicatie-net van de Franse regering al gegroeid tot 556 semaphore stations, die tezamen een afstand van 4800 kilometer konden overbruggen. Het principe werd bovendien overgenomen door een groot aantal andere landen, waaronder Engeland, Duitsland, Amerika en Rusland [3].

In het systeem van Chappe werd gebruik gemaakt van een eenvoudige sein-code voor het versturen van berichten en voor de onderlinge communicatie van de beambten. Door schade en schande heeft men in de loop der tijd ontdekt dat de snelle uitwisseling van gecodeerde berichten over communicatie lijnen een apart soort problemen oproept.
De elektro-magnetische telegrafie werd, bijvoorbeeld, al snel gebruikt ter beveiliging van het treinverkeer, jammergenoeg bij uitstek op gevaarlijke baanvakken, zoals enkelspoors verbindingen. Het is amusant om te lezen welke misverstanden er soms ontstonden tussen de seinwachters. We kunnen nu zien dat ze veroorzaakt werden door de ondeugdelijkheid van zowel de communicatie-middelen als van de afspraken over het gebruik van die middelen. Maar de trein-ongelukken die hier soms het gevolg van waren doen ons het lachen weer snel vergaan. Ze tonen aan hoe noodzakelijk het was om een goed inzicht te krijgen in de problematiek van een eenduidige berichten-uitwisseling.

Computer Communicatie

In dit artikel zullen we het niet alleen hebben over de communicatie problemen van seinwachters of semaphore-beambten. Praktisch dezelfde problemen ontstaan wanneer we een berichten-uitwisseling tussen computers op gang willen brengen. De transmissie-snelheden zijn hierbij uiteraard veel groter dan in het systeem van Chappe. De eisen die aan de doelmatigheid van het gebruik van soms dure transmissie-kanalen worden gesteld liggen ook hoger.

Computer-communicatie is natuurlijk al lang geen science fiction meer; het is realiteit. Wie buiten de normale kantooruren een bedrijf belt loopt een goede kans een harde fluittoon te horen, in plaats van een vriendelijke stem. De beller krijgt dan onbedoeld een computer aan de lijn. uitgebreid "bij te praten" over de ontwikkelingen van de dag.

Figuur 2

Als de verbinding eenmaal tot stand is gebracht kunnen de computers met een duizelingwekkende snelheid gegevens uitwisselen. De tekst van dit artikel (ongeveer 35.000 lettertekens) kan over een gewone telefoonlijn binnen vijf minuten, en over een speciale data-lijn zelfs in vijf seconden van de ene computer naar de andere gestuurd worden.

Het aantal verbindings-netwerken dat speciaal voor computer-communicatie wordt ingericht groeit nog steeds. Een van de meest bekende computer netwerken is het Amerikaanse ARPA (Advanced Research Projects Agency) net, dat al ruim 15 jaar in gebruik is. Naast het telefoonnet en het telexnet kent Nederland sinds kort ook een eigen openbaar, nationaal datanet dat beheerd wordt door de PTT [4]: "Datanet no. 1". Veel

bedrijven (banken, luchtvaartmaatschappijen, grote industrieen) beschikken bovendien al jaren over eigen netwerken, die soms tot ver buiten onze grenzen reiken.

Dat niet alle problemen van de computer communicatie inmiddels zijn overwonnen hopen we met dit stuk duidelijk te kunnen maken. Fouten in de communicatie kunnen er bijvoorbeeld toe leiden dat hele netwerken onbruikbaar worden voor communicatie. Ze kunnen er toe leiden dat computers elkaar verlammen en volkomen stil komen te staan (deadlock). Fouten kunnen ook, net als bij het treinverkeer, schade aanrichten, al spreekt de botsing van twee data-stromen in een transmissiekanaal wellicht iets minder tot de verbeelding dan een botsing van twee treinen op enkelspoor.

In het kort zal nu uitgelegd worden welke problemen er precies spelen, aan welke voorwaarden de oplossingen moeten voldoen en in welke richting deze gezocht moeten worden.

Het Communicatie-Protocol

De regels die de computers volgen bij het opbouwen van verbindingen en het tot stand brengen van de gegevens-uitwisseling worden een "protocol" genoemd [5]. Formeel gezien bestaat zo'n protocol uit drie onderdelen. Het specificeert:

- de *berichtenset*: alle berichten waarvan de computers de betekenis moeten kennen,
- de *bericht-struktuur*: de wijze waarop een bericht wordt gecodeerd en verzonden, en
- de *spelregels* (procedure regels) die voor de overdracht zijn vastgesteld.

De *berichtenset* bestaat in de regel uit tenminste twee soorten berichten: controle-codes (voor lijnbesturing) en data. De "betekenis" van berichten is vaak context afhankelijk en moet worden afgeleid uit de procedure-regels.

De *struktuur* van berichten is afhankelijk van het abstractie niveau waarop het protocol wordt gedefinieerd. Als we een telefoon verbinding als voorbeeld nemen dan moeten we in de eerste plaats een codering van berichten in geluidssignalen bepalen. We zullen daarbij rekening moeten houden met de eigenschappen van de telefoonlijn. Zo heeft het weinig zin om geluidsfrequenties te kiezen die boven de 3400 Hz of onder de 300 Hz liggen, omdat die signalen niet of sterk verzwakt door een telefoon verbinding komen.
Nu werkt een digitale computer natuurlijk zelf ook met codes. De computer codeert informatie in nullen en enen. Dit vereenvoudigt het coderingsprobleem in de computer communicatie aanzienlijk, want we hoeven nu alleen nog maar codes voor nullen en enen te kiezen om transport mogelijk te maken.

Een signaal-element met maar twee mogelijke waarden heet een "binary digit" afgekort tot "bit." [6] Maar alleen met nullen en enen beginnen we nog niet veel. We kunnen aan die beperking ontkomen door bits samen te voegen in groepen en aan elk groepje een aparte betekenis toe te kennen. Een groep van acht bits wordt een "octet" of een "byte" genoemd. Met een byte kunnen we nu al $2 \times 8 = 256$ verschillende codes maken. Een niveau boven de codering in geluidssignalen kunnen we nu een codering in bytes definieren. Bytes kunnen op hun beurt weer gegroepeerd worden in "frames" en frames kunnen weer worden samengevoegd in "packets." Zo wordt stap voor stap een steeds

hoger abstractie-niveau in de codering bereikt.

Dezelfde hierarchie die voor codering wordt gebruikt kunnen we toepassen voor het opstellen van de *spelregels* voor het gebruik van communicatie-lijnen. De meeste spelregels zijn eenvoudig. Zo'n regel kan zijn dat een telefoon verbinding alleen verbroken kan worden als hij eerst gemaakt is. Voor een mens spreekt dat vanzelf, voor een computer natuurlijk niet. Het venijn van de communicatie problemen zit ook niet zozeer in de complexiteit van de spelregels, maar in de combinatie van een overvloed aan regels. Een voorbeeld kan dit illustreren.

Deadlock

Computer "Alice" is bezig met het versturen van de eerder genoemde tekst naar een computer genaamd "Rabbit" [7]. Alice en Rabbit reserveren beiden een deel van hun geheugen voor de communicatie (een "buffer"). Als Alice de tekst sneller zendt dan Rabbit die kan verwerken loopt de buffer van Rabbit vol. In dit geval kan Rabbit aan Alice met behulp van controle-codes vragen het transport tijdelijk stil te zetten. Alice hoeft vervolgens niet stil te zitten terwijl Rabbit zijn buffer leegt. Zij kan van de situatie gebruik maken door haar buffer alvast te vullen met de tekst die nog naar Rabbit moet.

Nu kan het gebeuren dat Alice zover op Rabbit vooruit gaat lopen dat zij alle vrije bufferruimte verbruikt zodat zelfs het bericht van Rabbit dat het transport hervat kan worden niet meer opgenomen kan worden. Dit laatste bericht gaat verloren en als gevolg daarvan zullen de twee computers onherroepelijk vastlopen.

De oorzaak van deze deadlocks is soms erg moeilijk te achterhalen, niet in de laatste plaats omdat ze van zoveel toevalligheden afhangen: in dit geval de relatieve snelheden van de twee computers. Een complicerende factor is ook dat de twee computers vaak door verschillende mensen geprogrammeerd zijn, en dus verschillende opvattingen kunnen hebben over wat hun communicatie-partners behoren te doen om een probleem tot een oplossing te brengen.

Het komt maar zelden voor dat een deadlock het gevolg is van het falen van de computer hardware. In verreweg de meeste gevallen ligt de fout besloten in de procedure beschrijvingen die de protocol ontwerper heeft opgesteld.

Protocol Funkties

Een protocol ontwerper moet over een goed voorstellingsvermogen beschikken om de computer in staat te kunnen stellen adequaat te reageren op alle bizarre storingen die een berichten uitwisseling kunnen treffen. Geen enkel transmissie kanaal is foutloos. De verzonden berichten kunnen vervormd of gestoord worden, en het kan voor een ontvanger soms onmogelijk zijn om aan een binnenkomend signaal nog enige betekenis toe te kennen. In zo'n geval moet de communicatie toch weer snel hersteld kunnen worden. De ontvanger moet de zender kunnen laten weten welke berichten nog wel en welke berichten niet meer correct ontvangen zijn, en het gegevens transport moet weer feilloos op gang kunnen komen alsof er in het geheel geen verstoring is geweest.

Een communicatie protocol moet dus een groot aantal verschillende funkties vervullen. Het moet zorgen voor:

- het ordelijk opbouwen en weer afbreken van verbindingen (bijvoorbeeld het volgens de regels inkiezen op het telefoonnet, met de juiste interpretatie van de diverse signalen zoals informatietonen, bezettoon, kiestoon etc.),

- de codering van berichten in het afgesproken formaat (bijvoorbeeld in packets, frames, octets, of bits),

- de synchronisatie van zender en ontvanger (bijvoorbeeld door ervoor te zorgen dat een snelle zender een trage ontvanger niet kan overspoelen met berichten),

- het detecteren en herstellen van transmissie-fouten (foutbeheersing),

- de adressering en routering van berichten in netwerken.

Bovendien moet het protocol de computer in de gelegenheid stellen om adequaat op meer ingrijpende fouten te reageren (het uitvallen van een verbinding, een "crash" van de computer aan de andere zijde van de lijn, het falen van een randapparaat, etc.)

Verrassingen

Het verradelijke van protocol problemen is dat oplossingen in veel gevallen zo voor de hand schijnen te liggen. Een eenvoudige oplossing voor een eenvoudig probleem werkt in de meeste gevallen ook uitstekend. Maar het werkt nu juist niet in de uitzonderlijke gevallen. Onvolledigheid is een van de grootste problemen voor de protocol ontwerper. De ontwerper levert een ware slag met een onvoorstelbaar grote hoeveelheid interaktie patronen die bij de samenwerking van onafhankelijk werkende systemen kan voorkomen.

Het aantal mogelijke gebeurtenissen in een protocol van enige omvang blijkt zelfs zo overweldigend groot te zijn dat ook de krachtigste computersystemen bij lange na niet in staat zijn om door middel van een eenvoudige analyse alle mogelijkheden stuk voor stuk te bekijken. Ze zouden er vele tientallen jaren voor nodig hebben. Het verschijnsel heet "state space explosion": een explosie van het aantal toestanden waarin samenwerkende processen terecht kunnen komen. Het probleem voor de protocol onwerper is dan ook dat er veel meer kan gebeuren dan hij kan voorzien.

Goedbedoeld maar onvoorzien

Een aardige illustratie van dit soort "verrassingen" is het treinongeluk dat op de avond van de 8ste november 1900 in Duitsland plaatsvond [8]. In een dichte mist rijdt een sneltrein (no. 42) op het baanvak Muhlheim - Offenbach door een rood sein. Gelukkig ziet de treinbestuurder het signaal in het voorbijgaan en stopt alsnog zijn trein. Langzaam rijdt hij vervolgens terug om te wachten tot het sein op groen zal springen.

Het treinverkeer op deze lijn was beveiligd met "blok-signalering." Bij zo'n systeem is het vrijwel uitgesloten dat twee treinen zich tegelijkertijd in hetzelfde "blok" bevinden (een "blok" is een stuk spoorlijn). Botsingen zijn dus normaal gesproken niet mogelijk.
De toegang tot elk blok wordt bewaakt door een sein. Wanneer een trein een blok

binnenrijdt springt dit sein op rood, en het wordt pas weer groen als de trein het blok aan de andere zijde verlaat. Met nadruk vermelden we dat er niets mis is met deze signaleringsmethode. Het systeem heeft in de loop der jaren bewezen bijzonder veilig te zijn en wordt nog steeds toegepast.

Maar toch gebeurde er op die bewuste avond van de 8ste november een ongeluk waarbij 12 doden vielen. Hoe kon dat gebeuren ?

Figuur 3

Het rode sein was voor de bestuurder van trein 42 een aanwijzing dat in het blok dat hij per abuis was binnengereden nog een trein moest zijn. De bestuurder wist dat dit niet de bedoeling was en hij verliet dit blok dan ook weer zo snel mogelijk door achteruit te rijden. Strikt genomen voorzagen de regels niet in deze manoevre; voor de treinbestuurder lag hij niettemin voor de hand.

Met het passeren van het rode sein (vooruit rijdend) verliet trein 42 een blok. We zullen dit "blok A" noemen. Bij het verlaten van blok A reed trein 42 natuurlijk ook een nieuw blok binnen. We noemen dit "blok B".

De situatie wordt nog iets ingewikkelder omdat de signalering maar ten dele was geautomatiseerd. Bij het sein dat trein 42 passeerde zat een seinwachter die onder meer tot taak had om te noteren wanneer een trein een blok verliet en het daarop vrij te geven voor de eerstvolgende trein. De seinwachter voerde deze taak getrouw uit en gaf blok A vrij voor de achterop komende trein no. 238, op het moment dat trein 42 hem voorbij reed. Die seinwachter kon door de dichte mist niet zien dat trein 42 was gestopt. Erger nog, de seinwachter kon ook niet zien dat trein 42 blok A weer achteruit was binnengereden.

Er waren nu dus twee treinen in blok A (trein 42 en trein 238) en een trein in blok B (de voorganger van trein 42). Uit de informatie die de seinwachter had binnengekregen (het passeren van trein 42) kon hij echter alleen maar concluderen dat er twee treinen in blok B (trein 42 en zijn voorganger) zaten en maar een trein in blok A (trein 238),

Hij realiseerde zich dat dit gevaarlijk was, maar ook weer niet rampzalig: er waren gewoon twee treinen in een blok die allebei met ongeveer dezelfde snelheid in dezelfde richting reden. Als trein 42 zijn voorganger niet zou inhalen zou alles vanzelf weer goed komen bij het eerstvolgende blok dat de twee treinen moesten passeren. Wat voor hem wel duidelijk moet zijn geweest was dat het rode sein bij zijn wachtpost voorlopig ook rood moest blijven totdat zeker was dat alle "twee" de treinen blok B weer zouden hebben verlaten.

De seinwachter verhinderde trein 42 de toegang tot blok B met het vaste voornemen die blokkering te handhaven totdat diezelfde trein 42 blok B aan de andere zijde zou verlaten.

Vanaf dat moment was de botsing tussen de wachtende trein 42 en de aanstormende trein 238 niet meer te vermijden, en ze vond dan ook plaats. Het meest tragische is nog dat niemand iets verweten kon worden. Het gedrag van alle betrokkenen was begrijpelijk en

voor zover er regels waren ook volgens de regels. Het was eenvoudig niet voorzien dat er ooit een situatie kon ontstaan waarin een treinbestuurder zich gedwongen zag om eerst een blok te verlaten en er vervolgens weer in terug te keren.

Standaardisatie

Het spreekt vanzelf dat niet voor elke mogelijke combinatie van computers een ander protocol gedefinieerd kan worden. Een aantal organisaties houdt zich dan ook al geruime tijd bezig met de standaardisatie van de procedures voor informatie uitwisseling tussen computer systemen.

De twee belangrijkste standaard-organisaties op dit gebied zijn de ITU en de ISO.

De International Telecommunication Union (ITU) kent als belangrijkste orgaan het Comite Consultatif International Telegraphique et Telephonique (CCITT). De International Standards Organization (ISO) heeft als leden o.a. het National Bureau of Standards (NBS) en het American National Standards Institute (ANSI). Jammergenoeg werken de verschillende instanties niet altijd even goed samen. Er zijn vele "standaard protocollen" in omloop gebracht die door even zovele verschillende organisaties worden ondersteund. Daarnaast hanteert vrijwel elke grotere computer-industrie (IBM, Xerox, DEC) weer eigen afwijkende standaarden.

Niettemin bestaat op een aantal belangrijke punten overeenstemming. Zo is men het er over eens dat protocollen bij voorkeur een "gelaagde struktuur" moeten hebben. Protocol funkties kunnen daartoe gesplitst worden over een aantal niveaus, zodanig dat de protocol funkties op de hogere niveaus kunnen voortbouwen op de funkties die op een lager niveau zijn gerealiseerd.
Dit is heel goed te vergelijken met het principe van de "gelaagde machine" uit de theorie van het gestruktureerde programmeren. De protocol funkties op de hogere niveaus worden geimplementeerd met behulp van de lagere niveaus en verlenen een dienst, zoals foutbeheersing of synchronisatie, die de service van de lagere niveaus aanvult of uitbreidt.

De ISO heeft volgens dit principe een referentie model uitgewerkt voor de splitsing van protocol funkties over een zevental hierarchische niveaus. Dit "OSI-model" [9] wordt veel gebruikt als uitgangspunt voor nieuwe protocol-standaarden, al neemt een ieder de vrijheid het model op eigen wijze te interpreteren.

Een belangrijke standaard in wording is ook de bekende CCITT aanbeveling X.25 [10], die volgens hetzelfde principe is opgesplitst in een drietal lagen, globaal overeenkomend met de eerste drie niveaus in het OSI model.

Figuur 4

Specifikatie Methoden

Naast protocol standaarden zijn er ook specifikatie methoden nodig. Voor de specifikatie van protocollen wordt gebruikt gemaakt van speciale "talen." Zo’n taal beantwoordt natuurlijk alleen aan zijn doel als er volledige, ondubbelzinnige, en liefst ook gestruktureerde

beschrijvingen mee gemaakt kunnen worden. In het bijzonder zal de taal geschikt moeten zijn voor de geintegreerde beschrijving van de drie basis-elementen van het communicatie protocol: *berichtenset*, *bericht-struktuur*, en *procedure regels*.

In de kleine dertig jaar ervaring die het vakgebied inmiddels rijk is heeft men een goed inzicht kunnen verwerven in de eigenschappen die een taal moet hebben voor het doelmatig aanbrengen en hanteren van de gevraagde struktuur. Methoden waarmee complexe data en control-flow [11] strukturen kunnen worden opgesplitst in kleinere, overzichtelijke modules die net als protocol funkties over een aantal hierarchische lagen kunnen worden verdeeld, mogen worden verondersteld tot de standaard uitrusting van elke goed opgeleide programmeur te behoren.

Deze kennis komt bij het protocol specifikatie probleem goed van pas. Maar een protocol beschrijft meer dan data-strukturen en control-flow. Het belangrijkste deel van het protocol betreft immers de *interaktie* van machines, en juist dit interaktie aspect blijkt nu moeilijk te vangen in een beschrijving.

De Remmende Voorsprong

Er zijn enkele pogingen gedaan om protocol specifikatie talen te ontwikkelen waarmee het interaktie aspekt beschreven kan worden. Jammer is dat zulke aanzetten tot nieuwe specifikatie talen soms veel te snel tot standaard worden verheven. Iets soorgelijks dreigt te gebeuren met een grafische taal die door een studiegroep van de CCITT wordt bestudeerd [12]. Ondanks het feit dat deze taal, nog aan geen van de hierboven genoemde eisen van volledigheid, ondubbelzinnigheid en struktuur beantwoordt en zelfs twee van de drie basiselementen van het protocol ongespecificeerd laat [13], blijken veel instanties de taal inmiddels als "de facto" standaard te hebben omarmt.

Een onbedoeld neveneffekt van deze voortvarendheid in het het standaardiseren op een proeftaal is dat de noodzakelijke verdere ontwikkeling van specifikatie talen ernstig dreigt te worden belemmerd. Noodgedwongen zal men immers trachten om de nieuwe specifikatie methoden compatibel te doen zijn met de eerste voorbarige standaard. In de techniek staat dit verschijnsel bekend onder de naam "neerwaartse compatibiliteit." In de industrie wordt ook wel de uitdrukking "de remmende voorsprong" gebruikt [14].

De computer wetenschappen hebben al eens eerder met de gevolgen van de neerwaartse compatibiliteit te maken gehad. Toen de gestruktureerde programmeertalen van de Algol-familie in het begin van de zestiger jaren beschikbaar kwamen bleken velen zich al zozeer aan de eerste probeersels van de vijftiger jaren te hebben gebonden dat zij de nieuwe ontwikkelingen in het geheel niet meer mee konden maken. Nu, zo’n twintig jaar later, moeten we constateren dat onnoemelijk veel bedrijven zich gedwongen zien om door te blijven werken met de producten van de allereerste generatie, uitsluitend omdat zij te vroeg standaardiseerden [15].

Protocol analyse

Het belangrijkste probleem dat opgelost moet worden om een protocol te kunnen analyseren is het eerder genoemde probleem van de "toestands explosie." Om dit te bereiken moeten we een vereenvoudigd *model* van het protocol kunnen maken dat alle eigenschappen die we willen analyseren omvat, en dat ons tevens in staat stelt te abstraheren van de overige aspekten.

De twee modellen die hiervoor het meest worden gebruikt zijn "Finite State Machines" [16] (FSM) en "Petri Netten" [17]. Beide modellen stellen ons in staat om te abstraheren van alle interne operaties die nodig zijn om een protocol te doen funktioneren, ten gunste van het interaktie-aspekt.

Bij het _Petri Net_ wordt deze abstraktie zelfs zover doorgevoerd dat het model beperkt wordt tot het gebruik van een enkel signaalelement. De berichtenset van het protocol kan zo worden beperkt tot een enkel element: het "token". De verwachting is dat dit grotere abstraktie vermogen van het Petri Net tot een krachtiger analyse methodiek zal voeren. De onderzoekers die zich aan dit probleem hebben gezet zijn er echter nog niet in geslaagd om deze ook belofte waar te maken.

Er zijn wel interessante resultaten geboekt met analyse methoden die zich baseren op het _FSM_ model. Het FSM model is welhaast klassiek te noemen: er is erg veel studie aan verricht en het wordt dan ook behandeld in elk handboek over automaten. De verwachting is hier dat het grote arsenaal aan kennis dat beschikbaar is op de een of andere wijze moet kunnen worden toegepast op het protocol analyse probleem. Voor het probleem van de toestands explosie bieden de klassieke resultaten echter nog maar weinig houvast.

Validatie Algebra

Om een protocol te kunnen analyseren kan men ook trachten een wiskundig model te vormen in de vorm van een algebra. Het protocol analyse probleem wordt dan "vertaald" in een algebraisch probleem dat met de hulp van computers op te lossen is. Er is nog weinig bekend over de preciese struktuur die zo’n protocol validatie algebra moet hebben en over de rekenregels die men in de protocol analyse nodig heeft. Toch zijn er al enkele bemoedigende resultaten met deze benadering geboekt [18]. Vooral deze laatste methode belooft zich dan ook tot een krachtig hulpmiddel voor protocol analyses te kunnen ontwikkelen.

Reikwijdte van de Analyse

Een belangrijke vraag voor protocol analyse betreft het soort fouten dat we op het spoor willen komen. Ook hier steekt het probleem van de onvolledigheid weer de kop op. Het zou al te naief zijn om een protocol dat is geanalyseerd op de aanwezigheid van deadlock "correct" te noemen. De kans is immers groot dat de protocol fouten juist zitten in punten die niet in de analyse werden betrokken, zoals een buffer overflow, of de eindeloze herhaling van een vruchteloze berichten uitwisseling.

Er is in de loop der jaren een soort consensus gegroeid over de fouten die een bruikbare analyse methode in een ontwerp protocol moet kunnen vinden. Daaronder vallen

uiteraard de deadlocks, maar ook specifieke gevallen van onvolledigheid, zoals de ontvangst van een bericht dat niet verwacht wordt, de onbereikbaarheid van gedefinieerde systeemtoestanden, aannames over de relatieve snelheden van computers, het niet kunnen herstellen van de protocol fouten van een communicatie partner, of het niet kunnen herstellen van transmissie fouten.

Lang niet elke bestaande analyse methode is in staat om een protocol op al deze aspekten te beoordelen. Vaak zal men de complexiteit van de modelvorming moeten afwegen tegen de kracht van een analyse en de tijd die ervoor nodig is.

Het Testen van een Protocol

Het ontwerp van comunicatie protocollen is een specialisme aan het worden waar alleen grotere software leveranciers zich nog aan wagen. Een belangrijk nieuw probleem voor protocol gebruikers is nu om te beoordelen in hoeverre deze produkten aan de door hen gestelde eisen voldoen. Meestal zal men eisen dat een protocol implementatie zich conform de specifikaties van een internationale standaard gedraagt. Het kan echter buitengewoon lastig zijn om door middel van een praktijk test te controleren of conformiteit werkelijk in alle opzichten is gegarandeerd.

Engeland en Frankrijk hebben het probleem niettemin aangevat en zijn onlangs gestart met een gezamelijk poging om "testcentra" op te zetten waar producenten hun protocol op de conformiteit met een standaard kunnen laten controleren [19]. Juist omdat zo’n controle zelfs bij benadering niet volledig zal kunnen zijn (een rechtstreeks gevolg van het eerder genoemde probleem van de "toestands explosie") bestaat de vrees dat de testcentra snel hun geloofwaardigheid zouden kunnen verliezen.

De Nederlandse PTT wordt wel eens verweten dat zij de problemen op dit gebied een beetje onderschat. Zo stelde zij tegenover de lijvige rapporten van de Franse en Engelse onderzoekers een publikatie van 14 paginas, waarin binnen 2 bladzijden wordt uiteengezet hoe de werking van de protocol-funkties op de tweede en derde hierarchische laag van het gecompliceerde X.25 protocol zou kunnen worden getest [20]. Het doorstaan van deze test is voorwaarde voor de toelating tot het Nederlandse datanet.

De conclusie is haast onvermijdelijk dat het nieuwe PTT netwerk DN-1 binnenkort zal kunnen worden verlamd met officieel door de PTT goedgekeurde produkten. Om dit te voorkomen zou de PTT zich natuurlijk heel goed aan kunnen sluiten bij de Frans-Engelse onderzoeksgroep, of (liever nog) zelf het initiatief kunnen nemen tot het verrichten van gericht onderzoek naar protocol testmethoden. [21]

Hierboven is uiteengezet welke problemen zullen moeten worden opgelost om effektief communicatie protocollen te kunnen ontwerpen en analyseren. De minimale voorwaarden waaraan bijvoorbeeld specifikatie talen zullen moeten gaan voldoen zijn hierboven al heel in ’t kort aangegeven (’specifikatie methoden’).

De problemen zijn echter niet alleen van technische aard. We hebben het gevaar van de voortijdige standaardisatie al uitvoerig besproken. In Nederland wordt nog relatief weinig onderzoek naar communicatie protocollen gedaan. Voor een deel is dit het gevolg

van de bezuinigingen die het wetenschappelijk onderwijs worden opgelegd. Maar voor een deel komt het ook doordat de problemen worden onderschat. Met name voor de Nederlandse PTT, die met de in bedrijfstelling van het eerste openbare Nederlandse Datanet DN-1 rechtstreeks met de problemen wordt geconfronteerd, is een taak weggelegd bij het initieren, stimuleren, en zo mogelijk financieren van nieuw onderzoek.

NOTEN

1) Claude Chappe leefde van 1763 - 1805.

2) Volgens sommige bronnen stond de initialen "C.M." voor een zekere Charles Marshall uit het plaatsje Renfew.

3) "From semaphore to satellite," International Telecomunication Union, jubileumboek, Geneve 1965.

4) Een beschrijving van het netwerk is opgenomen in het blad "Het PTT-Bedrijf" van februari 1982.

5) Een meer uitgebreide behandeling van communicatie protocols kan worden gevonden in: L. Pouzin, and H. Zimmerman, "A tutorial on protocols," Proceedings of the IEEE, Vol. 66, No. 11, November 1978, pp. 1346-1370.

Voor een overzicht van specifikatie en validatie methoden wordt verwezen naar: P.M. Merlin, "Specification and validation of protocols," IEEE Transactions on Communications, Vol. COM-27, pp. 1761-1780.

6) De afkorting "bit" stamt al uit 1948. Hij werd bedacht door J.W. Tukey van Bell Laboratories. Zie "A History of Computing Research at Bell Laboratories (1937-1975)," B.D. Holbrook & W.S. Brown, Bell Laboratories, Computing Science Technical Report No. 99, Maart 1982.

7) De meeste computers hebben namen. "Alice" en "Rabbit" zijn de namen van twee VAX-750 computers bij Bell Laboratories.

8) Dit ongeluk wordt beschreven in: A. Schneider & A. Mase, "Railway accidents of Great Brittain and Europe, their causes and consequences," David & Charles, Newton Abbot, (1968).

Meer voorbeelden zijn te vinden in:
- O.S. Nock, "Historic railway disasters," Ian Allan, London, (1966).
- L.T.C. Rolt, "Red for danger," David & Charles, Newton Abbot, (1976).
- R.B. Shaw, "A history of railway accidents," Vall Ballov Press Inc. (1978).
- W.S. Griswold, "Train wreck," Stephen Greene Press, Vermont, (1969).

9) ISO Reference Model for Open Systems Interconnection, meestal afgekort tot "OSI model". CCITT studygroup VII, "Draft Recommendation," March 1982, Melbourne, (101 pgs.).

10) Het protocol staat bekend onder de naam X.25, het nummer van de aanbeveling waarin de CCITT de specifikaties publiceerde. CCITT Yellow Book, Vol. VIII, Geneve, 1980.

11) De control-flow bepaald in welke volgorde en onder welke voorwaarden programma onderdelen door de computer moeten worden uitgevoerd.

12) "Specification and Description Language" (SDL), CCITT Yellow Book, Vol. VI, Fascicle VI.7, Z.101 - Z.105, Geneve, 1980, 1982.

13) Met de grafische vorm van SDL kan alleen een globale control-flow specifikatie van een protocol worden gemaakt. Niet gespecificeerd kunnen worden de berichtenset, de berichten struktuur en de protocol datastrukturen. SDL ondersteunt ook geen hierarchische opsplitsing van een protocol in deelfunkties.
Er wordt gewerkt aan een programmatische vorm die in principe krachtiger zou kunnen zijn.

14) Ook bij de invoering van de industriele robots kunnen bedrijven te maken krijgen met de gevolgen van zo'n "remmende voorsprong." Het is mogelijk dat de robots van de "eerste generatie" nog lang niet zijn afgeschreven als de tweede of derde generatie beschikbaar komt. Niet alleen zijn snelle investeerders dan in het nadeel ten opzichte van de trage, de markt voor nieuwe robots kan zo klein worden dat een verdere ontwikkeling geremd wordt.

15) Een uitvoeriger motivatie van dit punt is ook te vinden in de inaugurele rede van Dr.ir. R.T. Boute, "Van gisteren op morgen", uitgesproken op vrijdag 22 oktober 1982, bij de aanvaarding van het ambt van gewoon hoogleraar in de Informatica aan de Katholieke Universiteit te Nijmegen (zie bv p. 10 ev.).

16) Een overzicht van deze methoden kan gevonden worden in IEEE Trans. on Communications, "Special issue on computer network architectures and protocols," Vol. COM-28, No. 4, April 1980.

17) Het Petri Net model is genoemd naar de maker C.A. Petri, die in 1962 op dit onderwerp promoveerde aan de Universiteit van Bonn.

18) "A theory for protocol validation," G.J. Holzmann, IEEE Trans. on Computers, Vol. C-31, No. 8, Augustus 1982, 730-738.

19) Proc. 2nd International Workshop on Protocol Verification, Idyllwild, CA, USA, ed. C. Sunshine, North-Holland Publ., May 1982.

20) "Technische voorwaarden voor de aansluiting van data-eindapparatuur op Nederlandse pakketschakelende datanetten (X.25 koppelvlak)." Publikatie Td 381 van de PTT - Centrale Afdeling Telegrafie en Datacommunicatie, bureel TG1, Den Haag, Juni 1982.

21) Aan de Technische Hogeschool in Delft wordt wel al, in samenwerking met het dr.

Neher Laboratorium van de PTT, onderzoek verricht naar protocol validatie methoden. Zie bijvoorbeeld:
"The Pandora system - an interactive protocol development system", G.J. Holzmann, Rapport no. 39, Vakgroep AVS, Afdeling Elektrotechniek, TH Delft, Augustus 1982.