Så simulerar man ett helt IoT-nät

EXPERTARTIKEL

Hur långt kan du skala upp din IoT-konstruktion? En simulator kan ge svaret.

Av Jakob Engblom, Wind River

Jakob Engblom är Product Line Manager för simulatorverktyget Simics på Wind River. Han har jobbat med simulering av inbyggda (och andra) datorsystem med Simics sedan 2002, först på startup-bolaget Virtutech, och sedan 2010 på Wind River. Jakob Engblom har en doktorsexamen i datorsystem och en magisterexamen i datavetenskap, båda från Uppsala universitet.

Det är en stor utmaning att utveckla och testa tillämpningar för Internet-of-Things (IoT) av den enkla anledningen att det handlar om stora system som innehåller väldigt många enheter. Det är svårt att trycka in hundratals noder i ett vanligt mjukvarulab för testning, och det är svårt att förse alla dessa noder med intressanta och realistiska indata. Frågan är hur man rent praktiskt ska gå tillväga för att testa mjukvara när den ska köras på hundratals eller kanske till och med tusentals IoT-noder?

Simulering är ett bra svar på den frågan.

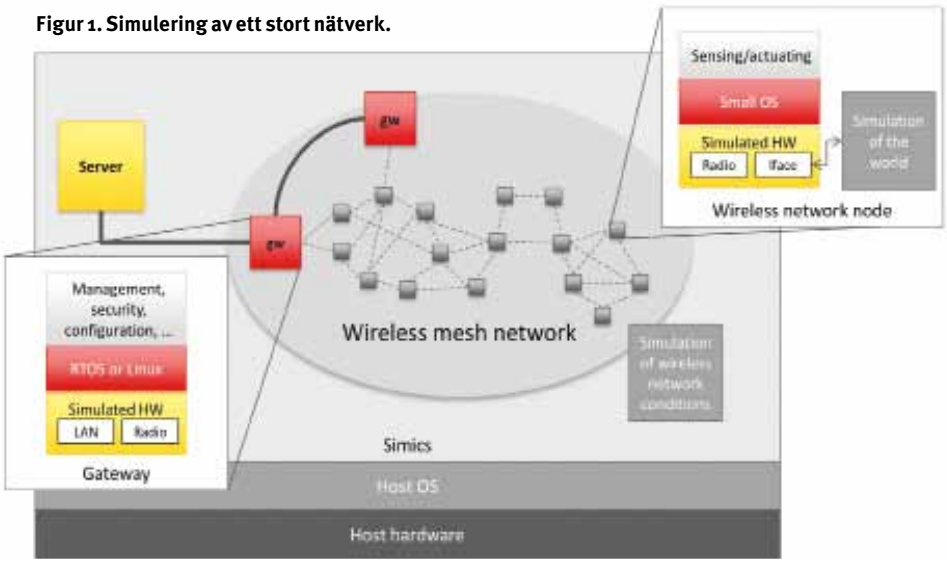
IOT-SYSTEM AV IDAG använder ofta en arkitektur i tre lager enligt figur 1. Systemet består av många små noder kopplade till varandra över ett lokalt trådlöst nät. Det finns en eller flera ”gateway”-noder på det trådlösa nätet som även är anslutna mot Internet eller ett annat nätverk, för att ansluta sensornätet till en server eller till molnet.

De små noderna kan vara sensorer, som temperatursensorer, elmätare, kameror eller strömbrytare, eller så kan de vara styrdon, som termostater, styrda strömbrytare eller dörrlås.

En gateway eller en koncentrator kopplar IoT-systemet till omvärlden och upprätthåller säkerhet. En backend-server, som ofta finns i molnet, hanterar affärslogik och styrning av systemet.

FÖR ATT TESTA ETT SYSTEM av det här slaget, behöver du sprida ut de trådlösa noderna över ett såpass stort område att alla inte har direktkontakt med varandra. Detta kräver typiskt att du använder en hel byggnad eller ett campus som labb. Att sätta upp och underhålla ett så stort nätverk innebär mycket jobb –arbetskostnaderna dränker snart kostnaderna för själva noderna.

Figur 1. Simulering av ett stort nätverk.



I en simulator, som i figur 1, är det enkelt att sätta upp ett godtyckligt nätverk. Du skriver ett program som virtuellt skapar och sprider ut noderna över allt det virtuella utrymme du kan tänkas behöva. Därefter modellerar du hur noderna kan nå (och inte nå) varandra. Istället för att manuellt hantera hundratals fysiska objekt har du reducerat jobbet till att hantera litet programkod – som är väldigt lätt att ändra.

En simulatorlösning som Wind River Simics simulerar hårdvaran i varje nod, vilket inkluderar processorer, minne, timers, LEDdar, trådlös radio och allt annat som behövs. De simulerade noderna kör riktiga operativsystem och riktiga tillämpningar – samma binärer som den riktiga hårdvaran skulle köra. De olika typerna av noder simuleras med olika modeller, även om de alla körs i en och samma simulering.

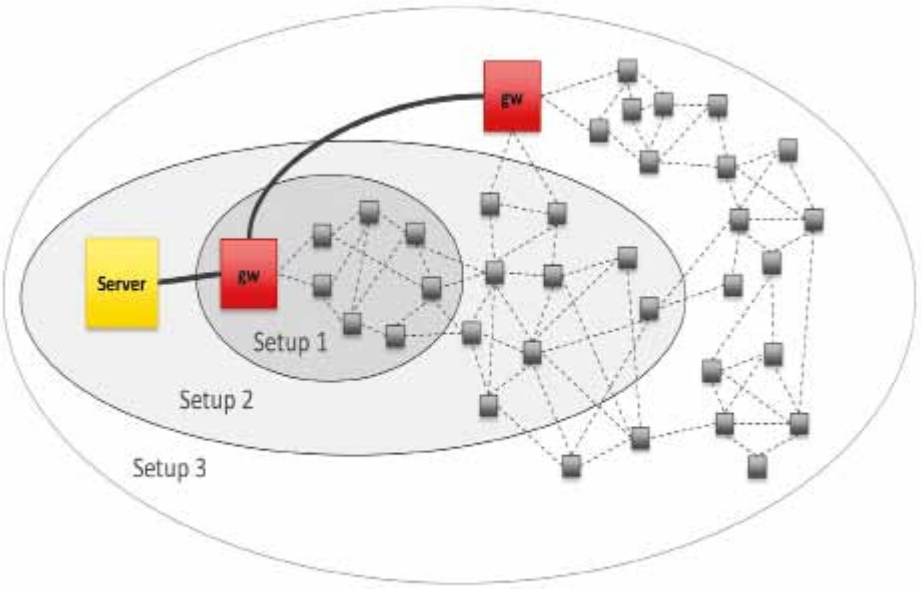
NÄR DU SIMULERAR hela IoT-systemet på det här sättet, kan du testa alla aspekter på mjukvaran, inklusive:

- den trådlösa nätverksstacken och dess hantering av nätverksproblem,
- programkoden för sensorer och styrdon, inklusive hur den fungerar i sin omgivning, och
- nodernas vilolägen och vakenhetsintervall och hur mycket ström de sparar

Man kan testa dataflödet från noderna till servern via gateway, inklusive:

- den ”middleware” som hanterar noderna och uppdaterar deras mjukvara (inklusive mjukvaruuppdateringar över det trådlösa nätet),
- säkerheten och säkerhetsfunktionen i noder och gateway,
- datahanteringssystemets skalbarhet när antalet noder växer.

SIMULERING ÄR SÄRSKILT BRA på att testa det sistnämnda – hur IoT-systemets hård-



Figur 2. Genom att skala upp nätverket kan man simulera större nätverk.

och mjukvara beter sig när systemet skalas upp.

Som man ser i figur 2 går det med hjälp av simulering att bygga system av godtycklig storlek – från små till gigantiska. Det betyder att systemets beteende kan testas i många olika skalor, från små enhetstester och tester av delsystem ända upp till de största konfigureringar du kan tänka dig.

Det är inte ovanligt att olika nivåer av test avslöjar olika typer av problem. Problemet är inte bara att säkerställa att systemet fungerar när det är maximalt stort, utan även att det fungerar effektivt över hela skalan från litet till stort.

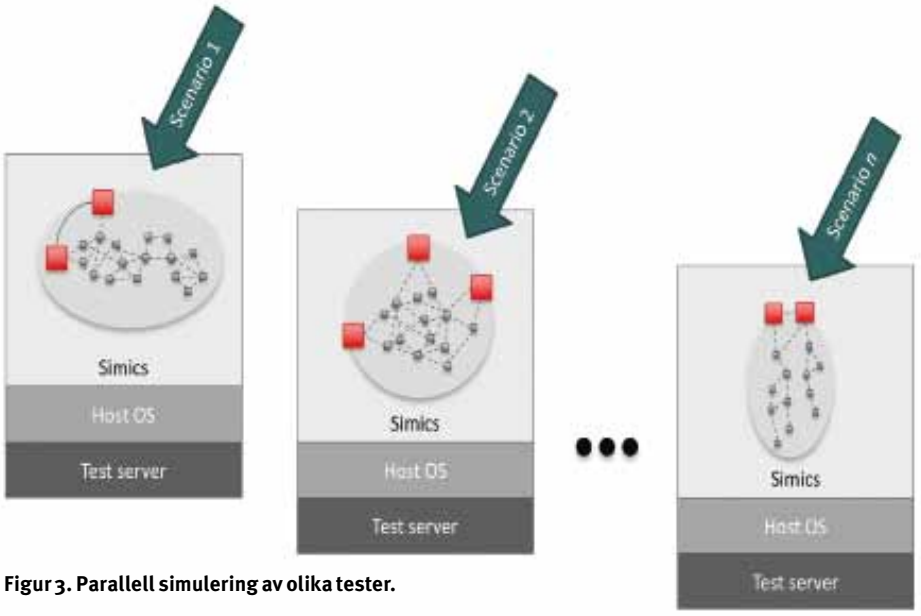
Figur 1 visar även hur man kan koppla in en simulering av den miljö i vilken IoT-systemet verkar. Varje sensornod behöver kopplas till en simulering av dess omgivning – så att den har data att skicka tillbaka till gateway och server. En IoT-nod utan omgivning är inte särskilt användbar eller intressant.

En aspekt av systemtest är att variera mottagningsförhållandena i det simulerade radionätet. I en simulator är det trivialt att bestämma vilken signalstyrka som gäller mellan två godtyckliga par av noder, och man kan införa simuleringsregler som slumpmässigt tappar fler och fler paket allteftersom signalstyrkan går ner, för att simulera effekten av dålig kontakt.

DET GÅR ATT VARIERA mottagningsförhållandena under det att testet pågår för att kontrollera hur noderna beter sig när förhållandena ändras, till exempel när ett tåg passerar mellan två noder och avbryter radiokommunikationen för ett kort ögonblick.

Det bästa är att sådana test kan styras exakt och upprepas exakt, till skillnad från den verkliga världen där man kan vara glad om man överhuvudtaget kan manipulera radioförhållandena, och då med stor ansträngning. En vanlig metod verkar vara att använda slumpmässiga störsändare såsom mikrovågsugnar, vilket inte är särskilt lätt att upprepa på ett kontrollerat sätt.

Den simulatorbaserade testningen kan också skalas ut horisontellt, som visas i figur 3. Det är enkelt att skapa olika varianter av nätverk för att kunna testa mjukvaran med olika sätt att konfigurera och lokalisera ett visst antal noder. Olika förhållanden mellan gateways och sensornoder kan testas, såväl som olika nättopologier. Figur 3 visar hur simulatorn låter dig köra flera olika tester parallellt på olika värdmaskiner, vilket betyder att den totala tidsåtgången för test blir mycket mindre än om testerna hade körts seriellt på fysisk hårdvara.



Figur 3. Parallell simulering av olika tester.

MEN KAN DET VERKLIGEN fungera i praktiken att simulera hundratals eller tusentals noder på en enda dator?

Svaret är ja. IoT-sensornoder har typiskt en mycket kort aktiv period. Sensorerna läser inte av världen omkring sig kontinuerligt, utan tenderar att vakna upp med jämna intervall för att göra en mätning och rapportera resultatet. Efter en mätning, som kanske bara tar några millisekunder, kan systemet gå i vila i flera minuter eller till och med i timmar. Detta är en energisnål lösning som gör det möjligt att fysiskt ha noder i drift under långa tidsrymder utan att behöva byta batterier.

Sålunda vilar systemet större delen av tiden, och detta kan utnyttjas för att accelerera simuleringen med hjälp av en teknik som kallas hypersimulering. Istället för att beta av dötiden cykel för cykel kan en simulatorlösning som Simics hoppa direkt till tidpunkten för nästa intressanta händelse. På detta sätt kan man simulera systemet flera gånger snabbare än realtid. Detta möjliggör väldigt stora IoT-simuleringar. Jag gjorde faktiskt detta själv för tio år sedan när Simics simulerade 1 000 IoT-noder på en 32-bitars Windows XP-dator med en enda processor. Det gick betydligt snabbare än i den verkliga världen. Då kändes det fantastiskt att det ens var möjligt – men idag är det vardagsmat.

MAN FÅR INTE GLÖMMA BORT att det i slutänden krävs fysiska labb för sluttestningen av ditt system – som Jack Ganssle brukar säga måste man testa det man levererar och leverera det man testar. Men det är också nödvändigt att använda simulering för att testa allt det man inte kan eller hinner göra med det fysiska testlabbet. Simuleringen låter dig täcka in fler testfall och garantera att kvaliteten upprätthålls och att systemet är robust under olika förhållanden. Med simulering kan man bygga bättre IoT-system snabbare och enklare. ■