En översikt över kantdatorsystem

Andreas Levander

Handledare: Andreas Lundell

1. Introduktion

Telefoner, vitvaror, TV, övervakningskameror, termostater och industriella maskiner – vad har alla dessa gemensamt? De har alla blivit ”smarta”. Genom att koppla upp dessa till internet kan man skapa ny funktionalitet eller få ökad effektivitet.

Antalet enheter inom sakernas internet (eng. Internet of Things) väntas tredubblas från ca 10 miljarder år 2020 till 30 miljarder år 2030 [1] och antalet data som skapas väntas öka från 64.2 ZB år 2020 till 180 ZB år 2025 [2]. Dessa enheter har oftast begränsad beräkningskapacitet och lagring så vanligen skickas de data som genereras till stora centraliserade datacenter som sköter lagring och beräkningar. Problemet med detta är att dessa datacenter ofta är långt bort från enheterna och detta medför hög latens och nätverksanvändning. För att lösa dessa problem har en ny modell kallad kantdatorsystem (eng. Edge Computing) uppstått. Kantdatorsystem innebär att man lagrar data och gör beräkningar så nära enheterna som skapar data som möjligt, alltså vid nätverkets kant. Denna modell ämnar inte ersätta molntjänster utan samverka med dem. Genom att sköta tidskritiska beräkningar och filtrering av data vid nätverkets kant och större beräkningar och analys i molnet kan man övervinna molntjänsters begränsningar och på så sätt skapa nya möjligheter.

I denna avhandling ämnar jag ge en översikt över kantdatorsystem och deras applikationer.

2. Bakgrund

2.1. Molntjänster

Det finns många definitioner på molntjänster men enligt amerikanska National Institute of Standard and Technology (NIST) innebär molntjänster en modell som på ett lättillgängligt sätt över internet ger kunder datorresurser (t.ex. nätverk,

lagring, beräkningskapacitet) enligt efterfrågan från en delad mängd hårdvara [3]. Denna modell består av fem väsentliga egenskaper, tre modeller för tjänster och fyra driftsättningsmodeller.

De fem väsentliga egenskaperna är självbetjäning, nätverksåtkomst, resursdelning, dynamisk allokering och uppmätt tjänst. En konsument kan få datorresurser enligt behov utan att behöva interagera med en människa. Resurserna är tillgängliga över nätverket genom standardiserade sätt och på så sätt tillgängliga på en mängd olika enheter. Datorhårdvaran delas av alla konsumenter och konsumenterna har ingen kontroll över exakt var resurserna finns och resurserna kan dynamiskt allokeras till de konsumenter som behöver dem. För konsumenterna verkar resurserna vara oändliga och de kan få vilken mängd som helst när som helst. All resursanvändning övervakas och mäts. Konsumenter får information om vad de använder och betalar enligt det.

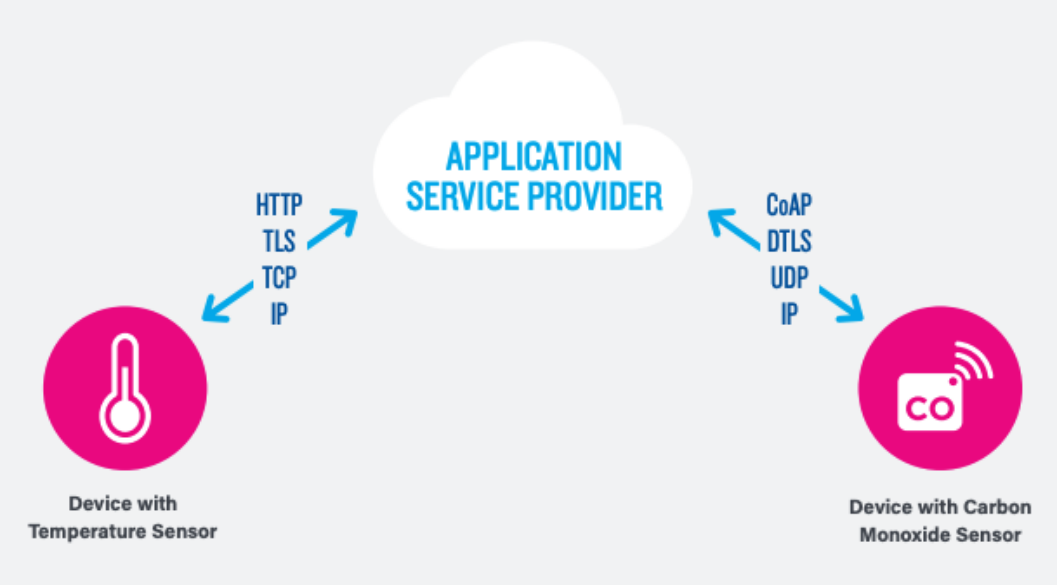
De tre tjänstemodellerna är mjukvara som tjänst, plattform som tjänst och infrastruktur som tjänst. I mjukvara som tjänst modellen erbjuder leverantören konsumenter hela applikationer som körs på molntjänstens hårdvara. Konsumenter har inte kontroll över hårdvaran eller mjukvaran (möjligen några små konfigurationsalternativ) utan använder bara applikationen. Med plattform som tjänst modellen sköter leverantören hårdvaran och en del av mjukvaran som operativsystemet, medan konsumenten kan skapa och använda sin applikation med verktyg, programmeringsspråk, bibliotek och tjänster som leverantören stöder. Infrastruktur som tjänst modellen betyder att leverantören sköter bara underliggande hårdvaran medan konsumenten sköter all mjukvara som operativsystem och applikationer.

De fyra driftsättningsmodellerna är privat moln, gemensamt moln, allmänt moln och hybridmoln. I ett privat moln använder en organisation exklusivt all infrastruktur och denna infrastruktur ägs och hanteras av organisationen själv eller extern leverantör. I ett gemensamt moln däremot används infrastrukturen exklusivt av en gemenskap av organisationer som har gemensam oro över till

exempel säkerhet. Infrastrukturen kan ägas och hanteras av en eller några i gemenskapen eller en extern leverantör. Allmänt moln är en driftsättningsmodell där infrastrukturen är öppen för användning av vem som helst över internet. Hybridmoln är en kombination av de andra modellerna. Konsumenter kan välja att använda flera olika driftsättningsmodeller på samma gång och på så sätt skapa unika lösningar.

2.2. Sakernas internet

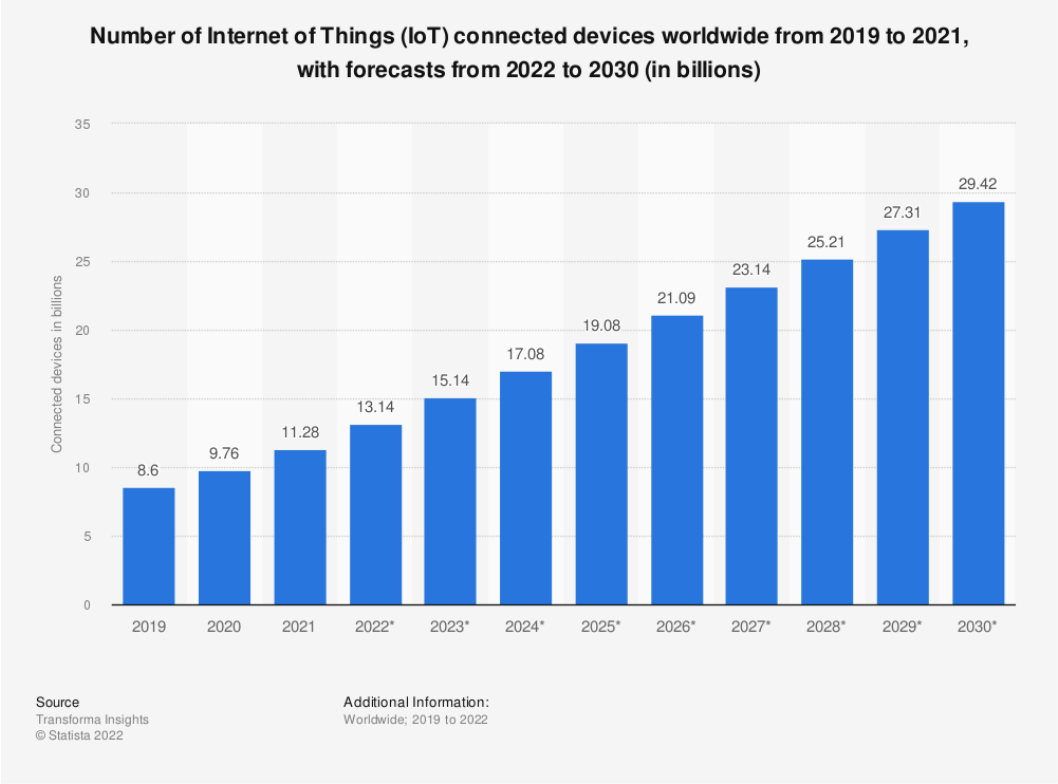
Medan det inte finns en universell definition på sakernas internet (eng. Internet of Things) avser man oftast vardagliga objekt som med hjälp av inbyggda system kopplas till internet [4]. Dessa objekt blir således ”smarta” och kan skapa, dela och hantera data. Ständiga förbättringar i datanätverk, dataanalys, datakomponenter och uppkomsten av molntjänster har möjliggjort detta. Inbyggda system måste ofta vara så små och effektiva som möjligt för att kunna placeras i vardagliga objekt och mobila enheter som till exempel mobiltelefoner, smarta TV:ar eller termostater. Detta medför att dessa har en väldigt begränsad mängd datorresurser. För att lösa detta används oftast molntjänster [4]. Figur 2.1 visar enhet till moln modellen. I denna modell skickar enheten data som den skapar till molnet via internet. Beräkningar, analys och lagring sker då i molntjänstens datacenter i stället för på enheten.



Figur 2.1 Enhet till moln kommunikationsmodellen [4].

Molntjänstens roll i sakernas internet är oftast att aggregera och bearbeta data från många enheter och skapa insikter eller göra beslut baserat på denna data [5]. Låg latens är därför viktigt för att kunna snabbt kunna göra beslut.

Cisco uppskattade att hälften av alla enheter på internet år 2023 kommer att vara så kallade maskin till maskin (M2M) enheter med smarta hemenheter största delen av dessa [6]. Som figur 2.2 visar förväntas antalet enheter inom sakernas internet tredubblas från år 2020 till år 2030. Detta kombinerat med den stora mängd data som dessa enheter skapar betyder en enorm ökning i datatrafik. Detta skapar utmaningar för nuvarande enhet till moln modellen. På grund av dess distribuerade arkitektur och närhet till enheterna är kantdatorsystem bra lämpade att lösa dessa utmaningar.



Figur 2.2 Antalet enheter inom sakernas internet och förväntad tillväxt [1].

3. Kantdatorsystem

3.1. Varför kantdatorsystem?

En ökad mängd enheter (huvudsakligen inom sakernas internet) och nya krav som de skapat har lett till brister i den traditionella molntjänstarkitekturen [7]. Problem som kantdatorsystem ämnar att lösa är följande [8] [5]:

Latens: Latens betyder tiden det tar för en enskild bit eller paket information att nå sin destination. Den fysiska distansen mellan källan och destinationen är den största orsaken till ökad latens men också nätverkstopologin inverkar.

Onödig datatransport: Mycket av de data som skapas är ”skräpdata” men det kan vara svårt att veta exakt vilken del det gäller utan att analysera data och enheten som samlar in data kanske inte har kapaciteten att göra det.

Tidsvärde: Mycket av de data som genereras måste behandlas inom en viss tid eller så minskar värdet exponentiellt. Till exempel realtidssystem som robotar behöver en så snabb behandling av data som möjligt för att kunna utföra handlingar i realtid.

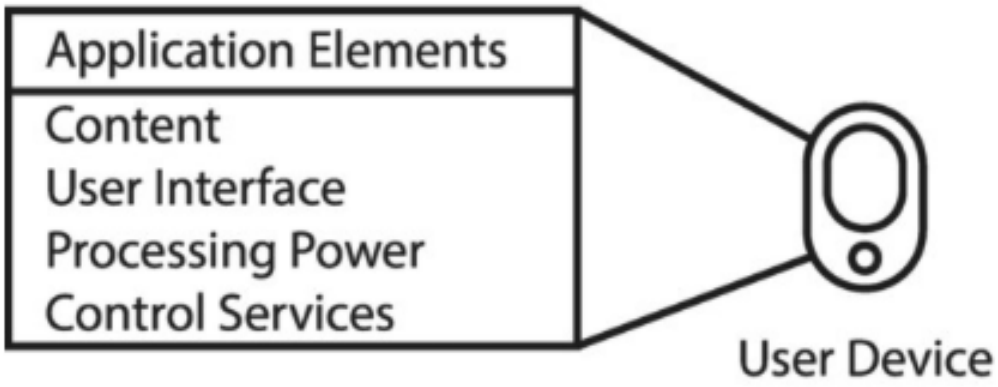
Transportkostnader: Att transportera en stor mängd data från enheterna som skapar dem till enheter som kan lagra eller utföra beräkningar kostar mycket energi och kräver mycket bandbredd. Detta skapar hög belastning på nätverket vilket ökar kostnaderna.

Säkerhet och integritet: Att transportera känsliga data över publika internet till molntjänster innebär säkerhetsproblem. Kantdatorsystem kan undvika detta eftersom de finns på samma privata nätverk som enheterna som skapar data. Kantdatorsystem kan också användas för att filtrera ut känsliga data innan det skickas till molnet.

Den största drivande faktorn bakom kantdatorsystem är lokalitet. För att kunna driva nya system som kräver komplexa beräkningar i realtid krävs beräkningsresurser nära enheterna som behöver den.

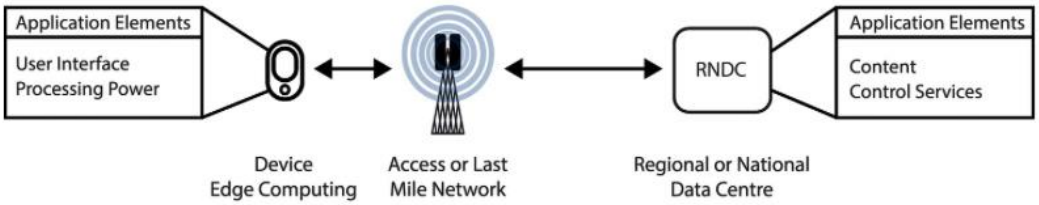
3.2. Grundläggande operation i kantdatorsystem

För att illustrera hur kantdatorsystem fungerar i praktiken tänkte jag gå igenom ett exempel [8]. Först tänk dig en applikation som körs bara på en och samma enhet som till exempel en smarttelefon. Alla resurser som applikationen behöver som beräkning och lagring finns på samma enhet. Figur 3.1 visar detta scenario. Applikationen är på så sätt begränsad av smarttelefonens förmåga att utföra dess funktioner. Detta kanske inte är ett problem med mindre applikationer men för sådana applikationer som behöver en större mängd beräkningskapacitet och lagring så räcker troligtvis inte kapaciteten på en batteridriven enhet till.



Figur 3.1 En applikation som körs på en ensam enhet [8].

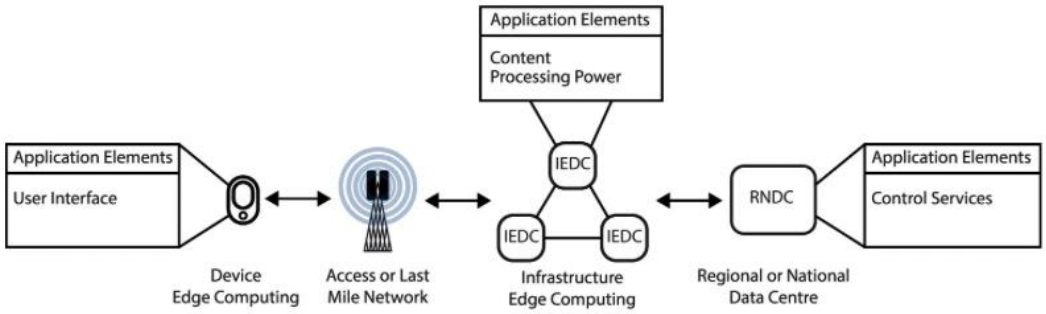
För att försöka lösa dessa begränsningar läggs nu till en molntjänst som ger applikationen tillgång till en stor mängd ny funktionalitet och möjligheter men också några nackdelar. Applikationen är ansluten till ett accessnät, i fallet med mobiltelefonen ett fjärde eller femte generationens mobilnät, och sedan över internet till molntjänstens servrar. Applikationen har nu tillgång till resurser (huvudsakligen beräkning och lagring) som finns på molntjänstens servrar, vilket möjliggör ny funktionalitet. Figur 3.2 visar en applikation som använder en molntjänst.



Figur 3.2 En applikation som använder resurser i molnet [8].

På grund av att molntjänstens servrar ofta befinner sig långt borta från enheten medför detta också två huvudsakliga utmaningar för applikationen. Om applikationen kräver stora mängder dataöverföring mellan enheten och molnet så skapar detta för det första högre nätverksanvändning vilket kan överbelasta nätverket. Det andra problemet är latensen. Realtidsapplikationer kräver tillräckligt låga latenser för att fungera, vilket på grund av fysiska begränsningar inte är möjligt med detta system.

Genom att lägga in ett kantdatorsystem mellan accessnätverket och molntjänsten kan en del av beräkningskapaciteten flyttas närmare enheten och på så sätt lösa problemen med att använda enbart molntjänster. Denna typ av system illustreras i figur 3.3.



Figur 3.3 En applikation som använder ett kantdatorsystem [8].

Applikationen har nu tillgång till en ökande mängd resurser i en ökande distans från enheten. Enheten själv, kantdatorsystemet några tiotals kilometer från enheten och molntjänsten som kan befinna sig tusentals kilometer från enheten. Resurserna som applikationen behöver kan nu delas upp på bästa möjliga sätt mellan dessa och på så sätt möjliggöra ny funktionalitet.

För att upprätthålla låga latenser krävs det att kantdatorsystemen är tillräckligt nära enheterna (optimalt några tiotals kilometer). Detta betyder att det behövs många av dessa distribuerade över stora områden.

3.3. Dataresurser vid nätverkets kant

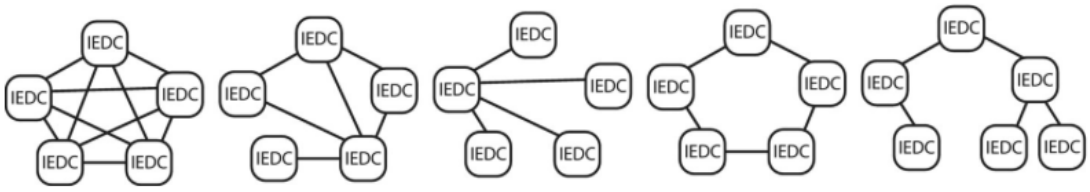
För att göra beräkningar vid nätverkets kant krävs enheter som gör dessa beräkningar. Dessa enheter är av varierande storlek med olika funktioner och beräkningskapacitet. Jag kommer fortsättningsvis kalla denna typ av enhet kantdatacenter.

Kantdatacenter har inte samma stordriftsfördelar som datacenter i molnet vilket betyder att kostnaden för dessa är högre. Detta medför att ända fördelarna kantdatacenter har över molntjänster är dess plats. Därför är det väldigt viktigt, för att kunna erbjuda låg latens och transportkostnad, att dessa finns så nära (topologiskt och fysiskt) kundernas accessnätverk som möjligt.

3.3.1. Nätverkstopologi

Kantdatacenter måste kopplas till användarnas accessnätverk för att kunna tillbringa värde [8]. För att detta ska kunna ske måste det finnas enheter i kantdatorsystemets nätverk vid platser där man kan sammankoppla användarens accessnätverk och kantdatorsystemets nätverk. Dessa enheter och kantdatacenter i ett område kopplas ihop och skapar ett nätverk. Jag kommer i detta kapitel kalla alla enheter och kantdatacenter i kantdatorsystemets nätverk för noder. Nätverkstopologin för kantdatorsystemets nätverk av noder ska utforskas i detta kapitel.

Det finns fem nätverkstopologier som är relevanta för kantdatorsystem [8]. Dessa är full mesh, partiellt mesh, stjärna, ring och träd. Figur 3.4 visar dessa.



Figur 3.4 full mesh, partiellt mesh, stjärna, ring och träd nätverkstopologier [8].

I full mesh nätverkstopologin kopplas varje nod till alla andra noder. Detta ger största möjliga redundans men ökar också kostnaden avsevärt för varje ny nod. Detta leder till att det inte är praktiskt att använda denna nätverkstopologi förutom i väldigt små nätverk.

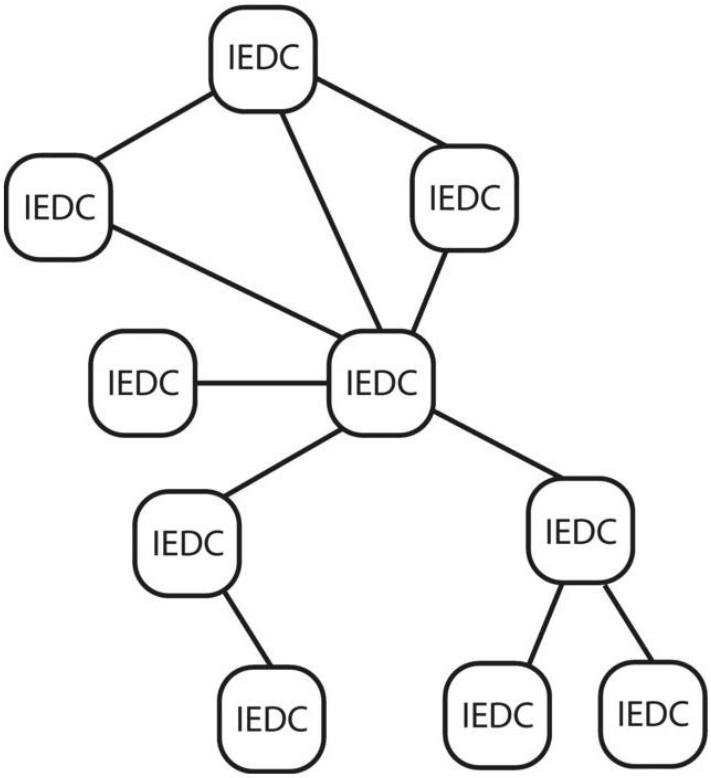
I ett partiellt mesh kopplas varje nod inte till alla noder som i full mesh utan bara några andra noder. Det är logiskt att designa detta mesh så att alla kantdatacenter kan nå alla andra kantdatacenter genom minst två vägar och på så sätt kan trafiken fortsätta om en väg har problem. Denna topologi har inte lika stor redundans som full mesh men är också betydligt mer kostnadseffektiv.

I stjärntopologin finns en nod som alla andra noder kopplas till. Detta leder till att all trafik går genom denna nod. Denna nod måste därför kunna hantera denna trafik och slutar noden fungera så skulle hela nätverket sluta fungera. Dessa problem gör att denna nätverkstopologi inte är lämplig för ett kantdatorsystem.

I ring nätverkstopologin kopplas noderna ihop som en ring. Detta betyder en viss redundans eftersom trafik kan ta två vägar till en nod dock inte på samma nivå som fullt eller partiellt mesh.

Träd nätverkstopologin betyder att större noder kopplas med flera mindre noder som ett träds grenar. Detta leder dock till samma problem som i stjärntopologin men gör det också lätt att lägga till nya mindre noder.

Den optimala nätverkstopologin för ett nätverk med större kantdatacenter är partiellt mesh kombinerat med användningen av träd för mindre noder [8]. Figur 3.5 visar denna nätverkstopologi. Detta ger största möjliga balans mellan redundans och kostnad och ger möjligheten att enkelt lägga till nya noder.



Figur 3.5 Partiellt mesh kombinerat med träd [8].

Enklare ring nätverkstopologin kan också användas i ett mindre nätverk för att minska kostnaderna.

3.3.2. Storleksklasser

Kantdatacenter kan designas i många olika storlekar beroende på användningsfall och behov i ett område [8]. Alla kantdatacenter kommer dock att vara betydligt mindre än ett datacenter i en molntjänst. Tabell 3.1 visar olika storleksklasser för kantdatacenter och deras kapacitet för kundutrustning.

En viktig egenskap att tänka på när man designar ett kantdatacenter är modularitet [8]. Modularitet betyder att man delar upp någonting i olika funktionella delar och medför större flexibilitet. Detta ger operatörer av kantdatacenter möjligheten att lägga till kapacitet i ett senare skede om så behövs.

Tabell 3.1 Olika storleksklasser av kantdatacenter [8].

Storleksklass	Typisk kapacitet (kW)	Storlek (Bredd x Höjd x Djup i meter)	Antal i en typisk stad	Exempelroll
1	<1	<1x1x0.5	10	Nätverkstermination
2	1–10	1.5x1.5x1.5	20	Nätverksfunktioner
3	10–50	3x2x3	5	Nätverksaggregation
4	50–100	3x2x5	3	Låg latensarbeten
5	100–200	3x2x8	2	Låg latensarbeten
6	200–250	5x2x8	1	Regional kantdatacenter

Storleksklass 1 är den minsta storleksklassen och tack vare den lilla storleken kan placeras på många olika platser som till exempel trafikljus. På grund av storleken används denna klass oftast för nätverkstermination och sammankoppling till kantdatorsystemets nätverk.

Storleksklass 2 är större än klass 1 och kan därför inte placeras lika flexibelt. Denna klass har som funktion att sammankoppla accessnätverk med kantdatorsystemets nätverk och skicka trafiken vidare till större kantdatacenter. Applikationer förutom nätverksfunktioner kunde ske i denna klass men oftast är det bättre att göra det i de större klasserna.

Storleksklass 3 är där traditionella multitenans datacenter modellen blir möjlig. Denna klass är väldigt flexibel i storlek och innehåll. Man kan börja med ett mindre kantdatacenter med en enskild kund för att sedan öka kapaciteten för att stöda flera kunder.

Storleksklass 4 och 5 är som ett traditionellt datacenter i en molntjänst med relativ kompakt storlek. Dessa klasser tillsammans ger största delen av beräkningskapaciteten. Till skillnad från klass 3 är dessa kantdatacenter

designade och byggda på förhand och är därför inte lika flexibla för förändring över tid men de har också betydligt mera kapacitet. Det är i dessa klasser som största delen av applikationerna kommer köras.

Storleksklass 6 är den största klassen och kan fungera som ett regional kantdatacenter som aggregerar all trafik från de mindre klasserna och kopplas samman med ett datacenter i molnet.

Storlekar större än dessa är inte flexibla nog och har därför svårt att fungera som kantdatacenter. Behöver man mera kapacitet kan man lägga till flera av dessa klasser i ett område i stället.

3.4. Skillnader mellan molntjänster och kantdatorsystem

Både molntjänster och kantdatorsystem har många likheter. Båda använder samma resurser (nätverk, lagring och beräkning) och tekniker (virtualisering, multitenans) [9] men det finns också vissa skillnader som ska utforskas i detta kapitel.

Molntjänster är baserade på stordriftsfördelar och på så sätt är deras datacenter stora och placeras på platsen med mest ekonomiska fördelar [7]. I ett kantdatorsystem däremot är resurserna av varierande storlek och distribuerade över ett stort område vilket gör att distansen till användaren minskar och på så sätt också latensen och nätverksanvändningen. Storleken på ett kantdatacenter är betydligt mindre än ett datacenter i en molntjänst. Tabell 3.2 visar de viktigaste skillnaderna mellan molntjänster och kantdatorsystem.

Tabell 3.2 Skillnader mellan molntjänster och kantdatorsystem [9].

	Molntjänster	Kantdatorsystem
Arkitektur	Centraliserad (servrar i datacenters)	Distribuerad (kantnoder)
Plats på noder	Internet (global)	Vid kanten av nätverket (lokal)
Kännedom om plats	Nej	Ja
Latens	Medel	Låg
Antal noder	Få (stora servrar)	Många (kantnoder)
Rörlighetsförmåga	Begränsad	Högre
Beräkningskapacitet	Högre	Lägre
Anslutning	Internet	Flera olika protokoll och standarder
Analysförmåga	Långsiktig/djup	Kortsiktig
Risk för avbrott	Medel	Låg
Säkerhet	Medel	Hög (p.g.a. distribuerad arkitektur)

4. Applikationer

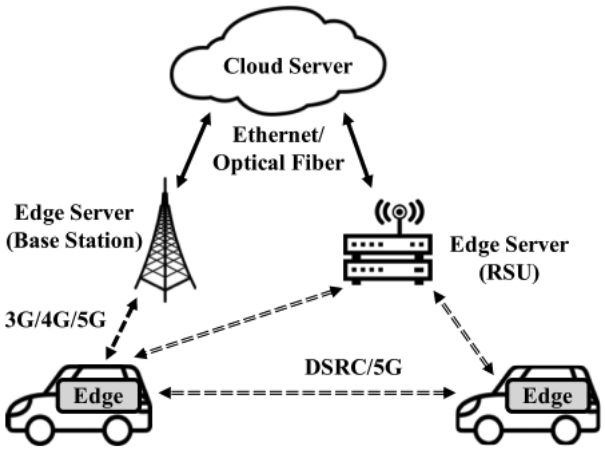
4.1 Mobilspel

(https://sci-hub.se/10.1109/jiot.2018.2805263)

4.2 Autonoma bilar

Säkerhet är en av de viktigaste aspekterna i en autonom bil. Bilen måste kunna reagera på faror väldigt snabbt och ju snabbare bilen kan reagera desto säkrare är färden. Datorsystemet i en autonom bil måste också kunna hantera en enorm mängd data (så mycket som 2gb/s) [10]. Detta kombinerat med dessa bilar är oftast batteridrivna medför väldigt strikta krav på latens, beräkningskapacitet och energieffektivitet. Kantdatorsystem möjliggör låg latens och hög datahanteringsförmåga och är därför ett bra val för en autonom bil.

Som figur 4.1 visar innehåller varje autonom bil ett kantdatorsystem som hanterar alla bilens funktioner. Bilen är också kopplad till kantdatacenter med 4G/5G och sedan till molnet via internet. Bilen kan också kontakta enheter nära vägen (RSU) och andra bilar via 5G eller DSRC som är en teknologi för trådlös kommunikation med kort räckvidd.



Figur 4.1 Ett kantdatorsystem i en autonom bil [11].

…

(https://sci-hub.se/10.1109/jproc.2019.2915983)

4.3 Sakernas internet för industri

(IIoT, Industrial internet of things)

5. Sammanfattning

Referenser

[1] Statista, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030," July 2022. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/. [Accessed February 2023].

[2] Statista, ”Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025,” June 2021. [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/. [Använd February 2023].

[3] T. G. Peter Mell, ”The NIST Definition of Cloud Computing,” September 2011. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. [Använd 2023].

[4] S. E. L. C. Karen Rose, ”The Internet of Things (IoT): An Overview,” The Internet Society (ISOC), 2015.

[5] K. S. C. S. Pethuru Raj, Edge/Fog Computing Paradigm: The Concept Platforms and Applications, 2022.

[6] Cisco, ”Cisco Annual Internet Report (2018–2023) White Paper,” Cisco, 2018.

[7] R. M. J. Z. S. A. Flavio Bonomi, ”Fog Computing and Its Role in the Internet of Things,” i *SIGCOMM*, Helsinki, 2012.

[8] A. Marcham, Understanding Infrastructure Edge Computing: Concepts, Technologies, and Considerations, Wiley, 2021.

[9] A. Synyaev, Internet Computing Principles of Distributed Systems and Emerging Internet-Based Technologies, Springer, 2020.

[10] S. a. T. J. a. Z. Z. a. G. J.-L. Liu, ”Computer architectures for autonomous driving,” *Computer,* vol. 50, nr 8, pp. 18-25, 2017.

[11] S. a. L. L. a. T. J. a. Y. B. a. W. Y. a. S. W. Liu, ”Edge Computing for Autonomous Driving: Opportunities and Challenges,” *Proceedings of the IEEE,* vol. 107, nr 8, pp. 1697-1716, 2019.