Applicatie Support Laag en synchronisatie

Database-replicatie in militaire systemen (2)

Marco van der Meijden en Berry Jansen

Zoals in het eerste deel in DB/M 3 is beschreven, bestaan ISIS applicaties over het algemeen uit losse processen die binnen de console worden geïntegreerd. Ondanks de scheiding in processen moeten die processen tezamen voor de eindgebruiker een consistente en geïntegreerde applicatie zijn.

Voor wat betreft de Man Machine Interface (MMI) zijn de console-integratie en de MMI design guidelines de belangrijkste middelen hiertoe. Echter, de integratie behelst ook data-integratie, aangezien de verschillende processen vaak views zijn op dezelfde data en wijzigingen in het ene proces direct in het andere zichtbaar moeten zijn. Om deze data-integratie te bewerkstelligen zijn er in ISIS zeer uitgebreide data caching- en business objectlagen gebouwd, zie afbeelding 1. Deze lagen zorgen voor een belangrijke performance-verbetering, data-integratie en -consistentie in business logica.

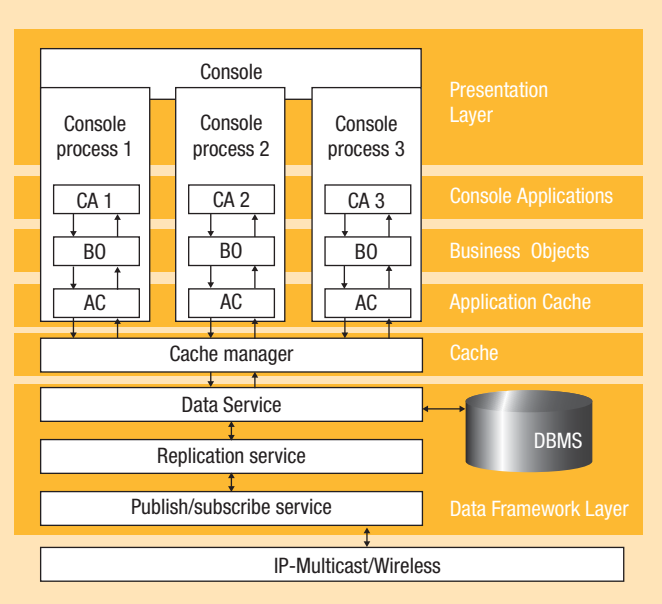
Elk ISIS applicatieproces heeft zijn eigen applicatie-cache waarin de voor dat proces relevante data vastgehouden worden. De coördinatie tussen de verschillende caches is in handen van een centrale cache manager. Elke applicatie-cache biedt faciliteiten zoals data querying, data indexering, filtering en proces-lokale transacties. Applicatie-caches voeren hun transacties uit op de cache manager. De cache manager verspreidt de transactie vervolgens naar de andere applicatie-caches die op hun beurt weer hun applicatie notificeren.
De verschillende applicatie-cache transacties worden door de cache manager vastgehouden totdat deze gezamenlijk als één transactie naar de data service gestuurd wordt. Vervolgens wordt deze transactie op de informatiebus gepubliceerd. Het is de eindgebruiker zelf die dit door een 'save' actie in gang zet.
Zolang de transactie nog open is kan de cache manager acties als undo/redo/restore uitvoeren.
Ook wijzigingen die van de 'replicatiebus' binnenkomen worden door de cache manager verwerkt en aan de applicatie-caches doorgegeven. Het resultaat van het geheel is dat de eindgebruiker zowel zijn eigen wijzigingen als die vanaf de replicatiebus near-real-time in verschillende viewers gereflecteerd ziet, zonder dat deze functionaliteit door iedere applicatie apart hoeft te worden geïmplementeerd.

Vooral de near-real-time automatische notificaties van wijzigingen vanaf de replicatiebus zijn cruciaal. Indien de eindgebruiker zelf een 'refresh' actie zou moeten doen of als de refresh timer gebaseerd is, heeft dat zeer nadelige gevolgen voor het gemeenschappelijke beeld in tijdkritische situaties.

Bovenop deze data caching-laag is de business objects-laag gebouwd. Deze laag vormt de grens van het data framework en zorgt voor een consistente instantiatie en gebruik van business-objecten. Het ISIS data framework en applicaties kunnen met verschillende datamodellen overweg en om dit te ondersteunen kunnen business-objecten potentieel een reeks aan interfaces implementeren.
De business objects-laag biedt enkele vereiste standaard interfaces (zoals bijvoorbeeld voor geografische representatie) en zorgt ervoor dat de juiste business logica uitgevoerd wordt onafhankelijk van de interface waarlangs een object benaderd wordt.

Data Service

ISIS moet te allen tijde beschikbaar zijn in een omgeving waarin netwerkverbindingen niet aanwezig of instabiel kunnen zijn en waarin het cruciaal is dat eindgebruikers ondanks hun enorme



Afbeelding 1: Applicatie Support-laag in detail.

geografische verspreiding een gezamelijk operationeel beeld behouden. Dit stelt specifieke eisen aan de architectuur als het gaat om beschikbaarheid, consistentie van informatie en conflictresolutie.

'Optimistische' Replicatie

Het gebruik van locking-mechanismen zoals two-phase commit is in ISIS niet mogelijk omdat het de beschikbaarheid van alle nodes vereist [11]. Omdat iedere node in principe een replica is van alle andere nodes, zijn traditionele 'pessimistische' replicatie-algoritmen in het algemeen niet bruikbaar [15]. Pessimistische replicatie kenmerkt zich door het feit dat een replica niet kan worden gebruikt tenzij kan worden gegarandeerd dat deze consistent en up-to-date is. Een voorbeeld van 'pessimistische' replicatie is 'floating master' replicatie, waarbij alle transacties op een 'master' worden gedaan en deze de geslaagde transacties synchroon naar alle replica's verspreidt. Als de master wegvalt wordt een van de replica's tot nieuwe master gekozen. Deze oplossingen lopen tegen grenzen aan bij grote aantallen replica's, onbetrouwbare verbindingen of hoge latency's.

> Het alternatief was om convergentie plaats te laten vinden op basis van transactietijd

Optimistische replicatietechnieken (zoals bijvoorbeeld gebruikt door Usenet en DNS) daaraantegen gaan er vanuit dat concurrent transacties op replica's plaatsvinden en dat eventuele conflicten af en toe voorkomen en deze achteraf worden gecorrigeerd.
Optimistische replicatie heeft als voordeel dat het betere beschikbaarheid, schaalbaarheid en collaboratiemogelijkheden biedt.
Het nadeel is dat moet worden omgegaan met eventuele conflicten en inconsistenties.
In het geval van ISIS is de keuze gemaakt voor 'optimistische' replicatie op basis van de afweging dat beschikbaarheid belangrijker is dan het a priori afdwingen van consistentie. In de praktijk zijn de hoeveelheid inconsistenties en conflicten die optreden binnen de ISIS applicaties minimaal, mede door de inrichting van autorisatie en gebruikersprocedures.

Consistentie en convergentie

Het replicatiemechanisme van ISIS werkt volgens het principe van 'uiteindelijke consistentie'. Dit wil zeggen dat de replicatie iedere node naar dezelfde actuele data convergeert als er geen nieuwe transacties meer plaatsvinden. Op ieder moment in de tijd kan iedere node echter een andere state hebben afhankelijk van de volgorde en subset van transacties die is ontvangen van andere nodes.
Uiteindelijke convergentie wordt bereikt door middel van een deterministische keuze op basis van de lokaal beschikbaar data van een node. De replicatie-service zorgt ervoor dat door middel van synchronisatie iedere node uiteindelijk dezelfde informatie lokaal beschikbaar heeft om deze deterministische keuze te kunnen maken. Dit betekent dat ervoor gezorgd wordt dat bij het verwerken van meerdere wijzigingen op hetzelfde attribuut van hetzelfde object dezelfde wijziging als actief wordt verklaard op alle nodes en de andere wijzigingen worden gearchiveerd.
Ieder attribuut van een object is daartoe voorzien van een eigen logisch volgnummer dat wordt opgehoogd bij iedere transactie.
De keuze van de 'actieve' attribuutwaarde gebeurt op basis van 'Thomas Write Rule' [15] (bijvoorbeeld; de waarde met het hoogste volgnummer wint). Dit is de meest populaire techniek voor het bereiken van consistentie voor optimische replicatie en wordt bijvoorbeeld ook gebruikt door MS Active Directory. Pas als de nummers gelijk zijn wordt de winnaar op basis van transactietijd bepaald. De heuristiek hiervan is dat er vanuit wordt gegaan dat degene die de meeste wijzigingen heeft gepleegd (hoogste nummer) ook de meest actuele informatie heeft.
Het alternatief dat in eerste instantie werd overwogen was om convergentie plaats te laten vinden op basis van transactietijd. Dit betekent eenvoudigweg dat de laatste transactie wint. Het nadeel van deze strategie is dat het tijdsynchronisatie vereist. Immers, indien een node zijn klok een jaar vooruit zet zullen zijn transacties gedurende een jaar alle conflicten winnen. Tijdsynchronisatie is in de ISIS-omgeving in de praktijk onmogelijk af te dwingen en daarom is er voor de andere strategie gekozen.

Voorbeelden (zie afbeelding 2):

- Node A wijzigt het positie-attribuut van eenheid X. Deze wijziging heeft voor het attribuut het wijzigingsvolgnummer 1. Node B ontvangt deze wijziging en wijzigt hetzelfde attribuut vervolgens ook. Deze wijziging krijgt dan nummer 2. Node A ontvangt die wijziging vervolgens en verklaart het actief op basis van het hogere volgnummer;
- Node A en B zijn volledig gesynchroniseerd op het moment dat de verbinding tussen hen wegvalt. Node A wijzigt vervolgens het positie-attribuut van eenheid X twee keer. Deze wijzigingen krijgen wijzigingsvolgnummers 2 en 3. Node B ontvangt deze wijziging niet en wijzigt hetzelfde attribuut zelf ook enkele keren. Deze laatste wijziging krijgt nummer 4. Als nu de nodes weer bij elkaar komen zal de laatste wijziging van node B actief verklaard worden. Indien node A nu weer een wijziging zou doen dan krijgt het nummer 5.

Applicaties bewust van logische inconsistenties

Hoewel ISIS van elke lokale transactie de logische consistentie afdwingt, is het heel goed mogelijk dat het ontvangen van twee overlappende consistente transacties een logische inconsistentie oplevert in termen van referentiële integriteit of business-logica. De convergentiestrategie op basis van volgnummers per attribuut zorgt immers alleen voor conflictresolutie op attribuutniveau. Gegeven de autonome manier van werken en het feit dat het aantal nodes in het totale netwerk onbekend is, kon er in het

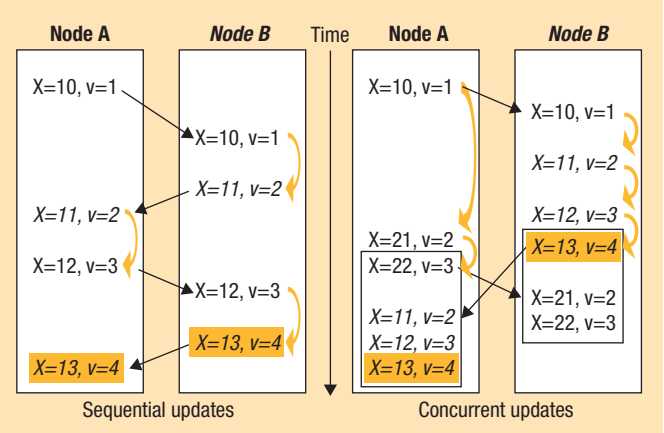
ontwerp van ISIS ook geen andere keus gemaakt worden dan te accepteren dat de volgorde waarin transacties verwerkt worden op geen enkele manier af te dwingen is. De 'correcte' volgorde kan immers niet eens bepaald worden.
Een gefingeerd voorbeeld zou de volgende situatie kunnen zijn: Node A en B staan niet met elkaar in verbinding. In het systeem kennen zij beiden de eenheden X en Y en een informatieregel in het systeem is dat eenheden niet onder bevel van een al uitgeschakelde eenheid geplaatst kunnen worden. De eindgebruiker op node A stelt nu eenheid X onder bevel van eenheid Y.
Tegelijkertijd meldt de eindgebruiker op node B dat eenheid Y uitgeschakeld is. Beide wijzigingen in het systeem zijn op zich compleet consistent, maar zodra de nodes met elkaar repliceren zal het resultaat zijn dat eenheid X onder bevel staat van de uitgeschakelde eenheid Y, hetgeen een informatieregel breekt.
Een van de belangrijkste keuzes in ISIS is daarom het accepteren van potentiële logische inconsistentie van informatie. Overigens geldt dit alleen voor informatie die door meerdere eindgebruikers gelijktijdig gewijzigd wordt. De praktijk is dat de meeste informatie, vanwege functiescheidingen, door maar één persoon gewijzigd wordt, waardoor de logische consistentie behouden blijft.
Het accepteren van logische inconsistenties betekent dat ook applicaties er mee om moeten kunnen gaan en de eindgebruikers moeten assisteren in het signaleren en oplossen van dit soort inconsistenties. Dit is een moeilijke taak die een zeer defensieve manier van programmeren met zich meebrengt, zeker als het applicaties betreft die werken met relaties tussen objecten.

Scheiding van logisch en fysiek datamodel

In ISIS is het fysieke datamodel gescheiden van de logische datamodellen.
Dit betekent dat logische ISIS objecten (eenheden, materieel, etcetera) geen representatie hebben in de fysieke databasetabellen, maar zijn uitgedrukt in een fysiek datamodel dat in essentie bestaat uit drie tabellen:

- Objecttabel.
  - ObjectId
  - ObjectType
- Attribuutwaardetabel.
  - ObjectId
  - AttribuutType
  - Attribuutwaarde
  - Wijzigingsvolgnummer
  - TransactieId
- Transactietabel.
  - TransactieId
  - Transactiegegevens

In dit fysieke model heeft elke attribuutwaarde dus zijn eigen record in de attribuutwaardetabel met daarbij een verwijzing naar een record in de Objecttabel en een verwijzing naar de Transactietabel.
Een belangrijk voordeel van deze benadering is dat het Data

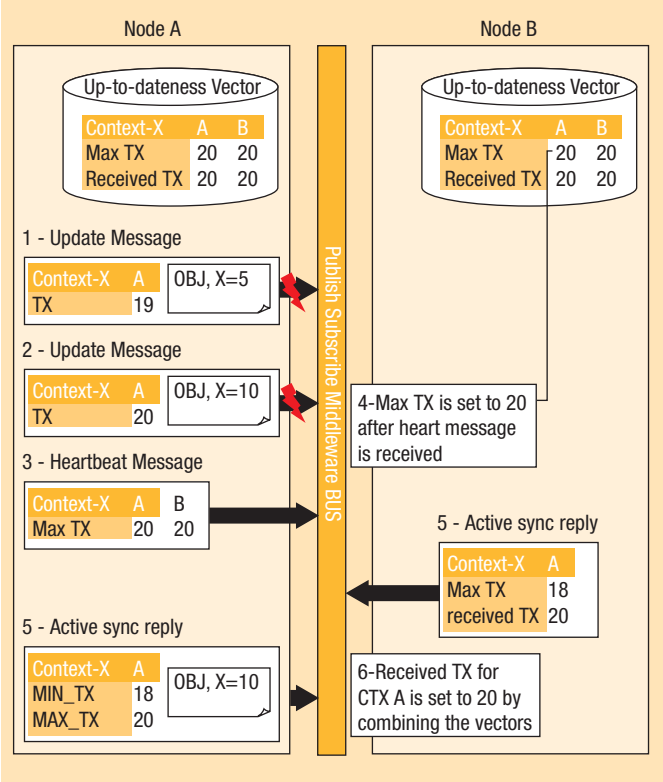


Afbeelding 2: Convergentie van attribuutwaarden.

Framework (de business-objecten uitgezonderd) onafhankelijk is van het logische datamodel omdat de werking gebaseerd is op het generieke fysieke model. Wijzigingen of uitbreidingen van het logische model hebben daardoor geen gevolgen voor de implementatie van de cache manager of andere services.
De logische datamodellen van ISIS zijn vastgelegd in aparte meta-tabellen (voor objecttypes, attribuuttypes, enumeratiewaardes etcetera). In de nabije toekomst zal ook het metamodel in de vorm van object-attribuutwaarde uitgedrukt gaan worden, zodat het ook kan worden gerepliceerd. Het gaat te ver om hier de precieze datamodellering te beschrijven, bovendien is het geheel vergelijkbaar met de modellering die MS Active Directory toepast. Het is de al beschreven applicatielaag die de individuele attribuutwaarden met behulp van de logische datamodellen tot logische objecten assembleert en aan de applicaties in de vorm van business objecten ter beschikking stelt. Het logische datamodel stelt die applicatielaag ook in staat om datatype- en consistentiecontroles uit te voeren op de wijzigingen die applicaties doen.

De redenen om tot deze scheiding te komen zijn de volgende:

- Wijzigingen op objecten moeten in ISIS per attribuut beschouwd worden. Andere methodes zoals het bij een wijziging confirmeren van alle andere attributen van een object werken niet in een gedistribueerd systeem;
- ISIS moet, onder andere vanwege de convergentiestrategie, voor elke attribuutwaarde additionele data bijhouden (volgnummer, referentie naar de transactie);
- ISIS moet logische inconsistenties accepteren. Dit maakt een klassieke genormaliseerde modellering met referentiële integriteit niet onmogelijk: foreign key constraints bijvoorbeeld kunnen niet worden afgedwongen waardoor de meerwaarde van het gebruik van een fysiek relationeel schema afneemt;
- ISIS moet als platform voor toekomstige systemen dienen die elk een (deels) nieuw datamodel kunnen hebben. Ook moeten de bestaande datamodellen uitgebreid kunnen worden. Ook dit maakt een representatie van het logische datamodel in het fysieke model erg moeilijk, zeker als daarbij replicatie in ogenschouw genomen wordt.



Afbeelding 3: Voorbeeld van actieve synchronisatie. Node B mist twee transacties. Door ontvangst van een heartbeat weet het dat het twee transacties gemist heeft. De actieve synchronisatie zorgt ervoor dat alleen de laatste waarde (X=10) wordt gestuurd, door middel van de 'up-to-dateness' vectoren wordt bepaald dat verder geen transacties gemist worden en de nodes weer in sync zijn.

Behalve het moeten accepteren van logische inconsistenties heeft de operationele omgeving en de daaruit voortvloeiende keuzes in het systeemontwerp nog een aantal andere belangrijke nadelen. Door de scheiding tussen logisch en fysiek datamodel kunnen database query's zeer complex en traag zijn. Dit is ook direct een van de redenen voor het bestaan van de applicatie-caches met hun filtering- en indexering-functionaliteit. Logische datamodellen mogen alleen uitgebreid worden. Het weghalen van bijvoorbeeld een attribuut uit een objectdefinitie mag in een gedistribueerde omgeving niet omdat er altijd nog nodes in het net kunnen zijn met applicaties die gebaseerd zijn op het oude model.

Replicatie Service

De replicatie service zorgt voor de selectieve distributie en synchronisatie van transacties die op de lokale database worden gedaan. Het resultaat is dat de data service zich naar de applicaties toe als een gedistribueerde database gedraagt. De werking van de replicatie-service is gebaseerd op een drietal functies; distributie van transacties, detecteren of er transacties gemist zijn en synchronisatie, zie afbeelding 3.
Transacties worden gerepresenteerd als Inserts op de fysieke Object-, Attribuut Waarde- en Transactie-tabellen. Hierbij wordt een geoptimaliseerd binair formaat gebruikt. De grootte van een insert van een nieuwe locatie voor een object is bijvoorbeeld niet groter dan 20 Bytes. Fysieke Deletes zijn niet mogelijk. In plaats daarvan hebben Objecten een 'deleted' attribuut dat wordt gezet. Bij synchronisatie wordt daarmee automatisch ook de informatie dat een object is verwijderd gesynchroniseerd. In principe is het in een gedistribueerde omgeving niet mogelijk objecten te verwijderen zonder de informatie vast te houden dat het object verwijderd is! De enige oplossing om te vermijden dat alle 'delete' informatie moet worden bewaard is het instellen van een 'tombstone lifetime' [15] die bepaalt hoe lang verwijderde objecten fysiek worden bewaard. Als een systeem langer dan de 'tombstone lifetime' uit het netwerk is geweest kan synchronisatie echter niet meer worden gegarandeerd.

Om te kunnen bepalen of een node informatie mist (out-of-sync detectie), wordt per context een 'up-to-dateness' vector bijgehouden. Deze vector houdt bij hoeveel transacties door iedere node op de context zijn gepleegd en welke van de transacties al door de eigen node zijn verwerkt. Iedere node vergelijkt zijn eigen 'up-to-dateness' vectoren met de vectoren die door andere nodes periodiek worden gepubliceerd om te ontdekken of het transacties mist. ISIS ondersteunt zowel volledige als actieve synchronisatie. Bij actieve synchronisatie wordt alleen de actuele informatie gesynchroniseerd. Bij volledige synchronisatie worden naast de actuele informatie ook historische transacties gesynchroniseerd. Dit gebeurt incrementeel, beginnend bij de meest recente transacties tot een bepaald punt terug in de tijd.
Als een node detecteert dat het informatie mist van een context wordt een synchronisatie-request gepubliceerd. In het geval van actieve synchronisatie bevat dit request de eigen up-to-dateness vector voor deze context. Op een node die het request beantwoordt (door middel van een sync-reply bericht) wordt het verschil bepaald tussen de eigen actieve data en de in het request uitgedrukte actieve data. Dit verschil bestaat uit alle (deel)transacties die voorkomen in de actieve data van de node die niet voorkomen in de 'up-to-dateness' vector van het request.
Naast data bevat het antwoord op een request ook een 'delta up-to-dateness-vector' die de nodes die het bericht ontvangen in staat stelt af te leiden of het inserten van de (deel) transacties leidt tot het volledig of gedeeltelijk gesynchroniseerd worden. Bij actieve synchronisatie kan het dus voorkomen dat het antwoord geen data bevat omdat de gemiste transacties niet meer in het actuele beeld voorkomen. Door het combineren van de delta 'up-to-dateness' vector en de eigen 'up-to-dateness' vector kan een node dan wel herleiden dat het geen actuele informatie meer mist en dus gesynchroniseerd is!

Replicatie en bandbreedte

Synchronisatie kan potentieel veel bandbreedte vergen en er is daarom veel aandacht besteed aan het minimaliseren van de protocol overhead. Verder wordt er maximaal gebruik gemaakt van de voordelen die multicast biedt (één reply kan bijvoorbeeld vele nodes in sync brengen).

Zoals beschreven publiceren nodes periodiek hun eigen 'up-to-dateness' vector ten behoeve van out-of-sync detectie. Een voorbeeld van het terugdringen van protocol overhead is een suppressieprotocol dat ervoor zorgt dat de meest actuele vector door slechts één node in het netwerk wordt gepubliceerd. Eenmaal gepubliceerd wordt deze vector periodiek in een klein bericht met een hashcode geconfirmeerd door de publicerende node. Wijzigingen op de vector worden vervolgens met een relatief klein deltabericht gecommuniceerd. Dit leidt ertoe dat de volledige vector alleen initieel en in het geval van 'net entry' van een nieuwe node gepubliceerd hoeft te worden.
De synchronisatie-service maakt alleen gebruik van Publish/Subscribe-operaties en andere nodes worden nooit direct geadresseerd. Hierdoor is mogelijk dat er meerdere nodes in het netwerk aanwezig zijn die een synchronisatie-request kunnen beantwoorden. Een goed voorbeeld hiervan is een 'late net entry' van een node die informatie mist die alle andere nodes in het netwerk wel hebben. Door middel van een 'voting' protocol wordt bepaald welke node(s) het antwoord geeft/geven zonder dat deze overlappen. Ook kan het voorkomen dat meerdere nodes tegelijkertijd hetzelfde request willen doen, bijvoorbeeld bij een 'late net entry' van een node met nieuwe informatie. In deze situaties worden identieke requests onderdrukt (door middel van back-off timers). Ook wordt ervoor gezorgd dat meerdere identieke of overlappende requests beantwoord worden met slechts één reply.

Verder houdt synchronisatie rekening met de beschikbare bandbreedte. Dit is vooral een uitdaging omdat ISIS point-to-multipoint synchroniseert met een protocol dat in essentie timer gebaseerd is. Indien een node bijvoorbeeld elke minuut een request herhaalt terwijl het antwoord 10 minuten nodig heeft om volledig gepubliceerd te worden, dan bestaat het gevaar dat na 9 minuten 9 nodes bezig zijn hetzelfde antwoord te publiceren. ISIS stelt daarom bijvoorbeeld requests uit, indien er geen bandbreedte beschikbaar is en bij de suppressie van reply's wordt rekening gehouden met de throughput-tijd van reeds door andere nodes aangekondigde reply's. Hierdoor kan ISIS de beschikbare bandbreedte maximaal benutten zonder overbodig berichtenverkeer. Door de beschreven voorzieningen is de protocol overhead lineair ten aanzien van het aantal gebruikte contexten en nodes en is efficiënte synchronisatie mogelijk in netwerken met een zeer lage bandbreedte.

Replicatie en netwerktopologie

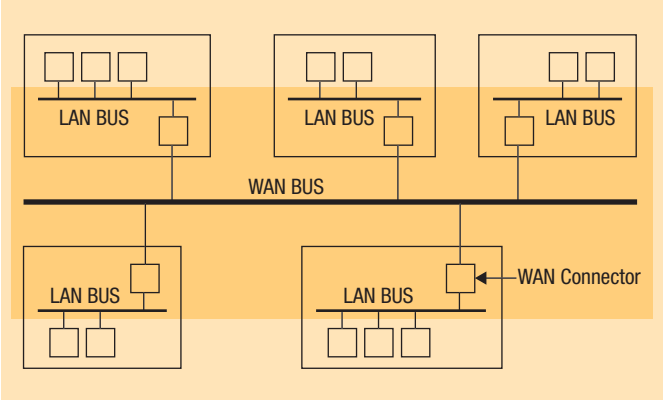
Omdat de replicatie-service alleen gebruik maakt van publish/subscribe middleware, is het onderliggende netwerk transparant. In de meest simpele configuratie kan de middleware worden gebruikt om de hele netwerkinfrastructuur (WAN's, LAN's) transparant te maken. Dit is echter niet erg schaalbaar en kan onnodig bandbreedte gebruiken als synchronisatie over het WAN verloopt terwijl op het LAN deze informatie ook beschikbaar is. Daarom wordt meestal gebruik gemaakt van de hiërarchische configuratie die is afgebeeld in afbeelding 4. Over het WAN wordt een bus geconfigureerd waar alle LAN's mee verbonden zijn via een WAN-connector. Een WAN-connector is een ISIS systeem dat op meer dan één informatiebus actief is en zorgt voor de (software) routing van ISIS-verkeer tussen deze informatiebussen. Op LAN en WAN wordt hetzelfde replicatieprotocol gebruikt. Synchronisatie gebeurt alleen binnen een bus, maar nieuwe transacties worden direct door WAN-connectoren 'geforward' tussen LAN en WAN. Als een WAN-connector door middel van synchronisatie data beschikbaar krijgt, wordt dit via synchronisatie weer op een andere bus verspreid. Omdat synchronisatie timer gebaseerd is kan hier dus latency optreden. Voor de WAN-connector zijn verschillende protocollen ontwikkeld om de routering van transacties en synchronisatie over verschillende replicatiebussen heen mogelijk te maken. Met het instandhouden van de routering van informatie op basis van receiver interest per context is een zekere overhead geassocieerd [12].
In netwerken met een lage bandbreedte is het daarom nodig een trade-off te maken tussen de hoeveelheid contexten die wordt gebruikt om de informatie in onder te verdelen en bandbreedte die het kost om de selectieve replicatie uit te voeren.
Verder kan het voorkomen dat er tussen informatiebussen meerdere WAN-connectoren geconfigureerd zijn, bijvoorbeeld ten behoeve van fail-over. In zo'n geval zorgt een selectieprotocol ervoor dat slechts één WAN Connector actief is.

> Ook protocollen specifiek voor combat net radio worden ondersteund

Publish/subscribe service

De replicatie-service is ontworpen voor publish/subscribe-operaties op een informatiebus. De publish/subscribe-service biedt hiervoor de implementaties en verbergt de specifieke details van de netwerk transportlaag. Het concept van publish/subscribe leent zich uitstekend voor broadcast-achtige implementaties zoals multicast en radio, maar desnoods kan een informatiebus ook bestaan uit een distributieboom van point-to-point verbindingen. Naast de replicatie-service maken ook andere ISIS services zoals messaging (chat/mail) gebruik van de publish/subscribe service.

De huidige publish/subscribe service ondersteunt een aantal verschillende COTS publish/subscribe middleware-producten zoals TIBCO Rendezvous [7] [9], daarnaast biedt het directe ondersteuning voor IP gebaseerde netwerk-transportprotocollen zoals UDP en PGM. Verder worden ook protocollen specifiek voor combat net radio ondersteund. Over het algemeen wordt geen zogenaamde 'certified delivery' vereist van de transportlaag, maar slechts een best effort reliable delivery. De ISIS replicatie-service biedt een meer bandbreedte-efficiënt alternatief voor certified delivery, omdat bij het verlies van berichten niet alle informatie



Afbeelding 4: De logische replicatiebus wordt fysiek gerealiseerd door het koppelen van LAN-bussen aan een WAN-bus.

wordt herzonden maar alleen de actuele informatie wordt gesynchroniseerd.
In de standaardconfiguratie wordt gebruik gemaakt van PGM/IP-multicast. PGM [14] is een nack-based reliable multicast protocol, wat betekent dat het packet repair-voorzieningen heeft. PGM biedt op dit moment de beste mix van reikwijdte (multicast), reliability en protocol overhead. Verder biedt PGM rate control en wordt het ondersteund door (CISCO) routers wat zorgt voor een zeer efficiënte packet repair.
ISIS gebruikt de PGM-implementatie van TIBCO Rendezvous op Windows 2000 en die van Microsoft voor Windows XP en hoger.

De publish/subscribe service kan gelijktijdig meerdere informatie-bussen bedienen en een groot deel van de al eerder beschreven WAN-connectorfunctionaliteit is in de publish/subcribe service geïmplementeerd. Operationeel kan dit er toe leiden dat een ISIS station in een voertuig enerzijds over een satellietverbinding het PGM protocol gebruikt om met de hogere echelons te communiceren ten behoeve van omringende voertuigen, waarmee het gelijktijdig met een radiospecifiek protocol over combat net radio communiceert.

Netwerklaag

ISIS is ontwikkeld om onafhankelijk van netwerkinrichting te zijn. Operationeel moet ISIS immers functioneren als een logisch overlay-netwerk dat een een scala aan verschillende netwerken en transmissiemiddelen overspant. Hierdoor is ISIS niet afhankelijk van netwerkgerelateerde voorzieningen als DNS, domain controllers en user directory's. Echter, ISIS prefereert een netwerk dat multicast ondersteunt omdat het met name is ontwikkeld om de bandbreedtevoordelen van multicast te benutten en om 'plug and play' (plug and fight) te werken.
Het TITAAN netwerk biedt hiervoor een fouttolerante en schaalbare ondersteuning van IP Multicast gebaseerd op PIM. Daarnaast is ISIS op kleine schaal ingezet in Afghanistan, Irak en Liberia, waarvoor telkens op ad hoc-basis een op satellietverbindingen gebaseerd VPN-netwerk werd opgezet.
Voor andere toepassingen wordt gekeken of gebruik kan worden gemaakt van peer-to-peer technologieën om plug & play deployment over het Internet mogelijk te maken [8], omdat IP Multicast over het Internet niet wordt ondersteund.

Conclusie

De ISIS architectuur is geoptimaliseerd voor beschikbaarheid, near-real-time selectieve replicatie van informatie, samenwerking, uitbreidbaarheid en flexibele plug & play gebaseerde deployments in omgevingen met beperkte bandbreedte. Met name de 'Replicatie Bus' en Data Service zijn hiervoor geheel dedicated ontwikkeld omdat er geen producten bestaan die al deze eigenschappen combineren [13]. De ISIS architectuur zal daarom in de komende jaren verder worden ontwikkeld om aan de eisen van toekomstige KL-toepassingen en communicatie-infra-structuren te blijven voldoen.

Referenties

[1] *Network Centric Warfare, Developing and Leveraging Information Superiority, David S. Alberts, John J. Garstka and Frederick P. Stein.*
[2] *Network Centric Warfare concepts in the Royal Netherlands Army C2 Architecture, M.G. van der Meijden, Nato RTO SCI-137 paper.*
[3] *ATCISS Filtering proposals: Report of Filter Task Group, Working paper 31-1, draft 5.0.*
[4] *MIP Multilateral Interoperability Program briefing notes (MBN), March 2003. www.mip-site.org/Public_documents/MBN-SH-PMG-Edition1.1.pdf.*
[5] *ATCCIS Working paper 14-3, ATCCIS Replication Mechanism (ARM) Consolidated Specification, Edition 3.0, 5 June 2000.*
[6] *Tactical Information Distribution using XML, M.G. van der Meijden, TNO report FEL-01-A124, 2001.*
[7] *Tactical information distribution with Message Oriented Middleware, Julian de Wit, TNO report FEL-01-S234, 2001.*
[8] *Peer-to-peer technology in a C2 environment: a preliminary study, M. van Lent, TNO-FEL Traineeship report 20030716, 2003.*
[9] *Automatic reconfiguration of TIBCO Rendezvous Routing Daemons in dynamic networks, D.J. Noort, TNO report FEL-03-S075, April 2003.*
[10] *TITAAN Phase 1 (HRF HQ): Detailed Technical Design, J. Vytopil. Royal Netherlands Army, C2SC, 2002. Version 1.01.*
[11] *Mastering agreement problems in distributed systems, M.Raynal and M. Singal, IEEE Software July/August 2001.*
[12] *Consideration of receiver Interest for IP Multicast Delivery, B.N. Levine, J. Crowcroft, C. Diot, J.J. Garcia-Luna-Aceves, J.F. Kurose.*
[13] *Data Replication in Low Bandwidth Military Environments - State of the Art Review, A. Gibb and S. Chamberlain, TTCP, C3I Group Technical Panel 10.*
[14] *PGM Reliable Transport Protocol Specification, www.ietf.org/rfc/rfc3208.txt?number=3208.*
[15] *Optimistic replication, Yasushi Saito, Hewlett-Packard Laboratories, Alto, CA (USA) and Marc Shapiro, Microsoft Research Ltd., Cambridge (UK), September 2003, Technical Report, MSR-TR-2003-60.*
[16] *Web Site Command & Control Support Centre, www.c2sc.org.*

Marco van der Meijden en Berry Jansen
Drs. Marco G. van der Meijden (vanderMeijden@fel.tno.nl) is Wetenschappelijk Medewerker bij de afdeling Command & Control en Informatievoorziening van TNO-FEL. Berry Jansen (berry.jansen@capgemini.com) is Managing Consultant bij Capgemini.