UNIVERSITETET I OSLO
Institutt for informatikk

Sikkerhet i Sensornettverk

Hvordan innføring av sikkerhet i trådløse sensornettverk påvirker strømforbruket, funksjonaliteten og levetiden til sensornodene

Masteroppgave

Ole Kristian Bergseth

1 August 2005

Abstract

This master thesis presents the effect that security has on a sensor network. A wireless sensor network is composed of a magnitude of sensors that together form a sensor network through self- organizing and self- configuring capabilities. A sensor node is small unit that functions as a transformer which converts physical signals into electrical signals. The sensor unit itself is composed of a sensor, a processor, a transceiver and a power supply like a battery. This battery is a small sized battery, so it is important to optimize all the components and software, so that the sensor nodes consume as little power as possible. Introducing security in a system which has so many restrictions as a sensor network is very challenging. All these challenges rules out the use of asymmetric encryption techniques, which make symmetric encryption techniques more relevant for a sensor network.

This report gives an overall description of the sensor network technology. We present security techniques that can be suitable for a sensor network. Additionally we look at the degree of security that we can obtain with the different security techniques. It is important that security is chosen carefully, so that the cost associated the particular security techniques don't drastically degrade the lifetime of a sensor node. Then we are going to illustrate the problems that arise when security is to be introduced in a sensor network. Further we have studied the overhead associated with known symmetric techniques, and also how this overhead effects the packet growth, throughput, transmission time, and how this effects the power consumption. The power consumption has been calculated through several scenarios, where we investigated how the lifetime of the sensor nodes will degrade when different security overhead is introduced into the data packet, and also how the node density can effect the power consumption in the sensor network. We have also tested how a sensor network operates at different load, and we have calculated metrics like power consumption, end to end delay, and average throughput in the sensor network. All these results have been obtain through simulations with the use of NS-2 network simulator and Sensorsim, and the results shows that our suggestion for a security framework can be implemented in a sensor network.

Forord

Denne masteroppgaven er skrevet som en del av studiene mine ved Universitetet i Oslo.

Jeg vil gjerne takke min interne veileder Frank Yong Li ved UNIK, og min eksterne veileder Knut Grythe ved SINTEF for deres veiledning.

Figuroversikt

Tabelloversikt

Formler

1 Introduksjon

Ad hoc og sensornettverk er to forskningsområder som har fått mye oppmerksomhet de siste årene, og det er tenkt at disse teknologiene vil representere fremtidens trådløse kommunikasjon. Det som først og fremst gjør disse teknologiene revolusjonerende er at kommunikasjonen foregår via multihopp og uten fast infrastruktur, samt at nettverket er selvkonfigurerende og selvorganiserende. Et sensornettverk er i realiteten en type ad hoc nettverk, men det er allikevel en del faktorer som skiller dem, og da spesielt ressursene og applikasjonene. En sensornode er en relativt liten enhet som skal kunne kombinere sensoregenskaper, prosessering, og kommunikasjon for å utføre sine oppgaver. De største forskjellene er at sensornettverket består av mange flere noder. Disse nodene er utplassert relativt tett, har store ressursbegrensninger (grunnet små i størrelse), og det at sensorene er beregnet til å utføre målinger og observasjoner.

De siste årene er det blitt gjort store fremskritt innen mikroelektromekaniske systemer (MEMS) [1][2]. Dette har ført til store gjennombrudd innen områder som digitalteknologi, batteriteknologi, og trådløs kommunikasjon, noe som har gjort sensorene mindre, billigere, mer allsidig, mer pålitelig, og mer levedyktig. Denne utviklingen gjør en teknologi som trådløse sensornettverk realiserbar, spesielt siden det er mulig å redusere strømforbruket i sensornodene, og på denne måten forlenge levetiden til sensornettverket. Et trådløst sensornettverk skal kunne bestå av et stort antall (hundrevis/tusenvis) autonome noder som til sammen danner et Ad hoc nettverk. Utplasseringen av disse sensornodene trenges ikke bestemmes på forhånd, noe som gjør at sensornodene kan utplasseres tilfeldig i forskjellige miljøer. Derfor må protokollene og algoritmene i et sensornettverk ha funksjonalitet som gjør dem i stand til å være selvorganiserende og selvkonfigurerende. Sensornettverket har dessuten en unik egenskap hvor sensornodene i sensornettverket skal ha mulighet til å samarbeide seg imellom, og sensornodene er derfor utstyrt med prosesseringsegenskaper, slik at rådata kan prosesseres lokal, for så å videresendes til sinken når nødvendig. Disse sensornodene karakteriseres ved at de har begrensninger i form av batterilevetid, liten strømenhet, lav båndbredde, liten lagringsplass, og lite minne.

Det er mange fordeler med et selvorganiserende trådløst sensor nettverk. De kan benyttes i mange sammenhenger, blant annet til helsemessige og medisinske formål, til militære formål, i hjem, samt i miljø- og katastrofe- områder. Levetiden til et sensornettverk er særdeles viktig ettersom det selvorganiserende trådløse sensornettverket skal kunne utføre forskjellige oppgaver og målinger avhengig av applikasjon. Sensornodene må i de fleste sammenhenger kunne klare seg uten vedlikehold, og må derfor være i stand til å tilpasse seg det miljøet den befinner seg i. De fleste protokollene som benyttes i tradisjonell nettverk eller i ad hoc nettverk kan ikke benyttes i et sensornettverk. Dette er på grunn av de karakteristiske forskjellene mellom disse to teknologiene.

Det å innføre sikkerhet i trådløse nettverk er en stor utfordring ettersom mediet som data transmitteres over er radio. Det innebærer at det er et delt medium hvor alle kan høre en transmisjon så lenge enhetene er innenfor hverandres transmisjonsradius. Det å tilby autentisering, integritet og konfidensialitet er viktig for å hindre en motstander i å kompromittere sikkerheten i sensornettverket. Men det å benytte tradisjonelle sikkerhetsfunksjoner for å oppnå god sikkerhet er ikke enkelt på grunn av karakteristikken til sensornettverket. Vi skal se nærmere på ulike sikkerhetsmål, ulike sikkerhetsfunksjoner, samt hvilke protokoller som kan være passende for et sensornettverk. Hovedsaklig vil det bli fokusert på symmetriske krypteringsalgoritmer ettersom krypteringsskjema av typen

fellesnøkkel er vanskelig å realisere i nåværende sensorer [12]. Dette skyldes ikke bare kodestørrelsen (lagring) og prosesseringsbegrensningene, men også sensornodenes høye krav til strømforbruket, noe som begrenser levetiden til sensornodene betraktelig.

En av de største begrensingene i en sensornode er kravet om lavt strømforbruk. Siden sensornoden er utstyrt med en liten strømforsyning, vil det være nødvendig med en balanse mellom batterilevetiden og funksjonaliteten til sensornoden. Vi skal se på hvordan innvirkningen sikkerhet har i et sensornettverk, og da spesielt på strømforbruket. Vi skal også se på hvordan funksjonaliteten til sensornodene reduserer levetiden. Altså desto oftere sensornodene er i funksjon, desto kortere vil sensornettnodenes levetid være. Dette er noe som må taes hensyn til ved design av et sensornettverk. Det vil først og fremst stilles store krav til hvordan sikkerheten designes med hensyn på batterikapasiteten. Fremfor å prioritere høy sikkerhet, må sensornettverk først og fremst satse på strømbevaring grunnet begrenset batterikapasitet. Det vil derfor også være en balanse mellom sikkerheten og strømforbruket; desto bedre sikkerhet, desto mer prosessering og sikkerhetsoverhead tilføres i pakkene, samt mer transmisjonstid kreves for å transmittere pakkene, noe som igjen vil føre til høyere strømforbruk og kortere levetid. Det må her taes hensyn til hvor god sikkerhet som egentlig trengs i ulike sammenhenger, slik at man ikke tilfører mer overhead enn nødvendig. Sikkerhetsmål, samt forskjellige varianter av sikkerhet og trusler, og løsninger for disse vil bli diskutert i kapittel 3.

Hovedbidragene i denne rapporten er å se på hvordan innføringen av sikkerhet i sensornettverket påvirker strømforbruket, funksjonaliteten og levetiden til sensornodene. Vi skal foreslå 2 passende pakkeformat for et sensornettverk. Deretter skal vi så på hvordan sikkerhetsoverheaden i disse pakkeformatene påvirker pakkeveksten, transmisjonstiden, gjennomstrømningen, samt hvordan strømforbruket øker ettersom ulik grad av sikkerhet innføres. Vi skal også se på hvordan strømforbruket endres ut ifra flere ulike scenarioer som involverer ulike driftssykluser. Til slutt vil vi se på hvordan levetiden til et sensornettverk degraderes over tid ved hjelp av en simulator. Vi vil benytte en nettverkssimulator med navn NS-2 for å simulere ulike scenarioer i et sensornettverk.

Oppgaven er organisert i 7 kapitler:

- Kapitel 2 gjennomgår sensornodenes og sensornettverkets arkitektur og virkemåte. Kapitelet er ment å gi en bedre forståelse for hva et sensornettverket er, hvordan det fungerer, hva som skiller et trådløst sensornettverk fra et tradisjonelt sensornettverk, hva som skiller et sensornettverk fra et ad hoc nettverk, samt hva et sensornettverket kan brukes til.
- Kapitel 3 tar for seg sikkerheten i sensornettverket. Både sikkerhetsmål og trusler blir presentert, samt aktuelle sikkerhetsmekanismer som kan være passende for et sensornettverket. Til slutt blir det presentert noen sikkerhetsprotokoller som det har blitt forsket på i seneste tid.
- I Kapitel 4 kartlegger vi overheaden som oppstår ved innføring av sikkerhet. Først illustreres problematikken med å innføre sikkerhet i et ressursbegrenset sensornettverk, deretter foreslåes et passende sikkerhetsformat som vi skal bruk videre i oppgaven. Vi skal se på 2 scenarioer, et som tar for seg autentisering og integritet, samt et som tar for seg konfidensialitet.
- Kapitel 5 presenterer estimerte beregninger og resultater som har blitt oppnådd som et resultat av valgt sikkerhetsformat. Dette er resultater som økt pakkevekst, økt transmisjonstid, gjennomstrømning, samt strømforbruket assosiert med å sende og

motta en pakke. Tilslutt skal vi se på hva slags sikkerhetsnivå man kan forvente å oppnå med valgt sikkerhet.

- Kapitel 6 tar for seg ulike scenarioer i sensornettverket, både estimerte og simulerte scenarioer. Dette inkluderer ulike driftssykluser, og hvordan levetiden til en sensornode varierer ettersom driftssyklusen til en sensornode endres. Her ser vi på strømforbruket over tid, både for sensoren, prosessering, og kommunikasjon. Visse antakelser vil bli foretatt her. Til slutt foretaes det en del simuleringer for å kartlegge strømforbruket ved ulike situasjoner som kan oppstå i et sensornettverket.
- Kapitel 7 består av hovedkonklusjonen, samt forslag til fremtidig arbeid.

2 Sensornettverkets arkitektur og virkemåte

Sensornettverk er en spesiell type Ad hoc nettverk som består av et stort mangfold av små sensornoder som til sammen danner et selvorganiserende trådløst sensornettverk uten bruk av fast infrastruktur [1][2]. Det at sensornettverket er et Ad hoc nettverk betyr at alle sensornodene har ruteregenskaper, og at sensornodene kan kommunisere direkte med hverandre, eller via hverandre hvis destinasjonen er utenfor rekkevidden. Sensornettverket dannes rundt en eller flere sink, som i realiteten er en basestasjon, som har som oppgave å fungere som et grensesnitt mot omverdenen. Periodiske transmisjoner av signaler gjør det mulig for sensornodene å opprette en rutingstopologi. Enhver node kan videresende en melding til sinken ved å identifisere adressen på meldingen, samt håndtere kringkasting av meldinger. Sinken har mye mer minne enn sensornodene, noe som innebærer at sinken enkelt kan lagre kryptografiske nøkler.

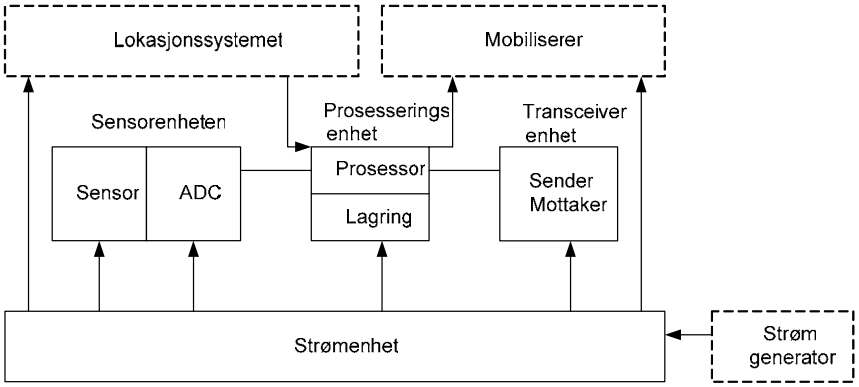
Sensornodene har som primæroppgave å foreta målinger og datainnsamlinger fra miljøet den er utplassert i, og dette kan skje konstant, eller periodevis [1][2]. Dette innebærer at sensornodene samler inn data, for så å rute data til sinken, som igjen ruter data videre til sluttbrukeren. Sensornettverket benytter seg av en eller flere sink, som funger som et grensesnitt mellom det infrastrukturløse sensornettverket og sluttbrukerne. Sinken kontrollerer alle sensorene, og den er såpass kraftig og ressurssterk, at den vil kunne overgå levetidene til alle sensornodene. Siden sensornettverket er et nettverket uten infrastruktur, må kommunikasjonen mellom nodene foregår via multihopp for å kunne nå sinken [1][2]. I tillegg til å samle inn data, kan sensornettverket også utføre prosessering innad i sensornettverket, som blant annet dataaggregering og kompresjon.

2.1 Sensornodens arkitektur

Sensornodene i sensornettverket er billige små sensorer med relativt lavt strømnivå. Hver sensornode i et slik nettverk er utstyrt med sensormekanismer, dataprosesseringsegenskaper og antenner for trådløs kommunikasjon. Disse sensornodene skal til sammen være i stand til å produsere og utveksle informasjon av god kvalitet over korte avstander. Sensornodenes primæroppgave er å samle inn data, enten alene, eller i samarbeid med andre sensornoder.

2.1.1 Sensornodens komponenter

Selve sensornoden består av 4 basiskomponenter (se figur 1) [1][2][3][48]. En sensorenhet, en prosesseringsenhet, en transceiverenhet (sender/mottaker- enhet), samt en strømenhet. Sensornoden kan også utstyres med tilleggsenheter, som blant annet et lokasjonssystem, en strømgenerator, eller en mobiliseringsenhet (er noen ganger nødvendig for å flytte sensornoder når den er påkrevd å utføre de tildelte oppgavene)[1]. Alle slike tilleggsenheter er applikasjonsavhengig, og må vurderes avhengig av nødvendighet ettersom de vil øke sensornodens strømforbruk.



Figur 1. Basiskomponentene til sensornoden.

2.1.1.1 Sensorenheten

Sensorenheten består av to deler, en sensor og en analog/digital konverter. Sensornoden samler inn data ved hjelp av målinger og observasjoner foretatt av sensor. Denne innsamlingen skjer ved at sensoren konverterer fysiske hendelser til elektriske signaler. Når en hendelse detekteres kan sensornoden rapportere hendelsen til en sink eller konsultere med de andre nodene i nettverket. Rapportene kan leveres umiddelbart når en hendelse detekteres eller lagres lokalt for senere levering. De analoge signalene produsert av sensorene basert på sensorenes observasjoner blir konvertert til digitale signaler, og så videresendt til prosesseringsenheten for prosessering.

2.1.1.2 Prosesseringsenheten

Prosesseringsenheten er utstyrt med en mikroprosessor som prosesserer data som kommer inn. I tillegg er også prosesseringsenheten assosiert med en liten lagringsenhet, hvor prosessoren kan lagre prosessert data. Prosessoren er dessuten utstyrt med en form for intelligens, slik at den kan håndtere prosedyrer som gjør at sensornoden kan samarbeide med andre sensornoder. Sensornoden kan kommunisere direkte med de andre sensornodene, eller sende data til sinken, enten direkte hvis sinken er innenfor sensornodens rekkevidde, eller via multihopp hvis sinken er utenfor sensornodens rekkevidde, og da må kommunikasjonen rutes via de andre sensornodene. Data som prosesseres lagres lokalt, for så å videresendes når nødvendig. Hendelser kan prosesseres på 3 nivåer [35]: Internt i sensornoden, lokalt av nabonoder, eller globalt. Internt i sensornoden da foretar sensornoden selv datainnsamling og prosessering av data, uten å konsultere med noen av de andre sensornodene. For lokalt og globalt nivå, samles rådata eller forhåndsprosessert data fra flere sensornoder, hvor all innsamlet data sendes til en enkelt lokasjon for prosessering for så å sammenslå data til mer komplett informasjon. Dette foregår da via intern forbindelse mellom disse samarbeidende sensornodene. Avhengig av hvor mye data som skal sendes, så kan det være mer strømsparende å først prosessere data lokalt før data sendes. Dette fordi kostnaden for å sende 1 bit er mye større enn kostnaden for å prosessere 1 bit [12]. Derfor bør rådata først prosesseres lokalt, for så å utveksle resultatet med andre sensornoder eller med sink. Dette gjør at færre bits blir transmittert, og mindre strøm forbrukes. Ettersom data propagerer i sensornettverket, kan andre sensornoder sammenslå sine data med data fra andre nabonoder. Tabellen [39] på neste side (tabell 1) viser en oversikt over hva slags type prosessorer ulike fabrikanter har utviklet. I tillegg viser tabellen når prosessorene først ble utgitt, samt strømforbruket, og hvor mye RAM og flashminne hver prosessor er utstyrt med. En veldig populær sensornode, Mica2 mote, benytter en prosessor av typen Atmel Mega128, som er utstyrt med 4 kb Ram, 128 kb Flashminne.

Leverandør	Enhet	RAM (kB)	Flash (kB)	Aktiv (mAh)	Sove (µAh)	Utgitt
Atmel	AT90LS8535	0,5	8	5	15	1998
	Mega128	4	128	8	20	2001
	Mega165/325/645	4	64	2.5	2	2004
General instruments	PIC	0,025	0,5	19	1	1975
Microchip	PICModern	4	128	2,2	1	2002
Intel	40044-bit	0,625	4	30	N/A	1971
	80518-bitClassic	0,5	32	30	5	1995
	805116-bit	1	16	45	10	1996
Philips	80C5116-bit	2	60	15	3	2000
Motorola	HC05	0,5	32	6,6	90	1988
	HC08	2	32	8	100	1993
	HCS08	4	60	6,5	1	2003
Texas	TSS4004-bit	0,03	1	15	12	1974
Instruments	MSP430F14x16-bit	2	60	1,5	1	2000
	MSP430F16x16-bit	10	48	2	1	2004
Atmel	AT91ARMThumb	256	1024	38	160	2004
Intel	XScalePXA27X	256	N/A	39	574	2004

Tabell 1. Ulike typer prosessorer fra ulike leverandører som er passende for små enheter.

2.1.1.3 Transceiverenheten

Transceiverenheten kobler sensorenheten til nettverket ved hjelp av trådløs kommunikasjon. Kommunikasjonen mellom sensornodene foregår ved at meldinger kringkastes ut i nettet, noe som betyr at alle sensornodene som er innenfor rekkevidden til signalet kan lese datapakkene. Tabell 2 viser en oversikt over ulik radioer som kan benyttes i et sensornettverk. Tabellen viser blant annet ulike leverandører, hvor hver av disse transceiverenhetene har ulike karakteristikk. Denne karakteristikken er kartlagt i tabell 2 [18].

Type	**Smalbånd (Narrowband)**				**Bredbånd (Wideband)**		
Leverandør	RFM	Chipcon	Chipcon	Nordic	Chipcon	Motorola	Zeevo
Modellnummer	TR1000	CC1000	CC2400	nRF2401	CC2420	MC13191/92	ZV4002
Maks Datarate kbps)	115,2	38,4	1000	1000	250	250	723,2
Motta effekt (mAh)	3,8	9,6	24	18(25)	19,7	37(42)	65
Sendeeffekt (mAh/dBm)	12/1,5	23/10	19/0	13/0	17,4/0	34(30)/0	65/0
Sovemodus (µAh)	1	1	1.5	0,4	1	1	140
Tiden det tar å slå på radioen (ms)	0,02	2	1,13	3	0,58	20	*
Modulasjon	OOK/ASK	FSK	FSK, GFSK	GFSK	DSSS-O-QPSK	DSSS-O-QPSK	FHSS-GFSK
Pakkedetektering	Nei	Nei	Programmerbar	Ja	Ja	Ja	Ja
Adressekoding	Nei	Nei	Nei	Ja	Ja	Ja	JA
Støtte for kryptering	Nei	Nei	Nei	Nei	128-bitAES	Nei	128-bitSC
Feildeteksjon	Nei	Nei	Ja	Ja	Ja	Ja	Ja
Feilkorrigering	Nei	Nei	Nei	Nei	Ja	Ja	Ja
Kvittering	Nei	Nei	Nei	Nei	Ja	Ja	Ja
Grensesnitt	bit	byte	Pakke/byte	Pakke/byte	Pakke/byte	Pakke/byte	Pakke
Bufring (bytes)	No	1	32	16	128	133	Ja*
Tids-synkronisering	bit	SFD/byte	SFD/Pakke	Pakke	SFD	SFD	Bluetooth
Lokalisering	RSSI	RSSI	RSSI	Nei	RSSI/LQI	RSSI/LQI	RSSI

Tabell 2. Ulike typer transceivere som kan brukes i et sensornettverk.

2.1.1.4 Strømenheten

Strømenheten består av 1 eller flere batterier som forsyner komponentene i sensornoden med strøm. Dette er den viktigste enheten til sensornoden, ettersom det er batterikapasiteten som bestemmer levetiden til en sensornode. Et sensornettverk må designes med hensyn på batterikapasiteten til sensornodene. Denne batterikapasiteten vil også avhenge av hvor stort batteri, og hvor mange batterier sensornoden er utstyrt med.

2.1.2 Trådløse teknologier

I sensornettverket benyttes et trådløst medium for kommunikasjon mellom nodene hvor kommunikasjonen foregår over en gitt frekvens. Forskjellige trådløse teknologier for kommunikasjon kan benyttes, blant annet radio og infrarød [1][2]. Hvilken av disse som egner seg best i et sensornettverk avhenger av kravene som stilles til hvert enkelt system. Radio er foretrukket ettersom sensorene er små, og dataratene er lave, og frekvens gjenbruken er høy grunnet korte kommunikasjonsavstandene. Foretrukket frekvens er ISM (Industrial, Scientific and Medical) bånd, som er lisensfrie frekvensbånd som kan benyttes for kommunikasjon i mesteparten av verden. De fleste sensornodene benytter frekvenser som 315, 433, 868, og 915 MHz frekvensbånd. En stor fordel med ISM er fri bruk av radioen, det enorme frekvensspekteret og global tilgjengelighet. De er ikke tilknyttet en spesiell standard, noe som gir større frihet ved implementering av radio i et sensornettverk. Noen av disse frekvensene brukes også av andre trådløse systemer (blant annet trådløst LAN). Dette gjelder nyere teknologier som er basert på 802.11 og 802.15 standarden. Infrarød er lisensfri og robust mot interferens fra andre elektriske enheter, og dessuten har de relativt billige

transceivere som er enkle å bygge. Den største ulempen er at infrarød krever fri sikt mellom sender og mottaker.

2.1.3 Forskjellige typer sensornoder

Det finnes mange sensornoder på markedet. Her er en sammenligning av forskjellige sensornoder [39]. Sjekk [46][50] for mer detaljert oversikt over sensornodene.

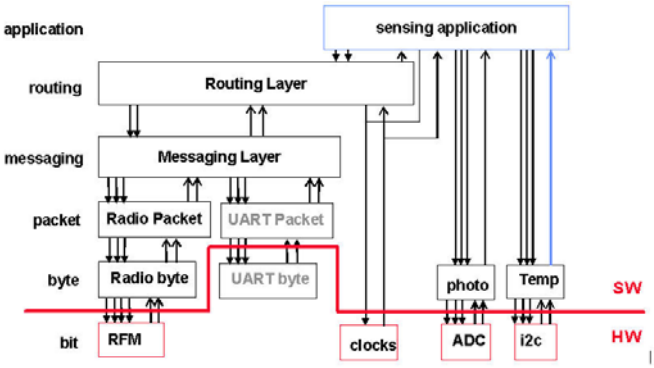
Mote Type Year	*WeC* 1998	*René* 1999	*René 2* 2000	*Dot* 2000	*Mica* 2001	*Mica2Dot* 2002	*Mica 2* 2002	*Telos* 2004
Microcontroller								
Type	AT90LS8535		ATmega163		ATmega128			TI MSP430
Program memory (KB)	8		16		128			60
RAM (KB)	0.5		1		4			2
Active Power (mW)	15		15		15		60	3
Sleep Power (μW)	45		45		75		75	6
Wakeup Time (μs)	1000		36		180		180	6
Nonvolatile storage								
Chip	24LC256				AT45DB041B			ST M24M01S
Connection type	I^2C				SPI			I^2C
Size (KB)	32				512			128
Communication								
Radio	TR1000				TR1000	CC1000		CC2420
Data rate (kbps)	10				40	38.4		250
Modulation type	OOK				ASK	FSK		O-QPSK
Receive Power (mW)	9				12	29		38
Transmit Power at 0dBm (mW)	36				36	42		35
Power Consumption								
Minimum Operation (V)	2.7		2.7		2.7			1.8
Total Active Power (mW)	24				27	44	89	41
Programming and Sensor Interface								
Expansion	none	51-pin	51-pin	none	51-pin	19-pin	51-pin	10-pin
Communication	IEEE 1284 (programming) and RS232 (requires additional hardware)							USB
Integrated Sensors	no	no	no	yes	no	no	no	yes

Figur 2. Oversikt over ulike typer sensornoder.

Figur 2 [50] viser hvilke typer prosessorer, lagring, transceiver de forskjellige sensornodene benytter. I tillegg er det også oppført forventet strømforbruk. Disse sensornodene er som figuren viser av ulike størrelser, avhengig av hva sensornodene skal brukes til. Lagringsplassen brukes til å lagre forskjellig informasjon. ROM/flash eller EEPROM (programmerbart minne) brukes til å lagre programmeringskoden til operativsystemet, sikkerhetsfunksjoner og nettverksegenskaper, mens RAM brukes til å lagre applikasjoner, sensordata og i midletidig prosessert informasjon [12].

2.1.4 Operativsystem

Sensornettverket benytter et open-source operativsystem som kalles TinyOS [43][50][52]. Dette er en hendelsesbasert operativsystem som utviklere kan jobbe ut ifra. TinyOS er designet slik at det skal oppta minst mulig plass i minnet. Dette gjør at TinyOS er spesielt beregnet for enheter som er av liten størrelse, og som har samme ressursbegrensninger som et sensornettverk. Datalasten i TinyOS sitt pakkeformat er på 0 til 29 bytes (Max 29 bytes), og hele pakkeformatet er på 36 bytes, noe som tilsier at pakken kun har 7 bytes med overhead. TinyOS tilbyr dessuten ingen sikkerhetsbeskyttelse, så dette er noe som må komme i tillegg. Sjekk [46] for mer info om TinyOS. TinyOS sin radiostakk [43][57][58] ser ut som i figuren nedenfor.



Figur 3. TinyOS sin radiostakk.

Stakken til TinyOS fungerer på følgende måte: RFM radioen mottar bits og sender bytes bit for bit. Mottak skjer ved at mottaker detekterer et startsymbol, mens ved transmisjon, så sendes det ut et startsymbol, etterfulgt av data. Bitsene omgjøres til eller fra bytes i Radio Byte, avhengig om den mottar eller sender. All kommunikasjon foregår i halv dupleks, ved at radioen mottar hele tiden, bortsett fra når den sender. Radioen kan altså ikke både sende og motta samtidig. Radio Packet setter disse bytesene sammen til en pakke. Det er i Radio Packet at medium aksesskontrollen utføres. Active Message Handler danner en melding av alle pakkene, for så å avgjøre hvilken applikasjon data er beregnet for. Videre viser figuren at sensoren kan utføre flere forskjellige typer applikasjoner, som blant annet temperaturovervåking. Mer om dette i [43][57][58].

2.1.5 Krav til sensornodene

Det stilles strenge krav til sensornodene [65]. Sensornodenes hovedoppgave er å måle og detektere visse hendelser, og derfor er det særdeles viktig at sensornode gjør dette med en tilfredsstillende nøyaktighet og hyppighet, samtidig som den skal kunne kommunisere med andre sensornoder i sensornettverket. Sensornoden må være utstyrt med en viss form for intelligens, slik at den kan ta egne beslutninger basert på målinger som den har oppnådd, men sensornoden må også kunne ta beslutninger basert på data den mottar fra andre sensornoder. Sensornoden må dessuten kunne operere tilfredsstillende ved lavest mulig strømforbruk, slik at sensornoden kan kunne ha en levetid på opptil flere år.

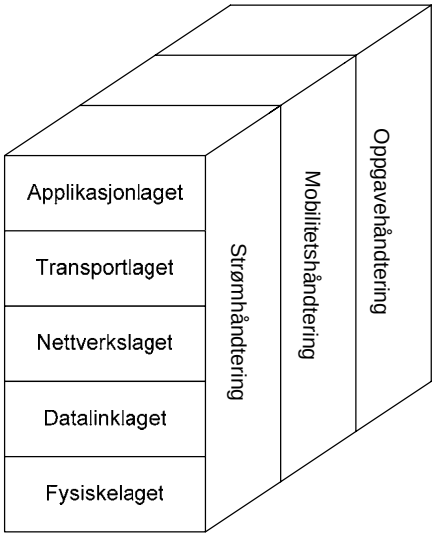
2.2 Sensornettverkets arkitektur

Sensornettverket er et Ad hoc nettverk hvor medlemskap (Sensornodene kan deles inn i grupper hvis nødvendig) og roller til sensornodene først bestemmes når nodene er utplassert [1][2]. Sensornodene vil selv foreta disse bestemmelsene siden de er selvorganiserende, samt kartlegge rutingstopologien. Reorganisering sørger for at rutingstopologien holder seg oppdatert. Det er antatt at sensornettverket (både sink og sensornoden) vil bruke en lagdelt protokollstakk som illustrert i 2.2.1.

2.2.1 Sensornettverkets protokollstakk

Protokollstakken består av tilsvarende lag som TCP/IP- protokollen, men tar i tillegg også parallelt for seg strøm-, mobilitets-, og oppgave- håndtering i alle lagene, ettersom disse funksjonene påvirker alle de 5 lagene [1][9]. De 3 parallell-lagene er nødvendig slik at sensornodene kan samarbeide på en strømeffektiv måte, og for å kunne rute data i et sensornettverk, samt dele ressursene mellom sensornodene. Det vil si at uten disse lagene, så

ville sensornodene kun jobbet individuelt. Det er viktig at sensornettverket kan samarbeide med hverandre, slik at levetiden til sensornettverket kan forlenges lengst mulig.



Figur 4. Sensornettverkets protokollstakk.

2.2.1.1 Fysiske lag

Det fysiske laget [1][67] tar for seg modulasjon, koding, valg av frekvens, samt sending og mottak av data. Sensornettverket er beregnet til å håndtere lave datarater siden det er batterilevetiden som er det viktigste. Designet av et fysisk lag bør fokusere på energieffektive modulasjonsmetoder som er enkle og robuste.

2.2.1.2 Datalinklaget

Datalinklaget [1] tar seg av kollisjoner og eventuelt ny overføring, effektkontroll, medium aksesskontroll, samt multipleksing av datastrømmer slik at kommunikasjonsressursene kan deles rettferdig og effektivt mellom sensornodene. Det innebærer medium aksesskontroll, samt feilkorrigering og feildeteksjon. Medium aksesskontroll (Medium aksesskontroll bruker ofte forkortelsen MAC, men siden denne rapporten tar for seg sikkerhet, så er det også en sikkerhetsfunksjon som bruker samme forkortelse. Derfor vil Medium aksesskontroll skrives rett frem uten bruk av forkortelse, og hvor MAC vil bli benyttet som forkortelse for en sikkerhetsfunksjon) må sørge for at sensornettverket har selvorganiserende egenskaper, samt at kommunikasjonsressursene deles rettferdig og effektivt mellom sensornodene. Et sensornettverk kan ikke benytte seg av samme medium aksesskontroll som benyttes i tradisjonelle trådløse nettverk siden disse mekanismene ikke er egnet for begrensningene som finnes i et sensornettverk[1][2]. Disse mekanismene må i hvert fall tilpasses et sensornettverk før de kan benyttes. Protokoller som er beregnet og designet for sensornettverk er følgende:

- SMACS (Self-organizing Medium Access Control for Sensor Networks) [1][6][9][35]
    - SMACS danner flat topologi (det motsatte gruppering) for sensornettverket. SMACS er en distribuert protokoll som tar seg av oppstart og organisering av sensornettverket etter utplassering. Dessuten lar SMACS sensornodene oppdage sine nabonoder, for så å opprette prosedyrer for kommunikasjon, uten bruk av lokal eller global master noder. SMACS fungerer best i statiske sensornettverk.
- EAR (Eavesdrop And Register) [1][9]

    - EAR algoritmen muliggjør sømløss sammenkobling av mobile sensornoder, samt kontinuerlige tjenester til sensornodene. EAR er transparent for SMACS, så SMACS er operativ helt til mobile sensornoder utplasseres.
- CSMA basert multippel aksess (Carrier Sensing Multiple Access) [1][9]
    - CSMA basert multippel aksess er en mekanisme som benytter seg av 2 komponenter: en for å lytte på linken, og en annen for å trekk seg unna hvis kollisjoner oppstår eller hvis andre benytter linken [1]. SMAC (Sensor Medium Access Control) er en slik type protokoll. Mer om SMAC i [1][6]

I noen tilfeller kan feilkontrollerende koding være veldig nyttig i et sensornettverk, og da spesielt hvis sensornettverket opplever høy mobilitet. Dette blant annet for å kunne kontrollere om integriteten til data den mottar er korrekt. For å kunne bevare levetiden, må en sensornode i perioder gå inn i en tilstand hvor aktivitetsnivået er redusert ettersom den begynner å gå tom for strøm [1]. Dette gjøres ved at sensornoden opererer med lave driftssykluser.

For at Medium aksesskontrollen skal være strømeffektiv, så må Medium aksesskontrollen adressere de ulike nivåene som forbruker strøm [72]. Dette innebærer kollisjoner (tapte pakker krever i noen tilfeller at pakken sendes på nytt), overhøring av andres kommunikasjon (mottak av pakker som er adressert til en annen node), overhead i form av kontrollpakker (overhead fører til reduksjon av nyttig data), samt lytting i forbindelse med ledigmodus (lytter etter mulig trafikk som aldri kommer).

2.2.1.3 Nettverkslaget

Hovedoppgavene til nettverkslaget [1][72] er å rute og videresende datapakker. Ruting foregår ved hjelp av multihopp ruting, slik at data rutes fra sensornode til sensornode gjennom sensornettverket helt til data kommer frem til riktig destinasjonen. Nettverkslaget tar også for seg organiseringen av en rute. Ettersom sensornodene har lite strøm og lite tilgjengelig minne, må mest mulig strømeffektive ruter lokaliseres. Sensornettverket kan utveksle rutingsmeldinger seg i mellom for å opprettholde en korrekt rutingstabell. Det er de mellomliggende sensornodene som foretar ruting avgjørelsene ut i fra rutingstabellen, og en sensornode må foreta et valg om hvilken sensornabo den skal rute data til. Ettersom nettverkstopologien er under konstant endring grunnet at sensornoder kommer og går, så må enhver sensornode selv foreta nettverksoperasjoner som videresending av pakker, ruting, og lignende nettverksoperasjoner. Protokollene på nettverkslaget må være selvorganiserende, samt kunne foreta periodisk reorganisering av nettverkstopologien. Det er dessuten veldig viktig at rutingen bruker minst mulig strøm. En annen viktig funksjon er å tilby internettkobling mot eksterne nettverk, som blant annet andre sensornettverk. I visse sammenhenger kan en sink fungere som en kommunikasjonsvei mot andre nettverk.

Flere teknikk kan benyttes for å sende data [71].

- Flooding
    - Hver sensornode kringkaster data til alle nabonodene sine, uansett om dem har mottatt data før eller ikke. Datapakkene kringkastes til alle nabonodene, og propagerer gjennom sensornettverket helt til dem når destinasjonen. Hvis det ikke eksisterer noen destinasjon, så droppes pakkene når TTL feltet (TTL feltet sier noe om hvor mange hopp en pakke kan foreta før den droppes) når null. Fordelen er at dette er en veldig enkel måte å rute pakker på, samt at ingen tilstand eller rutingsprotokoll trengs å vedlikeholdes. En annen fordel er at hvis

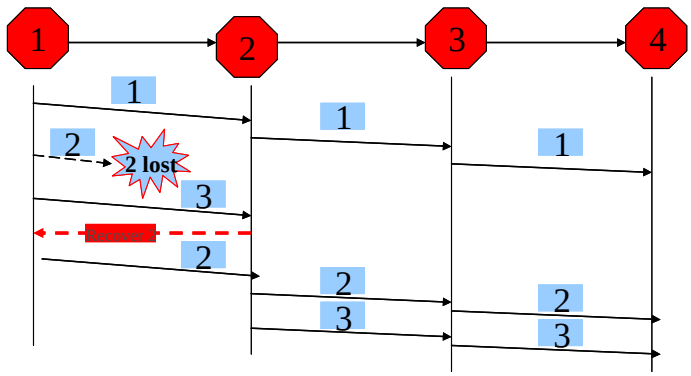
en destinasjon eksisterer, så blir den funnet. Ulempene er at flooding er ikke spesielt strømeffektiv siden det forekommer mye trafikk. En annen ulempe er at mange kopier av samme melding forekommer i sensornettverket, samt at den samme kopien kan sendes til den samme sensornoden flere ganger.

- Gossiping
  - Ved bruk av sladring, så sender en sensornode data til en tilfeldig valgt nabo, som igjen gjør det samme. Fordelen med sladring er at den løser problemet med at flere meldinger blir sendt til samme sensornode. Ulempen er at det kan ta lang tid å propagere meldinger gjennom sensornettverket og til destinasjonen, noe som fører til høy forsinkelse.
- SPIN (Sensor Protocol for Information via Negotiation)
  - SPIN foretar en slags 3-veis håndtrykk. SPIN sender data kun hvis sensornodene er interessert i å motta data. Før utveksling foretar SPIN en forhandling med sensornodene, slik at den finner ut hvilke sensornoder den kan sende data til. 3 typer meldinger benyttes: ADV, REQ og DATA. En sensornode bruker ADV for å annonsere at den har data å sende. REQ brukes for å be om å få spesifikke data, mens DATA er selve data som sendes. Dette gjøres for hver gang en sensornode ønsker å sende data.
- LEACH (Low-Energy Adaptive Clustering Hierarchy)
  - LEACH er en hierarkisk rutingsprotokoll som danner grupper av sensornoder for å minimere strømforbruker. Det er gruppelederne som kommuniserer direkte med sinken, mens sensornodene kommuniserer med gruppeleder. Gruppeledere kan velges tilfeldig, og gruppelederne kan også foreta dataaggregering og videresending av data til sinken. Sensornodene bestemmer selv om dem skal bli gruppeledere ved å kringkaste ADV meldinger, hvor andre sensornoder bestemmer hvilken gruppeleder dem tilhører ved å måle styrken til signalet. Fordelen er at LEACH ikke trenger noen global kjennskap til sensornettverket. Ulempen er ekstra overhead for å utføre dynamisk gruppering, og ekstra overhead oppstår også ved at gruppelederne endres ofte.
- SMECN (Small Minimum Energy Communication Network)
  - SMECN er en lokasjonsbasert protokoll som danner grafer av sensornettverket som inneholder informasjon om ruter som krever minimalt med strøm. Mest benyttet i Ad hoc nettverk, men kan også benyttes i sensornettverk.
- Directed Diffusion
  - Denne type datalevering skjer i 3 steg. Sinken annonserer interesse via en kontrollpakker som kringkastes i hele sensornettverket. Ut ifra denne kringkastingen, så settes det opp gradienter (ruter). Når en sensornode observerer en hendelse som er spesifisert i kontrollpakken, så sender den data tilbake til sinken via gradientene (ruten). Sensornoden sender data tilbake direkte via unikast. Fordelen er at ruter dannes når det er behov for dem, samt at hver enkelt sensornode foretar lagring av kontrollpakker, samt dataaggregering. Dessuten er denne måten å rute data på mer strømsparende, samt at den har lav forsinkelse. Ulempen er at sinken må kringkaste kontrollpakker om at den er interessert i å motta data, noe som gjør at dataleveringen ikke er kontinuerlig. Altså, sensornodene sender ikke data tilbake til sinken før sinken ber om det. En annen ulempe er den ekstra overheaden som er assosiert med all forespørslene som sinken kommer med.

2.2.1.4 Transportlaget

Transportlaget er nødvendig for at systemet skal kunne aksesseres gjennom Internett eller mot andre eksterne nettverk [1][2]. Et problem med sensornettverket er at det benytter attributt adresserbare sensornoder, hvor hver sensornode ikke trenger å ha unik identifikasjon. En mulig løsning på dette er å benytte TCP splitting (TCP akselerasjon kalles det også i andre sammenhenger), hvor UDP brukes inni sensornettverket mellom sink og sensornodene, mens TCP/UDP brukes mellom det eksterne nettverket og sinken. Det vil si at TCP forbindelsen kuttes ved sinken, for så å etablere en spesiell forbindelse som håndterer kommunikasjonen mellom sinken og sensornettverket. Resultatet blir at kommunikasjonen mellom brukeren og sinken foregår via internett eller satellitt med UDP eller TCP, mens kommunikasjonen mellom sinken og sensornodene kan være ren UDP. Grunnen til at UDP er mer passende enn TCP i et sensornettverk, er fordi UDP har mindre overhead, og dessuten har UDP mye lavere forsinkelse. Man kan derfor anta at all kommunikasjon i sensornettverket er upålitelig, ettersom mesteparten av kommunikasjonen foretaes med UDP.

For å sørge for en sikker og pålitelig transmisjon, så kan nye overføringer (ENG: Retransmission) eller feilkorrigerende koding innføres. Dette kan skje ved hjelp av multihopp ruting og mellomlagring [60]. Hvis for eksempel en rute består av fire noder, hvor sensornode1 må kommunisere gjennom 2 og 3 for å nå 4. Først sender sensornode1 pakke1, som kommer frem til sensornode4. Men pakke2 blir borte før den når sensornode2. Pakke3 sendes så, men når sensornode2 sjekker sekvensnummeret og ser at den mangler pakke2, så holder sensornode2 igjen pakke3, og sender en forespørsel til sensornode1 om at pakke2 må sendes på nytt. Når sensornode2 mottar pakke2, videresendes denne til sensornode3. Deretter sendes pakke3 til sensornode3. Hvis dette skulle skje hos noen av de andre sensornodene, så gjentaes prosessen. Figur 5 illustrerer prosessen.



Figur 5. Eksempel på retransmission ved hjelp av mellomlagring.

2.2.1.5 Applikasjonslaget

Flere applikasjoner kan lages og brukes på applikasjonslaget. Softwaren vil variere avhengig av oppgavene som sensornoden skal utføre. Eksempler på protokoller kan blant annet være [1]:

- Sensor Management Protocol (SMP)
  - System administrator kan benytte for eksempel SMP for å aksessere sensornettverket. Ettersom sensornettverket ikke benytter global identifikasjon, må SMP aksessere sensornodene ved å benytte attributtbaserte navn og lokasjonsbasert adressering. Andre oppgaver som SMP utfører er blant annet innføring av regler som er relatert til navnet, dataaggregering, og gruppering, samt utveksling av data som er relatert til lokasjonsalgoritmene,

tidssynkronisering, forflytting av sensornoder, kontrollering av at sensornodene slåes av og på, sende forespørsler om sensornettverkets konfigurasjon og statusen til nodene, samt rekonfigurering av sensornettverket og autentisering, nøkkeldistribusjon, samt sikkerhet under kommunikasjon[1]. SMP tillater altså software til å utføre administrative oppgaver som blant annet flytting av sensornoder og tidssynkronisering av sensornodene.

- Task Assignment and Data Advertisement protocol (TADAP)
  - TADAP benyttes blant annet til å fordele oppgaver til sensornodene i sensornettverket og for å annonsere om at data kan sendes. Dette kan skje ved at brukeren sender en kontrollpakke til sensornodene om at sensornodene kun skal måle visse hendelser. En annen mulighet kan være at sensornodene annonserer til sinken om at de har data som de ønsker å levere.
- Sensor Query and Data Dissemination Protocol (SQDDP)
  - SQDDP gjør det mulig å sende ut spørringer, eller svare på spørringer, samt samle inn innkommende svar.

2.2.1.6 Parallelt

De 3 parallell-lagene er ment å ta seg av kontrollering av strøm, mobiliteten og oppgavefordelingen i alle lagene i protokollstakken [1]. Parallell-lagene er nødvendige i alle lagene slik at sensornodene kan samarbeide på en mer strømeffektiv måte, samt rute data i sensornettverket, og dele ressurser mellom sensornodene. Uten disse parallell-lagene ville hver enkelt sensornode operert individuelt. Litt om hvert av parallell-lagene:

- Strømhåndteringen
  - Tar seg av strømforbruket til en sensornode, altså hvordan en sensornode forbruker strømkapasiteten. For eksempel, så kan en sensornode slå av mottakeren etter at den har mottatt en melding fra en nabo, for å unngå å motta flere kopier av samme melding. Sensornoden kan også kringkaste en beskjed til sine naboer om at den ikke lenger kan delta i utvekslingen på grunn av lav strømkapasitet, og kan på denne måten bevare resten av strømmen for sensoren.
- Mobilitetshåndteringen
  - Mobilitetshåndtering tar seg av mobiliteten til en sensornode ved å detektere sensornodens bevegelser og mobilitet. Det vil si at den detekterer og registrer bevegelsene. Dette for at det alltid skal vedlikeholdes en rute tilbake til sluttbrukeren, samt for at sensornoden kan holde oversikt over hvem som er naboene dens. Ved at en sensornode vet hvem som er naboene dens, så kan den kontrollere og balansere strømforbruket.
- Oppgavehåndteringen
  - Oppgavehåndteringen tar for seg balansering og fordeling av oppgaver mellom sensornodene. Det kan være oppgaver som målinger og observasjoner som foretaes i et område. Altså kontrollering av et område slik at ikke alle sensornodene i dette området foretar samme gjøremål, samt at gjøremålene fordeles mellom sensornodene avhengig av batterinivået til hver enkelt sensornode.

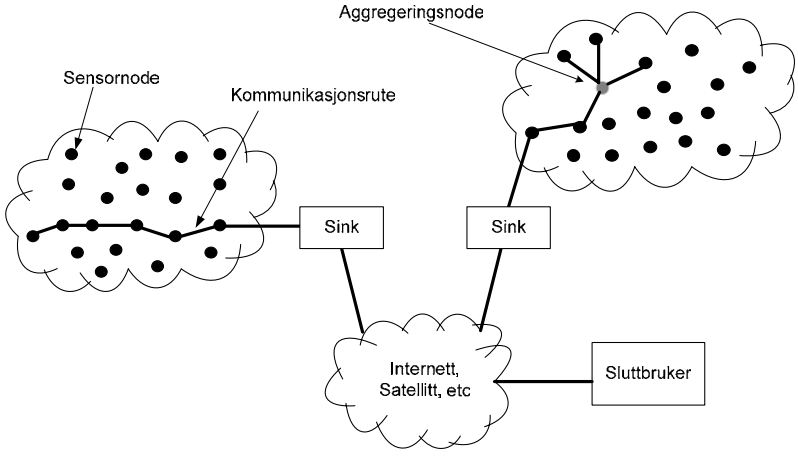
2.2.2 Konfigurering og utplassering av sensornodene

Konfigurering og vedlikehold av sensornettverkets kan foregå på 3 måter [1]: 1)før utplassering, og ved selve utplasseringen, 2) etter utplassering, og 3) ved utplassering av

ekstra sensornoder. Før utplassering vil det kun kreves minimalt med konfigurering, og sensornodene kan utplasseres i form av masseutplassering, eller manuelt en og en. Etter utplassering vil sensornodene være plassert relativt tett sammen, og nodene vil være omprogrammerbare, selvorganiserende, og støtte et strømreduserende trådløst sensornettverk. Posisjonen til sensornodene trenges ikke å bestemmes på forhånd, noe som gjør at sensornodene kan utplasseres tilfeldig i forskjellige miljøer. I tillegg innebærer dette også at protokollene og algoritmene i sensornettverket har selvorganiserende egenskaper. Det er ofte slik at når sensornodene først er utplassert, så har sensornodene i seg selv ingen mobilitet. Men dette er applikasjonsavhengig, for i praksis vil enkelte sensornoder være statiske, mens andre vil være mobile dersom objektet som sensornoden er festet på, beveger seg. Nettverktopologien vil endre seg ettersom nye noder blir lagt til, eller at eksisterende noder dør eller fjernes. Det kan også forekomme at ekstra sensornoder tilføres sensornettverket for å skalere sensornettverket, eller for å erstatte de eksisterende nodene som enten dør eller fjernes. Det er flere grunner til at sensorer må erstattes, f.eks. hvis batteriet er oppbrukt, eller at en sensornode blir ødelagt. Dette fører til hyppig topologiendring ettersom pakker må rutes om igjen, samt at nettverket må reorganiseres slik at topologien forblir oppdatert til enhver tid.

2.2.3 Multihopp og dataaggregering

Mye av trafikken som propagerer gjennom sensornettverket skjer over flere hopp, med muligens noe prosessering i form av dataaggregering underveis hos andre sensornoder [2]. Dataaggregering forekommer når data kommer fra flere sensornoder som benytter en felles rute eller en felles aggregeringsnode. Dessuten forekommer dataaggregering kun hvis sensornodene har samme attributter som fenomenet som blir overvåket eller analysert. Det vil si at sensornodene tilhører samme gruppe og utfører samme applikasjon. Det er ofte gruppelederen som opererer som aggregeringsnode. Figur 6 illustrerer en mulig topologi av et sensornettverk. Figuren inneholder 2 sensornettverk, som hver er koblet opp mot en sink. Innad i sensornettverkene rutes data via multihopp, og i noen tilfeller benyttes også aggregering for å sammenslå data til mer komplett data, for så å videresende data mot sinken. Videre, så benytter sinkene internett, eller satellitt for å nå sluttbruker.



Figur 6. Sensornettverks topologi med multihopp og dataaggregering.

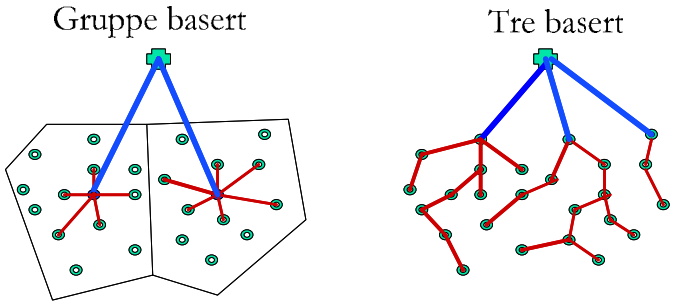
Flere sensorer innenfor samme gruppe kan utføre samme oppgaver for å oppnå bedre resultat. Sensornodene utfører selv prosessering av rå data, for så å videresende data til aggregeringsnoden, som slår sammen informasjonen, og videresender det til sink. Sensornodene må forholde seg til rutingsinformasjonen fra aggregeringspunktet.

Dataaggregering er nyttig for å eliminere duplikatmeldinger, samt for å redusere kommunikasjonsbelastningen. Slike aggregeringsnoder kan for eksempel være vanlige sensornoder som kan velges dynamisk, eller statisk, men dette kan avgjøres ved implementering eller ved utplassering. Et sensornettverk har dessuten behov for protokoller som kan ta seg av gruppehåndtering, som blant annet innmelding av nye sensornoder [1][2][31].

2.2.4 Gruppering av sensornoder

Det kan være nødvendig å gruppere sensornodene i grupper [2][54] for å danne en hierarkisk organisering av sensornettverket. I hver enkelt av disse gruppene er det en eller flere sensornoder, hvor en av disse nodene opererer som gruppeleder. Gruppelederen har som oppgave å håndtere operasjonene til de andre sensornodene i gruppen. Dette er operasjoner som blant annet dataaggregering, sikker kommunikasjon, og lignende. Gruppelederen fungerer også som et kommunikasjonsledd mot resten av sensornettverket, og da mot de andre gruppelederne, eller eventuelt direkte mot sinken.

Dannelse av grupper kan skje på 2 måter: Gruppebasert eller trebasert. En gruppebasert topologi opererer mer som en stjerne enn som en hierarkisk struktur, og er avhengig av at avstanden fra sensornodene er relativt kort. Dette innebærer at det er mindre behov for multihopp, og mer aktuelt med enkelt hopp (kort avstand) frem til gruppeleder. Gruppeleder sammenslår data, for så å videresende data direkte til sinken. Gruppelederne kan enten være en kraftigere enhet, eller en vanlig sensornode. Men en vanlig sensornode vil gå fortere tom for strøm, og må da byttes ut med en annen sensornode. Dette vil føre til at roller må endres, ettersom en sensornode kan bli valgt til gruppeleder når en annen sensornode går tom for strøm. Det er kun gruppeleder som foretar dataaggregering. I en trebasert struktur, så foregår kommunikasjonen ved hjelp av multihopp frem til gruppelederne, hvor gruppelederne kommuniserer direkte med sinken. Dessuten kan dataaggregering foretaes flere steder i sensornettverk underveis. Figuren [54] nedenfor viser en illustrasjon av gruppebasert og trebasert gruppering.



Figur 7. Gruppebasert versus trebasert gruppering av sensornoder.

2.2.5 Strømforbruk i et sensornettverk

Det stilles strenge krav til sensornodene når det gjelder strømforbruket. I tillegg til at sensornodene må være små, så må de også forbruke minimalt med strøm, ha lav produksjonskostnad, være tapbare og selvstendige, samt kunne operere uten vedlikehold og kunne tilpasse seg miljøet de opererer i. Siden en trådløs sensornode er såpass liten, så er det begrenset hvor stor strømenhet en sensornode kan utstyres med. Avhengig av applikasjon, så er det ikke alltid sensornoden har mulighet til å fornye strømressursene [1], og derfor er sensornodens levetid avhengig av batteriets levetid. Derfor er strømbesparelse og

strømhåndtering veldig viktig. Hvor stort strømforbruket som forekommer har en stor betydning for sensornodens levetid. Strømforbruket forekommer hovedsakelig innenfor 3 områder [35]:

2.2.5.1 Transceiveren

Det er i selve kommunikasjonen at strømforbruket er høyest. Dette gjelder både ved sending og mottak av data. Derfor må det foretaes tiltak for å redusere dette. Det er viktig å slå av transceiveren når transceiveren verken sender eller mottar data.

2.2.5.2 Prosessoren

Strømforbruket under dataprosessering er ikke så merkbart. Men flere ting spiller inn her, som for eksempel hvor mye strøm en prosessor forbruker for å prosessere data, hvilken klokkefrekvens prosessoren kjører på (desto høyere, desto mer krevende), samt hvor mye prosessorkraft som trengs for å prosessere data og sikkerhetsfunksjoner. Lokal prosessering er selve nøkkelen for å kunne redusere strømforbruket i et sensornettverk, fordi det er mer strømbesparende å lagre prosessert informasjon, istedenfor å sende informasjon med en gang.

2.2.5.3 Sensoren

Radiusen til sensoren varier avhengig av hva slags applikasjon man benytter. Blant annet vil periodiske målinger som sensoren foretar forbruke mindre strøm enn konstant måling. Hvordan sensoren foretar målingene har innvirkning på hvor mye strøm som forbrukes. Slike faktorer må taes til etterretning. Et tiltak kan være å sette sensornoden i sovemodus, for så å aktivere sensornoden når sensoren detekterer at en hendelse inntreffer.

2.2.6 Designkriterier for sensornettverket

Det er viktig at sensornettverket designes ut i fra disse kriteriene [1][2]:

- Et sensornettverk bør ha god feiltoleranse mot strømfeil, fysiske skader, samt interferens fra omgivelsene. I tillegg bør sensornettverket være pålitelig. Feil i en sensornode skal ikke utsette hele sensornettverket.
- Skalerbarheten er viktig ettersom sensornettverket kan bestå av flere hundre/tusen sensornoder. Ettersom antall sensornoder blir mange, blir det desto viktigere å bruke båndbredden enda mer effektivt og rettferdig. Her er det nyttig å benytte teknikker som dataaggregering. Rettferdighet er viktig med tanke på å samle inn data fra alle sensorer og unngå utsulting av visse sensornoder [7].
- Produktkostnader må være lave ettersom sensornodene må være billigere enn tradisjonelle sensorer for at det skal være lønnsomt med et sensornettverk. Hvis et sensornettverk skal kunne bestå av tusenvis av sensornoder, så må prisen være relativt lav for at dette skal være lønnsomt.
- Et sensornettverk har store hardwarebegrensninger grunnet sensornodens små størrelse, lite minne og prosesseringskraft, og ekstremt lave strømkapasitet, noe som må taes til etterretning ved design av et sensornettverk.
- Sensornettverkets topologi er under konstant endring ettersom nye sensornoder legges til, og andre dør.
- Sensornodene skal operere uten oppsyn over avstand i forskjellige miljøer, og må konstrueres ut i fra kravene som miljøet stiller til sensornoden.

Strømforbruk er det viktigste designkriteriet siden sensornodene har begrenset strømkapasitet (Mica2 mote benytter 2 AA batterier som hver har en kapasitet på 2500 mAh). Dessuten er det viktig å designe systemet med hensyn på batteriets levetid, slik at strømbevaring og strømhåndtering foretaes på en effektiv måte.

2.3 Forskjeller mellom Tradisjonelle sensornettverk og trådløse sensornettverk

Tradisjonelle sensornettverk [1] har tidligere bestått av et mye mindre antall sensornoder, hvor utplasseringen av sensornoder har vært nøye planlagt. Det har heller ikke vært vanlig med prosessering innad i sensornodene. Sensornodene bare detekterte hendelser, for så å rapportere dette direkte til en basestasjon. Denne rapporteringen har foregått via kabling, ikke trådløst. Ettersom sensorene har vært kablet, så har de også hatt tilgang til mye større strømforsyning. Dataaggregering har heller ikke skjedd innad i sensornettverk, men foretatt utenfor nettverket. I fremtiden vil vi se et mer fleksibel trådløst sensornettverk, som vil bestå av et mangfold av sensornoder. Disse sensornodene vil kunne prosessere data internt, før kommunikasjonen vil skje trådløst via ad hoc multihopp. Et trådløst sensornettverk skal dessuten kunne fungere uten tilsyn, ved at den har protokoller som er selvorganiserende og selvkonfigurerende. Radioen vil ha både en sender og en mottaker, men strømforsyningen vil være veldig begrenset. I tillegg vil sensornettverket kunne foreta dataaggregering innad i sensornettverket.

2.4 Forskjeller mellom Ad hoc og Sensornettverk

Et sensornettverk er som sagt et ad hoc type nettverk, men allikevel er det en del karakteristikker som skiller dem [2][11][12][32]:
Sensornettverket kan bestå av et mangfold av selvorganiserende trådløse sensornoder (mange tusen sensorer). Sensornettverket må benytte en eller flere sinker som funger som et kommunikasjonsledd/grensesnitt mellom sensornettverket og omverdenen. Sensornettverket foretar andre typer oppgave som blant annet målinger og lokalisering. All data som sensornodene samler inn videresendes til sinken, ettersom det er sinken som kontrollerer alle sensornodene i sensornettverket. Sensornodene bør være tett utplassert på grunn av at rekkevidden er relativ korte for sender og mottaker. Sensornodene er ofte utsatt for feil siden de må operere i forskjellige miljøer, så derfor må de være billig for at det skal være lønnsomt. De fleste sensornodene er statiske, noe som gjør at topologien ikke endrer seg så ofte. Topologien endrer seg kun når en node dør. Ad hoc nettverk vil ha høyere topologiendring siden enhetene er mobile. Sensornodene samarbeider, og utveksler data for å oppnå mer komplett informasjon, og denne informasjonen prosesseres først lokalt i sensornoden, for så å utveksles med andre sensornoder. Sensornodenes batterikapasitet, prosessering, og minne er mye mer begrenset på grunn av at nodene er såpass små. Dessuten trenger ikke sensornodene å ha global Identifikasjon som for eksempel IP adresser slik som ad hoc enheter, ettersom de vanligvis opererer lokalt i et eget område eller innad i et eget nettverk. En annen årsak til at sensornettverket ikke benytter global Identifikasjon er på grunn av den store overheaden siden det er såpass mange sensornoder. Kommunikasjonen i sensornettverket foregår hovedsaklig i form av kringkasting, mens et rent ad hoc benytter punkt- til punkt- kommunikasjon. Ved gruppering i et sensornettverk, så kan kommunikasjonen også foregå via multikasting. Dessuten er sikkerhetstrusler i et sensornettverk forskjellig fra trusler i et mobilt ad hoc nettverk [11]. Sensornoder kan også aggregere data ved at grupper av sensornoder sender data til et aggregeringspunkt som er tilknyttet denne gruppen i sensornettverket. Dette aggregeringspunktet kan være en sensornode. Kommunikasjonen i sensornettverket kan foregå på 3 måter: 1) Mange til en, det vil si at mange sensornoder sender data til sink eller et

aggregeringspunkt i sensornettverket. 2) En til mange: En sink eller en aggregeringsnode kringkaster til en liten gruppe av sensornoder, eller kringkaster forespørsler eller kontrollinformasjon til flere sensornoder. 3) Lokal kommunikasjon: Nabonoder sender meldinger seg imellom for å oppdage eller koordinere med hverandre. En sensornode kan kringkaste meldinger som er beregnet for alle naboer, eller sende en melding direkte til en enkelt nabo.

Disse forskjellene oppstår hovedsakelig på grunn av at sensornodene har mye større ressursbegrensninger enn tradisjonelle ad hoc systemer. Hvis sensornettverket ikke hadde hatt så store ressursbegrensninger, så ville karakteristikkene til sensornettverk og ad hoc nettverk vært mer lik.

2.5 Bruksområder

Et ad hoc nettverk er tiltenkt i alle miljøer hvor det ikke er tilgjengelig infrastruktur, eller hvor infrastrukturen er utilgjengelig på grunn av katastrofe. For et sensornettverk er målet å detektere og rapportere inn hendelser som oppstår innenfor et vist område (sensornettverkets område), eller foreta lokalisering av objekter innenfor et vist område. Sensoren til hver enkelt sensornode er veldig begrenset, men gjennom samarbeid med de andre nodene muliggjør dette en mer pålitelig sensorfunksjon. Sensornettverket kan brukes i flere områder [1][3][52]. Mange av disse miljøene kan være en stor påkjenning for sensornodene, derfor må de være ganske robuste. Et Sensornettverk kan brukes i flere typer områder og miljøer. I del kapitel 2.5.1 finnes en mer konkret beskrives av en av disse bruksområdene, mens i del kapitel 2.5.2 finnes en oversikt over noen andre aktuelle bruksområder.

2.5.1 Sensornettverk i medisinsk sammenheng

Noen av applikasjonene i medisinsk sammenheng vil være pasientovervåking, medisinadministrasjon i sykehus, overvåking av bevegelser, samt fjernovervåking av fysisk menneskelige data, og lokalisering og sporing av leger og pasienter inni og utenfor sykehuset [1][54]. Ved hjelp av fjernovervåking kan sensornettverket samle inn data om individer som har en eller annen form for medisinsk tilstand, samt lagre data over en lang periode. Dette gjøres ved at sensorer festes på kroppen til pasienten for å samle inne informasjon om pasientens helsetilstand, for så å videresende data til legen som eventuelt er utstyrt med en bærbar pc eller en håndholdt enhet som for eksempel en PDA. Sensornettverket kan også overvåke og detektere eldres oppførsel og væremåte i forskjellige situasjoner. Sensornodene vil gi pasienter større frihet, samt tillate leger å identifisere forhåndsdefinerte symptomer på et tidligere stadige. Det å utstyre leger og pasienter med små sensornoder gjør det mulig å foreta målinger (puls, hjerterate, og glukosenivå [65]) av helsen til pasienter, samt lokalisere leger og pasienter inni og utenfor sykehuset. Hver av disse sensornodene på for eksempel en pasient vil da ha sin egen spesifikke oppgave. Dette er oppgaver som blant annet detektering av hjerterytmen, måling av blodtrykk og oksygenmåling. Det å overvåke leger gjør at legene kan lokalisere hverandre inni sykehuset. Administrasjon av medikamenter kan foregå ved at sensornoder festes til medikamentene (dette krever at sensorene er veldig små) for å minimalisere sjansen for at pasienter får feil dosering, eller feilaktig får utskrevet ukorrekte medikamenter. Dette muliggjøres fordi pasientene vil være utstyrt med sensornoder som inneholder informasjon om pasientens identitet, og helsetilstand. Derfor kan det detekteres om pasienten for eksempel har astma og allergi, og lignende, samt hva slags medisiner og doseringer pasienten skal ha [1]. Hvis det for eksempel oppstår en krise situasjon som involverer mange skadde personer, kan helsepersonell ved hjelp av trådløse monitorer

overvåke tilstanden til pasientene, og ut i fra data som hentes ut, få en oversikt over hvilke pasienter som har mest, og raskest behov for hjelp, samt hva slags tilstand pasienten er i. Det er viktig med tanke på sikkerheten at helsepersonell autentiserer seg overfor nettverket før de får tilgang til å hente ut informasjon [14]. Helsepersonell må også kunne gi andre rettigheter til å hente ut informasjon når dette måtte bli nødvendig, for eksempel hvis et nytt helsepersonell ankommer et ulykkesstedet.

CodeBlue [14][61] er et eksempel på en infrastruktur for sensornettverk som er beregnet for bruk i medisinsk sammenheng. CodeBlue har som formål å integrere sensorer, PDA`er og PC systemer, hvor kommunikasjonen skjer trådløst via radio. Dette gjør jobben lettere for helsepersonell ved at de kan lese ut informasjon som er lagret i sensorene ved hjelp av en PDA eller en annen bærbar trådløs enhet. På denne måten kan helsepersonellet raskt få viktig informasjon om pasientens helsetilstand (informasjon om blant annet helsehistorie, hva slags medikamenter pasienten bruker, hva pasienten ikke tåler, hvordan pasienten skal behandles og lignende). Men for at dette skal være realiserbart må kommunikasjonen være pålitelig, slik at feil informasjon ikke forekommer. Denne løsningen setter store krav til robusthet og sikkerhet som kreves i medisinsk sammenheng [14]. Firmaer som leverer sensorer for bruk i medisinsk sammenheng er Nonin [14][15], og Numed [14][16]. Disse bruker Bluetooth, mens Radianse [14][17] leverer sensorer basert på radio. Radianse er beregnet for lokalisering og sporing inni sykehus. Disse leverer sensornoder som både er kablet og trådløse. Andre prosjekter er MobiHealth prosjektet, som tilbyr kontinuerlig overvåking av pasienter utenfor sykehusmiljøet. Fordel med dette er å redde liv, oppnå verdifulle data som kan benyttes til videre medisinskforskning, samt kutte kostnader i forbindelse med medisinske tjenester. Mer om dette i [14][37]

2.5.2 Andre bruksområder

Et sensornettverk kan brukes i mange andre scenarioer også [1][10][65]. Her følger en liste over en del bruksområder:

- Overvåking av miljøet (luftkvalitet, forurensning, skogbrann, krisesituasjoner)
- Innad i bygninger (detektere brann, fuktighet, temperatur, bygningsstruktur (vibrasjoner), luftkvaliteten)
- Overvåking (Sikkerhet, trafikk, naturen)
- Militært (Detektere fiender, overvåking av troppens helse og posisjonering)
- Lager (overvåke pakker)
- Medisin/helse (Overvåke pasienter, samt lokalisering av pasienter og leger.)
- Industri (prosesskontroll)
- Katastrofeområder (Sensornettverket kan utføre observasjoner og målinger i vulkaner)

2.6 Del konklusjon

Vi har nå sett på flere aspekter i et sensornettverk, samt hva som skiller et sensornettverk fra andre typer nettverk. Et sensornettverk er en teknologi som har som formål å detektere og observere hendelser som kan oppstå i flere forskjellige miljøer og sammenhenger. Hvis teknologien klarer å overkomme de store ressursbegrensinger i form av prosesseringskraft, lite lagring, og lav strømkapasitet, så vil dette bli en nyttig teknologi i fremtiden. Disse begrensingene gjelder spesifikt for sensornettverk, så derfor må nye teknikker utvikles ettersom tradisjonelle ad hoc teknikker ikke kan benyttes. Men for at vi trykt skal kunne ta i bruk denne teknologien, må vi også vurdere sikkerhetsutfordringene med en slik teknologi.

3 Sikkerhet i sensornettverk (trusler og løsninger)

Det har lenge vært kjent at sikkerheten i trådløse nettverk ikke har vært god nok [11]. Det er spesielt det at kommunikasjonen foregår via kringkasting som gjør trådløse nettverk sårbare. Det finnes mange forskjellige algoritmer som kommer med forskjellige protokoller. De forskjellige applikasjonene har forskjellige sikkerhetsinteresser og krav til sikkerhet. Selv om sikkerhet har vært et aktivt forskningsområde i trådløse nettverk, fører den unike karakteristikken til sensornettverk til store utfordringer innen sikkerhet. Det er faktorer som topologiendring, og at sensornettverket ikke benytter en sentralisert administrasjon som gjør sikkerheten vanskelig. Dessuten er ikke sikkerhetsmekanismene for tradisjonelle nettverk aktuelle for sensornettverk ettersom sensornodene er utstyrt med ekstremt begrensede ressurser. Hovedmålet med sikkerhet i sensornettverk er å kunne tilby sikkerhetstjenester som autentisering, konfidensialitet og integritet til sensornodene. Sikkerhet er som sagt en stor utfordring i et sensornettverk. Det at hundrevis eller tusenvis av sensornoder danner et selvorganiserende trådløst nettverk hvor sensornodene har så begrenset prosesseringskraft, lagring, båndbredde og strøm, samt at det heller ikke er noen garanti for den fysiske sikkerheten til nodene, gjør det vanskelig å tilby sikkerhet i slike nettverk. Det er viktig at de rette sikkerhetsteknikkene brukes for å hindre motstandere mot å utføre ulovligheter mot sensornettverket. Derfor er det flere faktorer man må vurdere. Altså hvor god sikkerhet som kan oppnåes i sensornettverk, og hvor god sikkerhet man trenger. Det er også viktig å integrere sikkerhet i hele systemet slik at ikke et punkt kan bli et angrepspunkt.

3.1 Sikkerhetsutfordringer i sensornettverk

I et sensornettverk er det mange sikkerhetsutfordringer. Faktorer som må taes i betraktning er at mellom sensornodene foregår det en åpen kommunikasjon, samt at kommunikasjonen foregår over et delt medium, og at det er høy dynamisk nettverkstopologi. Dessuten er den viktigste faktoren at sensornettverkets store ressursbegrensninger må tas i betraktning. I tradisjonelle nettverk, så benytter nodene dedikerte rutere, mens i sensornettverk må sensornodene selv operere som rutere, og videresende pakker til andre sensornoder. Siden trådløse sensornettverk benytter et delt medium, så er mediet tilgjengelig for alle. Derfor er det en nødvendighet å innføre sikkerhet i trådløse sensornettverk. Men god sikkerhet har en pris. Desto større og kraftigere sikkerhet som innføres i nettverket, desto høyere prosesserings-, og kommunikasjons- kostnader vil ramme systemet. Dessuten er mange av de tradisjonelle sikkerhetsalgoritmer uaktuelle for sensornettverk, siden de er designet for enheter som stasjonære og bærbare enheter med gode ressurser [12].

3.2 Trusler mot sensornettverket

Sensornettverket vil stå overfor mange trusler. Vi kan skille mellom passive og aktive angrep [4], hvor passiv betyr at motstanderen kun lytter etter informasjon, mens aktiv betyr at motstanderen kan benytte seg av informasjonen som den har fått via passive angrep til å utføre aktive angrep mot sensornettverket. Passive angrep er vanskelig å detektere fordi de sjelden involverer noen form for modifikasjon av data. Meldingene sendes og mottaes av kommunikasjonspartene på vanlig måte, uten at de er klar over at en tredje part lytter til kommunikasjonen. Kryptering kan løse dette problemet. Aktive angrep involverer modifikasjon av data eller generering av falske data. Sensornettverket står overfor trusler både mot sensornodene fysisk, og mot kommunikasjonen mellom sensornodene der en motstander både kan utføre passive og aktive angrep.

3.2.1 Trusler mot sensornoden

Når først sensornoden er utplassert kan man anta at sensornoden ikke er fysisk sikret [55]. Dermed kan en sensornode plukkes opp, og informasjonen som er lagret i den kan hentes ut (informasjon som blant annet krypteringsnøkkel), eller så kan motstanderen omprogrammere sensornoden til sitt eget formål, for så å utplassere noden i sensornettverket igjen. Derfor må det være en mekanisme som kan forhindre at dette skjer, for eksempel at en sensornode slår seg av når en motstander begynner å tukle med sensornoden. Ved å omprogrammere sensornoden, kan motstanderen utplassere en sensornode som sprer falske data (for eksempel falsk ruting informasjon), hindrer passering av riktig data, uthenter data fra nettverket, eller som utfører angrep som går ut på at en sensornode ikke kan utføre sine faste tjenester (DoS). Derfor er det viktig å sikre sensornoden fysisk.

3.2.2 Trusler mot kommunikasjonen

All form for kommunikasjon må holdes hemmelig slik at lyttere ikke kan motta, studere eller analysere data som transmitteres. En motstander kan få aksess til privat informasjon ved å overvåke kommunikasjonen mellom sensornoder. Kryptering vil delvis løse dette problemet, slik at motstanderen ikke kan lese data. Her krypteres data før transmisjon, for så å dekrypteres ved mottak. En motstander kan utføre angrep mot alle lagene i protokollstakken [4][32]. Videre følger en oversikt over aktuelle angrep mot kommunikasjonen i sensornettverket.

3.2.2.1 Passive angrep

Passive angrep går ut på at en motstander prøver å skaffe seg informasjon ved å lytte til kommunikasjonen, og på denne måten tilegne seg informasjon som kan brukes til å kompromittere sikkerheten i sensornettverket. Passive angrep involverer vanligvis lytting og analysering av datatrafikken [2][11][4][24]:

- Lytting (Eng: Eavesdropping)
    - Motstanderen lytter til kommunikasjonen mellom to sensornoder, og kan fange opp hemmelig informasjon (for eksempel ved bruk av en kraftig mottakerantenne), og kan på denne måten samle informasjon ved å fange opp pakkens kilde, destinasjon, størrelse, nummer, og tidspunktet pakken ble sendt. Problemet kan delvis løses ved å innføre kryptering i kommunikasjonen, men for at krypteringsnøkkelen skal kunne utleveres eller fornyes, så kreves det nøkkelutvekslings- og distribusjons- teknikker.
    - Løses ved bruk av kryptering.
- Trafikkanalysering (Eng: Traffic analysis)
    - Motstanderen får ikke vite all info, men noe data kan motstanderen få tak i. For eksempel ved å observere visse mønstre i meldingene, for så å kunne avgjøre hvor de kommuniserende partene befinner seg, samt identiteten til partene, samt lengden til meldingene.
    - Løses ved bruk av kryptering.

3.2.2.2 Aktive angrep

Ved aktive angrep, så utfører motstanderen angrep direkte mot sensornettverket. Motstanderen bruker ofte informasjon som motstanderen har tilegnet seg ved et passivt angrep, til å utføre et aktivt angrep. Det er flere former for aktive angrep [4][11][24][32][38]:

- Tilføring og endring av meldinger (Eng: Message alteration and injection)
    - Hvis motstanderen lytter til datatrafikk mellom flere sensornoder, så kan motstanderen tilegne seg nok informasjon til å kunne tilføre egne meldinger inn i sensornettverket, eller kunne snappe opp meldinger som sendes mellom sensornodene. På denne måten kan motstanderen sende falske data til både sluttbrukeren og sensornodene. Dette kan føre til høyere strømforbruk i sensornodene slik at levetiden reduseres betraktelig. Et annet angrep kan være tilføring av falsk rutingsinformasjon i sensornettverket, noe som resulterer i inkonsistente rutingstabeller.
    - Autentisering kan forhindre tilføring av falske pakker i sensornettverket. Pakkene blir da ikke akseptert. Ved bruk av integritetssjekk kan man detektere om en pakke har blitt endret under transmisjon.
- Angrep basert på svar av meldinger (Eng: Replay attack)
    - Motstanderen kan lytte til kommunikasjonen siden kommunikasjonen foregår via kringkasting, for så å svare på meldinger som er beregnet for en annen part, ved at motstanderen utgir seg for å være en av de kommuniserende partene. Det er vanlig å benytte en teller som inkrementeres for hver melding som sendes, for så å avvise meldinger med gamle tellerverdier. Dette kan være vanskelig å oppnå ettersom sensornettverket har begrenset med minne til å lagre tellerverdier i en tellertabell, men ved bruk av grupperinger blir det lettere. Dessuten trenger hver sensornode kun å lagre tellerverdien til kommuniserende parter, det vil si, at sensornodene kun trenger å lagre tellerverdier for sine nærmeste naboer.
    - Angrep kan forhindres ved bruk av en teller, tidsstempel eller et tilfeldig tall.
- DoS angrep (Eng: DoS attack)
    - En motstander som utfører DoS angrep [8][55] har som hensikt å sette hele funksjonaliteten til sensornettverket ut av spill ved å for eksempel kringkaste et såpass høyt signal at det jammer sensornettverket. I et sensornettverk som benytter fast radiofrekvens, så vil angrep av typen DoS være relativt enkelt å utføre, ettersom det bare er å jamme radiofrekvensen. Dette vil sette sensornettverket totalt ute av spill.
    - Angrepet er beregnet mot det fysiske laget, og man kan benytte teknikker som spredt spektrum og frekvenshopping [30] for å beskytte seg mot DoS angrep. Dessuten kan DoS angrep også rettes mot rutingsprotokollen, slik at kommunikasjonen forhindres.
- Maskerade angrep (Eng: Masquerade attack)
    - En mellomliggende sensornode utgir seg for å være en gyldig sensornode, men har i realiteten opprettet et falskt mellomledd mellom to gyldige sensornoder, eller det at en sensornode utgir seg for å være en annen sensornode [4].
    - Slik angrep kan forhindres ved at partene autentiserer seg overfor hverandre, for eksempel ved at de benytter symmetrisk kryptografi.
- Selektiv videresending (Eng: Selective forwarding)
    - En motstander kan omprogrammere en sensornode til kun å droppe visse type pakker, for så å videresende resten. Sensornoden kan også modifiseres pakkene med feilaktig data, for så å videresende datapakken til sinken.
    - Kryptering hindrer slike angrep, eller ved bruk av redundante ruter.

- Slukhull angrep (Eng: Sinkhole attack)
    - Motstanderen prøver å tiltrekke seg all trafikk fra et vist område ved å opprette et sluk/slukhull i sensornettverket, slik at trafikken rutes gjennom en sensornode som styres av motstanderen. Den prøver å tiltrekke andre sensornoder til å sende data gjennom den. Dette ved for eksempel å annonsere at den har ruter av høy kvalitet mot sinken. Ved et slukhull angrep, så forkastes alle pakkene. Slukhull angrep er veldig lett å få til i sensornettverk på grunn av at kommunikasjonsmønsteret er relativt forutsigbart. Det vil si at alle nodene sender data til sink, og ved å opprette en rute til sinken gjennom motstanderen, så vil motstanderen kunne avgjøre selv hva han vil gjøre med trafikken.
    - Hierarkisk og geografisk ruting vil redusere omfanget.
- Sybil angrep (Eng: Sybil attack)
    - En sensornode som styres av motstanderen kan utføre et sybil angrep ved at den benytter flere identiteter, for så å utgi seg for å ha flere identiteter overfor andre sensornodene.
    - Dette kan forhindres ved at hver nabo tildeles en eller flere gyldige naboer, og trafikken kan gå igjennom både verifiserte og ikke verifiserte sensornoder. Det er sinken som kontrollerer antall naboer for hver sensornode. Hvis antall naboer er høyere enn forventet, antar sinken at det sybil angrep har forekommet. På dette tidspunktet kan trafikken kun rutes gjennom verifiserte sensornoder.
    - For å verifisere naboene, må asymmetrisk kryptografi, eller sertifikater benyttes. Men disse vil ikke bli vurdert i denne sammenhengen.
- Ormehull angrep (Eng: Wormhole)
    - Denne type angrep krever minimum 2 sensornoder som styres av en motstander. Meningen er å sende meldinger som mottas i en del av sensornettverket over i et annet sensornettverket.
    - Geografisk ruting reduserer problemet, men innfører et nytt problem i form av at sensornodene må ha tillitt til hverandre.
- Hello angrep (Eng: Hello flooding)
    - Mange protokoller krever at nodene kringkaster HELLO pakker for å annonsere sitt nærvær til naboene sine. Når en sensornode mottar en slik melding, vil den anta at den er innenfor normal radio rekkevidde til senderen. En motstander kan kringkaste slike pakker med en kraftig transceiver, slik at andre sensornoder tror motstanderen er i nærheten av dem. De andre sensornodene vil da prøve å sende data til motstanderen, men disse vil ikke nå frem.
    - Angrepet forhindres ved å bruke 3-veis håndtrykk. Her må alle nye sensornoder sende HELLO meldinger. En sensornode som mottar en HELLO melding, svarer med en tilfeldig generert melding. Den nye sensornoden må da sende denne tilfeldige melingen tilbake, noe som garanterer at linken er gyldig.
- Falsk bekreftelse (Eng: Malicious acknowledgement))
    - Målet er å overbevise en sensornode om at en svak link er en god link, eller at en dø/ødelagt sensornode faktisk er en sensornode som fungerer. Dette gjøres gjennom å forfalske linklagskvitteringer.
    - Angrepet kan forhindres ved bruk av et tilfeldig tall, sammen med kryptering. Når en sensornode mottatt og dekryptert en melding, sendes det tilfeldige tallet tilbake i kvitteringen.

3.3 Målet med sikkerhet

I tradisjonelle nettverk omhandler sikkerhet autentisering, integritet, konfidensialitet, og ikke-benektelse [4]. I et sensornettverket gjelder de 3 første, men også andre krav spiller inn. Nedenfor kommer en kort forklaring på sikkerhetsbegrepene som er nødvendig for å oppfylle kravene for sikkerhet [4] [36][42][72]:

3.3.1 Integritet

En sensornode må kunne være sikker på at data den mottar fra andre sensornoder ikke har blitt endret underveis og at integriteten er bevart. Det at integriteten er bevart betyr at sensornoden har en garanti om at det den mottar faktisk er det som ble sendt, og at data ikke har blitt endret underveis av en motstander. Integritet oppnåes gjennom dataautentisering. Integritet og dataautentisering kan kombineres i form av digitale signaturer. Signaturen legges med dataene og fungerer som et fingeravtrykk av dataene. Signaturen er igjen låst på en slik måte at man vil se dersom den har blitt endret underveis, eller dersom en annen har laget den. Det er vanlig at en MAC eller en MD5 benyttes for å bevare integriteten [4].

3.3.2 Autentisering og aksesskontroll

Autentisering kan deles opp i 2: Autentisering av en sensornode (entitet), og autentisering av en melding. En sensornode i sensornettverket må kunne sikre seg om at den sensornoden eller sinken den kommuniserer med faktisk er den man utgir seg for å være. Dette krever at sensornodene beviser identiteten sin, slik at mottaker kan verifisere at mottatt data faktisk kommer fra riktig sender. Hvis ikke autentisering forekommer, kan en motstander utgi seg for å være en gyldig sensornode, og på denne måten tilegne seg sensitiv informasjon eller tilføre egne meldinger inni sensornettverket. En sensornode må derfor identifisere seg ovenfor en annen sensornode. I en to parts kommunikasjon [12] kan dataautentisering oppnåes gjennom ren symmetriskteknikker. Denne prosessen forgår ved at både sender og mottaker kjenner til en hemmelig nøkkel som benyttes til å generere en meldings autentiserings kode (MAC) av alle data. Denne MAC mekanismen legges med i alle meldingene som sendes, og på denne måten kan mottaker forsikre seg om at den er kommet fra en gyldig sender. Problemet er at slik autentisering kan ikke benyttes i kringkasting, i hvert fall ikke så lenge det ikke forekommer god tillit mellom sensornodene. For eksempel hvis en sender ønsker å sende autentisert data til en gjensidig sensornode uten tillitt, er bruken av symmetrisk MAC ikke sikkert. Dette fordi enhver av mottakerne kjenner MAC nøkkelen, og på denne måten kan de da utgi seg for å være senderen, for så å forfalske meldinger til andre mottakere. Derfor benyttes asymmetrisk mekanismer for å oppnå autentisert kringkasting i tradisjonelle nettverk, men dette kan bli vanskelig å realisere i et sensornettverk. Hvis det ikke foreligger noen form for autentisering eller konfidensialitet, da er det ingen mekanismer som beskytter nettverket fra å bli tatt over av en motstander. For eksempel kan en motstander sette opp sin egen sink som kringkaster en bedre rute enn den gyldige sinken, slik at sensornodene heller vil sende data til falske sinken [7].

Med aksesskontroll er det ønskelig å begrense tilgangen til sensornettverket ved å begrense sensornodenes evne til å få tilgang til visse ressurser, noe som gir økt kontroll over ressurstilgangen. Kun de med tilgang skal kunne benytte seg av en viss ressurs. Ved å benytte en MAC eller MD5, så hindrer man uautoriserte sensornoder i å sende meldinger inni sensornettverket.

3.3.3 Konfidensialitet

Informasjon som utveksles mellom to noder må i noen tilfeller være konfidensiell. Det vil si at kun de partene informasjonen er beregnet for skal kunne lese informasjonen. En sensornode må kunne benytte seg av krypteringsteknikker slik at data som sendes først krypteres ved hjelp av en hemmelig nøkkel som bare de involverte partene kjenner til, slik at informasjonen ikke leses av uautoriserte. Dette forhindrer passive angrep. Dessuten er det algoritmene som avgjør hvor sterk kryptering man har [4]. Desto større krypteringsnøkkel, desto mer prosesseringskraft kreves. Det vil si at desto lengre krypteringsnøkkelen er, desto vanskeligere vil det være å finne riktig kryptering. Hvis krypteringsnøkkelen er liten, altså mindre enn 128 bits, vil det være relativt enkelt å knekke krypteringen gjennom å prøve ut alle mulige nøkkelverdier. Nøkkelstørrelsen, samt hvor lang tid det tar å knekke krypteringen er en måte å måle hvor god kryptering man har. Det er vanlig at data krypteres før forsendelse, for så å dekrypteres ved mottak.

3.3.4 Data ferskhet

Det er ikke alltid tilstrekkelig med dataautentisering, integritet og konfidensialitet. Vi må også kunne forsikre oss om at hver melding er fersk [12]. Det at data er fersk innebærer at man har en slags garanti for at ingen motstander har svart på gamle meldinger. Det finnes 2 typer ferskhet: svakferskhet, som tilbyr delvis sortering av rekkefølgen til meldingene, men inneholder ingen informasjon om meldingsforsinkelsen. Den andre typen er sterkferskhet, som tilbyr en fullstendig sortering av rekkefølgen til meldingene, samt estimert forsinkelse. Svakferskhet er nyttig for sensormålinger, mens sterkferskhet er nytting for tidssynkronisering innenfor sensornettverket. Ferskhet oppnåes gjennom at meldingenes rekkefølge taes i betraktning, samt tidstempelet. Det vil si at man noterer når en pakke blir sendt, slik at man kan estimere om en melding har blitt fanget opp eller ikke. En teller, som egentlig er en strøm av bits som genereres av sender, benyttes vanligvis for å oppnå ferskhet. Telleren sørger for at meldingenes rekkefølge er korrekt, og at meldingene ikke er kopier av tidligere sendte meldinger. Dataferskhet kan oppnåes ved hjelp av teller som er en ikke repeterende verdi.

3.3.5 Tilgjengelighet og ikke-benektelse

Sensornettverket skal være tilgjengelig for kommunikasjon til enhver tid, uansett om en gruppe sensornoder er blitt kompromittert. Ikke-benektelse innebærer at en sender av en melding ikke kan nekte for å ha sendt informasjonen, samt at mottaker ikke kan nekte for å ha mottatt informasjonen. Ikke-benektelse er dessuten ikke nødvendig å innføre i et sensornettverk så lenge man kan verifisere at alle pakker som sendt av gyldige sensornoder.

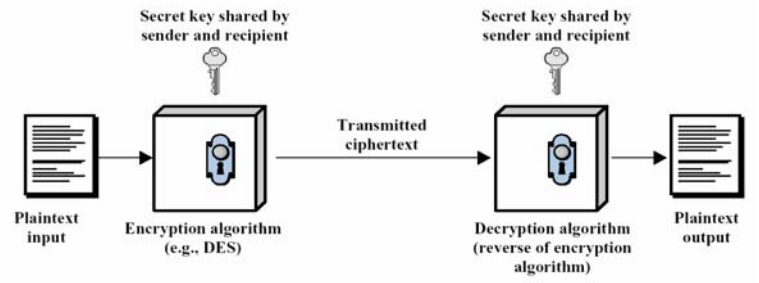
3.4 Krypteringsalgoritmer

Det finnes 2 typer krypteringsalgoritmer: symmetriske og asymmetriske [4][34]. Det som først og fremst skiller disse, er at ved symmetrisk kryptografi benyttes samme krypteringsnøkkel både til kryptering og dekryptering, mens asymmetrisk kryptografi benytter et par av krypteringsnøkler. Dette nøkkelparet er en felles krypteringsnøkkel, og en privat krypteringsnøkkel, hvor den private krypteringsnøkkelen alltid holdes hemmelig. Innenfor disse finnes det igjen mange forskjellige typer algoritmer.

3.4.1 Symmetrisk versus asymmetrisk kryptografi

Det er slik at asymmetriske algoritmer egner seg best til nøkkelutveksling og autentisering av brukere som kun skjer en gang per sesjon, mens symmetriske passer best til kommunikasjonen som foregår i etterkant av brukerautentisering. Ettersom asymmetriske krypteringsalgoritmer ikke er passende for et sensornettverk på grunn av at slike krypteringsalgoritmer er mye større, og krever mer prosesseringskraft, noe som igjen gjør det mer strømkrevende [13], så vil det kun være symmetriske krypteringsalgoritmer som er aktuelle [13]. Dessuten krever asymmetriske mekanismer at sender og mottaker har flere nøkler. Dette vil føre til at man må lagre mange nøkler, noe som igjen gjenspeiler seg på lagringsplassen. Dessuten er asymmetriske mekanismer mer matematiske, og krever mer prosessering ettersom nøklene er større (1024 bits nøkler). Symmetriske teknikker har mange fordeler foran asymmetriske: Høyere gjennomstrømning, kortere nøkler, raskere asymmetrisk kryptografi, samt at symmetriske teknikker kan benyttes til å konstruere andre sikkerhetsmekanismer. Dessuten kan symmetriske chiffer sammenslåes for å danne kraftigere chiffer. Ulempene er at nøkkelen må forbli hemmelig for alle andre parter som ikke er involvert, og i store nettverk trengs mange symmetriske nøkler, samt at nøklene burde endres ofte. Dessuten er utvekslingen av den symmetriske nøkkelen er komplisert.

Symmetriske kryptografi baserer seg på at både sender og mottaker har samme hemmelignøkkel som både benyttes til kryptering og dekryptering, hvor sender krypterer klartekst med den hemmelige nøkkelen før transmisjon, og mottaker dekrypterer pakken ved mottak, med den hemmelige nøkkelen. Med andre ord så betyr det at A og B har samme krypteringsnøkkel, det vil si at alle data som A krypterer kan dekrypteres av B, og motsatt. Selve krypteringsprosessen for symmetrisk kryptering er illustrert i figuren [4] nedenfor.



Figur 8. Symmetrisk kryptografi.

Symmetriske kryptografi kan benytte seg av krypteringsnøkler som blant annet DES, 3DES, Skipjack, Blowfish, IDEA, RC4, RC5, RC6, AES. For sikre algoritmer, så er det nøkkelstørrelsen som bestemmer hvor lang tid det tar å knekke algoritmen /arbeidsmengden. Hver nøkkel har 2^n muligheter. Symmetrisk kryptografi kan få problemer med nøkkelhåndtering ettersom hver node må lagre n-1 nøkler, noe som gir et total av n^2 nøkler. Det har blitt testet ut en del symmetriske krypteringsnøkler i sensornettverk, hvor RC5, AES og Skipjack har vist seg å fungere tilfredsstilende for sensornettverket [36].

Fordelene med asymmetrisk kryptografi er at den er bedre og mer egnet til distribuering av nøkler, samt nøkkelhåndtering [4]. Til nøkkelhåndtering kreves færre nøkler, men felles nøkkeldistribusjon er et problem. Andre fordeler er at den har bedre skalerbarhet for større systemer, ved at den kan tilby verifisering og ikke-fornektelse. Kun den private nøkkelen trenges å holdes hemmelig, dessuten trenger man ikke endre nøklene så ofte. Brukes mest til digital signaturer. Ulemper er at den er treg, og at den er matematisk intensiv. Dessuten har den dårligere gjennomstrømning, krever store nøkler (1024 bit), og er ikke like sikker som

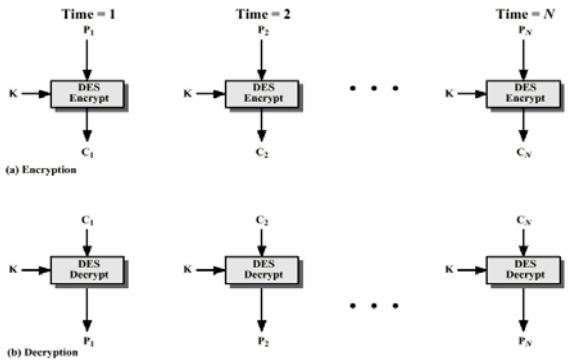
symmetriske teknikker. Asymmetrisk kryptografi benytter RSA, ECC, Diffie-Hellman, El Gamal, DSA, Knapsack, PGP. [5] har tatt for seg asymmetrisk kryptografi, hvor de har testet et sensornettverk som benytter krypteringsprotokoller som RSA, DSA og El Gamal. Disse har fungert, men strømforbruket var uansett høyt. Mer om dette i [5].

Chiffer er en teknikk som bruker matematiske algoritmer for å transformere data inni noe som ikke er lesbart, og kan sees på som et sett med regler for hvordan klarteksten skal transformeres inni chiffertekst. Denne transformasjonen og gjenvinningen av data avhenger av algoritmen, samt null eller flere krypteringsnøkler. Symmetriske algoritmer kan deles inn i to typer chiffer [36]: Blokk chiffer og strøm chiffer.

3.4.1 Blokk chiffer

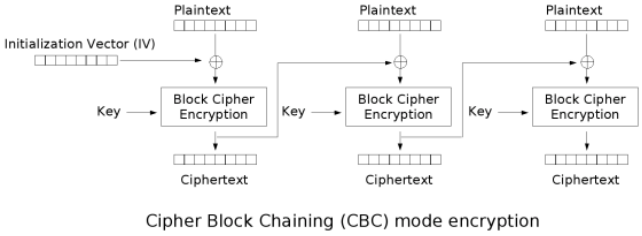
En blokk chiffer [4][36] er en krypteringsmetode som har som funksjon å dele opp en klartekst inni blokker av fast størrelse, for så å kryptere hver av disse blokkene en og en om gangen. Resultatet blir blokker av chiffertekst på samme størrelse som hver blokk av klartekst. Typisk blokkstørrelser er 64 og 128 bit for tradisjonell sikkerhet [4], men i et sensornettverk vil denne størrelsen sannsynligvis avhengig av pakkestørrelsen og pakkeformatet ([63] har implementert sikkerhet i et sensornettverk som bruker AES, samt en blokkstørrelse på 128 bit, noe som har fungert bra). En blokk chiffer deler opp klartekst i blokker av fast størrelse. I krypteringsalgoritmen kommer en blokk av klarteksten, samt en hemmelig nøkkel som input. Det er krypteringsalgoritmen som utfører ulike substitusjoner og transformasjoner på klarteksten, men eksakt hvordan dette skal gjøres avhenger av den hemmelige nøkkelen. Ut kommer en chiffertekst, som avhenger av klartekst og den hemmelige nøkkelen. 2 forskjellige nøkler vil danne 2 forskjellige chiffertekster. I motsatt ende foretaes dekryptering, ved å kjøre krypteringsalgoritmen i revers, hvor den benytter den hemmelige nøkkelen på chiffertekster for å gjenvinne klarteksten. Sikkerheten i symmetrisk kryptering avhenger av at nøkkelen forblir hemmelig, ikke at algoritmen forblir hemmelig. Vi trenger kun å holde nøkkelen hemmelig. Det finnes flere forskjellige operasjonsmoder som kan benyttes:

- ECB (Electronic Codebook Mode):
    - I ECB blir hver blokk av 64 bits klartekst kryptert hver for seg ved hjelp av den samme nøkkelen. ECB er bra til transmisjon av enkeltverdier som for eksempel krypteringsnøkler.
    - Hvis den samme 64 bit chiffertekst blokken forekommer mer enn en gang i meldingen, så vil den produsere den samme chiffertekster, så derfor er ikke ECB sikker. Hvis meldingsstrukturen er lik, så kan motstanderen utnytte dette.
    - Figuren under [4] illustrerer prosessen.

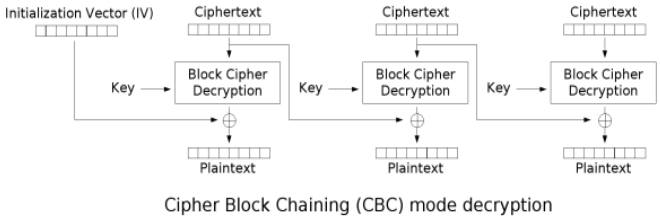


Figur 9. ECB operasjonsmoder.

- CBC (Cipher Block Chaining)
    - Input i krypteringsalgoritmen blir XOR`ed med den neste 64 bits klarteksten, og de foregående chiffertekstene. CBC brukes til autentisering, og vanlig blokkorientert transmisjon.
    - Ved kryptering blir IV og første klartekstblokk XOR`ed sammen, og resultatet blir sendt som input i krypteringsalgoritmen for så å produsere chiffertekstblokk som output. Deretter blir foregående chifferblokk og nåværende klartekst XOR`ed, og resultatet av dette blir sendt som input i krypteringsalgoritmen.
    - Samme nøkkel i krypteringsalgoritmen brukes for hver blokk
    - Ved dekryptering blir hver chifferblokk kjørt gjennom dekrypteringsalgoritmen. Første gang blir IV XOR`ed med outputen fra dekrypteringsalgoritmen, for så å gjenopprette første klartekstblokk. Deretter blir foregående chifferblokken XOR`ed med outputen fra dekrypteringsalgoritmen for å produsere resterende klartekstblokkene.
    - IV må være kjent av både sender og mottaker. For å oppnå best mulig beskyttelse, så bør også IV beskyttes like bra som nøkkelen.
    - Figurene under illustrerer krypterings og dekrypteringsprosessen.



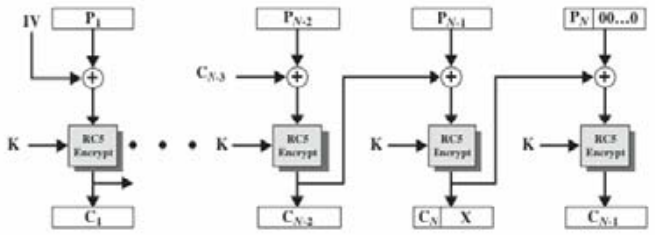
Figur 10. CBC krypteringsprosessen.



Figur 11. CBC dekrypteringsprosessen.

- CTS (Cipher Text Stealing) [36] [62] er en variant av CBC som ikke krever noen form for padding.
    - Krypteringsprosessen:
        - Hvis lengden på klartekst ikke er like stor som blokk størrelsen, så legges det til tilstrekkelig med nuller slik at den blir lang nok. Deretter krypteres klarteksten med CBC, for så å bytte ut de to siste CBC blokkene, og så å kutte chifferteksten ned til lengden av den opprinnelige klarteksten. Det eneste CTS krever er at data er minimum en blokkstørrelse, for eksempel 8 bytes, ellers er den på samme størrelse som klartekst.

- Dekrypteringsprosessen:
  - Hvis chifferteksten ikke er like stor som blokkstørrelsen, ved for eksempel at den er n bit for kort, så paddes den med de siste n bitsene til blokkchiffer dekrypteringen av den siste fulle chiffertekst blokken. Bytt ut de to siste chiffertekst blokkene, for så å dekryptere chifferteksten med CBC, for så å kutte klarteksten ned slik at den blir like lang som chifferteksten.
- CTS krever dessuten av klarteksten er større enn en blokkstørrelse
- Figuren under illustrerer CTS prosessen.

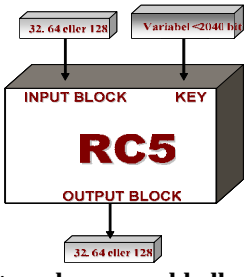


Figur 12. CTS krypteringsprosessen.

Aktuelle blokkchiffer [4] er RC5, Skipjack og AES for et sensornettverk. Disse er det blitt benyttet i tidligere prosjekter [28] [42] med stor suksess, og kan fint benyttes i et sensornettverk. [28] har dessuten testet flere sikkerhetsprotokoller, for så å evaluere deres tilpasning for et sensornettverk. Mer om dette finnes i [28].

3.4.1.1 RC5

RC5 [4][42][45][56] er en relativ enkel og rask symmetrisk blokkchiffer som tilbyr høy sikkerhet. RC5 benytter parametere som variabel blokkstørrelse, variabel nøkkelstørrelse, variabelt antall gjennomganger, og kan i tillegg brukes på prosessorer av variabel ordstørrelse. Disse parameterne kan endres, og tillater en balanse mellom høyere hastighet og bedre sikkerhet, slik at sikkerhetsnivået kan endres ut i fra nødvendigheten. Hvis parametere er for lave, så vil ikke gi noen som helst sikkerhet, mens veldig høye parametere gir kraftig sikkerhet, men kan være drepende for teknologier som blant annet sensornettverk. RC5 tar imot en fast blokkstørrelse som input, og produserer en fast blokkstørrelse som output, ved å benytte en hemmelig nøkkel. Fordel med RC5 siden den bruker lite kodestørrelse, samt at den er veldig effektiv. En annen fordel med RC5 er dens lave minnekrav, og dens relative høye krypteringsytelsen, noe som gjør den passende til enheter med begrenset minne, som blant annet sensornoder [7][13]. Sikkerhetsparameterne i RC5 kan endres ettersom ut i fra systemkravene som et systemet har, og på denne måten kan den balansere mellom sikkerhetsnivå og strømforbruket. Ulempene med RC5 er at den er patentert, samt at RC5 krever at nøkkelen er kalkulert på forhånd, noe som bruker 104 ekstra bytes RAM pr nøkkel.



Figur 13. Oversikt over blokkstørrelsen og nøkkellengden til RC5, samt output.

Antall runder i en krypteringsalgoritme indikerer hvor god krypteringsalgoritmen er. Det er anbefalt minimum 16 runder for å få tilfredsstillende sikkerhet [70].

3.4.1.2 Skipjack

Skipjack [36][44] er en blokk chiffer som har lavt minneoverhead som fint kan brukes til 8 bits enheter. Skipjack krypterer og dekrypterer data inni 64 bits blokker ved hjelp av en 80 bits nøkkel. Den tar en 64 bit klartekstblokk som input, og gir en 64 bit chiffertekst som output. Sikkerheten i skipjack avhenger av antall ganger algoritmen gjentaes for å produsere chifferteksten. Sikkerheten øker eksponentialt med antall ganger algoritmen gjentaes. Skipjack har dessuten minimalt med kostnader assosiert med nøkkeloppsett og er ikke patentert som blant annet RC5.

3.4.1.3 AES

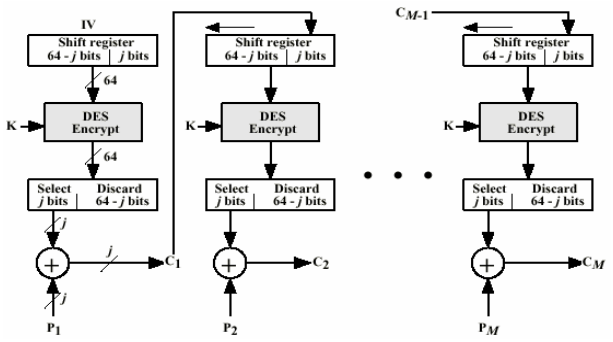
AES kan benyttes, og det er testet at denne algoritmen fungerer nesten like bra som RC5 [7]. En negativ ting med AES er at størrelsen på blokk er litt større enn på RC5. AES bruker fast blokkstørrelse på 128 bit, og en nøkkelstørrelse på 128, 192 og 256 bit. Nøkkelen og blokkstørrelsen kan være i størrelsesorden på 32 bit fra minimum 128 bit til maksimum på 256 bit. Det er ingen svakheter på nøklene, og ingen kjente angrep mot algoritmen når algoritmen benyttes med 10 runder. Implementeringen er relativ effektiv, og dette inkluderer også smartkort [4].

3.4.2 Strømchiffer

En strøm chiffer prosesserer input kontinuerlig, og benytter en IV og en nøkkel som input [4]. Denne nøkkelstrømmen XOR`es med klarteksten. Strømchiffer er vanligvis raskere enn blokkchiffer, men strømchiffer er en stor ulempe ved at hvis den samme IV benyttes flere ganger til å kryptere to forskjellige pakker, så kan begge klartekstene gjenvinnes. For å kunne garantere dette, så må IV være relativt lang (8 til 16 bytes). Det å benytte kortere IV enn 8 - 16 bytes vil være veldig usikkert, og da må man være forberedt på at IV gjenbrukes relativt fort, og at sikkerheten utsettes deretter. En strøm chiffer bruker en nøkkel K og en IV som en tellerverdi, og utvider det til en stor, tilfeldig nøkkelstrøm $G_K(IV)$. Nøkkelstrømmen blir så XOR`ed med meldingen: $C = (IV, G_K(IV) \text{ XOR } P)$. De raskeste strømchiffer er raskere enn den raskeste blokkchiffer, noe som gjør den virker veldig passende for et sensornettverk. Problemet med strømchiffer er at hvis den samme IV brukes til å kryptere to forskjellige meldinger, så er det mulig å gjenvinne begge klartekstene. For eksempel hvis vi har gitt: $C = (IV, G_K(IV) \text{ XOR } P)$ og $C' = (IV, G_K(IV) \text{ XOR } P)$, så kan man gjenvinne $P \text{ XOR } P'$, noe som gjør at man kan gjenvinne deler, eller all informasjon i P og P´. Dette er en kjent svakhet som kan illustreres i WEP protokollen [18]. For å kunne garantere at IV ikke brukes flere ganger, så må IV være tilstrekkelig lang, for eksempel på 8-16 bytes. Dette kan vise seg å være for mye i et sensornettverk som har begrenset størrelse på pakkeformatet. Et alternativ er å bruke kortere IV, for så å akseptere gjenbruk av IV. Et annet alternative er å bruke blokkchiffer. Strøm chiffer benytter seg av følgende operasjonsmoder [4]:

- CFB
    - Input blir prosessert J bit om gangen. Foregående chiffertekster brukes som input i krypteringsalgoritmen for å produsere en tilfeldig output, som igjen blir XOR`ed med klarteksten for å produsere neste enhet av chifferteksten. CFB brukes til autentisering og strømorientert transmisjon.

- CFB kan konvertere fra blokkchiffer til strømchiffer.
- 1 strømchiffer trenger ikke å padde en melding for å få den til å bli en fast størrelse. Den kan operere i real time.
- En fordel med strøm chiffer er at chifferteksten ikke trenger å være av samme lengde/størrelse som klarteksten. Hvis et 8 bit tegn sendes, så blir hvert tegn kryptert med 8 bit. Hvis mer enn 8 bit brukes, så vil det sløse båndbredde.
- Figuren [4] under illustrerer prosessen:



Figur 14. CFB operasjonsmoder.

- OFB
  - Veldig lik CFB, bortsett fra at input i krypteringsalgoritmen er den forgående outputen til krypteringsnøkkelen. CFB er bra til strømorientert transmisjon over en kanal med mye støy.

3.4.2.1 RC4

RC4 er en symmetrisk strøm chiffer av variabel nøkkelstørrelse med byte orientert operasjon. Selve algoritmen baserer seg på tilfeldig ombytting.

3.4.3 Sammenligning av blokkchiffer og strømchiffer

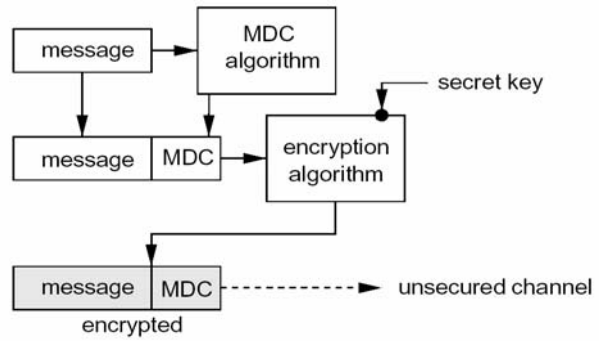
Blokkchiffer benytter en operasjonsmoder som deler meldingene inn i små segmenter som kalles blokker. Hver av disse blokkene krypteres blokk for blokk. For eksempel vil en CBC kun lekke små mengder av informasjon ved gjenbruk av IV, noe som er en stor forbedring over strømchiffer. CBC er designet til bruk sammen med en IV. Hvis vi bruker en 8 byte blokkchiffer resulter det i chiffertekst bygges opp av blokker på 8 bytes. Dette kan resultere i en forlengning av meldingen hvis meldingen ikke er like stor som 8 bytes. Man kan for eksempel benytte CTS som sørger for at chifferteksten er like lang som underliggende klartekst. Kryptering av datalast mindre enn 8 byte vil produsere en chiffertekst på 8 byte, fordi CTS krever minst en blokk av chiffertekst. Det å benytte en blokk chiffer har flere fordeler. Det at den mest effektive meldings autentiseringsalgoritme (MAC) bruker blokkchiffer, så må alle sensornodene implementere blokkchiffer. Det å benytte blokkchiffer besparer kodeplass.

3.5 Hash funksjoner

En teknikk som hashing kan brukes til å bevare integriteten til meldingene som utveksles mellom sensornodene. MDC [4] og MAC [7] er aktuelle teknikker ettersom disse ikke benytter så store blokkstørrelser.

3.5.1 MDC (Manipulation Detection Codes)

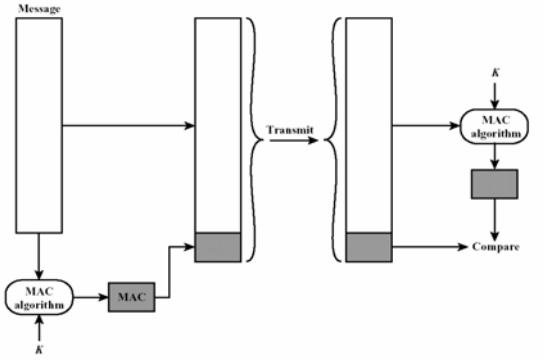
MDC [4] sikrer dataintegritet ved hjelp av tilleggsmekanismer, men benytter ingen nøkkel til å kryptere MDC mekanismen. Det dannes en MDC ved hjelp av en MDC algoritme. Denne MDC taggen sendes sammen med meldingen. Mottaker danner så en egen MDC, og sammenligner sin egen MDC med den som ble sendt med pakken. Hvis de er like, så er integriteten bevart. For å få kryptert MDC, så må hele pakken krypteres. Figuren [4] nedenfor illustrerer prosessen.



Figur 15. Selve prosessen hvor MDC genereres og sendes.

3.5.2 MAC (Message Authentication Codes)

MAC [4][36] er enklere enn MDC, ettersom den ikke er avhengig av tilleggsmekanismer, og forskjellig fra MDC, så bruker MAC en krypteringsnøkkel for å kryptere MAC mekanismen. Selve MAC er en liten informasjonsdel som sikrer autentisering av meldingen og senderen, samt bevarer integriteten til meldingen. MAC mekanismen krypteres ved hjelp av en symmetrisk nøkkel som sender og mottaker deler. Vi oppnår både autentisering og integritet ved å legge med en MAC i datapakken som transmitteres. MAC kan sees på som en kryptografisk sikkerhets sjekksum som legges på datapakken. For å generere en MAC, må både sender og mottaker dele samme hemmelige nøkkel. Senderen genererer en MAC over pakken med den hemmelige nøkkelen som begge deler. Denne MAC mekanismen sendes sammen med datapakken. Mottakeren foretar samme prosess ved å generere en MAC av datapakken ved hjelp av den delte, symmetriske nøkkelen, for så å sammenligne denne MAC delen med MAC mekanismen som sender la med i pakken. Hvis MAC mekanismene er like, da er pakken gyldig, og kan mottaes av mottaker, ellers så forkastes den hvis MAC mekanismene er ulike. Dette betyr at hvis en motstander endrer noe av data i pakken, eller hvis for eksempel interferens/støy under transmisjonen endrer noen bit pakken, så vil det føre til at MAC mekanismene blir ulike, og pakken forkastes. MAC mekanismene kan lages ved å benytte blokkchiffer. Selve prosessen kan illustreres slik [4]:



Figur 16. Selve prosessen hvor MAC genereres og sendes.

3.5.3 SHA-1 (Secure Hash Algorithm) og MD5 (Message Digest 5)

Disse hash funksjonene kan brukes til å danne en hash av en melding. Et problem med disse er at SHA-1 danner en 20 bytes hash av meldingen, mens MD5 danner en 16 bytes hash av meldingen. Derfor vil ikke disse bli evaluert i denne sammenhengen. Det er foretatt testing av ulike hash algoritmer om deres anvendelighet i et sensornettverk. Mer om dette kan leses finnes i [13].

3.6 Nøkkelutveksling

Nøkkelhåndtering [4][36] i form av nøkkelutveksling er en stor utfordring ettersom asymmetriske teknikker ikke er passende for sensornettverk et sensornettverk. Dette er på grunn av karakteristikken til ad hoc nettverk, samt alle ressursbegrensningene i et sensornettverk. Som nevnt tidligere, så benytter ikke ad hoc nettverk noen form for sentral administrasjon siden nettverket er distribuert. For å effektivisere nøkkelhåndteringen er man avhengig av en sentralisert administrasjon. Nøkkelhåndtering sier noe om hvordan kryptografiske nøkler distribueres og deles i sensornetteverket. I tradisjonelle nettverk er det vanlig å bruke forskjellige nøkler for forskjellige oppgaver. Det er flere teknikker på hvordan dette kan gjøres. Uansett hvilken teknikk som brukes, så vil det være en stor fordel å ofte endre den symmetriske nøkkelen, slik at motstanderen ikke får tak i den delte nøkkelen.

3.6.1 Utfordringer med symmetrisk nøkkelhåndtering

Det er kjent at den beste måten å foreta nøkkelhåndtering er gjennom asymmetrisk teknikker [4]. Men som nevnt tidligere, så vil ikke denne typen teknikker ikke bli vurdert i denne sammenhengen grunnet karakteristikken til sensornettverket. Kun symmetriske teknikker vil bli vurdert. Men selv med symmetriske teknikker, så kan et effektivt nøkkelhåndteringsskjema være vanskelig å oppnå. Dette er på grunn av sensornodenes begrensede ressurser, og da spesielt lagringsplassen. I en sensornode vil krypteringsnøklene bli lagret i RAM. Det er vanlig i tradisjonelle nettverk at kommuniserende parter deler en symmetrisk nøkkel i form av en sesjonsnøkkel seg i mellom som ingen andre besitter. Denne nøkkelen kan bli utlevert på flere måter, enten fra en sentralisert enhet som begge kommuniserende parter stoler på, eller så kan de ha fått tildelt den fysisk. Men i et sensornettverk kan dette bli vanskelig å realisere. Hvis hver sensornode skal få utlevert en sesjonsnøkkel (separat nøkkel som ingen andre har) til hver enkelt sensornode den kommuniserer med, så betyr det i praksis at hver sensornode må lagre 2^n sesjonsnøkler avhengig av n antall naboer. Dette blir mange nøkler, avhengig av hvor mange naboer sensornoden har. Men hvis for eksempel hver sesjonsnøkkel er på 128 bit hver, og med en RAM størrelse på 4KB, så er det klare begrensninger på hvor mange sesjonsnøkler hver sensornode kan lagre. I tillegg vil sensornodene ikke ha 4KB ledig ettersom mye av lagringsplassen går til blant annet operativsystemet. Dette er en av grunnene til at asymmetriske teknikker ikke kan brukes, ettersom en asymmetrisk krypteringsnøkkel er på 1024 bit hver.

3.6.2 Nøkkeldistribusjon med symmetrisk nøkkelhåndtering

Symmetrisk teknikker gir ingen god nøkkelhåndtering. Men i et sensornettverk så kan det vise seg å være den beste måten å oppnå en strømeffektiv nøkkeldistribusjon. Det er flere måter å oppnå dette på.

3.6.2.1 Fysisk nøkkeldistribusjon

Den enkleste formen for nøkkelutveksling er å fysisk utlevere [4] en hemmelig nøkkel til hver sensornode allerede ved utplassering, slik at alle autoriserte sensornoder kan kommunisere sammen ved hjelp av denne nøkkelen (alle bruker samme nøkkel). Fordelen er at det gir en relativt enkel nøkkelhåndtering, samt minimalt med konfigurering. Dessuten kan man foreta lokal kringkasting ettersom alle sensornodene kan lese pakker som er sendt fra gruppeleder. Problemet med en slik nøkkelhåndtering, er at hvis en motstander klarer å gjenvinne nøkkelen, så er hele sikkerheten i sensornettverket utsatt. Dette gjør at motstanderen kan lytte til all trafikk, eller tilføre egne meldinger inni nettverket. Alle uautoriserte som prøver å kommunisere med sensornodene, vil bli avvist.

3.6.2.2 Gruppeleder foretar nøkkeldistribusjon

En sentral administrasjon ville dannet en kraftigere nøkkeldistribusjon [4]. En fremgangsmåte vil være å organisere sensornodene i grupper [69], og hvor sensornodene innad i hver av disse gruppene deler samme nøkkel, enten med hverandre, eller ved at alle sensornodene deler samme hemmelige nøkkel med gruppelederen. Dette vil kun føre til at en motstander kan lytte til trafikk i kun et lite område, samt tilføre egne meldinger her, og ikke i hele sensornettverket, men kun en liten del av sensornettverket. En kompromittert sensornode kan dekryptere alle meldinger fra sensornoder som er medlem av samme gruppe som motstanderen, men kan ikke dekryptere meldinger som er sendt fra andre grupper, eller tilføre egne meldinger hos dem. Det kan være en fordel å organisere disse gruppene ut i fra hvilke oppgaver de har, slik at data fra en gruppe kan gå direkte fra gruppeleder, og til sinken. Hvis data må rutes gjennom andre gruppeledere, så gjøres dette gjennom ende til ende kryptering, slik at ingen andre kan lese data enn sinken. Gruppering vil dessuten løse problemet med at sensornodene i en gruppe slipper å lagre så mange sesjonsnøkler. Gruppeleder vil opprette nye krypteringsnøkler, for så å distribuere den nye nøklene til sensornodene i gruppa ved å kryptere den nye nøklene med den gamle. Problemet er at dette vil føre til mye trafikk ettersom det vil kreve periodisk nøkkeldistribusjon, noe som igjen vil redusere levetiden til sensornettverket. Dette gir bedre sikkerhet, og en motstander som klarer å tilegne seg en nøkkel vil da kun ha begrenset adgangsmuligheter, slik at en motstander kun kan lytte til og tilføre egne meldinger innad i gruppa.

Nøkkelen i en gruppe kan utleveres på følgende måte [69]:

- Ved å benytte en felles nøkkel i hver gruppe. En part med tillitt hos alle sensornodene kan velge nøkkelen, og fysisk levere den til en gruppe sensornoder, for eksempel ved at en gruppeleder genererer en ny nøkkel, og krypterer denne nye nøkkelen med den gamle nøkkelen. Fordelen er at kompromitterte sensornoder får begrenset tilgang. En motstander får kun tilgang til pakker som kommer fra sensornoder som tilhører samme gruppe. Det vil si at motstanderen kan dekryptere, samt lese alle meldinger som kommer fra sensornoder fra samme gruppe, men motstanderen kan ikke lese meldinger som kommer fra andre grupper. Ulempen er at hvis denne nøkkelen kompromitteres, så utsettes sikkerheten innad i gruppen. Problemet er hvis gruppelederen kompromitteres, ettersom gruppelederen har kommunikasjonsnøkkelen til andre gruppeledere.
- Man kan benytte ulik nøkkel for hver kommunikasjon. Hvis to parter ønsker å kommunisere, så kan den ene parten generere en ny nøkkel, for så å kryptere denne ved hjelp av den gamle nøkkelen. Fordelen er at hvis en motstander kompromitterer

sikkerhet, så får motstanderen kun begrenset tilgang til sensornettverket. Sensornoden får også veldig begrenset tilgang. Hvert par av sensornoder som kommuniserer bruker forskjellige nøkler. Ulempen er at denne type nøkkelhåndterning trenger en nøkkeldistribusjonsprotokoll.

3.7 Sikkerhetsprotokoller for sensornettverk

De siste årene har det blitt foretatt begrenset med forskning innenfor sikkerheten i sensornettverk, og da spesielt på sikkerhetsprotokoller. Siden protokollene som benyttes i tradisjonelle nettverk ikke egner seg for sensornettverk har det vært en nødvendighet å utvike protokoller som er mer passende for sensornettverk. Noen aktuelle protokoller vil bli presentert her.

3.7.1 SPINS

SPINS protokollen [2][12] er en sikkerhetsprotokoll som er designet basert på 2 faktorer: at sensornoden skal operere mer effektivt, samt at strømkapasiteten skal bevares lengst mulig. Selve SPINS protokollen baserer seg på to andre byggeblokker, nemlig SNEP og µTESLA.

3.7.1.1 SNEP (Secure Network Encryption Protocol)

Funksjonen til SNEP [2][12][30] er å tilby konfidensialitet, 2-parts autentisering, integritet, og dataferskhet, med lav overhead. Den legger bare til 8 bytes per melding, noe som sikrer lav overhead, samt at den benytter DES-CBC. Som enhver annen kryptografisk protokoll så bruker den en teller, men unngår å sende telleren med pakken ved at tilstanden ivaretaes hos begge parter. Det som skjer først er at sender og mottaker eksplisitt oppretter en teller ved å utveksel meldinger seg i mellom. Etter at telleren er satt opp hos hver av partene, så inkrementeres telleren for hver pakke som mottaes. Telleren brukes også sammen med krypteringsnøkkelen for å opprette en MAC. Telleren sendes altså ikke med pakken, noe som sparer pakken for overhead. Når en mottaker mottar en pakke, så inkrementeres den siste lagrede verdien. SNEP kan brukes både for kommunikasjon mellom en sink og en bestemt sensornode, eller for kommunikasjon mellom en bestemt node og en sink. Protokollen garanterer at to identiske klartekst meldinger ikke produserer samme chiffertekst melding ved hjelp av en teller ”C”, som deles mellom en sensornode og sink. Hver node deler en unik masternøkkel K med sinken. Masternøkkelen brukes til å generere alle andre nøkler. En engangskrypteringsnøkkel brukes av SNEP for å tilby datakryptering. Denne engangskrypteringsnøkkelen er generert ved å bruke nøkkelen som er generert fra masternøkkelen K og en inkrementell teller som input i RC5 krypteringsalgoritmen. RC5 gir en binær streng som brukes som en engangsnøkkel. Meldingen XOR`es med engangsnøkkelen, for så sendes, og telleren inkrementeres for å forberede neste melding. Sinken er klar over nodens tellerverdi og den genererte nøkkelen, slik at den genererer en identisk engangsnøkkel, for så å XOR`e den krypterte nøkkelen med engangsnøkkelen for å få frem klarteksten.

SNEP har som mål å tilby følgende egenskaper:

- Semantisk sikkerhet:
    - Den samme meldingen krypteres forskjellig for hver gang ettersom telleren inkrementeres etter hver melding. Telleren er tilstrekkelig lang nok til å aldri gjentas innenfor levetiden til noden. Det betyr at hvis man krypterer den samme meldingen flere ganger, så vil man få forskjellig chiffertekst hver gang.

- Sikre data autentisering:
    - En mottaker kan være sikker på at en melding opprinnelig kommer fra riktig sender, hvis MAC verifiseres korrekt.
- Beskytter mot angrep basert på svar av meldinger (Eng: Replay attack):
    - Tellerverdien i MAC mekanismen forhindrer svar av gamle meldinger. Hvis telleren ikke er tilstede i MAC mekanismen, kan en motstander enkelt svare på meldingen.
- Svak ferskhet:
    - En mottaker vet at meldingen ble sendt etter den forrige meldingen den mottok korrekt (den hadde en lavere tellerverdi), hvis meldingen verifiseres korrekt.
- Lav kommunikasjons overhead
    - Tellertilstanden bevares hos hvert endepunkt og trenger ikke å sendes med hver melding (I tilfeller der MAC mekanismen ikke stemmer, kan mottaker prøve ut et fast (lite) antall teller inkrementeringer for gjenopprettelse fra meldingstap. I tilfeller der dette feiler, må de 2 partiene engasjere seg i en teller utvekslingsprotokoll, som bruker sterkferskhet protokollen.

Vanlig SNEP tilbyr kun svak data ferskhet [12], fordi den kun opprettholder senderekkefølgen på meldingene for node B, men ingen forsikring til node A om at en melding har blitt opprettet av B i respons til en hendelse i node A. Node A oppnår sterk data ferskhet for en respons til en hendelse hos B gjennom en tilfeldig verdi N_A (som er et tilfeldig tall tilstrekkelig stort nok til å være uforutsigbart). Node A genererer N_A tilfeldig og sender den med en forespørselsmelding R_A til node B. Den enkleste måten å oppnå sterk ferskhet på, er for B å returnere den tilfeldige verdien med responsmeldingen R_B i en autentisert protokoll. SNEP beskytter dessuten mot lyttere. Dette gjør den ved at lytteren ikke har noe informasjon om klarteksten, selv om lytteren ser flere krypteringer av samme klartekst. En teknikk for å oppnå dette er å benytte tilfeldighet. For å oppnå 2 parts autentisering og dataintegritet, benyttes MAC.

SNEP fungerer bra ved lite pakketap. Hvis en pakke blir borte, så vil telleren til mottaker være mindre enn telleren til senderen, noe som vil føre til at MAC vil være ulike. Mottaker vil så inkrementere telleren sin, og vil så prøve å dekryptere meldingen med en ny teller. Mottaker vil prøve dette noen ganger. Hvis dette feiler flere ganger, så må partene foreta en ny oppsett av telleren ved hjelp av å utveksle meldinger. Dette kan føre til høyt strømforbruk, ettersom flere pakker må sendes for å utveksle en tellerverdi når en eller flere pakker går tapt. Dette fungerer bra i miljøer hvor pakketapet er lavt, men i miljøer hvor pakketapet er høyt, så vil det føre til strømforbruket øker.

3.7.1.2 µTESLA

Funksjonen til µTESLA (Micro Time, Efficient, Streaming, Loss-tolerant Authentication protocol) [2][12][30] er å tilby autentisert kringkasting av data, og protokollen kan benytte RC5. Den sørger for at sinken kan kringkaste autentisert informasjon til alle sensornodene. Protokollen krever at sinken og sensornodene er synkronisert i tid, og at hver node vet en øvre grense for maksimalt synkroniseringsavvik. Sinken genererer en MAC for hver pakke ved hjelp av en hemmelig nøkkel (nøkkelen er hemmelig ved dette tidspunktet). Dette gjør den for å kunne sende en autentisert pakke til en sensornode eller til sinken. En sensornode som mottar en pakke, kan på denne måten verifisere at den korresponderende MAC nøkkelen ikke enda er avslørt av sinken (basert på synkronisert klokke, dens maksimale feil synkronisering, og tidspunkt som nøkkelen er avslørt). Sensornoden som mottar pakken kan på denne måten

være sikker på at bare sender kjenner til MAC nøkkelen, og at ingen motstander har endret pakken underveis. Sensornoden lagrer pakken i buffer. Innen tiden det tar å avsløre nøkkelen, så kringkaster sinken en verifisert nøkkel til alle sensornoder. Når en node mottar den avslørte nøkkelen, kan den verifisere at nøkkelen er korrekt. Hvis nøkkelen er korrekt, kan sensornoden bruke nøkkelen til å autentisere pakken som er lagret i buffer [12]. Hver MAC nøkkel er en nøkkel i en nøkkelring. Nøkkelringen er generert av en felles enveisfunksjon F. Denne enveisnøkkelringen er generert ved at sender velger den siste nøkkelen K_n tilfeldig i nøkkelringen, og for så gjentatte ganger benytte F til å generere alle andre nøkler: $K_i = F(K_i+1)$. Hver sensornode kan enkelt utføre tidssynkronisering, samt kunne hente ut en autentiseringsnøkkel fra nøkkelringen på en sikker og autentisert måte, ved å benytte SNEP. Avsløring av nøkkelen er uavhengig av kringkastingen av pakker, og er bundet til tidsintervallet. µTESLA fungerer slik at sender kringkaster den nåværende nøkkelen periodevis i en spesial pakke.

µTESLA bruker kun symmetrisk mekanismer. Dette fordi bruken av asymmetriske mekanismer i sensornoder er upraktisk, ettersom asymmetriske mekanismer har høy overhead i forbindelse med prosessering, kommunikasjon og lagring.

µTESLA har flere faser for autentisert kringkasting [12]

- Oppsett hos sender
  - Sender genererer en sekvens av hemmelige nøkler: $K_i = H(K_i+1)$.
- Kringkasting av autentiserte pakker.
  - Tiden deles inn i tidsintervall
  - Sender assosierer hver nøkkel i en envis nøkkelring med hvert tidsintervall
  - Sender vil røpe nøkkel K_T etter en forsinkelse av et intervall etter tidsintervall T.
- Autentisering av pakker som kringkastes
  - Sjekker sikkerhetsforhold, for eksempel ved å sjekke om nøkkelen allerede har blitt avslørt. Hvis greit, da bufres pakken. Ellers forkastes den.
  - Når en node mottar en nøkkel fra et tidligere tidsintervall, blir nøkkelens pålitelighet sjekket. Hvis påliteligheten er grei, da kan mottaker autentisere alle pakker i dette tidsintervallet.
- Sensornodens kringkasting av autentisert data
  - Sensornoden kringkaster data gjennom sinken.
  - Noden kringkaster data uansett hvordan sinken oppbevarer den enveis nøkkelringen og sender nøklene til sensornoden som kringkaster når nødvendig.

µTESLA foretar kringkasting på to måter: kringkasting av autentisert informasjon mellom sinken og sensornodene, og kringkasting av autentisert informasjon mellom sensornodene. [7]

3.7.2 TinySec

TinySec [36][40] er en sikkerhetspakke som enkelt kan integreres i en sensornode. TinySec dekker fundamentale sikkerhetskrav. Dette innebærer autentisering, konfidensialitet og integritet av meldinger mellom nabonoder. TinySec er hovedsakelig laget for datalinklaget, og er transparent for applikasjonslaget. Dessuten skal utviklere kunne justere sikkerheten etter som dem selv ønsker. TinySec støtter to forskjellige sikkerhetsmuligheter: autentisert kryptering (TinySec-AE) og bare autentisering (TinySec-Auth). Med autentisert kryptering, krypterer TinySec datalasten og autentiserer pakken med en MAC. MAC mekanismen er

generert over det krypterte dataet og pakkehodet. Ved bare autentisering, autentiseres hele pakken med en MAC, men datalasten blir ikke kryptert.

3.8 Antakelser ved trådløst kommunikasjon

I forbindelse med sikkerheten i trådløse nettverk, og da ikke minst sensornettverk, bør man foreta følgende antakelser. Disse antakelsene har også [12] gjort:

- Trådløs kommunikasjon er i utgangspunktet ikke en sikker måte å foreta kommunikasjon
- Hver sensornode er ikke sikret fysisk
- Hver sensornode har ikke full tillit til andre sensornoder, men hver enkelt har full tillit til seg selv og sinken.
- Bruken av asymmetriske nøkler er upassende for sensornettverket, siden disse teknikkene er for ressurskrevende. Symmetriske krypteringsteknikker er mer aktuelle, men det er upraktisk å bruke mesteparten av de nåværende sikkerhetsalgoritmene. Uansett så kreves det at sikkerheten implementeres med minimal overhead.
- Kompromiss av en sensornode skal ikke utsette de andre sensornodene. Dette er mer ment som et krav til sikkerheten.
- Kompromiss av sinken kan gjøre hele sensornettverket ubrukelig, ettersom alle sensornodene har full tillit til sinken.
- Aggregeringsnoder kan muligens få tillit fra de andre sensornodene. Uansett antaes det at en motstander kan legge ut falske aggregeringsnoder, eller overta gyldige aggregeringspunkter. Det å gi tillit til den, samt sinken er viktig for å oppnå nøkkelhåndtering.
- Vanskelig for sensornettverket å ta i bruk mange av de samme sikkerhetsteknikkene som brukes i stasjonære enheter.
- Det å utføre felles nøkkel utveksling er veldig ressurskrevende, og kan være vanskelig å realisere.

3.9 Del konklusjon

Det er helt klart en del trusler mot et sensornettverk. Derfor er det viktig at man innfører sikkerhet slik at man kan bekjempe disse truslene. De spesielle, karakteristiske begrensningene som foreligger i et sensornettverk gjør det upraktisk å benytte mange av de sikkerhetsprotokollene som eksisterer i dag, ettersom de er designet til å operere med kraftige prosessorer og bedre lagringsplass. På grunn av de store begrensningene i sensornettverket må man være nøye med hvilke kryptografiske teknikker, og sikkerhetsprotokoller som man ønsker å bruke i sensornettverket. Det å innføre sikkerhet i trådløse sensornettverk er fullt mulig, men sikkerheten må velges på bakgrunn av egenskapene og begrensningene til sensornodene. Det kan dessuten være en fordel å begrense sikkerheten ut i fra hva som kreves av sikkerhet. I et sensornettverk er autentisering og integritet veldig viktig å oppnå. Avhengig av hva sensornettverket skal brukes til, og om det er nødvendig å holde data hemmelig for uautoriserte parter, så kan konfidensialitet innføres i form av kryptering.

4 Overhead ved innføring av sikkerhet

For å oppnå en sikker kanal, så er det vanlig at kanalen tilbyr konfidensialitet, dataautentisering, dataintegritet, dataferskhet, og ikke-benektelse. For et sensornettverk er det de 3 første som er mest aktuelle. Det som er interessant er hvordan innføring av sikkerhet påvirker sensornodenes strømforbruk, funksjonalitet og levetid. Derfor bør sikkerheten i et sensornettverk vurderes ut i fra hvor god sikkerhet man har behov for, slik at man heller vurderer sikkerheten ut i fra rimelighet, og på den måten prøver å holde overheaden lavest mulig. Vi skal først se på 2 forskjellige tilfeller med overhead, hvor i det ene tilfellet så innføres sikkerhet på bekostning av datalasten i en melding ettersom header og sikkerhets felt opptar mer plass, og et annet tilfelle der datapakken vokser ettersom header og sikkerhets felt legger til ekstra bytes i datapakken. Dette vil gi en illustrasjon på problematikken som oppstår når sikkerhet skal innføres i et ressursbegrenset sensornettverk. Deretter skal vi ta for oss 2 pakkeformat som kan være passende for et sensornettverk, hvor det ene tar seg av autentisering og integritet, mens det andre tar for seg konfidensialitet. Et sensornettverk kan som sagt ikke benytte seg av tradisjonelle sikkerhetsprotokoller ettersom disse har en tendens til å benytte 16 til 32 bytes til sikkerhet (128 til 256 bit). Sensornettverket må derfor nøye seg med en mindre, kanskje mer usikker variant av sikkerhet. Målet bør da være å redusere kostnadene mest mulig fremfor å oppnå tilsvarende sikkerhet som tradisjonelle nettverk benytter, men derimot en mer redusert variant av sikkerhet. Det er en balanse mellom sikkerhet, ytelse, og strømforbruk, så man bør vurdere sikkerheten ut ifra hvor god sikkerhet man har behov for. Et sensornettverk bør dessuten benytte pakker som er av minst mulig størrelse, noe i likheten av pakkeformatet til TinyOS som er på 36 bytes, inkludert header og datalast.

I tradisjonelle nettverk er det vanlig at autentisering, integritet og konfidensialitet oppnåes gjennom ende til ende sikkerhetsmekanismer, som blant annet SSH [25], SSL/TLS, eller IPSec [4]. Ruterne trenger kun å se på meldingshodet (header), og de trenger ikke aksessere datalasten. Dette er ikke alltid tilfellet i sensornettverk. I et sensornettverk er det vanlig at trafikken rutes fra mange til en, det vil si at trafikken rutes gjennom flere sensornoder før den når sinken, og ofte via sensornoder som opererer som aggregeringsnoder, hvor sensornodene utveksler sensordata via multihopp for å nå en sink. Dessuten kan det hende at nabonoder i et sensornettverk ofte detekterer de samme hendelsene, så hvis hver av disse sensornodene sender pakker til sink, så vil det føre til sløsing av både strøm og båndbredde. For å forhindre disse redundante meldingene, og for å redusere trafikken (spare strøm), så er det vanlig at sensornodene prosesserer data lokalt, samt at de benytter aggregering for å eliminere disse duplikatmeldingene. Dette fører til at sensornoder ofte må aksessere meldingene som mottaes, ved å sammenligne innholdet med eget prosessert data, for så å endre innholdet hvis nødvendig ved hjelp av aggregering. Sensornoder foretar kun aggregering hvis de utfører samme oppgaver. Ofte er det bestemte sensornoder som foretar disse aggregeringsoppgavene, men dette blir avgjort enten før utplassering, eller etter utplassering ved hjelp av sensornettverketes selvorganiserende egenskaper.

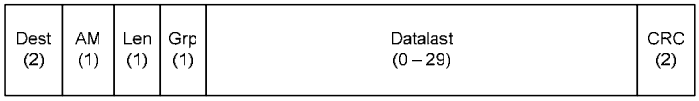
Sikkerheten skal forsikre oss om at all data vi mottar er autentisk, og at både dataets integritet og konfidensialitet er bevart. Men at data er konfidensiell er nødvendigvis ikke det viktigste i et sensornettverkt. I noen sammenhenger det ofte viktigere at data som mottas ikke er modifisert av en motstander, samt at dataet som mottas er fra den riktige senderen (at meldingens integritet er bevart og at senderen er autentisert). Men dette avhenger av hvordan man ønsker å implementere systemet, altså alt avhenger av hva man ønsker og hva man har behov for. For at det skal være lønnsomt å innføre sikkerhet i et sensornettverk, så er det

viktig at kostnadene i form av strømforbruk ved prosessering og ved transmisjon av pakker ikke drastisk degraderer levetiden til sensornettverket. Kostnader som oppstår ved innføring av sikkerhet er: større pakker fører til at radioen bruker lengre tid til å sende en pakke, mens mindre pakker krever hyppigere forsendelser, og at radioen må slåes av og på oftere. I tillegg kommer ekstra kostnader i form av prosessering, og strømforbruket som trengs til kryptografi (dette trenger ikke være så signifikant). Kostnadene vil selvfølgelig avhenge av implementeringen, og hvor stor grad av sikkerhet som innføres.

Vi skal først se på 2 forskjellige scenarioer hvor det antaes at systemet benytter TinyOS sitt pakkeformat [68]. Det første scenarioet tar for seg et system som benytter integritet og autentisering, mens det andre scenarioet tar for seg konfidensialitet i form av kryptering, i tillegg til at integritet og autentisering bevares. Dette kan da implementeres på 2 måter: Enten ved å ta i bruk en fast pakkestørrelse, men som har en variabel datalast. Det vil si at hvis det innføres ekstra overhead, så vil det skje på bekostning av datalasten. Eller så kan det implementeres ved å benytte variabel pakkestørrelse, hvor pakkestørrelsen vokser ettersom overhead innføres, men datalasten forblir den samme. Tilslutt vil vi foreslå et passende pakkeformat for integritet/autentisering, og et for konfidensialitet.

4.1 Scenario 1: Integritet og autentisering

Det er som sagt antatt at systemet benytter et pakkeformat som er tilsvarende likt som det pakkeformatet som TinyOS bruker. Pakkeformatet i TinyOS er på maks 36 bytes, og ser ut som følgende:



Figur 17. TinyOS pakkeformat.

Feltene i pakken har følgende betydning [68]:

- Dest (Destinasjonsadresse)
  - 2 bytes.
  - Inneholder identiteten til sensornodene som befinner seg som neste node.
- AM (Active Message Handler)
  - 1 byte.
  - Fungerer på samme måte som port nummer i TCP/IP, hvor AM spesifiserer en passende måte å hente ut og tolke meldingene hos mottaker. AM bestemmer ut ifra verdiene som er lagt i feltet, hvor på applikasjonslaget meldingene er adressert til.
  - Ved hjelp av AM feltet, så kan en sensornode avvise pakker som ikke har samme AM felt, noe som betyr at sensornodene ikke utfører samme oppgave.
- Len (lengden)
  - 1 byte.
  - Indikerer hvor stor datalasten i meldingene er.
- Grp (Gruppeidentitet)
  - 1 byte
  - Sørger for at flere sensornettverk kan operere i samme område uten å interferere med hverandre, ved at Grp feltet indikerer hvilket sensornettverk pakken er beregnet for. Den kan sees på som en svak aksesskontroll mekanisme.

    - Sensornodene kan kommunisere ved hjelp av multikasting ettersom Grp feltet indikerer hvilket sensornettverk som informasjonen skal adresseres til.
- Data (Datalasten)
    - Datalasten varierer fra 0 til 29 bytes, og inneholder selve data
- CRC (Sjekk sum)
    - 2 bytes
    - Benyttes for å sjekke om en datapakke er blitt endret underveis grunnet feil under transmisjon.

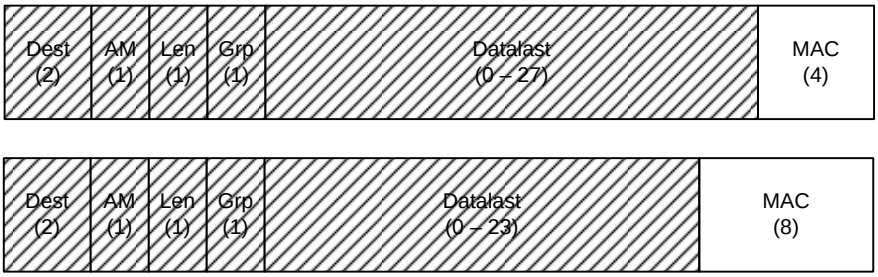
Som pakkeformatet til TinyOS illustrerer, så er ikke TinyOS utstyrt med noen form for sikkerhet. Dette betyr i utgangspunktet at all trafikk er usikker. Det eneste pakken er utstyrt med er en CRC. CRC er en hashfunksjon [41] som produserer en sjekksum av dataet i pakken, slik at den kan detektere om pakken er blitt endret på underveis. Men dette gir ingen form for autentisering, samt at det gir en svak form for integritet. Autentisering kan oppnåes ved hjelp av ren symmetrisk kryptering [4]. Hvis sender og mottar deler samme nøkkel, så kan bare sensornoder med denne gyldige nøkkelen kryptere og dekryptere pakker. Hvis meldingen inneholder en feildeteksjonskode, samt et sekvensnummer, da kan mottaker være sikker på at ingen endringer er foretatt, og at rekkefølgen er riktig. Hvis meldingen i tillegg inneholder et tidsstempel, da kan mottaker være sikker på at meldingen ikke er blitt forsinket utenom det vanlig. Men både sekvensnummer og tidsstempel fører til høyere overhead i datapakken. Med symmetriske nøkler, så er det nøkkelstørrelsen som bestemmer hvor lett det er å knekke nøkkelen. En n bit symmetrisk nøkkel har 2^n muligheter [4]. Større krypteringsnøkkel fører til at krypteringen blir vanskeligere å knekke. I sensornettverket tar vi ikke hensyn til meldingssekvensen, og vi vil heller ikke benytte noen form for tidstempel. Dette for å slippe det ekstra overheadet som er assosiert med disse mekanismene.

For å holde kostnadene i form av overhead lavest mulig er det ønskelig å bytte ut CRC med en kraftigere mekanisme som blant annet [36][51] har foreslått. En MAC mekanisme vil ta for seg både autentisering av meldingen, samt at den bevarer integriteten til meldingen. Denne MAC mekanismen utfører samme funksjon som CRC ved at den genererer en MAC verdi over hele datapakken (både datalasten og header), og hvor denne verdien legges med pakken før transmisjon, og når mottaker mottar pakken, så genererer han på samme måte en egen MAC over datapakken, for så å sammenligne sin egen MAC med den MAC mekanismen som senderen sendte med datapakken. Er de like, så er pakkens integritet bevart. Men MAC mekanismen kan ikke skille om endring i datapakken skyldes en motstander, eller om det skyldes feil under selve transmisjonen. Uansett, så forkastes pakken hvis integriteten ikke er bevart. På denne måten beskytter MAC hele datapakken mot modifisering (hindrer også en motstander i å omadressere en datapakke), inkludert header. Det er vanlig at en slik MAC mekanisme er på 8 til 20 bytes i tradisjonelle nettverk [4]. Et sensornettverk har ikke ressurser til å benytte seg av så lange MAC, så vi vil illustrere bruken av en 4 bytes og en 8 bytes MAC mekanisme til bevaring av meldingsautentisering og meldingsintegritet. MAC mekanismen krever som sagt at både sender og mottaker deler en hemmelig nøkkel. Hvis ikke MAC mekanismen krypteres, så er det ingen garanti for at meldingen er sendt av en autorisert sensornode. Kun meldingsintegriteten vil være bevart. Senderen må derfor i tillegg til å generere en MAC, også kryptere MAC med en hemmelig nøkkel som både sender og mottaker deler. Mottakeren som har samme nøkkel kan på samme måte generere en MAC over hele datapakken, for så å sammenligne sin egen MAC med MAC mekanismen som fulgte med pakken. Hvis de er like, kan mottaker se på pakken som gyldig, ellers vil mottaker forkaste pakken. Sikkerheten ved innføring av MAC er direkte relatert til lengden av MAC [4]. En MAC mekanisme på 4 eller 8 bytes gir ingen god sikkerhet, men tatt sensornettverkets

ressurser i betraktning, så kan ikke et sensornettverk ta i bruk en MAC som er større ettersom overheaden bør holdes lavest mulig. Sannsynligheten for at motstanderen skal kunne finne riktig MAC, så må motstanderen sende ut maksimum 2^n meldinger for å kunne finne den riktige MAC mekanismen. Men for at motstanderen skal kunne finne riktig MAC må han gjøre dette ved å sende pakker til en autorisert sensornode. Ettersom sensornettverket benytter en MAC mekanisme på henholdsvis 4 eller 8 bytes, så vil det gi 2^{32} og 2^{64} mulige kombinasjoner for å finne riktig MAC. Det kan antaes at siden sensornodene ikke vil sende data kontinuerlig og vil i store deler av tiden være i sovemodus, samt at båndbredden i sensornettverket er relativt lav (38400 bps), så vil ikke dette utgjøre såpass stor trussel. I et nettverk hvor båndbredden er høy, for eksempel 11 mb, og hvor data sendes relativt ofte, så vil en MAC på 8 bytes være altfor lite, men dette kan kompenseres med at nøkkelen må bytes ut veldig ofte. Størrelsen på MAC vil variere fra applikasjon til applikasjon, avhengig av hva som er viktig for sensornettverket. Videre skal vi se på scenarioer med ulike pakkestørrelser, og med ulike datalast, både med og uten kryptering.

4.1.1 Fast pakkestørrelse med variabel datalast

Med en fast pakkestørrelse på 36 bytes, så vil ekstra overhead innføres på bekostning av datalasten. Etter innføring av en 4 bytes MAC på bekostning av CRC, så vil datalasten kun reduseres med 2 bytes. Dette utgjør kun 6.9 %, og er akseptabelt. En 8 bytes MAC vil resultere i at datalasten reduseres til 23 bytes, noe som tilsier en reduksjon på 20.7 % i forhold til den opprinnelig datalasten. Figuren nedenfor illustrerer hvordan disse datapakkene vil se ut.

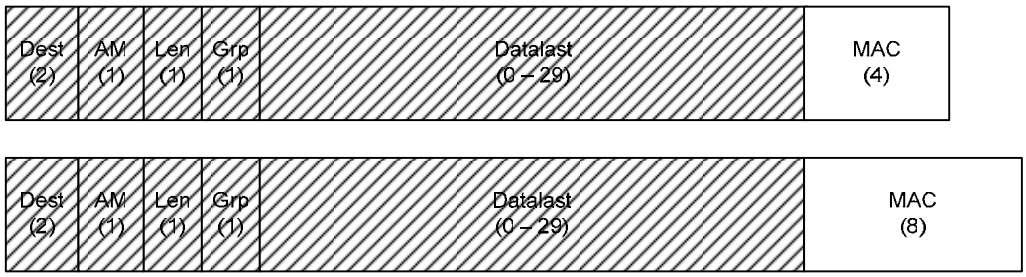


Figur 18. Oversikt over fast pakkestørrelse på 36 bytes.

Fordelen med å benytte små datapakker, som blant annet 36 bytes, er at små pakkestørrelser fører til mindre sannsynlighet for feil på pakkene under transmisjon. De mindre pakkestørrelsene har lavere feilprosent i motsetning til større datapakker. Dette fordi mindre pakkestørrelser har høyere sannsynlighet for å unngå feil i data. Bit feil raten har en tendens med å øke med 10^-3, så desto større pakken er, desto mer utsatt er den for feil [41]. Derfor vil det i mange sammenhenger være en fordel med å ha mindre pakker, ettersom et sensornettverk benytter en udp protokoll for å holde overhead lavest mulig. Siden sensornodene selv foretar prosessering lokalt, så kan det godt tenkes at data som transmitteres er såpass kompakt, at sensornodene ikke trenger å transmittere så altfor mye data på en gang. En fordel med å benytte større pakker er at den faste overheaden blir mindre i forhold til total pakkestørrelsen. Ulempen er at den generelle forsinkelsen for utveksling av informasjon øker, samt at små pakkestørrelser gir dårligere gjennomstrømning, noe som betyr at flere pakke må sendes. Som nevnt tidligere så er det viktig at både pakkestørrelse og sikkerhet velges ut i fra applikasjonen, altså ut i fra hva sensornettverket faktisk skal foreta seg. Dette avhenger av hva man ønsker å implementere, og bør vurderes nøye.

4.1.2 Variabel pakkestørrelse, med fast datalast

Ved å innføre variabel pakkestørrelse, så vil innføring av sikkerhet føre til at datapakken vokser ettersom ulik grad av sikkerhet innføres. Ved å innføre en MAC på enten 4 eller 8 bytes på bekostning av CRC, så har pakkestørrelsen økt til henholdsvis 38 og 42 bytes. Dette tilsvarer en økning av pakkestørrelsen på 5.5 % og 16.7 % i forhold til TinyOS sin pakkestørrelse.



Figur 19. Variabel pakkestørrelse, med datalast på 29 bytes

Selv om meldingsintegriteten i pakkene er bevart, så betyr det ikke at meldingskonfidensialitet er det. Meldingen i seg selv er ikke kryptert, men kun MAC mekanismen er kryptert med en hemmelig nøkkel, noe som betyr at alle kan lese innholdet i pakken. Fordelen med å ikke kryptere datalasten, er å spare kostnader assosiert med å måtte kryptere og dekryptere meldingene hver gang, noe som krever mer prosesseringskraft. Dessuten slipper man den ekstra forsinkelsen som oppstår ved kryptering. Ulempen er at alle kan lese innholdet i pakkene. For å kunne autentisere brukeren, så trenger de kommuniserende partene en sikkerhet om at kun autoriserte sensornoder kan lese pakker. Autentisering kan oppnåes ved at begge benytter den samme krypteringsnøkkelen. Hvis kun sensornode A og B har samme nøkkel, da vet sensornode B at pakken den mottar er fra A, siden kun A kan kryptere data med den hemmelige nøkkelen som sensornode A og B deler. Både A og B må da anta at krypteringsnøkkelen forblir hemmelig, og at en motstander ikke har kompromittert sikkerheten, men det er selvfølgelig ingen garanti for dette. Det vil ikke være behov for å forholde seg til størrelsen på datalasten, siden MAC bare beregnes ut i fra hele pakken, uavhengig av hvordan datapakken er. Hvis blokk chiffer benyttes, så må man forholde seg til eksakte blokkstørrelser, på for eksempel 4, 8, eller 16 bytes. Det vil selvfølgelig gi bedre sikkerhet desto større blokkstørrelse man bruker ettersom variasjonen i mønsteret øker [4]. En annen mulighet kan være å benytte CTS, som krypterer datalasten eksakt i samme størrelse. Eneste kravet CTS har, er at pakken inneholder data på minst 8 bytes (en blokkstørrelse).

4.2 Scenario 2: Konfidensialitet

Vi skal se at overheaden øker gradvis ettersom man innfører kryptering, og hvis man i tillegg også ønsker autentisering og integritet, så vil dette øke overheaden enda mer. Vi skal se på hvordan denne overheaden øker stegvis ettersom vi innfører disse sikkerhetskravene inni datapakkene. I tilfeller der konfidensialitet er viktig, så trenger vi å kryptere datalasten i meldingene. Det finnes 2 måter å kryptere på: link kryptering, eller ende til ende kryptering [4]. Med linkkryptering, er hver sensornode utstyrt med en krypteringsnøkkel slik at all trafikk over alle kommunikasjonsruter er sikret. Dette fører til mange krypteringsnøkler i sensornettverket, men gir god sikkerhet. En negativ ting er at meldingene må dekrypteres hos hver enkelt sensornode hver gang en sensornode mottar en melding. Sikkerheten ligger derfor hos sensornodene. Link krypteringen må designes slik at sensornodene skal kunne dekryptere data ved mottak, samt kunne endre datainnholdet hvis nødvendig. Ved ende til ende

kryptering, så foretaes krypteringsprosessen kun hos ende systemene, altså kun hos sender og hos mottaker. Data sendes uendret gjennom sensornettverket helt til destinasjonsnoden, og de mellomliggende sensornodene trenger ikke dekryptere meldingene, kun videresende dem mot destinasjonen. Destinasjonen deler samme nøkkel som senderen, og kan derfor dekryptere meldingen. Det å benytte ende til ende kryptering mellom sensornodene og sinken i et sensornettverk vil ikke være gunstig siden det er ment at de mellomliggende sensornodene skal kunne ha tilgang til pakkene som er sendt slik at de skal kunne endre innholdet i meldingen hvis nødvendig. Derfor er link til linkkryptering kanskje det beste alternativet. Men dette må vurderes ut i fra hvordan sensornettverket skal operere. Men hvis alle sensornodene benytter samme nøkkel, så kan sensornodene selv avgjøre om de ønsker å dekryptere datapakken, eller om de skal videresende datapakken umiddelbart. Link kryptering foregår på de lavere lagene (lag 1 og 2 i OSI modellen), mens ende til ende kryptering foregår på høyere lag (3, 4, 6 og 7 i OSI modellen) [4].

Innføring av kryptografi vil gi ekstra overhead i datapakken, samt påføre ekstra overhead i prosessoren og minne (RAM). Økt pakkestørrelse kan redusere meldingens gjennomstrømning, samt øke forsinkelsen, men viktigst for sensornettverk er hvordan dette vil påvirke strømforbruket i sensornoden. Avhengig av hva slags kryptering som innføres, så vil kryptering uansett gi ekstra forsinkelse og økt prosessering, noe som vil ha innvirkning på strømforbruket, både med tanke på transmisjon og prosessering [36].

Kryptering uten noen form for autentisering er usikkert, og kryptering alene sikrer ikke meldingsintegritet. Mottaker kan ikke detektere endringer i chifferteksten, og derfor bør MAC mekanismen også benyttes etter at kryptering er innført, slik at meldingen er autentisert, samt at meldingsintegriteten forblir bevart. MAC nøkkelen bør dessuten være forskjellig fra krypteringsnøkkelen. Det er dessuten ønskelig å oppnå semantisk sikkerhet [4][36][42], slik at hvis man krypterer samme klartekst 2 ganger, så får man 2 forskjellige chiffer tekster hver gang. Dette er ønskelig å oppnå slik at en motstander ikke har mulighet til å tilegne seg deler av klarteksten som er kryptert. På denne måten hindrer man at en lytter tilegner seg informasjon om klarteksten selv om motstanderen lytter til flere krypteringer av samme melding. For å oppnå semantisk sikkerhet, så trenger vi en teller i form av en IV, som brukes sammen med krypteringsmekanismen. Hovedpoenget med IV er som sagt å legge til variasjon i krypteringsprosessen når det er lite variasjon i meldingene. IV starter på 0, og inkrementeres så med 1 for hver melding som sendes. IV må opprettholdes for hver destinasjon, så hvis det er mange destinasjoner, så vil dette føre til høyt minneforbruk. Hvis det innenfor hver gruppe vil være begrenset med sensornoder, så er det begrenset hvor stor IV man trenger. IV trenger ikke være sikker, og sendes ofte fullstendig sammen med pakken med krypterte data ettersom mottaker må bruke IV til å dekryptere pakken. IV bør muligens droppes i sensornettverk for å slippe mer overhead, men vi velger altså å illustrere hvordan pakkeformatet endrer seg hvis IV taes med. Formatet til IV kan spesifiseres, og det vanlige er 8 til 16 byte [4], men i et sensornettverk bør denne størrelsen vurderes ut ifra hvilken grad av sikkerhet som er nødvendig. Lengden på IV og måten IV genereres på kan ha en stor betydning på både sikkerhet og ytelse. Hvis IV er for lang, får vi unødvendig mange bits i pakken, noe som igjen fører til mer prosessering og økt strømforbruk. Hvis IV derimot er for kort, så kan vi anta at IV gjentaes fortere, noe som igjen utsetter sikkerheten. Derfor er det viktig å finne en størrelse på IV som er passende for sensornettverket. En IV på n-bit er garantert å gjentaes etter at 2^n + 1 pakker er sendt. IV kan også gjentaes tidligere, hvis for eksempel IV velges som en tilfeldig verdi på n-bit, så kan vi forvente at IV gjentaes etter $2^{n/2}$ pakker har blitt sendt. Derfor kan det være en fordel å bruke en teller som IV, og denne telleren sendes også med pakken slik at mottaker blir informert om verdien til telleren. Det er altså størrelsen på denne tellerverdien

som bestemmer sikkerheten, det vil si at desto kortere IV, desto kortere tid tar det før IV gjentar seg. Konfidensialitet må bevares, og det å gjenbruke IV kan lekke ut informasjon. Med en 4 bytes IV, så vil gjenbruk av IV forekomme etter 2^{32} meldinger fra hver sensornode. Det å øke lengden på IV vil føre til større pakkeoverhead.

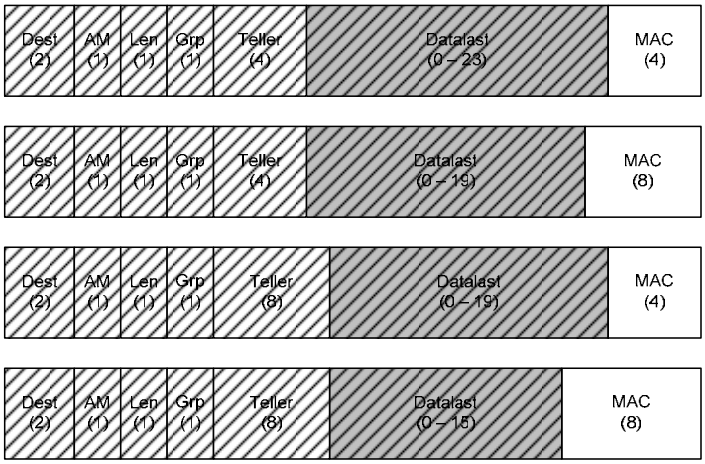
En ulempe med IV, er at hvis hver sensornode genererer IV basert på en teller som starter på 0, vil den første pakken som sendes fra en sensornode gjenbruke den samme IV som sendes i første pakken fra alle andre sensornoder. Derfor er det ofte en fordel å benytte felt i header for å kunne begrense dette. Dette kommer vi tilbake til senere når vi skal foreslå et passende pakkeformat som kan brukes i et sensornettverk. Det å generere IV tilfeldig er derimot en dårlig ide, ettersom bursdagsparadokset gjelder, og man da kan forvente at IV gjentaes etter at $2^{n/2}$ pakker er sendt i sensornettverket. I forhold til strømchiffer hvor gjentakende IV kan avsløre klarteksten til begge meldingene, så vil derimot gjenbruk av IV i CBC kun avsløre lengden (i blokkene) av det lengste delte prefikset av de to meldingene [4].

Til kryptering, så må en symmetrisk krypteringsalgoritme benyttes. Det kan være en fordel å bruke mange av de samme sikkerhetsprimitive, slik at man slipper å implementere så mange forskjellige, noe som bare vil oppta plass. CBC eller CTS kombinert med MAC er en bra kombinasjon siden begge er blokkchiffer. Det er en fordel å benytte den samme blokkchiffer til alle kryptografiske handlinger, dette for å kunne gjenbruke kode ettersom det er begrenset med lagringsplass i en sensornode. Dette er et forslag som [12] har fremstilt. Dessuten kan det være greit å bruke ulike krypteringsnøkler for ulike applikasjoner. Kan på denne måten ta i bruk gruppefeltet i header. For eksempel så kan man benytte blodtrykk, og hjerterytme som to ulike applikasjoner. Dessuten bør man benytte 2 forskjellige nøkler, en til kryptering av datalasten, mens den andre for kryptering av MAC. Det kan benyttes flere alternativer, som blant annet AES, Skipjack og RC5. Ytelsen til disse krypteringsalgoritmene er tilfredsstillende i henhold til [64]. RC5 er en relativt bra krypteringsalgoritme, og den tilbyr god sikkerhet. Den er allerede testet ut i mange sammenhenger av sensornettverk [36][42], samt at egenskapene er passende for sensornettverket. Zigbee [63] benytter derimot AES som krypteringsalgoritme. Det kan også benyttes flere operasjonsmodus, som blant annet Cipher Text Stealing (CTS), eller Cipher Block Chaining (CBC) som er 2 operasjonsmodus som kan benyttes sammen med RC5. CTS er som sagt veldig lik CBC, men fordelen med CTS fremfor CBC er at den håndter klartekst av vilkårlig lengde, og produserer en chiffer tekst som er av samme lengde som klarteksten. Dette er en fordel der datalasten ikke er nøyaktig lik blokkstørrelsen. CTS tillater chiffer teksten til å være av samme størrelse som klarteksten, selv de ikke er av samme blokk størrelse som chiffer. Ved å benytte IV, og CTS eller CBC, så oppnåes semantisk sikkerhet ved at gjentatte mønstre (som blant annet header) ikke oppstår, så lenge IV er unik for hver blokk. Hvis for eksempel datalasten alltid er fast, for eksempel 16, 24 eller 32 bytes, da kan CBC benyttes. Ellers kan det være en fordel med CTS. Vi skal se på hvordan pakken endrer seg med et pakkeformat uten IV, et pakkeformat med en 4 bytes IV, og tilslutt et pakkeformat med en 8 bytes IV. Vi skal se på ulike pakkestørrelser ettersom sikkerhetsoverhead legges til.

4.2.1 Variabel datalast, med fast pakkestørrelse

Ved å innføre kryptering i tillegg til MAC, så vil datalasten i fast pakkestørrelse reduseres yterligere. En pakkestørrelse på 36 bytes som benytter en teller på 4 bytes og en 4 bytes MAC, fører til at datalasten reduseres til 23 bytes, noe som tilsvarer en reduksjon på 20.7 % i forhold til datalasten på 29 bytes som opprinnelig benyttes. En 8 bytes MAC sammen med en 4 bytes IV vil redusere pakken ned til 19 bytes. Det vil den på samme måte gjøre hvis man

innfører en 8 bytes IV sammen med en 4 bytes MAC. Forskjellen er at en 8 bytes MAC gjør det vanskeligere for en motstander å finne riktig MAC fremfor en 4 bytes MAC, samt at det vil gi litt ekstra prosesseringsoverhead, mens en 8 bytes IV fremfor en 4 bytes IV gjør at sensornodene kan sende flere pakker før gjenbruk av IV forekommer. Dette betyr en sikkerhet hvor variasjonen i krypteringen avhenger av størrelsen på IV, og det samme gjelder for størrelsen på MAC. En IV på 8 bytes vil gi en motstander 2^{64} muligheter til å finne riktig IV, noe som gir en og semantisk sikkerhet tatt sensornettverkets båndbredde i betraktning. Pakkestørrelsen vil være 46 bytes for begge pakkestørrelsene, noe som indikerer at datalasten har blitt ytterligere redusert ned til 19 bytes, noe som utgjør en reduksjon på 34.5 %. Det kan også være en mulighet å benytte en 8 bytes IV sammen med en 8 bytes MAC. Dette vil gi en god sikkerhet i et sensornettverk å være, men med en overhead på 21 bytes, så vil det ikke være mye rom for datalasten (Maksimalt 15 bytes) Dette er en reduksjon på 48.3 % i forhold til opprinnelige pakkeformatet til TinyOS, noe som indikerer nesten en halvering av opprinnelig datalast. Det betyr i prinsippet at man må sende 2 pakker for å få gjennom den tilsvarende datalasten som en TinyOS datapakke, noe som indikerer at gjennomstrømningen er halvert. Dette passer fint i tilfeller der det ikke er behov for å sende så mye data, og hvor sikkerhet er viktig. Figuren nedenfor illustrerer hvordan datalasten minsker ettersom man innfører mer sikkerhet i en fast pakkestørrelse på 36 bytes.



Figur 20. Fast pakkestørrelse med variabel datalast.

Tabell 3 oppsummerer hvordan datalasten reduseres ettersom sikkerhetsoverhead legges til. Reduksjonen for integritet og autentisering er også tatt med. Det at datalasten reduseres vil påvirke gjennomstrømningen ettersom datalasten reduseres i forhold til en pakkestørrelse på 36 bytes. Dette vil også ha innvirkning på strømforbruket, hvis det er mye data som skal transmitteres, noe som vil føre til flere transmisjoner. Ettersom pakkestørrelsen er konstant, så vil det ikke ha noen innvirkning på forsinkelsen. Mer om dette i kapittel 5.

Pakkeformat	**Datalasten (bytes)**	**Datalastens Reduksjon (%)**
TinyOS	29	-
Integritet + autentisering	27	6.9
Konfidensialitet (4 bytes IV og MAC)	23	20.7
Konfidensialitet (4 bytes IV og 8 bytes MAC)	19	34.5
Konfidensialitet (8 bytes IV og 4 bytes MAC)	19	34.5
Konfidensialitet (8 bytes IV og MAC)	15	48.3

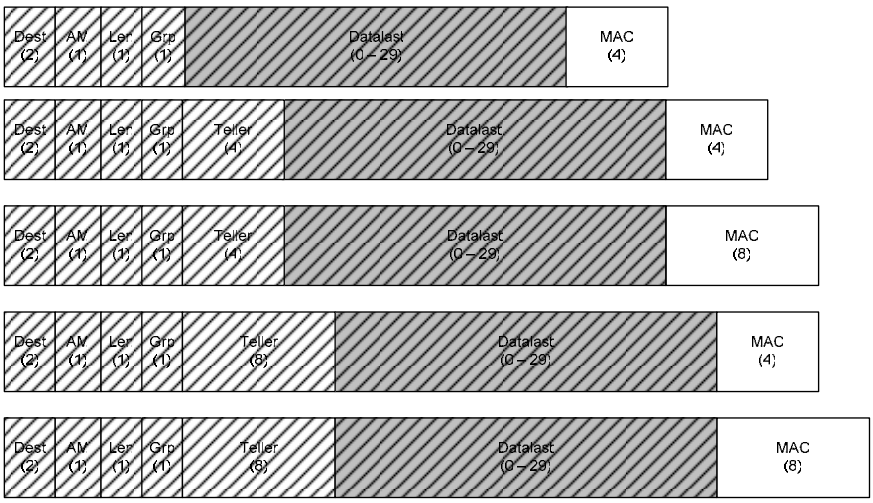
Tabell 3. Hvordan datalasten reduseres i forhold til TinyOS for variabel datalast.

4.2.2 Variabel pakkestørrelse, med fast datalast

Ved å benytte fast datalast, så vil overhead assosiert med sikkerhet føre til at pakkestørrelsen vokser. Ved kun å kryptere datalasten uten å benytte en eller annen form for IV, så slipper man sikkerhetsoverhead i form av IV. Dette krever at en operasjonsmoder som Electronic CodeBook (ECB) benyttes, hvor hver 8 bytes blokk av klartekst data krypteres individuelt med den samme nøkkelen. Dette er en metode som egentlig passer best til å sende enkelt verdier som blant annet krypteringsnøkler [4]. Problemet er at hvis man krypterer 2 helt like blokker med data, så vil ECB generere 2 helt like chiffer blokker, noe en motstander kan utnytte. Et annet problem er at den danner blokker på 8 bytes, og hvis siste blokk ikke er på 8 bytes, så legger den til ekstra bytes for å fylle opp hele blokken, så derfor kan det være en fordel å benytte en datalast slik at datalasten er nøyaktig slik at vi får helt fulle blokker i ECB, for eksempel en datalast på 16, 24, 32 eller 40 bytes (dette har vi ikke tatt hensyn til her). Dette gjør at ECB ikke er passende for variable datalast varianten. For å spare plass i datapakken, så er dette en metode som kan benyttes selv om dette ikke er en sikker måte å sende data på. Her vil vi kun oppnå en overhead i form av en MAC, men det vil uansett oppstå en forsinkelse i form av generering av både MAC og kryptering av datalasten, samt et vist strømforbruk assosiert med krypteringen av MAC og datalasten.

For å oppnå semantisk sikkerhet, så kan IV benyttes sammen med enten CTS eller CBC for å generere variasjon i krypteringen. Ettersom gjenbruk av IV bare er et problem når man gjenbruker samme IV med den samme nøkkelen, så kan vi enten anta at den hemmelige nøkkelen byttes innen den blir gjenbrukt, eller at gjenbruk av IV er akseptabelt. Problemet er at en IV som gjentar seg kan avsløre klarteksten i meldingen. Vi har valgt å se på en IV på 4 bytes og en på 8 bytes. Blokkstørrelsen og IV er alltid like store, noe som i dette tilfellet vil føre til at blokkstørrelsen er på 4 og 8 bytes.

Ettersom en IV på 4 bytes er innført, samt en 4 bytes MAC, så har pakkestørrelsen økt til 42 bytes. Dette gir en økning på 16.7 % i forhold til den opprinnelige størrelsen til TinyOS. Ved å innføre en 8 bytes teller og en 4 bytes MAC, eller en 4 bytes teller og en 8 bytes MAC, så vil pakkestørrelsen øke til 46 bytes, noe som tilsvarer en økning på 27.8 %. En ytterligere økning vil føre til en pakkestørrelse på 50 bytes som benytter en 8 bytes teller og 8 bytes MAC. Dette gir en økning på 38.9 % i forhold til den opprinnelige TinyOS pakkestørrelsen.



Figur 21. Pakkeformatene for variabel pakkestørrelse med fast datalast.

Tabell 4 oppsummerer hvordan pakkestørrelsen øker ettersom sikkerhetsoverhead legges til. Økning for integritet og autentisering er også tatt med. Denne økningen har innvirkning på forsinkelse, gjennomstrømning, og ikke minst, strømforbruket. Mer om dette i kapitel 5.

Pakkeformat	Pakkestørrelse (bytes)	Økning (%)
TinyOS	36	-
Integritet + autentisering	38	5.5
Konfidensialitet (4 bytes IV og MAC)	42	16.7
Konfidensialitet (4 bytes IV og 8 bytes MAC)	46	27.8
Konfidensialitet (8 bytes IV og 4 bytes MAC)	46	27.8
Konfidensialitet (8 bytes IV og MAC)	50	38.9

Tabell 4. Hvordan pakkene vokser i forhold til TinyOS for variabel pakkestørrelse.

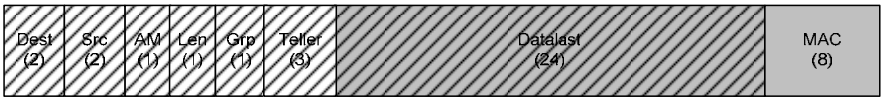
Det blir en del overhead i datapakken ettersom man øker størrelsen på både IV og MAC. Det er derfor viktig at man finner en passende størrelse som kan fungere i et strømbegrenset sensornettverk. Hvis det er nødvendig med IV, så bør denne effektiviseres best mulig. I kapittel 4.3 skal vi se på en mer effektiv variant for beregning av IV.

4.3 Foreslått pakkeformat for autentisering og konfidensialitet

I kapitel 4.1 og 4.2 har vi illustrert problematikken med å innføre sikkerhet i et system med store ressursbegrensninger. Et problem er at pakkeformatet som brukes er relativt lite, noe som gjør at innføring av sikkerhet vil føre til at det ikke blir så mye plass til rent data, eller at datapakken blir for stor i forhold til opprinnelig pakkeformat. En variant som TinySec [36] og 802.15.4 [63] benytter til implementering av IV virker lovende, og i tillegg reduserer den ressursbruken. Det innebærer at tellerverdien som legges med, også bygges på de andre feltene i header. TinySec bruker kun en 2 bytes teller. I tillegg til disse 2 bytene, så benytter den også destinasjonsfeltet (2 bytes), lengdefeltet (1 byte), AMfeltet (1 byte), men også et Src-felt (2 bytes) som indikerer avsender er lagt med. Disse feltene utgjør til sammen 8 bytes, og med en teller på 2 bytes, så kan hver sender og mottaker transmittere maksimalt 2^{16} pakker seg i mellom før gjenbruk av IV oppstår. I motsetning til måten TinySec har gjort det, så mener vi at det kan være en fordel å organisere sensornodene i grupper ettersom sensornettverk kan bli relativt store, hvor hver gruppe har en gruppeleder. Da er det nødvendig å beholde gruppefeltet, noe TinySec har valgt å ignorere. Ved å beholde dette gruppefeltet, så kan TinySec sin variant av IV virke både fornuftig, og ikke minst, effektiv. Det kan være en fordel å innføre en større IV, samt en MAC av tilfredsstillende størrelse, derfor vil vi introdusere en teller på 3 bytes istedenfor en IV på 2 bytes, noe som fort kan vise seg å være for lite. 802.15.4 gjør det på tilsvarende måte som TinySec, men har andre felt i header, samt at IV utgjør 16 bytes, inkludert header. Et problem som oppstår når alle nodene benytter seg av samme nøkkel, er at IV har en tendens til å repetere seg relativt raskt. Dette er en problematikk som har gjort sikkerheten i WEP protokollen dårlig [18]. Det som skjer er at alle nodene bruker samme nøkkel, og at alle pakkene som sendes fra de ulike nodene setter IV til å starte på 0. Det betyr at nettverket kun kan sende 2^{24} pakker før IV repeteres, og hvor fort IV repeteres avhenger av hvor mange noder som eksiterer, og hvor ofte disse nodene sender pakker. For å redusere denne problematikken, så kan det være greit å organisere sensornodene i grupper, hvor hver gruppe benytter en egen krypteringsnøkkel, samt at gruppen har en tilfredsstillende IV. Vi kan danne et rammeverk for et pakkeformat som kan være passende for et sensornettverk. Figur 22 illustrerer et slik format.

Figur 22. Et mulig rammeverk for et pakkeformat som kan benyttes i et sensornettverk.

Verdiene for n vil variere avhengig av hvilket formål sensornettverket har. For eksempel verdier for teller kan variere fra 2 til 8, avhengig av hvor mye trafikk sensornettverket skal foreta, og avhengig om hvor kritisk en repeterende IV er for sensornettverkets sikkerhet. Datalasten kan variere fra 16 til 32 bytes, mens passende verdier for MAC kan være opp til 16 bytes, men denne verdien må også vurderes (helst ikke større enn 8 bytes). Grp- feltet er satt opp som valgfritt, avhengig om man benytter gruppering, eller ikke. Det samme gjelder for Src- feltet. Dette feltet er veldig viktig hvis for eksempel gjensendelse av pakker (Eng: retransmission) benyttes. Figur 23 illustrerer et foreslått pakkeformat som kan brukes for å oppnå konfidensialitet.



Figur 23. Foreslått pakkeformat for konfidensialitet.

Ut i fra pakkeformatet, så har vi en pakkestørrelse på 42 bytes (tabell 4 viser at det er en økning på 16,7 %). Her er IV basert på en teller, samt verdier som ligger i header. Header er på 7 bytes, og ved å innføre en 3 bytes teller, så kan vi bygge en IV på totalt 8 bytes hvor IV baserer seg på Dest-, Src- og Grp- feltene. I realiteten betyr det at en sensornode, som befinner seg innenfor en gruppe, kan sende 2^{24} meldinger til en spesifikk destinasjonsnode. Det er sannsynlig at sensornoder som utfører forskjellige applikasjoner ikke vil være medlemmer av samme gruppe, men dette kan variere fra applikasjon til applikasjon. En slik sikkerhet kan ansees som tilfredsstillende for et sensornettverk, ettersom en sensornode mest sannsynlig vil dø ut før den har sendt 2^{24} meldinger (sensornoden kan da maksimalt sende 16,7 millioner meldinger til samme nabonode før IV er garantert å gjentaes). Hvis det derimot ikke er nødvendig å ha med sender adressen (altså Src- feltet), så kunne disse 2 bytene lagt til i telleren, slik at telleren blir 5 bytes. Fordelen med å ta med Src- feltet er at IV gjelder for kommunikasjonen mellom 2 sensornoder, og ikke i hele sensornettverket. Hvis vi ikke hadde hatt Src- feltet, så kunne hver node i sensornettverket sendt maksimalt 2^{40} meldinger til samme destinasjon før IV ville repeteres, noe det ville ført til at en motstander kunne lytte til trafikken, og fortere kunne funnet like felter i meldingene. Dette fordi det vil sendes mange flere pakker til samme destinasjon, som i dette tilfellet vil være sinken. Uansett hvor stor denne telleren er, så er alle sensornodene nødt til å lagre tellerverdiene til sine nabonoder. Dette opptar lagringsplass, samt at det øker prosesseringskostnadene, så det bør være en maksimal grense for hvor mange nabonoder en sensornode kan ha. Dessuten kan vi se for oss at MAC dannes ut i fra hele pakken, for så at MAC krypteres med en egen nøkkel. Deretter krypteres både datalasten, inkludert MAC, med krypteringsnøkkelen.

Ved å kun benytte integritet og autentisering, så kan man fjerne telleren og Src feltet for å spare plass. Disse utgjør 5 bytes, noe som gir et pakkeformat på 37 bytes. Figur 24 viser pakkeformatet.

Figur 24. Foreslått pakkeformat for integritet og autentisering.

Vi ønsker å sammenligne de foreslåtte pakkeformatene med TinyOS, slik at vi får en oversikt over hvor stor forskjellene utgjør. For å danne et korrekt bilde, så har vi redusert datalasten i TinyOS ned til 24 bytes, slik at datapakkene har lik datalast, samt like header. Det eneste som skiller dem er sikkerhetsoverheaden og det at TinyOS bruker en CRC. Pakkeformatet til TinyOS er nå på 31 bytes totalt. Det resulterer i at forskjellen på pakkestørrelsene nå er på 6 og 11 bytes i forhold til TinyOS. Etter at TinyOS sitt pakkeformat er redusert, ser det slik ut:

Figur 25. TinyOS med 24 bytes datalast.

4.4 Del konklusjon

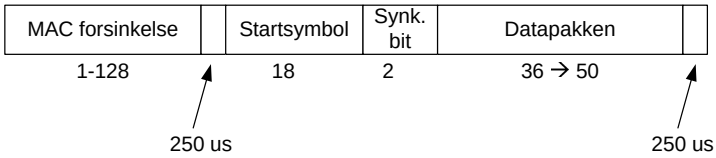
Et sikkerhetssystem bør designes ut i fra systemets krav til sikkerhet. Derfor bør brukeren ha flere moduser å velge mellom ut ifra sikkerhetskravene, slik at ulike nivå av sikkerhet kan velges manuelt. Det kan hende noen systemer er mer interessert i best mulig ytelse, samt lengst mulig levetid, og at sikkerhet ikke er så viktig. Hvis for eksempel krypteringen droppes, så vil dette sannsynlig redusere forsinkelsen, samt føre til bedre ytelse og lengre levetid for sensornodene. Kryptering er kun nødvendig hvis det er aktuelt å holde data hemmelig slik at andre ikke kan lese innholdet i meldingen. Kryptering burde derfor være valgfritt, og bør kun brukes hvis nødvendig, siden det medfører ekstra overhead og forsinkelse, samt økt prosesseringskraft og økt strømforbruk. Autentisering er derimot veldig viktig, slik at nettverket ikke tar imot uautoriserte meldinger som kanskje en motstander selv har tilført i sensornettverket. Det er viktig at sikkerheten i sensornettverket designes slik at det er muligheter for å justere sikkerhetsnivået ut i fra nødvendigheten ettersom det er en balanse mellom sikkerheten, forsinkelsen, og strømforbruket i sensornettverket. For eksempel bør en kunne justere sikkerhetsparameterne ut i fra de sikkerhetskravene et system krever. Derfor kan det være greit å ha flere moduser å velge ut ifra hvilken sikkerhet som er ønskelig. Følgende moduser kan være aktuelle: ingen kryptering, integritet og autentisering, samt konfidensialitet (inkludert integritet og autentisering), hvor man kan velge mellom ulike grader av integritet og autentisering, og konfidensialitet på bekostning av ekstra overhead i datapakken.

5 Resultater ved innføring av sikkerhet

Vi skal se på resultatet av innføring av sikkerhet i sensornettverket ut i fra de kriteriene og antakelsene vi brukte i kapitel 4. Vi skal se på hvordan overhead påvirker datalasten, gjennomstrømning, hvordan transmisjonstiden øker, og hvordan overheaden påvirker strømforbruket. Til slutt skal vi også kartlegge hvor god sikkerhet man oppnår. Vi kan se for oss et system som bruker tilsvarende spesifikasjoner som er blitt benyttet i Mica2 mote sensornodene [46]. Disse sensornodene er veldig populære å bruke. Radioen i Mica2 mote er av type CC1000 [29][46], og denne radioen benytter CSMA for å kontrollere aksessen mot mediet.

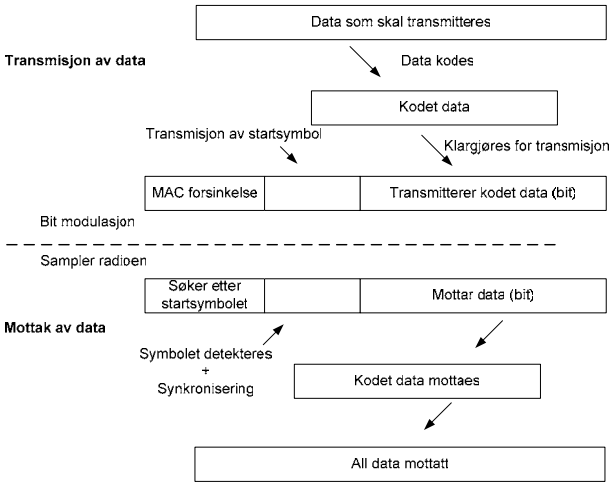
Konsekvensene med å øke pakkestørrelsen er at lengre pakker forbruker mer båndbredde ettersom det tar lengre tid å sende dem, samt at de gir økt forsinkelse på grunn av at kommunikasjonskanalen er relativ treg [36]. Dette vil føre til økt strømforbruket, fordi radioen må være slått på lengre når større pakker sendes. Konsekvensene ved variabel datalast, og fast pakkestørrelse, er at det fører til at radioen må slåes av og på hyppigere, ettersom flere pakker må sendes for å få gjennom alt data. I tillegg til pakkeformatene som har blitt analysert, så vil det ved kommunikasjon komme ekstra overhead i form av startsymbol, samt synkroniseringsbytes [33]. Dette startsymbolet er på 18 bytes, og i tillegg kommer ekstra 2 synkroniseringsbytes. Dette vil igjen øke overhead assosiert med å sende en pakke av en gitt størrelse. Hvis vi har en fast meldingsstørrelse på 36 bytes, så vil MAC rammen være på 56 bytes (36+18+2). Startsymbolet er det første som sendes ut på linken, og er nødvendig slik at radioen kan detektere om det er aktivitet på linken. I tillegg skal startsymbolet signalere de andre sensornodene om at en datapakke skal sendes, slik at de andre sensornodene kan beregne hvor lang tid det vil ta og transmittere denne pakken. Rett etter startsymbolet og synkroniseringsbyten, så følger datapakken som består av datalasten og overhead assosiert med datapakken. Hvis ikke synkroniseringen opprettholdes vil det føre til at data vil bli korrupte, noe som fører til en mislykket transmisjon. Mottaker som overvåker kanalen, vil ved hjelp av startsymbolet få beskjed om at en transmisjon skal påbegynne. For at mottaker i det hele tatt skal kunne detektere dette startsymbolet, så må mottaker sample kanalen med en hastighet som er dobbelt så rask som chipraten [33]. Vi kan se for oss at totalt data som transmitteres over linken ser slik som i figur 26. Figur 27 viser en oversikt over transmisjonsprosessen.

Figur 26. Illustrasjon over all overhead assosiert med å sende eller motta en pakke.



Figur 27. Selve transmisjonsprosessen for å transmittere 1 datapakke.

Figur 27 illustrerer at Mica2 bruker 250 µsekunder for å bytte mellom transmisjons- og mottaker- modus. Figuren viser også at transmisjonsprosessen for en pakke med fast pakkestørrelse på 36 bytes, pluss startsymbol på 18 bytes og synkroniseringsdelen på 2 bytes, utgjør 56 bytes totalt. En stegvis prosess kan illustreres slik:



Figur 28. Transmisjon og mottak av data steg for steg.

Figur 28 viser en stegvis hvordan kommunikasjonsprosessen foregår. Ideen er hentet fra [59]. Først kodes data før transmisjon. Under transmisjonen må først sender vente til kanalen er ledig, så sendes startsymbolet for å signalere de andre sensornodene om at en transmisjon skal starte. Mottakelse av data starter ved at startsymbolet detekteres, og for i det hele tatt kunne detektere startsymbolet, så må kanalen overvåkes. For å bevare strømkapasiteten lengst mulig, så må kanalen samples periodevis fremfor å sample kontinuerlig. Startsymbolet indikerer starten på pakken, og tilbyr timing informasjon for pakken. Ved mottak av startsymbolet må mottaker prøve å beregne den nøyaktige timingen til pakken, slik at mottaker kan sample datalasten på midten av hver bit helt til slutten av pakken. Startsymbolet etterfølges så av synkroniserings bytesene, deretter data. Viktig at transmisjon foregår med stor nøyaktighet slik at det ikke oppstår feil i transmisjonen grunnet dårlig synkronisering. MAC forsinkelsen som er illustrert er perioden hvor en sensornode lytter om kanalen er ledig før den sender. Mottaker får beskjed om at transmisjonen snart starter ved at den mottar startsymbolet. Ved mottak, så detekteres først startsymbolet. For at mottaker skal klare å skille startsymbolet fra støyet, så må mottaker sample kanalen med en hastighet som er dobbelt så rask som selve transmisjonshastigheten. Etter at mottaker har detektert startsymbolet, må mottaker synkronisere seg med den nøyaktige fasen til den innkommende transmisjonen. Nøyaktig hvordan samplingen og synkroniseringsprosessen foregår vil vi ikke gå mer innpå, men dette kan leses mer om i [59].

5.1 Overhead

Ved å innføre sikkerhet i et sensornettverk så forekommer det ekstra sikkerhetsoverhead i datapakken. Dette resulterte i en økning i pakkestørrelse i forhold til opprinnelig TinyOS pakkestørrelse for variabel pakkestørrelse, mens for fast pakkestørrelse så ble datalasten redusert ettersom sikkerhetsoverhead ble innført på bekostning av datalasten. Det første vil som sagt føre til høyere forsinkelse som et resultat av overhead. Forsinkelsen er tiden fra en sensornode sender en pakke fra applikasjonslaget til enten en sensornode eller sinken mottar pakken på applikasjonslaget (skal se mer på dette i kapitel 6). Tabellene nedenfor viser hvordan sikkerhetsoverheaden påvirker pakkeformatet i hvert av tilfellene, altså for variabel pakkestørrelse, for fast pakkestørrelse, og for foreslått pakkeformat. Det som er interessant, er hvordan datalasten endres i forhold til antall bytes som sendes. Dette er også noe som vil påvirke gjennomstrømningen, noe vi skal se på senere.

Pakkeformat (variabel)	Pakkestørrelse (bytes)	Datalasten (bytes)	Ekstra Overhead (bytes)	Datalasten Utgjør (%)
TinyOS	36	29	20	51,8
Integritet + autentisering	38	29	20	50,0
Konfidensialitet (4 bytes IV, 4 bytes MAC)	42	29	20	46,8
Konfidensialitet (4 bytes IV, 8 bytes MAC)	46	29	20	43,9
Konfidensialitet (8 bytes IV, 4 bytes MAC)	46	29	20	43,9
Konfidensialitet (8 bytes IV, 8 bytes MAC)	50	29	20	41,4

Tabell 5. Hvor mye datalasten utgjør i forhold til alt data som sendes for variabelt pakkeformet.

Pakkeformat (fast)	Pakkestørrelse (bytes)	Datalasten (bytes)	Ekstra Overhead (bytes)	Datalasten Utgjør (%)
TinyOS	36	29	20	51,8
Integritet + autentisering	36	27	20	48,2
Konfidensialitet (4 bytes IV, 4 bytes MAC)	36	23	20	41,1
Konfidensialitet (4 bytes IV, 8 bytes MAC)	36	19	20	33,9
Konfidensialitet (8 bytes IV, 4 bytes MAC)	36	19	20	33,9
Konfidensialitet (8 bytes IV, 8 bytes MAC)	36	15	20	26,8

Tabell 6. Hvor mye datalasten utgjør i forhold til alt data som sendes for fast pakkeformat.

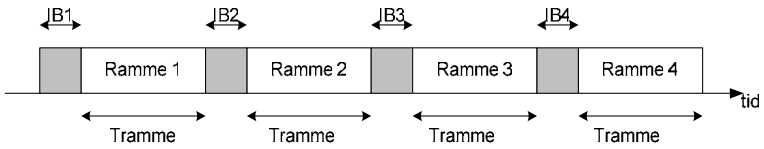
Pakkeformat (foreslått)	Pakkestørrelse (bytes)	Datalasten (bytes)	Ekstra Overhead (bytes)	Datalasten Utgjør (%)
Integritet + autentisering (8 bytes MAC)	37	24	20	42,1
Konfidensialitet (3 bytes IV, 8 bytes MAC)	42	24	20	38,7

Tabell 7. Hvor mye datalasten utgjør i forhold til alt data som sendes for foreslått pakkeformat.

Tabellene viser at desto mer overhead som innføres i sensornettverket, desto mindre data får man igjennom i forhold til det som sendes. Det betyr at hvis man har mye data å sende, så vil man måtte sende mange flere pakker for å gjennom ønsket data. Vi skal også se på hvordan dette påvirker gjennomstrømningen, transmisjonstiden, og strømforbruket.

5.2 Gjennomstrømning

Gjennomstrømningen sier noe om hvor mye data som kommer igjennom av det totale data som sendes. Gjennomstrømningen finner vi ved dividere datalasten med rammeoverføringstiden. Rammeoverføringstiden er totalpakkestørrelse dividert med overføringshastigheten. I tillegg så kommer også forsinkelsen assosiert med medium aksesskontrollen inn. Vi skal se på en medium aksessprotokoll som er beregnet for sensornettverk, nemlig S-MAC (Sensor MAC). S-MAC [6] utfører periodevis lytting og soving. Varigheten til disse periodene kan justeres av brukeren, og er avhengig av hva slags type applikasjon sensornettverket utfører. Gjennomstrømningen finner man ved å inkludere multippel aksess metoden. Det er antatt at CSMA/CA er den mest aktuelle multippel aksess metoden for sensornettverk. CSMA/CA protokollen ser slik ut for et sensornettverk:



Figur 29. Illustrasjon over hvordan Sensor medium aksesskontroll fungerer.

Figur 29 [49] viser kommunikasjonen mellom kun en sender og en mottaker. Det vil si at etter initial backoff perioden, så sendes pakken umiddelbart. Med S-MAC får man en balanse mellom forsinkelsen og strømforbruk. En slik protokoll kan føre til at båndbredde ikke deles rettferdig mellom sensornodene, ved at noen sensornoder får sende ofte, mens andre aldri får sende. Derimot er en slik protokoll strømeffektiv ettersom protokollen har byttet ut ledigmodus med sovemodus, ved at radioen slåes av når sensornoden er i sovemodus. Driftssyklusen kan dessuten variere fra applikasjon til applikasjon, for eksempel ved høy belastning kan driftssyklusen være så høy som 10 %, mens ved lav belastning, så kan driftssyklusen være mindre enn 1 %. Ved disse driftssyklusene så vil radioen være slått av i 90 %, samt i mer enn 99 % av tiden av tiden. Dette for vil bevare levetiden til sensornoden lengst mulig. Som sagt så benytter Chipcon Radio [29] CSMA/CA multippel aksess metode, og har båndbredde på 38400 bps (Mica2 mote). CSMA/CA mekanismen er modifisert slik at den er mer passende for sensornettverk som benytter forbindelsesløse protokoller. Som alle andre trådløse enheter, så opererer Chipcon også i half dupleks, noe som gjør at den ikke kan detektere kollisjoner under transmisjon, men kun prøve å unngå dem. For å unngå kollisjoner benyttes en Initial backoff mekanisme. Denne ligger i størrelsesorden 7,5 ms til 34.2 ms for en pakkestørrelse på 36 bytes, og benyttes av sensornoder til å sjekke om radioen er ledig. Hvis kanalen er ledig, da kan sensornodene sende, hvis kanalen er opptatt, da må sensornodene vente en viss tid. I våre tilfeller vil initial backoff perioden variere avhengig av pakkestørrelsene. Minimum initial backoff vil variere fra 6,5 ms for en pakkestørrelse på 31 bytes, og 10,4 ms for en pakkestørrelse på 50 bytes. Maksimum initial backoff vil derimot variere fra 33,1 ms for en pakkestørrelse på 31 bytes, og 37,1 ms for en pakkestørrelse på 50 bytes. Initial backoff verdiene fant vi på følgende måte:

$$IBmin = \frac{\text{Min MAC forsinkelse + pakkestørrelse (bits)}}{\text{Båndbredden (bps)}}$$

$$IBmax = \frac{\text{Max MAC forsinkelse + pakkestørrelse (bits)}}{\text{Båndbredden (bps)}}$$

Formel 1. Formel for beregning av minimal og maksimal initial backoff.

Hvis vi tar for oss gjennomsnittlig backoff periode, så kan vi finne gjennomstrømningen ved å ta datalasten dividert med tiden som trengs for å sende en total pakke addert med gjennomsnittlig backoff periode. Først finner vi rammeoverføringstiden. Altså hvor lang tid det tar å sende en hel ramme. Deretter finner vi gjennomsnittlig backoff. Til slutt dividerer vi datalasten med rammeroverføringstiden addert med gjennomsnittlig backoff periode. Formlene er oppsummert under:

$$\text{Gjennomstrømning} = \frac{\text{Datalasten (bit)}}{\text{Rammeoverføringstiden (sek)} + \frac{\text{IBmin} + \text{IBmax}}{2}}$$

$$\text{Rammeoverføringstiden (sek)} = \frac{\text{Total pakkestørrelse (bit)}}{\text{Båndbredde (bps)}}$$

Formel 2. Gjennomstrømning og rammeoverføringstiden.

Ved vanlig CSMA/CA, så benyttes også en DIFS som er en tilfeldig tid som sensornodene skal vente når kanal er opptatt. Gjennomstrømningsformelen her gjelder kun når det er en sender og en mottaker [49], så derfor kan vi ignorert DIFS ettersom ingen andre enn disse 2 sensornoden vil kunne sende. Vi har også inkludert den ekstra overheaden assosiert med å sende en datapakke i beregningen av rammeoverføringstiden.

Tabellene 8, 9 og 10 viser en oversikt over gjennomstrømning med hensyn på CSMA/CA.

Pakkeformat (variabel)	**Pakke størrelse (bytes)**	**Data lasten (bytes)**	**Ekstra overhead (bytes)**	**Båndbredde (bps)**	**Ramme overføringstid (ms)**	**Gjennom-strømning (bps)**
TinyOS	36	29	20	38400	11,7	7127,5
Integritet + autentisering	38	29	20	38400	12,1	6956,5
Konfidensialitet (4 bytes IV, 4 bytes MAC)	42	29	20	38400	12,9	6628,6
Konfidensialitet (4 bytes IV, 8 bytes MAC)	46	29	20	38400	13,8	6312,9
Konfidensialitet (8 bytes IV, 4 bytes MAC)	46	29	20	38400	13,8	6312,9
Konfidensialitet (8bytes MAC, 8 bytes IV)	50	29	20	38400	14,6	6049,5

Tabell 8. Oversikt over gjennomstrømning for variabel pakkestørrelse.

Pakkeformat (fast)	**Pakke størrelse (bytes)**	**Data lasten (bytes)**	**Ekstra overhead (bytes)**	**Båndbredde (bps)**	**Ramme overføringstid (ms)**	**Gjennom-strømning (bps)**
TinyOS	36	29	20	38400	11,7	7127,5
Integritet + autentisering	36	27	20	38400	11,7	6476,8
Konfidensialitet (4 bytes IV, 4 bytes MAC)	36	23	20	38400	11,7	5257,1
Konfidensialitet (4 bytes IV, 8 bytes MAC)	36	19	20	38400	11,7	4136,1
Konfidensialitet (8 bytes IV, 4 bytes MAC)	36	19	20	38400	11,7	4136,1
Konfidensialitet (8 bytes MAC, 8 bytes IV)	36	15	20	38400	11,7	3129,1

Tabell 9. Oversikt over gjennomstrømningen for fast pakkestørrelse.

Pakkeformat (Foreslått)	Pakke størrelse (bytes)	Data lasten (bytes)	Ekstra overhead (bytes)	Båndbredde (bps)	Ramme overføringstid (ms)	Gjennom-Strømning (bps)
TinyOS	31	24	20	38400	10,6	6315,8
Integritet + autentisering (8 bytes MAC)	37	24	20	38400	11,9	5827,0
Konfidensialitet (3 bytes IV, 8 bytes MAC)	42	24	20	38400	12,9	5485,7

Tabell 10. Oversikt over gjennomstrømning for foreslått pakkestørrelse.

Tabellene viser at desto mindre data man sender i forhold til den totale datapakken, desto lavere blir gjennomstrømningen. Dette er et problem med å innføre sikkerhetsoverhead i et sensornettverk som benytter såpass små datapakker. Det kan fort forekomme at overheaden i datapakken er av samme størrelse, eller kanskje større enn selve data som skal sendes. Dette gjør at det kreves at flere transmisjoner må foretaes for å få gjennom all data, hvis overhead er høyt. Derfor er det viktig at man benytter et passende pakkeformat i sensornettverket.

5.3 Økning i transmisjonstid

Ettersom sikkerhetsoverheaden innføres, så vil pakkestørrelsen øke. Det som vil være aktuelt her er pakkestørrelsen av variabel lengde ettersom pakkestørrelsen med fast lengde ikke vil ha noen form for pakkevekst i forhold til overhead. Det vil gå mer ut over gjennomstrømningen. Tabellene 11 og 12 viser en total oversikt over hvordan transmisjonstiden øker ettersom overhead assosiert med innføring av sikkerhet øker. Vi kan regne ut denne økningen ved å ta tiden det tar å sende en ramme basert på TinyOS, og sammenligne denne tiden mot tiden for de andre rammene. I denne beregningen har vi også tatt med den ekstra overheaden assosiert med startsymbolet og synkroniseringsbit.

Pakkeformat (variabel)	Pakke størrelse (bytes)	Data lasten (bytes)	Ekstra overhead (bytes)	Båndbredde (bps)	Ramme overføringstid (ms)	Økning i Transmisjonstid (%)
TinyOS	36	29	20	38400	11,7	0,0
Integritet + autentisering	38	29	20	38400	12,1	3,6
Konfidensialitet (4 bytes IV, 4 bytes MAC)	42	29	20	38400	12,9	10,7
Konfidensialitet (4 bytes IV, 8 bytes MAC)	46	29	20	38400	13,8	17,9
Konfidensialitet (8 bytes IV, 4 bytes MAC)	46	29	20	38400	13,8	17,9
Konfidensialitet (8 bytes MAC og IV)	50	29	20	38400	14,6	25

Tabell 11 Økning i transmisjonstid grunnet overhead for variabelt pakkeformat.

Pakkeformat (Foreslått)	Pakke størrelse (bytes)	Data lasten (bytes)	Ekstra overhead (bytes)	Båndbredde (bps)	Ramme overføringstid (ms)	Økning i Transmisjonstid (%)
TinyOS	31	24	20	38400	10,6	0,0
Integritet + autentisering	37	24	20	38400	11,9	11,7
Konfidensialitet (3 bytes IV, 8 bytes MAC)	42	24	20	38400	12,9	21,5

Tabell 12. Økning i transmisjonstid grunnet ekstra overhead.

Ut ifra tabell 11 og 12 så ser man at ettersom mer overhead innføres, så vil dette påvirke hvor lang tid det tar å sende eller motta en pakke. Det at rammeoverføringstiden øker er altså det største problemet ved innføring av sikkerhet, ettersom det er under transmisjon at strømforbruket er høyest, noe som vi skal se på senere. For variabel pakkestørrelse, så varierer denne økningen fra 3,4 til 24,8 % i forhold til det å sende en datapakke på 56 bytes, mens for foreslått pakkeformat så varierer den økningen med 12,3 til 21,7 % i forhold til et TinyOS format med en datalast på 24 bytes. Dette vil gi et mer korrekt økningsforhold ettersom datalasten, og header er like, og det eneste som skiller pakkeformatene er sikkerhetsoverheaden.

5.4 Strømforbruket

Trådløs kommunikasjon er beryktet for å være relativt uforutsigbar ettersom kvaliteten avhenger av miljøet som kommunikasjonen foregår i. Faktorer som spiller inn er hvilken frekvens kommunikasjonen foregår over, hva slags modulasjonsskjema som benyttes, og hva slags type enheter som kommuniserer sammen. Kvaliteten på kommunikasjonen kan variere på grunn av avstander. Dette gjelder også for sensornettverk, kanskje i en enda større grad ettersom sensornodene har mye større ressursbegrensninger. Komponentene i en sensornode som bruker strøm er hovedsakelig som nevnt tidligere, prosessoren, radioen, og sensoren. Sensornettverk bruker transceiverer som er designet for lavt strømforbruk i et multihopp miljø [47]. Dessuten kan multihopp være mer strømbesparende enn ett enkelt hopp som strekker seg over en lengre distanse. Dette skal vi test ut ved hjelp av simulering i kapitel 6.

For å kartlegge strømforbruket i et sensornettverk, så må vi se på strømforbruket over tid. Strømforbruket vil naturligvis øke ettersom ulik grad av sikkerhet fører til mer overhead. Det vil kreve mer prosessering for å ta seg av kryptering og dekryptering av datalasten og MAC. Strømforbruket vil også øke ved forsendelse av data grunnet sikkerhetsoverhead i form av IV og MAC. Dessuten er det viktig å fullføre blokkchiffer operasjonene raskt, ettersom de kryptografiske operasjonene overlappes med radiooperasjonene. Hvis chiffer operasjonen ikke blir fullført i tide, da vil ikke data som trengs for radio bli tilgjengelig i tide [36]. Raskere blokkchiffer forbruker dessuten mindre strøm. Blokk chiffer vil bruke prosessoren hyppig, noe som vil føre til strømtap mens pakken blir kryptert. Det er forventet at prosesseringsperioden er lengre når en benytter kryptering, ettersom det er flere blokkchiffer operasjoner. Det vil ta lengre tid å sende en pakke ettersom prosessen tar lengre tid ved at pakken først må krypteres, for så å bruke lengre tid ved transmisjon ettersom pakkestørrelsen har økt, eller at det krever flere forsendelser ettersom pakkestørrelsen har minket.

Øking av pakkestørrelsen vil som vi har sett, redusere gjennomstrømningen, samt øke transmisjonstiden, noe som igjen påvirker forsinkelsen. I tillegg vil også alt dette påvirke strømforbruket. Større pakker koster mer, for eksempel ved at de reduserer båndbredden, samt at de fører til høyere forsinkelse ettersom kommunikasjonskanalen er relativ treg. I tillegg

fører det til høyere strømforbruk, fordi radioen må være slått på over lengre tid når den sendere større pakker. Desto lengre radioen forblir på, desto mer strøm forbrukes. For å minimalisere dette strømforbruket, så må man minimere perioden radioen er på, og maksimere båndbredden. Vi antar en Mica2 mote med en båndbredde på 38400 bps. Mindre pakker med lav datalast krever at det sendes flere pakker for å få gjennom en viss datamengde. Dette krever at radioen må slåes av og på oftere, noe som igjen øker strømforbruket. Dessuten gir det lavere gjennomstrømning.

Vi kan se for oss at sensornettverket benytter sensornoder med tilsvarende spesifikasjon som Mica2 mote sensornoden. Hver sensornode benytter 2 AA batterier. Vi kan for eksempel anta at sensornoden er utstyrt med 2 Energizer oppladbare batterier [27], med en batterikapasitet på 2500 mAh pr batteri. Det vil si totalt 5000 mAh. Ah (Ampere per hour) indikerer hvor stor kapasitet et batteri har målt i timer. Desto høyere verdi, desto lengre levetid har batteriet. I en sensornode med 2 batterier, avhenger spenningen av hvordan batteriene er plassert. Hvis de kobles sammen serielt, da får vi høyere spenning (2 batterier à 1.5 V vil til sammen produsere 3 V, men en samlet batterikapasitet på 2500 mAh), men hvis parallell kobling benyttes, da får vi høyere strømkapasitet (Altså dobbelt så mye som ved et enkelt batteri, det vil si 5000 mAh, men med en spenning på 1.5 V). Enheten Ah indikerer hvor mye strøm i ampere et batteri maksimalt kan produsere i løpet av en time. Et batteri er heller ikke lineært, det betyr at et 500 mAh batteri kan ikke produsere 1000 mAh på 30 min. Ved å benytte milliamper pr time, så får man et tilnærmet estimat på hvor lenge batteriet lever under en gitt belastning. De viktigste faktorene som påvirker batteriets levetid er utladningsstrømmen eller mengden av nåværende tap et batteri opplever. Levetiden til et batteri finner vi ved å dividere den maksimale batterikapasiteten med nåværende utladningsstrøm. Hvis for eksempel en sensornode utstyres med 2 batterier som hver har en strømkapasitet på 2500 mAh per batteri, så kan batteriene kjøres serielt i 3 Volt, noe som tilsier at batteriene har en total kapasitet på 2500 mAh ved 3 volt. Sensornodene har komponenter som for det meste opererer på 2,7 Volt, så vi kan anta at den totale strømkapasiteten i en sensornode er på 2500 mAh. Senere i rapporten vil vi også operere med enhetene Watt/h og Joule/s. For å konvertere Ah om til disse verdiene, så tar vi å multipliserer total batterikapasiteten på 2500 mAh med voltstyrken, noe som gir oss 7500 mWatt/h. Joule er egentlig Watt per sekund, men vi velger å operere i timer, så da multipliserer vi antall sekunder per time, som er 3600 sekunder, med total batterikapasitet som er 7500 mWatt/h. Dette gir en total batterikapasitet på 27000 Joule per sekund.

For å beregne levetiden til et batteri, så kan formel 2 benyttes [53].

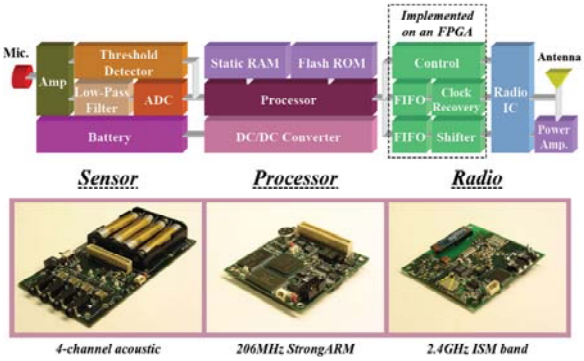
$$\text{Estimert levetid (timer)} = \frac{\text{Batterikapasitet (mAh)}}{\text{Utladningsstrøm (mA)}}$$

Formel 3. Estimert levetid for et batteri.

For at en sensornode av typen Mica2 mote skal kunne ha en levetid på 1 år, så må sensornoden ha et gjennomsnittelig strømforbruk på 285 µA per time. Dette er relativt lite, så for å maksimere levetiden, må sensornoden benytte driftssykluser som er relativt korte, ved at sensornoden slåes av eller settes i sovemodus så lenge den ikke sender, mottar, eller prosesserer data. Hvis sensornodene er i et overvåket miljø, så kan man anta at sensornodene enten kan lades, eller at batteriene kan skiftes ut. Hvis hver enkelt sensornode har en spesifikk oppgave, så er det muligens ikke nødvendig å innføre dataaggregering. Det vil si at hver

enkelt sensornode som mottar data ikke trenger å dekryptere data for å lese det. Den trenger bare å videresende data mot destinasjonen.

Først skal vi se hvordan sikkerhetsoverheaden øker strømforbruket ved transmisjon ettersom mer og mer overhead legges til. I kapittel 6 skal vi se på flere scenarioer hvor vi tar for oss flere tilfeller, blant annet en driftssyklus, hvor sensornoden periodevis er i sovemodus og i aktivmodus. For å kartlegge strømforbruket i en sensornode må vi ta for oss hver komponent i en sensornode, og kartlegge hvor stort strømforbruk hver enkelt av disse komponentene forbruker, for eksempel per driftssyklus, eller per transmisjon. Figur 30 viser en mulig oppbygging av en sensornode, og dens komponenter. Figuren [72] viser altså hva slags komponenter de ulike komponentene benytter for å kunne operere:

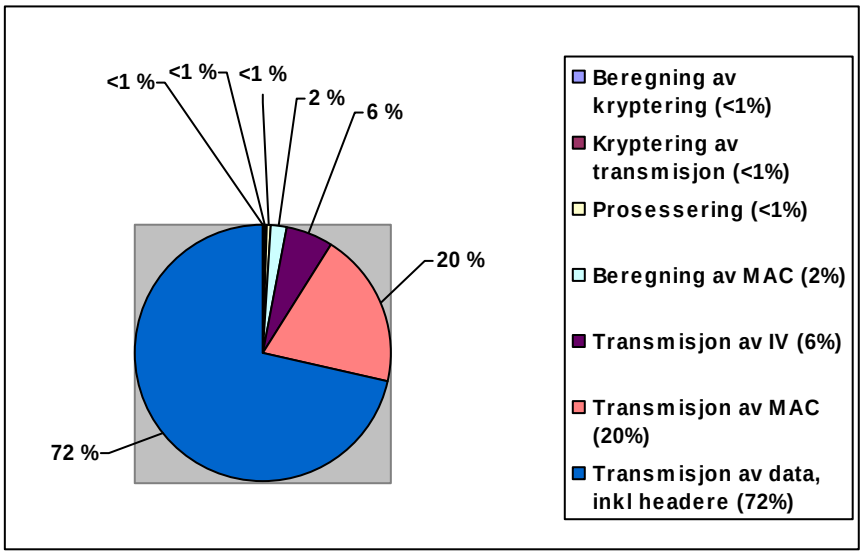


Figur 30. Sensornodens oppbygging.

Vi skal først og fremst ta for oss strømforbruket som oppstår i radioen, og i sensoren. I tillegg skal vi regne ut en antatt verdi som gjelder for prosesseringen, og andre komponenter som bruker strøm i sensornoden. I radioen er det som sagt flere ting som spiller inn på strømforbruket: type modulasjon, datarate, sendeeffekt (avgjør transisjonsrekkevidde) og driftssyklusen (Eng: duty cycle). Radioen kan operere i flere moduser: sendemodus, mottakermodus, ledigmodus, og sovemodus. Det er som sagt viktig å slå av radioen når sensornoden har sendt data, siden sensornoden opplever det høyeste strømforbruket i sendemodus. Dette gjelder også i mottakermodus og i ledigmodus. En annen viktig ting er at det forekommer et vist strømtap når sensornoden skifter mellom modusene. Strømforbruket er veldig avhengig av hvilken modus sensornoden opererer i. For eksempel, så forbruker sensornoden betraktelig mye mindre strøm når den er i sovemodus enn i aktivmodus. Prosessoren kan også være i flere moduser: Aktivmodus, ledigmodus og sovemodus. Hver enkelt av disse modusene har forskjellig strømforbruk, hvor aktivmodus er den dominerende strømforbrukeren, og hvor sovemodus har et minimalt strømforbruk. Det som avgjør levetiden til sensornoden er hvor raskt batteriet tømmes. Hvert batteri har en spesifisert batterikapasitet. Derfor er det viktig at sensornoden er designet slik at hvis ingen interessant hendelser forekommer, så slåes enkelte komponenter av, eller så går komponentene over i en strømbesparende tilstand. Slike tilstander er viktig for at sensornodene skal kunne bevare strømkapasiteten lengst mulig. Det er slik at det er ved sending og mottak av data at strømforbruket er størst, men mer om dette i kapitel 6.

Strømforbruket ved å starte opp radioen er relativt lite. Når det gjelder strømforbruket assosiert med kretsløpet, så kan vi anta en verdi i milliampere størrelsesordenen. Det er også antatt at strømforbruket i kommunikasjonsdelen utgjør ca 98 %. Dette er en verdi vi har hentet ifra [12]. Her er det totale strømforbruket illustrert i et kakediagram. Ut i fra kakediagrammet,

så ser vi at kommunikasjonen forbruker cirka 98 % av det totale forbruket. Det betyr at prosessoren, og de resterende komponentene forbruker mindre enn 2 % av det totale strømforbruket. I [12] er det antatt at strømforbruket ved å sende en byte tilsvarer det samme som om prosessoren skulle utføre 11000 instruksjoner, altså i størrelsesorden 1:1000. Figur 31 illustrerer en antatt fordeling av det totale strømforbruket som [12] har benyttet ved bruk av en datapakke på totalt 28 bytes. Denne fordelingen tar for seg det totale strømforbruket som oppstår ved forsendelse av en pakke.



Figur 31. SPINS fordeling av det totale strømforbruket i en sensornode.

Pakkeformatet i SPINS er ulikt pakkeformatet som vi benytter. Men uansett så kan vi også foreta en slik antakelse for de foreslåtte pakkeformatene, og på den måten kartlegge strømforbruket for hver enkelt av pakkeformatene opp mot TinyOS sitt pakkeformat.

5.4.1 Strømforbruket ved kommunikasjon

Ved kommunikasjon, så er det strømforbruket i transceiveren, samt strømforbruket ved å starte opp/vekke sensornoden som må taes i betraktning. Den siste er mer dominerende når pakkene er små. Ettersom pakkene er små, så må transceiveren slåes av og på hyppigere siden den må sende flere pakker. Radioen kan være i 4 moduser: av, sovemodus, mottakermodus, og i sendemodus [48][67]. Strømforbruket må kalkuleres for hver av disse modusene, og strømforbruket avhenger av hvor lang tid radioen har vært i hver av disse tilstandene, og hvor ofte radioen har vært i hver enkelt av disse modusene, altså hvor mange ganger radioen har byttet mellom hver av disse tilstandene.

For å regne ut teoretisk hvor mye strøm som trengs for å sende hver enkelt av disse pakkene, så kan vi danne oss en graf som illustrere utladningstiden, samt hvor lang tid det tar å sende pakken. Strømforbruket finner vi ved å ta integralet av området under grafen. Vi kan benytte følgende formel, men da må vi ta med strømforbruket som oppstår ved å slå på radioen, samt strømforbruket som oppstår ved det elektriske kretsene.

$$C = \int_{t=t_0}^{t_0+t_d} I(t)\,dt$$

C = Strømforbruket som oppstår over en viss tid (mAh)

I = Utladningsstrøm (mAh)

Formel 4. Formel for å beregne strømforbruket over tid.

Det er også andre måter man kan finne strømforbruket på. [48] benyttet følgende formel:

$$E_{bit} = \frac{E_{start}}{L} + \frac{E_{elec} + P_{RF}(M)}{R_S \times Log_2 M} \times (1 + \frac{H}{L})$$

E_{bit} = Strømforbruket ved å sende 1 bit

E_{start} = Strømkapasiteten ved oppstart

E_{elec} = Strømforbruket som oppstår i det elektriske kretsene

L = Selve pakken (bits)

P_{RF} = Strømforbruket som oppstår i radioen

R_S = Datarate

M = Modulasjonsmetode (antall bits pr symbol)

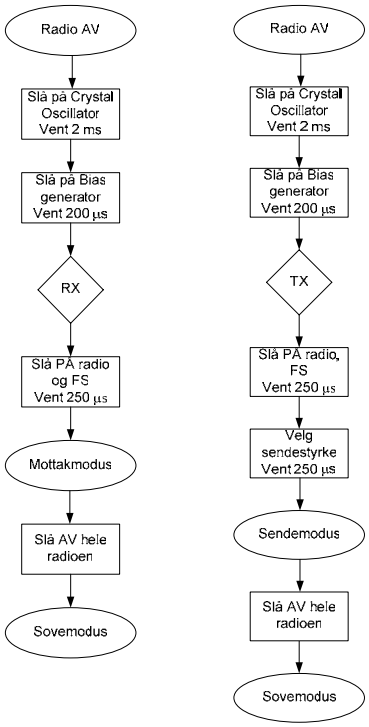
H = Ekstra overhead i pakken (antall bits)

Formel 5. Formel for å beregne strømforbruket pr bit.

Men ettersom spesifikasjonen til Mica2 mote inneholdt spesifisert hvor mye strømforbruket til hver av komponentene var på over tid, så benyttet vi heller formel 4, hvor C er det totale strømforbruket som oppstår når radioen sender i en vis tid, I er strømmen som forbrukes i løpet av tiden t. Denne formelen tar ikke med oppstart av radioen, samt strømforbruket ved det elektriske kretsløpet. Det må taes med i tillegg, noe vi skal kalkulere senere i kapitelet. Dessuten benytter også NS-2 tilsvarende kalkulering av strømforbruket. Derfor ønsker vi å benytte denne formelen.

Begge disse formlene kan brukes, men den nederste er vel den som gir et mest komplett kalkulering av strømforbruket, siden den tar med strømforbruket ved oppstart, samt strømforbruket som oppstår i det elektriske kretsene. Utladningen vil varier etter hvor mye sendeeffekt som benyttes. Tiden det tar å sende en pakke er t. Ettersom pakkestørrelsen øker, så vil naturligvis strømforbruket også øke siden sikkerhetsoverhead tilføres i pakken. Med fast pakkestørrelse, så vil selvfølgelig strømforbruket være konstant siden lik mengde data sendes.

Figur 32 viser tilstandsdiagrammet for aktivering av sender og mottaker modus.



Figur 32. Tilstandsdiagram for mottaker og sending av data.

Som vi ser, så kan vi kartlegge strømforbruket i radioens komponenter steg for steg [29]. Først startes Crystal oscillator, for så Bias. Avhengig om man skal sende eller motta, så vil systemet først slå på Frekvens synthesiser (FS), før radioen settes i mottaker- eller sende-modus. Når datapakken er mottatt eller sendt, så slåes hele radioen av. I tillegg til strømforbruket ved sending av data, så må vi også ta hensyn til Cr. Osc + Bias + Synth. Oppstarten av Cr. Osc. tar 2 ms, mens Bias bruker 200 µs på å starte opp. Når fre. synth har slått seg på, så vil disse 3 være på helt til radioen har mottatt/sendt alt data. Strømforbruket begynner med en gang hver av disse igangsettes, og helt til disse slåes av. Vi kalkulerte strømforbruket for hver enkelt av disse. Disse komponentene er ikke så viktig, men det er strømforbruket som er assosiert med dem som er viktig. Mer om disse komponentene i [29].

Strømforbruket kun i Crystal Osc. = 105 µAh, mens for Bias og Crystal Osc., så er det 860 µAh.
Bare Crystal Osc har et strømforbruk på 5.83E-11 Ah, mens Bias og Crystal Osc. har et strømforbruk på 6E-11 Ah. Frekvens syntesiser + Bias + Crystal Osc. har et strømforbruk på 4/5 eller 5/6 mAh, avhengig av om transceiveren mottar/sender i 433/868 MHz. Det er flere fordeler å operere med en høyere frekvens. For det første er antennevinningen høyere ved 868 MHz enn ved 433 MHz. For det andre kan man benytte seg av mindre antenne ved 868 MHz fremfor 433 MHz. I [86] er det anbefalt en antenne på 8,6 cm for 868 MHZ, mens en antenne på 17,3 cm for 433 MHz. [86] har brukt formelen for 75/Frekvens (MHz) for å komme frem til størrelsen på antenna.

Strømforbruket ved transmisjon avhenger av hvor langt man ønsker å sende. Strømforbruket til Mica2 mote forbruker følgende ved de forskjellige sendeeffektene:

Sendeeffekt (mWatt)	Strømforbruk 433/868MHz (mAh)
P = 0,01 (-20 dBm)	5,3/8,6
P = 0,03 (-15 dBm)	7,4/9,3
P = 0,1 (-10 dBm)	7,9/10,1
P = 0,3 (-5 dBm)	8,9/13,8
P = 1,0 (0 dBm)	10,4/16,8
P = 3,0 (5 dBm)	14,8/25,6
P = 10,0 (10 dBm)	26,7/NA

Tabell 13. Oversikt over sendeeffektene til Mica2 mote.

Det å sende med en effekt på 3mW, og med en båndbredde på 38400 bps gir oss et strømforbruk på 25,6 mAh med en frekvens på 868MHz. Det ekstra strømforbruket i radioen er på 6 mAh for kretsløpet, noe som gir et totalt strømforbruk ved transmisjon på 31,6 mAh. Men med såpass stor sendeeffekt, så vil man kunne nå relativt langt (ca 100 meter) [46]. Ved å for eksempel redusere strømforbruket ved sendeeffekt, og heller satse på flere hopp, så kan man redusere sendeeffekten til 1 mW, som gir et strømforbruk på 16,8 mAh + 6 mAh = 22,8 mAh. Forskjellen er at senderadiusen vil være mindre i siste tilfellet. Dette er noe sensornoden selv må regulere ut ifra hvor lang avstand det er mellom den og naboene. I kapitel 6 skal vi se på hvordan sendeeffekten kan reduseres avhengig av nodetettheten, samt avstanden mellom sensornodene.

Tabellene 14 og 15 gir en oversikt over strømforbruket som oppstår ved å transmittere de forskjellige pakkene ved de forskjellige tilfellene.

Pakkeformat (variabel)	Total pakkestørrelse (bytes)	Ramme overføringstid (ms)	Strømforbruk 38400 bps (nAh)	Økning (%)
TinyOS (29 bytes datalast)	56	11,7	102,4	0,0
Integritet + autentisering	58	12,1	106,1	3,6
Konfidensialitet (4 bytes IV, 4 bytes MAC)	62	12,9	113,4	10,7
Konfidensialitet (4 bytes IV, 8 bytes MAC)	66	13,8	120,7	17,9
Konfidensialitet (8 bytes IV, 4 bytes MAC)	66	13,8	120,7	17,9
Konfidensialitet (8bytes MAC, 8 bytes IV)	70	14,6	128	25
Pakkeformat (fast)	**Total pakkestørrelse (bytes)**	**Ramme overføringstid (sek)**	**Strømforbruk 38400 bps (nAh)**	**Økning (%)**
TinyOS (likt for alle)	56	11,7	102,4	0,0
Pakkeformat (Foreslått)	**Total pakkestørrelse (bytes)**	**Ramme Overføringstid (ms)**	**Strømforbruk (nAh)**	**Økning (%)**
TinyOS (24 bytes datalast)	51	10,6	93,3	0,0
Integritet + autentisering (8 bytes MAC)	57	11,9	104,2	11,7
Konfidensialitet (3 bytes IV, 8 bytes MAC)	62	12,9	113,4	21,5

Tabell 14. Strømforbruket ved de ulike tilfellene med en datarate på 38400 bps.

Tabellen viser som forventet at det er en direkte sammenheng mellom transmisjonstiden og strømforbruket. Desto lengre transmisjonstid, desto høyere strømforbruk vil sensornodene oppleve. Ettersom transmisjonen står for 98 % av det totale strømforbruket assosiert med å transmittere 1 datapakke, så er det viktig at man er nøye med innføring av overhead i en datapakke. Sikkerhetsoverheaden må dessuten vurderes nøye, slik at man får den sikkerheten en applikasjon krever.

Tabell 15 viser strømforbruket for de foreslåtte pakkeformatene ved de ulike sendeeffektene. Strømforbruket ved transmisjon vil variere ettersom mer sendeeffekt benyttes. Vi har allerede sett at den høyeste sendeeffekten for en Mica2 mote er på 31,6 mAh, noe som gir et strømforbruk på 104,2 og 113,4 nAh. Ved å benytte lavere sendeeffekt, så får man følgende strømforbruk:

Pakkeformat (Foreslått)	Total pakke størrelse (bytes)	Ramme Overførings tid (ms)	Strøm-forbruk P=0,01mW (8,6 mAh) (nAh)	Strøm-forbruk P=0,03mW (9,3 mAh) (nAh)	Strøm-forbruk P=0,1mW (10,1mAh) (nAh)	Strøm-forbruk P=0,3mW (13,8mAh) (nAh)	Strøm-forbruk P=1mW (16,8 mAh) (nAh)
TinyOS (24 bytes datalast)	51	10,6	43	45	47	58	67
Integritet + autentisering (8 bytes MAC)	57	11,9	48	51	53	65	75
Konfidensialitet (3 bytes IV,8 bytes MAC)	62	12,9	52	55	58	71	82

Tabell 15. Strømforbruk ved ulike sendeeffekt for en båndbredde på 38400 bps.

Resultatene i tabell 15 viser at strømforbruket reduseres ettersom sendeeffekten reduseres. Dette betyr i realiteten at høyere sendeeffekt vil gi større transmisjonsradius, mens mindre sendeeffekt, vil føre til lavere senderadius. Dette må kalkuleres på forhånd etter hva det er behov for. Det optimale for et sensornettverk ville vært om sensornodene selv kunne foreta en slik effektkontroll, noe som ville gjort dem i stand til å endre sendeeffekten ut ifra det som er mest lønnsomt. En slik intelligens ville vært særdeles nyttig, og ville vært meget strømeffektiv, noe som hadde gjort sensornodene i stand til å kalkulere den korrekte transmisjonsavstanden, for så å korrigere sendeeffekten deretter. Dette skal vi se mer på i kapitel 6.

5.4.2 Strømforbruket ved prosessering

Strømforbruket ved prosessering vil som sagt øke ved innføring av sikkerhet. Økt sikkerhet betyr økt prosessering og økt strømforbruk, samt økt forsinkelse. Men dette er egentlig neglisjerbart i forhold til strømforbruket ved transmisjon. Prosessoren kan på samme måte som radioen være i 4 moduser: av, sovemodus, ledigmodus, og i aktivmodus [48][67]. Det er selvfølgelig i aktivmodus at prosessoren forbruker mest strøm, mens i sovemodus forbruker den minimalt med strøm. I sovemodus er prosessoren og de fleste interne komponenter slått av. Den vekkes av et internt avbrudd, hvor den så entrer ledigmodus. I ledigmodus, så er prosessoren fremdeles inaktiv, men da er interne komponenter aktive, som blant annet intern klokke eller timer. I aktiv modus er både prosessoren og alle dens komponenter aktive. Strømforbruket kan kalkuleres avhengig av hvor lenge prosessoren er i hver enkelt av disse tilstandene, samt hvor ofte den bytter mellom tilstandene. Men dette kan bli relativt vanskelig uten bruken av gode strømmodeller. Derfor vil vi benytte samme antakelse som [12], med at strømforbruket ved prosessering er i størrelsesorden 1:1000 i forhold til strømforbruket ved kommunikasjon.

5.4.3 Strømforbruket ved sensormekanismen

Sensoren har som oppgave å konvertere fysiske fenomener om til elektriske signaler [48]. Strømforbruket i sensoren er viktig å kartlegge ettersom driftssyklusen til sensoren kan degradere levetiden til sensornoden ganske merkbart. En enkelt sensor kan være i 2 tilstander [72], enten av eller på. Strømforbruket kan kalkuleres etter hvor lenge sensoren er på. Når sensornoden er av, forbruker den ikke noe strøm. Avhengig av hvor avansert en sensor er, så kan det hende at den har flere tilstander. Men kun tilstandene av og på vil bli vurdert i denne sammenhengen. Strømforbruket i sensoren forekommer ved signalsampling og ved konvertering av fysiske signaler til elektriske signaler, samt signalbehandling, og analog til digital konvertering. Ettersom strømforbruket til sensoren forblir det samme uavhengig av innføring av sikkerhet eller ikke, så kan vi anse dette som en konstant verdi. Mer om dette i kapitel 6.

5.5 Sikkerhetsnivået ved de ulike scenarioene

Hvor god sikkerhet man har avhenger av flere faktorer: hva slags algoritme som benyttes, at nøkkelen forblir hemmelig, lengden til nøkkelen, IV, og hvordan disse delene fungerer sammen [4]. Men hovedsakelig, så er det lengden på krypteringsnøkkelen som avgjør hvor god sikkerhet man har. Det er vanlig å måle dette i arbeidsmengde, altså hvor lang tid det tar å finne alle kombinasjonene til krypteringsnøkkelen. Dessuten har lengden på krypteringsnøkkelen også en innvirkning på hastigheten på krypteringen/ dekrypteringen. På samme måte kan en si at desto større blokkstørrelse som benyttes, desto bedre sikkerhet har

man, men hastigheten det tar å kryptere/dekryptere reduseres deretter. En blokkstørrelse på 64 bits er en fornuftig størrelse, og er en slags standardstørrelse. Hvor god/sterk en algoritme er, avhenger av hvor stor blokkstørrelse den bruker, antall runder den foretar, og hvor stor nøkkel som brukes. For RC5 kan følgende parametere brukes, RC5(32/12/16) [66]. Det betyr at RC5 bruker en 8 byte blokkstørrelse med 12 operasjonsrunder, samt en 16 bytes (128 bits) nøkkel.

En krypteringsnøkkel kan for eksempel knekkes ved å utføre det som kalles ett Brute-force angrep [4]. Dette innebærer at motstanderen prøver ut alle mulige kombinasjoner helt til motstanderen har omgjort chifferteksten til klartekst. Ved gjennomsnitt, så må halvparten av alle mulige nøkler/kombinasjoner prøves ut for å oppnå suksess [4]. Hvor lang tid det tar å dekryptere data er avhengig av hvor rask prosessoren er. Prosessoren til en sensornode er relativt mye tregere enn en prosessor som for eksempel benyttes i en bærbar pc. Denne størrelsesordenen er på ca 1000 ganger, ettersom en sensornode er på 4 MHz, og en bærbar har en prosessor i underkant av 4 GHz. Hvis vi antar at en bærbar pc foretar en dekryptering i løpet av ett µs, så kan vi anta at en sensornode bruker betraktelig mer tid, kanskje 1000 ganger mer tid. Dessuten kan vi anta at en motstander ikke trenger å bruke en sensornode, men at det er realistisk at en mye kraftigere enhet benyttes til å dekryptere data. Denne dekrypteringen kan dessuten foretaes offline, noe som gjør jobben delvis enklere for motstanderen. Tabell 16 viser en oversikt over hvor vanskelig det er å avsløre en nøkkel av n størrelse, hvor n øker med 32 bits, fra 32 bits til 256 bits:

Nøkkelstørrelse (bits)	**Antall kombinasjoner**	**Tid til utføring av 1 kryptering/µs**	**Hvis 10^6 nøkler/µs**
32	$2^{32} = 4.3x10^9$	2^{31} µs = 35.8 min	2.15 ms
64	$2^{64} = 1.8x10^{19}$	2^{63} µs = $2.9x10^5$ år	106 dager
96	$2^{96} = 7.9x10^{28}$	2^{95} µs = $1.2x10^{15}$ år	$1.2x10^9$ år
128	$2^{128} = 3.4x10^{38}$	2^{127} µs = $5.4x10^{24}$ år	$5.4x10^{18}$ år
160	$2^{160} = 1.5x10^{48}$	2^{159} µs = $2.3x10^{34}$ år	$2.3x10^{28}$ år
192	$2^{192} = 6.3x10^{57}$	2^{191} µs = $9.9x10^{43}$ år	$9.9x10^{37}$ år
224	$2^{224} = 2.7x10^{67}$	2^{223} µs = $4.2x10^{53}$ år	$4.2x10^{47}$ år
256	$2^{256} = 1.16x10^{77}$	2^{255} µs = $1.84x10^{63}$ år	$1.8x10^{57}$ år

Tabell 16. Tiden det tar å finne riktig nøkkel ved å prøve ut alle kombinasjoner.

Tabell 16 viser altså hvor lang tid det vil ta en motstander å finne riktig krypteringsnøkkel avhengig av hvor lang tid enheten bruker på å utføre en kryptering. Når motstanderen først har funnet riktig krypteringsnøkkel, så kan motstanderen uten problemer lese data som sendes over sensornettverket. Tabellen viser også at en nøkkelstørrelse på 128 bit vil være vanskelig å finne, og vil gi en ganske god kryptering. Det er også anbefalt å benytte en krypteringsnøkkel på minimum 128 bit [4].

Tabell 17 [42] viser en oversikt over sikkerhetsfunksjonaliteten, og hva slags overhead de forskjellige sikkerhetstjenestene medfører.

Sikkerhetsfunksjon	**Funksjonalitet**	**Overhead**
Konfidensialitet	Kryptering	Økt prosessering og økt minneforbruk
Integritet og aksesskontroll	MAC	Økt prosessering og ekstra overhead i pakken
Semantisk sikkerhet	IV i pakke	Overhead i pakken i form av IV
Svar beskyttelse	IV i pakke	Overhead i pakken i form av IV

Tabell 17. Overhead som oppstår i forbindelse med sikkerhetsteknikkene.

For at en motstander skal klare å endre en MAC så må dette gjøres ved å sende 2^n pakker til en gyldig sensornode, og kan altså ikke gjøres offline. Det krever at en motstander må sende 2^n pakker til en bestemt sensornode. Vi har benyttet en MAC på 8 bytes i de foreslåtte pakkeformatene, noe som gjør at en motstander må sende maks 2^{64} pakker for å kunne få en pakke godkjent av en sensornode. Det samme prinsippet gjelder for IV, og med en 3 bytes IV, så kan en sensornode sende maksimalt 2^{24} pakker til samme nabonode før IV gjentaes med samme nøkkel. Som sagt vil gjenbruk av IV garantert oppstå innen 16,7 millioner pakker har blitt sendt mellom 2 sensornoder. Hvis det sendes 1 pakke i minuttet, så vil gjenbruk av IV garantert oppstå innen 11597 dager. Dette kan være tilfredsstillende avhengig av hvor mange pakker som sendes. Når det gjelder MAC, så kan en sensornode motta maks 2^{64} pakker før motstanderen har klart å sende riktig MAC. Så hvis en motstander sender pakker av størrelsen 31 bytes kontinuerlig til en sensornode med en datarate på 38400 bps, og hvor vi kun tar hensyn til de 31 bytesene + startsymbol og synkroniseringsbytesene på 20 bytes, så vil en motstander kunne sende cirka 94 pakker i sekundet. I løpet av en dag så vil en sensornode kunne motta 8,1 millioner pakker hvis den mottar pakker kontinuerlig. Strømkapasiteten til en sensornode avgjør hvor mange pakker en sensornode kan motta. Hvis radioen til sensornoden er på hele tiden, altså i mottakermodus (15,6 mAh), så vil sensornoden kunne leve maksimalt 160 timer, noe som tilsvarer litt over 6 dager. Det betyr at den kan motta maksimalt 48 millioner pakker, noe som fint er innenfor grensen for at MAC skal gjentaes. Dette gir en tilfredsstillende sikkerhet. Derimot er det ganske urealistisk at en radio vil være i aktivmodus kontinuerlig i et sensornettverk. Det er mer sannsynlig at sensornode vil gå over i sovemodus i perioder.

5.6 Del konklusjon og oppsummering av resultatene

En samlet oversikt over resultatene som er oppnådd kan oppsummeres i tabell 18.

Pakkeformat (variabel)	Total pakkestørrelse (bytes)	Ramme overføringstid (ms)	Gjennom-strømning (bps)	Økning i transmisjonstid/ strømforbruk (%)	Strømforbruk (nAh)
TinyOS	56	11,7	7127,5	0,0	102,4
Integritet + autentisering	58	12,1	6956,5	3,6	106,1
Konfidensialitet (4 bytes IV, 4 bytes MAC)	62	12,9	6628,6	10,7	113,4
Konfidensialitet (4 bytes IV, 8 bytes MAC)	66	13,8	6312,9	17,9	120,7
Konfidensialitet (8 bytes IV, 4 bytes MAC)	66	13,8	6312,9	17,9	120,7
Konfidensialitet (8 bytes IV, 8 bytes MAC)	70	14,6	6049,5	25	128
Pakkeformat (fast)	**Total Pakkestørrelse (bytes)**	**Ramme overføringstid (ms)**	**Gjennom-strømning (bps)**	**Økning i transmisjonstid/ strømforbruk (%)**	**Strømforbruk (nAh)**
Alle tilfellene	56	11,7	7127,5	0	104
Pakkeformat (Foreslått)	**Total Pakkestørrelse (bytes)**	**Ramme overføringstid (ms)**	**Gjennom-strømning (bps)**	**Økning i transmisjonstid/ strømforbruk (%)**	**Strømforbruk (nAh)**
TinyOS (24 bytes datalast)	51	10,6	6315,8	0,0	93,3
Integritet + autentisering (8 bytes MAC)	57	11,9	5827,0	11,7	104,5
Konfidensialitet (3 bytes IV, 8 bytes MAC)	62	12,9	5485,7	21,5	113,2

Tabell 18. Total oppsummering av resultatene.

Resultatene viser at innføring av sikkerhet vil føre til økt strømforbruk ved transmisjon av data. Desto mer overhead, og desto lengre tid det tar å transmittere en datapakke, desto høyere vil strømforbruket være. Det er også forventet at desto mer sikkerhet som innføres, og desto større pakkene blir, desto høyere vil forsinkelsen være. Overheaden assosiert med sikkerhetsteknikkene vil føre til at det tar lengre tid å transmittere en datapakke over kanalen, noe som også vil påvirke strømforbruket, båndbredden og forsinkelsen. Sikkerheten bør derfor innføres ut i fra restriksjonene til sensornettverkets ressurser, samt ut i fra viktigheten til data.

Det er flere kostnader assosiert med innføring av sikkerhet. Dette innebærer kostnader i form av enten at pakkestørrelsen øker, eller det at mindre pakker krever hyppigere forsendelser. En annen kostnad forekommer på grunn av den ekstra prosesseringstiden som er nødvendig med kryptering og dekryptering, samt det ekstra strømforbruket som er assosiert med både transmisjon og prosessering. Som tabell 18 viser, så er kostnadene ved lengre pakker at de reduserer gjennomstrømningen, øker forsinkelsen, og sist, men ikke minst, så øker strømforbruket ettersom radioen må forbli slått på over lengre tid ved større pakker. Strømforbruket oppstår hovedsakelig ved å det blir mer data å transmittere, og det at kryptering fører til mer prosessering. Mindre pakker derimot har lavere transmisjonstid, forsinkelse, og strømforbruk per pakke. Men gjennomstrømningen blir mye dårligere, noe som krever at flere pakke må sendes for å få igjennom ønsker data. Dette reduserer

gjennomstrømningen, og økt strømforbruk kan forekomme ved at radioen må slåes av og på oftere, ettersom man må foreta flere transmisjoner.

6 Ulike scenarioer og simuleringer av strømforbruket i sensornettverk

Vi skal nå se på flere ulike scenarioer et sensornettverk kan operere i. Det gjøres ved å benytte ulike driftssykluser for de ulike komponentene i sensornoden. En annen ting vil være å danne ulike scenarioer avhengig av hvilke situasjoner en sensornode kan komme opp i, for eksempel om sensornoden opererer som en detekterende sensornode, eller om en sensornode opererer som en ruter eller et aggregeringspunkt. Det vil også foretaes en del simuleringer av ulike scenarioer. Dette er scenarioer som involverer strømforbruket og levetiden til sensornodene, levetiden til en rute, samt strømforbruket ved ulik nodetetthet. Dette innebærer at dersom nodetettheten i et lite område øker, så kan det være interessant å se på hvordan dette påvirker strømforbruket ettersom kommunikasjonen vil foregå over flere hopp for å kunne nå sinken. Det at kommunikasjonen vil foregå over flere hopp har med at ettersom nodetettheten øker, så vil også avstanden mellom sensornodene vil bli kortere, noe som igjen gjør at sensornodene kan sende med lavere sendeeffekt.

For at en sensornode skal kunne bevare levetiden sin lengst mulig, så er det viktig at sensornoden i seg selv er veldig strømeffektiv. En sensornode er som sagt bygd opp av flere komponenter hvor komponentene kan være i ulike tilstander. Tabell 19 (hentet fra [73]) viser et eksempel på hvilke tilstander sensornoden kan være i.

Tilstand	**CPU**	**Minne**	**Sensor**	**Radio**
S_0	Aktiv	Aktiv	PÅ	TX, RX
S_1	Ledig	Sove	PÅ	RX
S_2	Sove	Sove	PÅ	RX
S_3	Sove	Sove	PÅ	AV
S_4	Sove	Sove	AV	AV

Tabell 19. Ulike sovemoduser en sensornode kan være i.

Som [73] indikerer, så vil dypere sovemodus, som for eksempel S_4, føre til høyere forsinkelse enn S_0 siden alle komponentene i S_4 er slått av, og er i sovemodus, mens i S_0 så er alle komponentene aktive, noe som vil føre til mindre forsinkelse. Derimot, så vil S_4 være mye mer strømeffektiv enn S_0, men det vil ta mye lengre tid å igangsette den. Dette gir en balanse mellom forsinkelse og strømforbruk, hvor sistnevnte helt klart er viktigst. Det trenges ikke ta så mye hensyn til forsinkelsen, så lenge data som sendes ikke har noe krav om dette. Det kan selvfølgelig være en fordel om disse tilstandene velges ut i fra aktivitetene til sensornodene. Har sensornoden lite å gjøre, så må den settes i sovemodus over lengre tid, men hvis sensornoden har mye gjøre så må den være aktiv oftere og lengre. Det som skiller disse tilstandene er strømforbruket som oppstår ved å gå fra sovemodus til aktivmodus. Hvis en sensornode er i en dypere sovemodus (S_4), så vil den forbruke mindre strøm, men den vil bruke lengre tid til å komme ut av sovemodus, og over i aktivmodus. Ut ifra tabellen, så ser man at sensoren alltid er på, bortsett fra i S_4. Dette gjør at sensornoden i tilstand S_4 selv må bestemme når den skal våkne. I tilstand S_3 så vil sensornoden ha slått av alle komponentene, bortsett fra sensoren. På denne måten kan sensornoden hele tiden detektere og måle hendelser, for så å vekke opp sensornoden, hvis en hendelse skulle inntreffe. Dette innebærer å gå fra tilstand S_3 til S_0. Det vil i denne overgangen oppstå en forsinkelse ved å gå fra tilstand S_3 til S_0, men dette er nødvendig for at sensornoden skal kunne spare mest mulig strøm. I perioder kan det være greit å sette sensornoden også i tilstand S_4. Problemet med S_4 er å vekke opp sensornoden når det faktisk er en hendelse som inntreffer. Sannsynligheten for at sensornoden

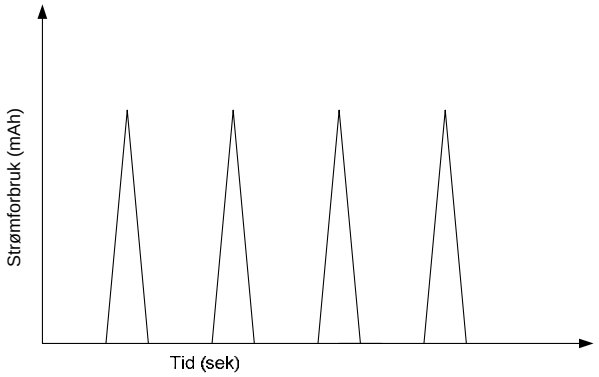
vekkes opp akkurat i riktig tidspunkt, er liten, så S_4 bør ikke benyttes altfor mye, ettersom det fort kan føre til at hendelser som forekommer, ikke detekteres av sensornoden. Derimot kan S_4 benyttes i de tilfellene hvor det ikke er kritisk at hendelser oversees. Prosessoren og minnet settes i en slags sovemodus hvor strømforbruket er minimalt, siden det er nødvendig at prosessoren følger med om det kommer signaler som skal vekke den opp.

Mica2 sensornoden er som sagt utstyrt med en ChipCon 1000 radio, en Atmega 128 prosessor, og en sensortype som avhenger av formålet. MTS310CA [46] er en type sensor som kan brukes sammen med Mica2 mote, og hvor MTS310AC kan detektere seismiske bevegelser og vibrasjoner, samt at den har et strømforbruk på mindre enn 1 mAh [46]. Det antes at radioen enten sender, mottar, eller så er den slått av. Radioen settes altså ikke i sovemodus. Det samme gjelder for sensoren. CPU kan derimot være i 3 moduser, enten i aktiv, ledig, eller i sovemodus. Men det er ikke ønskelig å slå av CPU. Minne kan være i 2 moduser, enten aktiv eller i sovemodus. Disse tilstandene er viktig å ta hensyn til.

6.1 Estimert levetid for en sensornode ved ulike driftssykluser

Vi kan opprette en komplett driftssyklus for å kartlegge levetiden til en enkelt sensornode over tid, ved å benytte både sovemodus og aktivmodus om hverandre. Når sensornoden ikke er i aktivmodus, må den settes tilbake i sovemodus. Vi kan kartlegge levetiden til en sensornode ved å se på hvor lenge en driftssyklus varer, og hvor ofte den forekommer. For å bevare sensornodens levetid lengst mulig, så er det viktig at sensornoden sover mesteparten av tiden. Når den først vekkes, så er det viktig at prosessering og kommunikasjon foregår på meste mulig effektiv måte, det vil si at når en komponent har foretatt sin oppgave, og ikke har flere oppgaver å utføre, så må den enten slåes av eller settes i sovemodus. Når sensornoden er ferdig med driftssyklusen, eller ikke har flere oppgaver å utføre, så går den tilbake i sovemodus. For at levetiden til en sensornode skal bevares lengst mulig, så er det nødvendig at sensornodens forblir slått av så lenge som 99 % av tiden. For eksempel ved at man foretar noen driftssykluser i minuttet. Det kan være greit at driftssyklusen er mindre enn 1 % for at levetiden til sensornoden skal optimaliseres, men dette er veldig avhengig av applikasjon og hensikten til sensornoden. Hvis sensornoden opererer i et miljø hvor hendelser forekommer veldig hyppig, og hvor det er antatt at hendelsene kun foregår over kort tid, da er det ikke så nødvendig å optimalisere levetiden, men heller at alle hendelsene detekteres.

En sensornode kan operere på 2 måter, enten ved at den vekkes opp ved faste intervaller, eller at sensornoden vekkes opp når en hendelse inntreffer. Det siste tilfellet vil sørge for at alle hendelser som inntreffer vil bli prosessert (S_3), mens ved det første tilfelle, så kan det i verste fall forekomme at signifikante hendelser ikke detekteres (S_4). I S_3, så er som sagt sensoren på hele tiden. Det mest optimale ville være at sensornoden slåes på hver gang en hendelse inntreffer, for så å slåes av når hendelsen stopper, men dette kan være vanskelig å realisere. Vi kan foreta periodisk driftssykluser, hvor sensornoden foretar datainnsamling via sensoren, deretter prosesserer data, og til slutt sende data, for så å gå tilbake i den ønskede tilstanden (S_0 til S_4). Figur 33 viser en illustrasjon over hvordan en periodisk driftssyklus kan foregå. Her vil sensornoden slåes på ved faste intervaller, og da prøve å detektere hendelser. Hvis hendelser forekommer, så vil sensornoden samle inn data, prosessere data, for så å sende data til sinken. Hvis ikke noen hendelser forekommer, så vil sensornoden slåes av etter en viss tid (intervallets varighet), for så å sove frem til neste intervall.



Figur 33. Periodisk driftssyklus.

Et annet eksempel kan være at sensornoden vekkes opp av en oppstått hendelse. Måten dette gjøres på har stor betydning for sensornodens levetid. Dette innebærer at sensoren forblir på hele tiden, slik at sensoren kan detektere de hendelsene som måtte oppstå. Et problem som oppstår ved begge tilfellene når man skal spare strøm, er at radioen slåes av. Dette gjør at hvis noen prøver å sende data via kringkasting, eller ønsker å rute data, så vil sensornoden ikke detektere dette. Det vanlige vil være å ha sensoren på hele tiden (en sensor som bruker lite strøm), hvor sensoren sender et signal til prosessoren om at en hendelse har inntruffet, og at prosessoren må våkne, eller at prosessoren våkner periodevis for å se om sensoren har data som må prosesseres, for så å prosessere data, og gå tilbake i sovemodus. Prosessoren kan vekke radioen når en hendelse som krever samarbeid om prosessering og transmisjon av data. Hvis sensornodene vekkes opp for sjelden ved periodevis driftssyklus, så kan det forekomme at pakker forkastes. Sensornoden kan stille inn en tidsmåler før den går i sovemodus, hvor tidsmåleren har som oppgave å vekke opp sensornoden etter en viss tid.

Det er ønskelig å regne ut strømforbruket ved å motta, samt strømforbruket ved å sende en pakke. Det kan antaes at det å kalkulere MAC koster cirka 2 %, og all prosessering (inkl kryptering og dekryptering) koster mindre enn 1 %, og deretter kan en prosentfordeling for transmisjon av pakkeformatet foretaes [12]. TinyOS har ikke noen form for MAC, men CRC vil nok ikke være så prosesseringsintensiv som MAC, men det antaes at CRC koster like mye som MAC i denne sammenhengen. Det vil si at kostnadene ved kommunikasjonen vil være på 98 % av det totale strømforbruket for begge tilfellene. De resterende 2 % er strømforbruket assosiert med de andre komponentene i sensornoden, som blant annet prosessoren og minnet. Til estimering av strømforbruket vil det bli benyttet en pakke på 31 bytes uten sikkerhetsoverhead, en på 37 bytes med autentisering og integritet, samt en pakke på 42 bytes med konfidensialitet. Faktorer som påvirker strømforbruket til en radio er som sagt hva slags type modulasjon som brukes, datarate, sendeeffekt (avgjøres av senderadiusen), og ikke minst, operasjonsdriftssyklusen, altså hvor lenge sensornoden er aktiv. Når den ikke er aktiv, så sover den. I aktivmodus, så har radioen et strømforbruk på 31,6 mAh ved transmisjon. Hvis 2 % utgjør det resterende strømforbruket, så blir det totale strømforbruket på 32,2 mAh. Ved en komplett driftssyklus, så må også strømforbruket assosiert med at en sensornode ikke er i aktivmodus taes med. Sensornoden bør være i S_3 når radioen ikke er aktiv, altså når den ikke sender data. Når sensornoden slåes på, og settes i aktivmodus, så foretar den prosessering på en rask og effektiv måte, for så å gå tilbake i sovemodus.

Vi vet at strømforbruket ved å sende pakker på størrelse 51, 57 og 62 bytes (inkl MAC overhead) er på 93,3, 104,2 og 113,4 nAh. Hvis dette utgjør 98 % av det totale strømforbruket assosiert med sikkerhetsoverhead, både ved kommunikasjon og prosessering, så kan vi regne ut det totale strømforbruket ved at det resterende utgjør 2 %. Det totale strømforbruket for

hvert av tilfellene er 95,2, 106,3, og 115,7 nAh, og det inkluderer det totale strømforbruket assosiert med å sende en pakke (inkludert det faste strømforbruket på 6 mAh). Dette strømforbruket er beregnet ut ifra verdier for strømforbruk som [29] har spesifisert er aktuelle. Ved mottak av data, så vil vi ha et strømforbruk på 15,6 mAh, og et strømforbruk på 15,9 mAh inkludert overhead assosiert med prosessering og beregning av sikkerhet. Dette vil utgjøre et strømforbruk på 46,0, 51,5, og 56,0 nAh for hvert av pakkeformatene ved mottak. Totalt strømforbruk, inkludert 2 % for prosessering, så utgjør dette 46,9, 52,5 og 57,1 nAh. Selv om 2 % virker lite, så vil det degradere levetiden til et sensornettverk. Vi skal se på noen forskjellige tilfeller av driftssykluser, både når sensoren er på hele tiden, og tilfeller der sensoren kun slåes på i perioder. Driftssyklusen bør som sagt være på mindre enn 1 %, men det kan være greit å kartlegge estimert levetid ved ulike driftssykluser pr minutt.

6.1.1 Ulike driftssykluser

For oss er det S_0, S_3 og S_4 som er de mest interessante tilfellene. Vi skal danne en driftssyklus hvor sensornoden foretar sitt formål ved detektering av hendelser, for så å prosessere data, og sende data. Når sensornoden ikke detekterer hendelser, eller transmitterer data, så sover den, men sensoren kan enda være aktiv (S_3). Hvis det ikke forekommer noen hendelser, så vil sensornoden ha et strømforbruk som kun omhandler sensoren i aktiv modus, pluss prosessoren og minnet som er i sovemodus. Vi kan også kartlegge et tilfelle der hele sensornoden sover (S_4), og hvor sensornoden kun vekkes opp i faste tidsintervaller, og sover resten. Vi kan kalkulere strømforbruket pr driftssyklus, for så å teste med forskjellige tidsintervaller.

Hvis vi har en eller flere driftssykluser hvert minutt, så kan vi kalkulere dette ut ifra det totale strømforbruket pr driftssyklus. Det er flere faktorer som spiller inn på strømforbruket i et ordentlig sensornettverk, blant annet kollisjoner, samt det å overhøre meldinger som skal til andre (unødvendig trafikk), samt lang ledig tid. Ledig tid innebærer at sensornoden er klar for hendelser som måtte inntreffe. Det kan for eksempel være at sensornoden er klar for å rute trafikk fra en annen sensor og videre mot sinken. S-MAC (Sensor Medium aksesskontroll) løser problemet med ledig tid ved at S-MAC periodevis sover og lytter [6]. Radioen slåes av når sensornoden er i sovemodus. Problemet med at sensornoden hører pakker som skal til andre kan løses ved at sensornoden sover når naboene kommuniserer. Varigheten på kommunikasjonen indikerer hvor lenge en sensornode skal sove. Det med overhøring kan bli et stort problem ved høy belastning (mye trafikk). Dette fordi mange sensornoder må lytte til trafikken fra mange andre sensornoder. Sensornodene er nødt til å benytte ledigmodus i perioder, ettersom et sensornettverk baserer seg på multihopp ruting. Hvis absolutt ingen sensornoder lytter, så er det heller ingen som kan videresende pakker fra en annen sensornode.

6.1.1.1 Sensoren i sovemodus versus aktivmodus

Når sensornoden sender eller mottar data, så vil den være i en tilstand tilsvarende S_0. Her vil alle komponentene være aktive i den perioden hvor radioen er aktiv. Vi kan illustrere strømforbruket i sensornoden når sensoren er i S_3, sammenlignet med når sensoren er i S_4. Vi kan kartlegge strømforbruket ved at sensoren kun slåes på i 1 sekund, mot at sensoren er på i 60 sekunder. I begge tilfellene vil radioen også sende 1 sekund av totalt 60 sekunder. Det totale strømforbruket ved transmisjon er som sagt 32,2 mAh, og strømforbruket til sensoren er på 1 mAh når den er på, og 0 mAh ellers. Når sensoren er på hele tiden, så vil dette føre til et strømforbruk på 1 mAh * 60 sekunder + 32,2 mAh * 1 sekund, og dette vil resultere i et totalt

strømforbruk på 25,6 µAh. Mica2 har en strømkapasitet på 2500 mAh, noe som fører til at sensornoden vil ha en levetid på cirka 67 dager. Hvis sensoren kun slåes på i 1 sekund, vil strømforbruket være på 1 mAh * 1 sekund + 32,2 mAh * 1 sekund, noe som resulterer i et strømforbruk på 9,2 µAh. Dette tilsvarer en estimert levetid på 188 dager når sensoren kun er på i 1 sekund pr minutt versus 67 dager hvis sensoren er på hele tiden. Her taes det ikke hensyn til hvilket pakkeformat som brukes, men kun at radioen transmitterer i 1 sekund pr minutt. Avhengig av hvordan driftssyklusen til sensoren opererer, og hvor ofte sensornoden sender eller mottar data, så vil dette ha stor innvirkning på sensornodens levetid. Tabell 20 viser 3 forskjellige driftssykluser for sensoren, hvor sensoren har ulike driftssykluser, mens radioen sender i 1 sekund i hvert av tilfellene.

Sensorens driftssyklus	Sensorens varighet (pr minutt)	Radioens varighet (pr minutt)	Estimert batterilevetid (antall dager)
Full detektering	60 sekunder	1 sekund	67 dager
Periodisk detektering	10 sekunder	1 sekund	148 dager
Periodisk detektering	1 sekund	1 sekund	188 dager

Tabell 20. Driftssyklus hvor radioen sender i 1 sekund

Dette tilfellet gjelder kun når sensornoden sender data. Det kan blant annet være en sensornode som detekterer hendelser, som sender data via andre sensornoder mot sinken. Ut ifra dette tilfellet, så ser man at måten sensoren opererer på har stor innvirkning på sensornodens levetid. Hvis sensoren har en driftssyklus på 60, 10 eller 1 sekund, så vil levetiden til sensornoden være på henholdsvis 67, 148, eller 188 dager hvis sensornoden kun sender i 1 sekund. Det at radioen er slått på så lenge som 1 sekund pr minutt er vel egentlig ikke helt sannsynlig hvis man ønsker å bevare levetiden til en sensornode. Men under disse omstendighetene så vil en sensornode ha et slikt strømforbruk. Det at sensoren er på hele tiden bør kun forekomme i nødvendige applikasjoner. Strømforbruket hos sensoren er på 1 mAh, noe som tilsvarer en levetid på 104 dager bare ved at sensoren er aktiv hele tiden, mens resten av sensornoden er av eller i sovemodus. Derfor er det viktig at sensoren også får tildelt passende driftssykluser ut ifra hvilken applikasjon sensornodene skal utføre.

6.1.1.2 Forskjellige tilfeller for strømforbruket i et sensornettverk

Vi kan forestille oss at det kan oppstå 3 tilfeller i et sensornettverk. Det første tilfellet kan være at en sensornode som detekterer en hendelse, kun sender pakker mot sinken. Det andre tilfellet kan være at sensornodene som ligger mellom den detekterende sensornoden og sinken, må operere som rutere, ved at de må videresende pakker fra den detekterende sensornoden, og videre mot sinken. I dette tilfellet vil sensornodene også måtte motta data, for så å videresende data til neste sensornode. Et siste tilfelle kan være at en aggregeringsnode vil motta flere pakker enn det den vil sende, siden den vil foreta lokal prosessering av data den mottar. Det er da antatt at en aggregeringsnode kun videresender 35 % av alle data den mottar, ettersom den kan sammenslå data til mer komplett data. I de andre tilfellene vil en sensornode som detekterer et fenomen sende i 50 % av 1 sekund, mens mellomliggende sensornoder som opererer som rutere vil motta i 50 % av 1 sekund, for så å transmittere i 50 % av 1 sekund. I alle disse tilfellene vil sensoren ha en driftssyklus på 1, 10 og 60 sekunder.

I det første tilfellet vil som sagt en detekterende sensornode kun sende i 50 % av et sekund, og vi kan tenke oss at sensoren kan ha en driftssyklus på 1, 10 og 60 sekunder. Tabell 21 viser at

en sensornode vil ha en levetid på 82, 239 og 365 dager, avhengig av driftssyklusen til sensoren. Desto lengre sensoren og radioen er på, desto kortere levetid har sensornoden.

Sensorens driftssyklus	Sensorens varighet (pr minutt)	Radioens varighet (sender) (pr minutt)	Estimert batterilevetid (antall dager)
Full detektering	60 sekunder	Halvt sekund	82 dager
Periodisk detektering	10 sekunder	Halvt sekund	239 dager
Periodisk detektering	1 sekund	Halvt sekund	365 dager

Tabell 21. Driftssyklus hvor sensornoden i 1 sekund.

Vi kan så anta at en mellomliggende sensornode vil motta data, for så å videresende data til neste sensornode mot sinken. Dette kan den gjøre ved å motta i 50 % av 1 sekund, for så å sende i 50 % av et sekund. Hver mellomliggende sensornode vil da ha en levetid på henholdsvis 74, 183 og 249 dager.

Sensorens driftssyklus	Sensorens varighet (pr minutt)	Radioens varighet (sender/mottar) (pr minutt)	Estimert batterilevetid (antall dager)
Full detektering	60 sekunder	0,5/0,5 sekunder	74 dager
Periodisk detektering	10 sekunder	0,5/0,5 sekunder	183 dager
Periodisk detektering	1 sekund	0,5/0,5 sekunder	249 dager

Tabell 22. Driftssyklus hvor sensornoden sender og mottar i 1 sekund.

I det siste tilfellet kan man anta at radioen mottar i 65 % av et sekund, mens radioen sender i 35 % av et sekund. Levetiden for hver enkelt aggregeringsnode vil ved de ulike tilstandene være på 76, 197 og 276 dager.

Sensorens tilstand	Sensorens varighet (pr minutt)	Radioens varighet (sender/mottar) (pr minutt)	Estimert batterilevetid (antall dager)
Full detektering	60 sekunder	0,35/0,65 sekund	76 dager
Periodisk detektering	10 sekunder	0,35/0,65 sekund	197 dager
Periodisk detektering	1 sekund	0,35/0,65 sekund	276 dager

Tabell 23. Driftssyklus hvor sensornoden sender i 0,35 sekunder, og mottar i 0,65 sekunder.

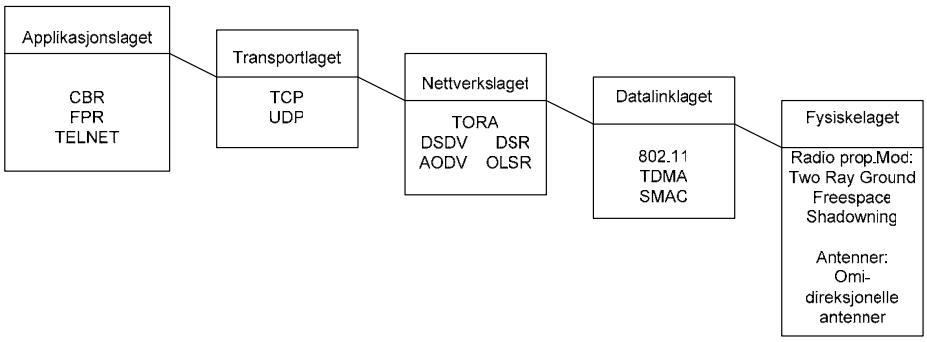
Ut ifra tabellene ser vi at måten sensoren og radioen opererer på, har stor innvirkning på levetiden til sensornodene. En annen ting er også hvilke type oppgaver en sensornode foretar seg, for eksempel om den detekterer en hendelse, for så å rapportere hendelsen til sinken, eller om sensornoden operer som ruter, eller om sensornoden opererer som en aggregeringsnode. I disse tilfellene har sensornodene operert med forskjellige driftssykluser for sensoren, og måten radioen har fungert på har variert avhengig funksjonen til sensornodene. En annen ting er at sensornodene som ligger mellom sensornoden som detekterer en hendelse, og sinken, vil ha høyere strømforbruk ettersom de både må motta data, for så å videresende data. Det samme gjelder for aggregeringsnoder. Men en aggregeringsnode vil derimot kunne prosessere data fra flere, for så å sammenslå data til mer komplett data. Derfor er det sannsynlig at en aggregeringsnode ikke trenger å videresende all data den mottar, men vil kunne sammenslå data til mer kompakt data, noe som gjør at aggregeringsnoden kan motta mer enn det den videresender. I neste kapitel skal vi se på hvordan sensornettverket oppfører seg i en simulering.

6.2 Simulering av sensornettverk

I simuleringen ble det benyttet en simulator med navn NS-2. Dette er en simulator som kan simulere mange forskjellige typer nettverk. Målet med simuleringen var å illustrere hvordan sikkerhetsoverheaden påvirker sensornodene i et sensornettverk. For å kunne simulere et sensornettverk var det nødvendig å installere en utvidelsespakke med navn SensorSim, som var ment å kunne simulere et sensornettverk i et NS-2 miljø. Et problem var at mange av protokollene som NS-2 benyttet, ikke var designet for et sensornettverk. For eksempel måtte simuleringen baseres på MAC 802.11 protokollen, samt ad hoc rutingsprotokollen AODV. En rutingsprotokoll som baserer seg på kringkasting var dessverre ikke tilgjengelig. Heller ingen rutingsprotokoller som er designet for et sensornettverk var tilgjengelig. Derfor måtte vi nøye oss med AODV og MAC 802.11 i alle simuleringene.

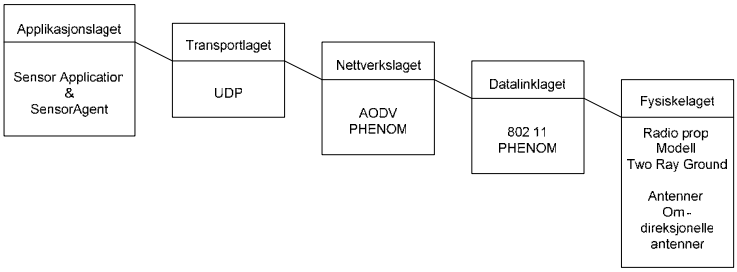
6.2.1 Litt om Ns-2 og Sensorsim

NS-2 [19] er et simuleringsverktøy som kan brukes til å simulere ulike typer nettverk. NS-2 er i utgangspunktet ikke beregnet til å simulere et sensornettverk, men mer beregnet for trådløse nettverk som 802.11, og Ad hoc nettverk. Dette fordi protokollene i NS-2 ikke er designet for et slikt nettverk. NS-2 har ulike protokoller for hvert lag som er tilgjengelig for et trådløst system. Figur 34 viser en oversikt over mulige protokoller som kan brukes til simuleringen av trådløse nettverk.



Figur 34. Protokoller som brukes i Ns-2 til å simulere trådløse nettverk.

For å kunne simulere et sensornettverk, så ble det som sagt benyttet en utvidelsespakke for NS-2 som heter Sensorsim [20] [21]. Denne utvidelsen ga oss mulighet til å simulere et sensornettverk, men den hadde noen begrensninger, ettersom den benyttet protokoller som er beregnet for 802.11 og Ad hoc. For simulering av et sensornettverk, så ble det benyttet noen andre protokoller, blant annet i applikasjonslaget (SensorApplication og SensorAgent), i nettverkslaget (PHENOM: ruting av fenomendata rutes over en egen kanal), samt i datalinklaget (PHENOM: En egen kanal som fenomenet kan sende data over). Figur 35 viser en oversikt over de protokollene som ble benyttet i hvert av lagene i simuleringene:



Figur 35. Protokolloversikt i simuleringen av sensornettverket.

Vi valgte å benytte SensorSim slik den er utviklet, hvor vi kun endre på parametere, som blant annet hendelser som naturlig vil oppstå i et sensornettverk. Dette innebærer blant annet hvordan en sensornode reagerer når den detekterer en hendelse. Sensoren kan da velge å sende en pakke (av en valgt størrelse) til sink periodevis, eller så kan sensornoden sende en pakke til sink umiddelbart etter at sensornoden har detektert en hendelse. Applikasjonslaget håndterer pakker som et fenomen sender ut. Dette gjøres ved at sensorapplikasjonen genererer pakker som sendes til sinken når en sensornode detekterer et fenomen. Hvis farging er slått på, så vil en sensornode som skal sende en pakke til sink bli rød. Hvis en sensornode blir grønn, så betyr det at sensornoden ikke har detektert noen hendelse innenfor en viss tid. Fargingen ble slått av i vårt tilfelle siden vi skulle beregne strømforbruket i sensornettverket. Beregning av strømforbruket lot seg ikke gjøre med farging slått på, ettersom sensornoder som går tom for strøm benytter andre farger. Følgende kan endres i sensorapplikasjonen: MESG_SIZE: Størrelsen på pakkene som den detekterende sensornoden sender til sinken. I vårt tilfelle var disse pakkene satt til 31, 37, og 42 bytes. TRANSMIT_FREQ: Hvor ofte en sensornode skal sende en melding til sinken etter å ha detektert et fenomen. I vår simulering så sendes 2 pakker hver gang et fenomen detekteres. Hvor ofte et fenomen oppstår, avhenger av pulsraten. SILENT_PHENOMENON: Antall sekunder som har gått uten at en sensornode har detektert en hendelse. I vårt tilfelle var denne satt til 0,2 sekunder, og hvis en sensornode ikke detekterer en hendelse innen 0,2 sekunder, så endrer sensornoden farge til grønn. En sensoragent festes til kanalen, slik at sensoragentene kan detektere hendelser som et fenomen genererer. Sensornodene bruker 2 agenter: en sensoragent som festes til fenomenkanalen for å kunne motta PHENOM pakker, og en UDP agent, som transporter pakkene til sinken. Etter at en sensoragent mottar en PHENOM pakke fra et fenomen, så videresender sensoragenten pakken til sensorapplikasjonen. PHENOM er en type pakker som et fenomen sender ut. Hvor ofte PHENOM pakker genereres og sendes ut avhenger av pulsraten. Det vil si hvor ofte et fenomen skal forekomme. 4 hendelser kan detekteres: Karbon Monoxide, høy eller lav seismisk aktivitet, og lyder. Disse hendelsene kan også genereres tilfeldig. Alle PHENOM pakker sendes via en egen kanal. Dette fordi at fenomenet bør operere i et kollisjonsfritt domene. Selve utsendingen av PHENOM pakker skjer via PHENOM ruting. Dette fører til at sensornodene trenger å ha 2 grensesnitt, en for 802.11 kanalen, og en for PHENOM kanalen. I tillegg har SensorSim gjort endringer i strømmodellen til NS-2, slik at detektering av et fenomen vil redusere strømkapasiteten med en fast verdi, avhengig av hvilken verdi som benyttes.

Simuleringen ble strukturert på følgende måte:

- Det ble konfigurert separate kanaler for fenomen og data, siden fenomennoden bør sende på en annen kanal enn den sensornodene benytter.
- Det ble foretatt simuleringer med et fenomen.
- Det ble opprettet 25, 49 og 100 sensornoder i de ulike simuleringene
- Kun 1 sinknode ble benyttet. Dette for at SensorSim har problemer med å takle flere sink.
- Det ble lagt til en 1 sensoragent, 1 UDP agent og 1 sensorapplikasjon til hver sensornode.

Avvik fra et ekte sensornettverk:
Vi vil bruke MAC 802.11 og AODV protokollene. Dette er protokoller som egentlig er utviklet for trådløse LAN, og ad hoc nettverk, og som ikke er spesielt beregnet for et sensornettverk. Disse protokollene har en del overhead assosiert med seg. 802.11 sender RTS/CTS/DATA/ACK for alle unikast pakker, og DATA for all kringkastingstrafikk.

Man kan fint bruke dem til å simulere et sensornettverk, men for å beregne strømforbruket riktig, så bør det utvikles mer realistiske protokoller. Et annet problem med disse er alle forespørslene som AODV rutingsprotokollen sender ut. Dessuten baserer AODV seg på den samme ruten hele tiden, mens de fleste rutingsprotokollene for et sensornettverk benytter kringkasting. AODV pakkene er uansett for store for et sensornettverk. Når det gjelder 802.11, så bruker den RTS-CTS. Disse pakkene er også av urealistisk størrelse, og bør derfor ikke benyttes. Men selv om protokollene ikke er optimale, så vil vi uansett få målinger som er konsistente i forhold til hverandre. Man kan anta at simuleringen har foregått i et åpent landskap, slik at radioens sendeeffekt blir utnyttet optimalt. En annen ting er at det antaes at oppførselen til simuleringen som varer i 60 sekunder, er konstant, og at den videre utviklingen ikke endres, slik at man kan estimere levetiden til sensornodene. På denne måten får vi 6 driftssykluser i minuttet, hvor vi sender 2 pakker per driftssyklus.

6.2.2 Forklaring på oppsettet av simuleringene

Simuleringene tar for seg strømforbruket som oppstår i et sensornettverk. En sensornode dør når strømkapasiteten er brukt opp, og i dette tilfellet vil sensornoden ikke kunne lades opp. Det vi først og fremst ønsker å se på er strømforbruket som oppstår når ekstra sikkerhetsoverhead blir tilført, sammenlignet med en situasjon der ingen sikkerhet benyttes. Et sensornettverk kan tenkes å ha en nodetetthet på flere tusen noder innenfor et lite område, men dette er ikke tilfellet i denne sammenhengen. Dette har vi valgt å overse, og kun fokusere på strømforbruket for et begrenset antall sensornoder innenfor et lite område.

Simuleringene forgikk under antakelse om at sensornodene var manuelt utplassert, og etter utplassering forble sensornodene statiske, og ville forbli i samme posisjon hele tiden. Det ble foretatt flere simuleringer, med 1 fenomennode, 1 sink, og 25, 49 og 100 sensornoder i et område på 550m x 550m. Fenomennoden som i dette tilfellet var CO gass, genererte hendelser hvert 10. sekund, som sensornodene i området ville detektere og rapportere tilbake til sinken. Strømmodellen som allerede var implementert i NS-2 ble benyttet for å regne ut strømforbruket ved transmisjon av en pakke, samt ved mottakelse av en pakke. SensorSim muliggjorde også at det ble mulig å regne ut strømforbruket som oppstår ved detektering av en hendelse, noe NS-2 ikke var utstyrt til å håndtere i utgangspunktet. Strømmodellen var relativt enkel, og kalkulerte strømforbruket ved å multiplisere varigheten en hendelse forekom, med strømforbruket for en slik hendelse. Ellers så har standardverdier blitt benyttet hvis ikke annet er spesifisert.

Følgende parametere og verdier ble benyttet, og hadde innflytelse på strømforbruket i sensornettverk:

- Transmisjonsavstand
    - Avstanden til nærmeste nabo, horisontalt og vertikalt ble avgjort av hvor stor transmisjonseffekt som ble benyttet. Transmisjonsavstanden ble i noen tilfeller satt slik at hver sensornode kan ha maksimalt 4 naboer. Det ble også foretatt noen simuleringer hvor hver sensornode hadde 8 naboer.
    - Carrier sensing og mottaker sensitiviteten ble satt til samme verdi, slik at sensornodene bare kunne lytte til trafikk fra nabonodene. Disse verdiene var konstante under de fleste simuleringene, og det var for det meste kun transmisjonseffekten som ble endret.
- Antall sensornoder
    - 25, 49noder og 100 sensornoder
- Hvor ofte en hendelse inntreffer (pulsrate)

    - hvert 10ende sekund blir det generert en hendelse av fenomennoden
- Transmisjonsfrekvens
    - Antall ganger pr pulsrate en udp pakke skal sendes til sinken. Denne pakken var forskjellig i de ulike simuleringene, men en pakkestørrelse på 31, 37 og 42 bytes ble benyttet.
- Strømforbruk
    - txPower: Strømforbruket for å sende en pakke
    - rxPower: Strømforbruket for å motta en pakke
    - sensePower: Strømforbruket for å detektere en hendelse
    - initialPower: Total strømkapasiteten ved start av simuleringen
    - idlePower: Strømforbruket ved ledig tid. Dette har vi ikke tatt hensyn til i rapporten, så vi satte denne til å være null.
- Transmisjonseffekt: Varierer avhengig av hvor langt en ønsker å sende. I simuleringene ble transmisjonseffekten satt til 3 mWatt, 1mWatt samt 0,01 mWatt.
- MAC 802_11
    - 802_11 bruker følgende format for unikast pakker: RTS/CTS/DATA/ACK. Vi slo av RTS/CTS for å unngå overhead assosiert med disse kontrollpakkene. I tillegg til alle datapakkene som sendes, så vil MAC returnere en kvittering i form av en ACK, for å indikere at pakken er mottatt.
    - Ellers ble standardverdier benyttet:
        - CW_{min} = 31, mens CW_{max} = 1023.
        - SIFS = 10us, mens SlotTime = 20us.
        - PLCPHeaderLength = 48 bits, mens PreambleLength = 144 bit.
        - ShortRetryLimit = 7 forsøk. Mens LongRetryLimit = 4 forsøk.
        - LL delay = 25 us.
- AODV
    - Standardverdien for hvor ofte AODV skal utføre forespørsler om en rute var satt til 10 sekunder. Det betyr at så lenge det forekommer trafikk på en link i løpet av denne perioden, så vil AODV ikke foreta ny ruteforespørsel. Våre simuleringer genererer trafikk hvert 10ende sekund, noe som gjør at vi akkurat befinner oss i grenseland. Det betyr at i simuleringene, så foretar AODV kun forespørsler om ruter i begynnelsen av simuleringen. Deretter benyttes den samme ruten for resten.

Hvor lenge en sensornode lever i denne sammenhengen, avhenger av hvilket pakkeformat den bruker, dataraten, og kostnadene assosiert med å sende og motta data, samt kostnadene assosiert med å detektere en hendelse. Vi tar ikke hensyn til kostnadene assosiert med ledigmodus. På denne måten kan vi vise at sensornodene har en driftssyklus hvert 10ende sekund hvor de detekterende sensornodene sender 2 udp pakker, hvor vi antar at sensornoden er i en strømsparende tilstand når den ikke sender eller mottar data. Som sagt vil en sensornode dø når den går tom for strøm. Vi vil senere i rapporten kartlegge strømforbruket som oppstår i sensornettverket ved hjelp av tabeller og grafer, som vil illustrere levetiden til hver enkelt sensornode som har blitt benyttet som rute under simuleringen. Vi vil derimot ikke ta for oss levetiden til sensornoder som ikke har blitt benyttet, men kun de sensornodene som har sendt eller mottatt en eller flere udp pakker. I tillegg skal vi se på ende til ende forsinkelsen, og den gjennomsnittlig gjennomstrømningen som oppstår i hver av simuleringene.

6.2.3 Simulering av strømforbruket for de 3 pakkeformatene

Denne simuleringen tok for seg hvordan strømforbruket endret seg over tid for hvert av de 3 pakkeformatene. Dette ville gi en fin indikasjon på hvordan levetiden til sensornodene ville endre seg ettersom pakkestørrelsen økte på grunn av sikkerhetsoverhead. Strømforbruksmodellen til NS-2 var som sagt relativ enkel, og benyttet seg av standardverdier for transmisjon og mottak av data, samt for detektering av et fenomen, og strømforbruket ble så beregnet over tid. Det ble benyttet verdier fra tidligere i rapporten, hvor kostnadene for å transmittere en pakke var 96,6 mWatt (3 volt x 32,2 mAh), og kostnaden for å motta en pakke var 47,7 mWatt (3 volt x 15,9 mAh). Kostnaden for å detektere en hendelse ble satt til 3 uWatt, ettersom SensorSim kun benytter en fast verdi for å detektere et fenomen. I simuleringen ble sinken plassert øverst i høyre hjørne, mens fenomennoden ble plassert øverst i venstre hjørne. Det var ønskelig at hendelsene i simuleringen var mest mulig like, slik simuleringen av strømforbruket ble mest mulig konsistent i forhold til hverandre. På den måten kunne man få et ganske nøyaktig svar på hvordan de ulike pakkestørrelsene endret seg i forhold til hverandre. Denne simuleringen ble også forsøkt med 2 fenomen, men problemet var at pakketapet økte, og målingene med hensyn på strømforbruket ble ujevne. Derfor benyttet vi kun 1 fenomen.

Ved hjelp av simulering skal vi kartlegge hvordan strømkapasiteten til sensornodene endrer seg over tid avhengig av hvilket pakkeformat sensornodene bruker. Et annet interessant tilfelle er å kartlegge prosentvis levetid til en sensornode når sikkerhetsoverhead blir innført, i forhold til en pakke uten sikkerhet. En sensornode dør som sagt når den ikke har noe strøm igjen. Et problem vi opplevde i simuleringen når vi økte trafikkintensiteten var at når hendelsene inntraff hyppig, så hadde sensornodene en tendens til å få fulle buffer. Desto oftere sensornodene genererte trafikk, samt flere sensornoder som detekterte samme hendelse, desto fortere gikk bufferne fulle, noe som førte til at all trafikk ved disse sensornodene ble dumpet. Dette skyldes blant at det ble generert mange udp pakker fordi vi satt trafikkintensiteten til å være urealistisk høy (flere udp pakker ble generert i sekundet), samt at AODV og 802_11 protokollene sender kontrollpakker. Vi slo av RTS-CTS i 802_11, og benyttet AODV slik den er designet, samt reduserte generering av udp pakker til hvert 10 sekund. Det å slå av RTS-CTS førte til at det ble mindre overhead i sensornettverket ettersom RTS-CTS sendes for hver udp pakke som skal sendes. RTS brukes for å spørre om å få lov til å sende, mens CTS sendes tilbake om at det er klart å sende. RTS-CTS hindrer skjulte node, noe som sannsynligvis vil redusere antall kollisjoner. Vi testet en simulering ved å benytte RTS-CTS, og det førte som forventet til høyere strømforbruk, så derfor valgte vi å ignorere RTS-CTS. Vi slo også av RTS-CTS i alle de andre simuleringene. MAC protokollen som 802.11 bruker heter CSMA/CA. Denne protokollen lytter først til kanalen for å se om den er ledig. Hvis ledig, så sender den pakken. Hvis opptatt, så venter den i en tilfeldig tid. Det vi gjorde var å sette carrier sensing rekkevidden lik mottaker rekkevidden, slik at en sensornode ikke kunne detektere trafikk utenfor sin egen mottaker radius. På denne måten kan man unngå overhøring av andres kommunikasjon.

Det hadde vært interessant å teste ut en rutingsprotokoll for et sensornettverk som baserer seg på kringkasting av datapakker. Hvilke rutingsprotokoller som er tiltenkt et sensornettverk, og hvordan de virker er forklart i kapitel 2. Men ettersom ingen slik rutingsprotokoller var utviklet for SensorSim, så ble det anbefalt av [19] å benytte AODV [26]. AODV benytter den samme stien i nettverk, og den sender ut forespørsler (RREQ) gjennom nettverket når den trenger å forhandle om en rute. De sensornodene som har en rute sensornoden kan benytte sender et svar tilbake (RREP) om at den har en rute tilgjengelig mot destinasjonen. RREQ pakkene kringkastes til sensornodens 1 hopps naboer. Slik fortsetter det gjennom hele

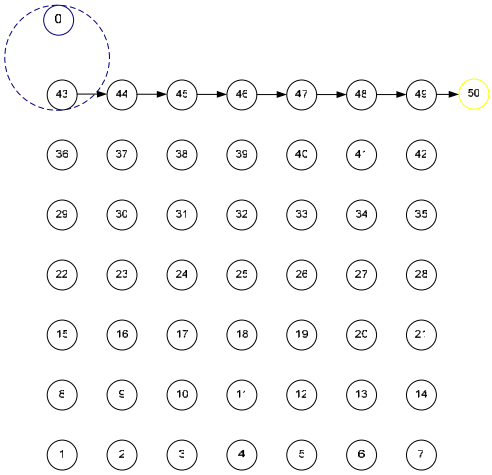
sensornettverket, helt til enten destinasjonsnoden mottar RREQ, eller hvis en sensornode som har en rute til destinasjonen mottar RREQ. Alle sensornodene som mottar en RREQ svarer med en RREP hvis de har en rute tilgjengelig mot destinasjonen. Dette fører til mye trafikk, men garanterer at en rute vil blir funnet, hvis det er en tilgjengelig rute. RREQ kringkastes ut i nettverket, mens RREP sendes tilbake via unikast. Disse meldingene sendes kun ut når en sensornode har behov for en rute. Straks en sensornode mottar en RREP, så kan den sende data til destinasjonen. Hver enkelt sensornode videresender en RREP tilbake kun en gang. Det vil si at hvis en sensornode mottar en RREP fra 3 andre sensornoder, så bare overser den disse. Dette førte til en del utveksling av kontrollpakker, og hvor hver pakke får en rute som pakken kan benytte. Denne ruten vedlikeholdes så lenge sensornoden har behov for den. Det vil si at det periodisk sendes pakker over linken (Så lenge det er aktivitet på linken innen 10 sekunder er gått, men hvis det går mer enn 10 sekunder uten aktivitet på linken, så må nye rutingsforespørsler sendes ut). I denne simuleringen så foretaes det kun rutingsforespørsler når første pakke sendes, deretter benyttes samme rute hele tiden. Bruken av AODV i vår simulering vil som nevnt tidligere avvike litt, men vil uansett gi oss en indikasjon på hvordan økt pakkeformat endrer levetiden til sensornodene. Det ble foretatt 3 simuleringer for å se på strømforbruket, en for en pakkestørrelse på 31 bytes, en for 37 bytes, samt en for 42 bytes.
I simuleringen ble det benyttet følgende parametere:

Verdier	**Simulering 1**	**Simulering 2**	**Simulering 3**
Avstand mellom noder Horisontalt/Vertikalt (meter)	74/63	74/63	74/63
Sendeeffekt (mWatt)	3,0	3,0	3,0
RX/CS sensitivitet (meter)	75,2	75,2	75,2
Datarate (bps)	38400	38400	38400
Antall sensornoder	49	49	49
Pulsraten (pr sekund)	10	10	10
Transmisjonshyppighet (pr sekund)	2	2	2
Frekvens (MHz)	868	868	868
TX power (mWatt)	96,6	96,6	96,6
RX power (mWatt)	47,7	47,7	47,7
SensePower (uWatt)	3,0	3,0	3,0
Strømkapasitet (J/s)	27000	27000	27000
Simuleringens varighet (sekunder)	60.2	60.2	60.2
Pakkestørrelse (bytes)	31	37	42
RTS-CTS	AV	AV	AV

Tabell 24. Parameterverdier som ble bruk for å simulere strømforbruket for de 3 pakkeformatene.

6.2.3.1 Resultatet av simuleringen av strømforbruket for de 3 pakkeformatene

I simuleringen tar vi for oss strømforbruket i sensornettverket langs en rute. AODV satte opp rutingstabellen på følgende måte:



Figur 36. Rutingstopologien ved beregning av strømforbruk for 31, 37 og 42 bytes pakker.

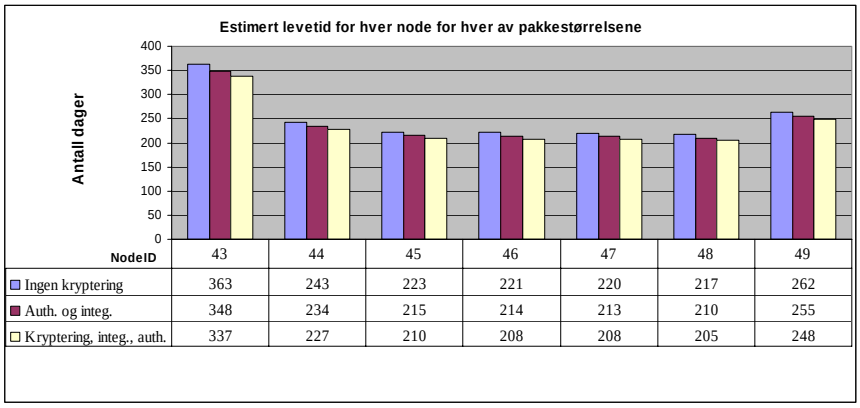
Figur 36 viser rutingstopologien, samt hvilken retning sensornodene sender pakkene som mottaes, eller sender ut på grunn av at de har detektert en hendelse. Den blå sirkelen viser hvilket område hendelsen kan detekteres i. Det vil si at det kun er sensornode 43 som detekterer hendelsene i denne simuleringen. Når sensornode 43 detekterer hendelser, så rapporteres hendelsene tilbake til sinken ved å sende 2 pakker av en gitt størrelse (31, 37 eller 42 bytes) til sensornode 44, som igjen videresender pakken mot sinken. Det er dessuten i dette området at aktiviteten og strømforbruket vil være høyest (langs ruten mellom sensornoden som detekterer hendelsen, og sinken). Den gule sensornoden (node 50) er sinken. Det er antatt at sinken har såpass høy batterikapasitet at den vil leve lengre enn alle sensornodene i sensornettverket, så derfor er ikke strømforbruket i sinken tatt med. Simuleringen gav oss følgende strømforbruk for hvert av pakkeformatene:

NodeID (langs ruten)	Simulert levetid for 31 bytes (ant. dager)	Simulert levetid for 37 bytes (ant. dager)	Simulert levetid for 42 bytes (ant. dager)	Nedgang i levetid fra 31 til 37 bytes (i %)	Nedgang i levetid fra 31 til 42 bytes (i %)
43	363	348	337	4,1	7,2
44	243	234	227	3,7	6,6
45	223	215	210	3,6	5,8
46	221	214	208	3,2	5,9
47	220	213	208	3,2	5,5
48	217	210	205	3,2	5,5
49	262	255	248	2,7	5,3

Tabell 25. Levetiden for hver sensornode, samt nedgang i levetid i sammenlignet uten sikkerhet.

Tabell 25 viser en oversikt over levetiden til hver enkelt sensornode langs ruten mellom sensornoden som detekterer en hendelse, og helt frem til sinken. Altså fra sensornode 43, og til sinken, som er node 50. Ut ifra tabellen kan vi se at levetiden for hver av sensornodene er avhengig av hvilken pakkestørrelse som benyttes. Disse verdiene fikk vi ved å benytte simulering med en varighet på 60,2 sekunder. Deretter antok vi at den videre utviklingen forble konstant, slik at vi kunne estimere levetiden til hver enkelt sensornode. I tillegg viser de 2 siste kolonene hvordan den prosentvise nedgangen i levetid ble for hvert av pakkeformatene i forhold til det opprinnelige pakkeformatet på 31 bytes. Det viste seg som først antatt, at en pakkestørrelse på 37 bytes (autentisering og integritet) og 42 bytes (konfidensialitet), gir en redusert levetid i forhold til et pakkeformat uten noen som helst sikkerhetsoverhead. Denne reduseringen varierte fra 2,7 % til 4,1 % ved innføring av

autentisering og integritet, mens nedgangen i levetid ved innføring av kryptering varierte fra 5,3 til 7,2 % for de forskjellige sensornodene i forhold til det opprinnelige pakkeformatet. Det er som forventet at det er de mellomliggende sensornodene som opplever høyest strømdegresjon, og det er også disse sensornodene som har lavest levetid. Ut ifra disse verdiene, så ser det ikke ut til at sikkerhet har så stor betydning på levetiden til en sensornode. For en pakkestørrelse som kun innfører autentisering og integritet, så er ikke dette så merkbart, men levetiden reduseres mer ved innføring av kryptering. Dette er ikke en ekstrem redusering, og man kan godt innføre denne type sikkerhetsgrad hvis det er nødvendig. Avhengig av hvor viktig dataet er, så kan man fint innføre sikkerhet på bekostning av mindre levetid. Men, så lenge konfidensialitet ikke er nødvendig, så kan det uansett være greit å innføre autentisering og integritet, siden det ikke vil drastisk degradere levetiden til en sensornode. Figuren under viser en grafisk illustrasjon over levetiden til de forskjellige sensornodene.



NodeID	43	44	45	46	47	48	49
Ingen kryptering	363	243	223	221	220	217	262
Auth. og integ.	348	234	215	214	213	210	255
Kryptering, integ., auth.	337	227	210	208	208	205	248

Figur 37. Sammenligning av levetiden til hver enkelt sensornode ved bruk av ulike pakkestørrelse.

I grafen ser man at for hvert av pakkeformatene, avhengig av om sikkerhet er innført, eller hvilken type sikkerhet som benyttes, så vil hver sensornode ha lavere levetid avhengig av sikkerhetsgraden. Y-aksen viser antall dager en sensornode lever, mens x-aksen viser hvilken sensornode det er snakk om. En annen interessant observasjon er at sensornodene 43 og 49 har lengre levetid enn sensornodene 44–48. Det at sensornode 43 har lengre levetid har med at det er sensornode 43 som detekterer hendelsene, noe som gjør at den sender ut 2 udp pakker for hver hendelse som inntreffer, mens de andre sensornodene imellom må operere som rutere. Det vil si at de må motta pakker, for så å rute pakkene videre mot sinken. Videre i figuren så ser man at sensornode 49 har høyere levetid enn sensornode 44–48, noe som egentlig ikke skulle vært tilfellet. Sensornode 49 skulle egentlig hatt tilsvarende levetid som node 44–48 ettersom det kun er 1 sensornode som detekterer fenomenet. Med det menes at hvis det hadde blitt generert hendelser flere steder i sensornettverket, så ville sannsynlig sensornodene nærmest sinken, deriblant sensornode 49, mottatt trafikk fra flere retninger, noe som ville ført til høyere trafikkintensitet i nærheten av sinken. En grunn til at sensornode 49 i denne sammenhengen hadde høyere levetid enn sensornodene 44–48, var at sensornodene 44–48 opplevde flere kollisjoner, samt at de sendte og mottok mer data enn sensornode 49. Dette er data som blant annet AODV pakker, og MAC pakker. Disse kontrollpakkene førte til økt strømforbruk ettersom AODV kringkaster ruting forespørsler til sine nærmeste naboer, og hvor MAC krever en ACK for hver pakke som sendes. Når en av disse kontrollpakkene blir tapt, så sender sensornoden ut en ny forespørsel. I vårt tilfelle, så benyttet vi bare en fenomennode. Hadde vi benyttet flere fenomennoder, så ville sensornoden nærmest sinken opplevd klart størst belastning. Dette har med at disse sensornodene må videresende pakker til sinken fra alle de andre sensornodene i sensornettverket. For å teste ut denne teorien, så prøve

vi å kjøre simuleringen med 2 fenomennoder, en oppe i venstre hjørne, og en nede i høyre hjørne. Dette gav forventet resultat, men pakketapet økte en god del, noe vi egentlig ikke ønsket å oppleve for mye av. Denne simuleringen viste at strømforbruket hos sensornodene nærmest sinken økte i forhold til de andre sensornodene, ettersom sensornodene nærmest sinken måtte videresende pakker også for andre sensornoder, noe vi også skal se på i kapitel 6.2.5.

I tillegg til strømforbruket, så beregnet vi også ende til ende forsinkelsen og gjennomsnittlig gjennomstrømning (gjennomstrømningen i hele sensornettverket dividert på tiden som først pakke ble sendt og til siste pakke ble mottatt (50.2 sekunder)). Ende til ende forsinkelsen inkluderte all forsinkelse som oppsto i sensornettverket fra det tidspunktet en pakke ble sendt fra applikasjonslaget til senderen, og helt til pakken ble mottatt i applikasjonslaget hos mottaker. Dette inkluderer blant annet bufring hos hver sensornode, forsinkelse på datalinklaget (medium aksess), samt tiden det tar for en pakke å propagere gjennom sensornettverket.

Det ble observert at ved forsendelse av 12 udp pakker i løpet av 60,2 sekunder, så forekom det 73 kollisjoner i hele sensornettverket i hvert av de 3 tilfellene. Langs ruten forekom det kun 10 kollisjoner totalt, mens det ikke forekom tapte udp pakker på applikasjonslaget. Sensornode 43 sendte ut 12 udp pakker mot sinken, hvor sinken mottak alle disse udp pakkene. De resterende kollisjonene forekom på andre steder i sensornettverket. Hvis simuleringen hadde benyttet protokoller av typene MAC og AODV som hadde vært spesielt beregnet for et sensornettverk, så ville sannsynligvis levetiden til hver enkelt sensornode vært litt høyere i forhold til det denne simuleringen har vist, ettersom at kontrollpakkene ville vært mindre. Hvis en rutingsprotokoll som kringkaster pakkene ut i sensornettverket hadde blitt brukt, så ville dette ført til mye høyere strømforbruk i hele sensornettverket, og ikke kun langs ruten, noe som vi har opplevd i vår simulering. Sensornettverket en del overhead på grunn av at AODV kringkastes ut i sensornettverket, men dette gjelder kun første gang en datapakke skal sendes. Etter dette er det for det meste overhead assosiert med MAC som oppstår, og da i forbindelse med ACK pakkene som sendes for hver pakke. Ved bruk av kringkasting så ville alle datapakker blitt kringkastet hos hver node som mottar en datapakke. Dette ville ført til høy trafikk, samt at de ville forekommet mange kopier av samme melding. Uansett ville vi opplevd en tilsvarende utvikling, ved blant annet at forsinkelsen for hvert av pakkeformatene ville økt, samt at strømforbruket også øker avhengig av hvilket pakkeformat som benyttes.

Tabell 26 viser en total oversikt over antall sendte, mottatte, videresendte og droppede pakker for hver av sensornodene langs ruten. I tillegg viser tabellen også gjennomsnittlig, maksimum og minimum ende til ende forsinkelse, samt gjennomsnittlig gjennomstrømning.

Pakke størrelse (bytes)	Ant. sendte pakker (udp)	Ant. mottatte pakker (udp)	Ant. droppede pakker (udp)	Ant. kollisjoner	Gj. snittlig ende til ende forsinkelse (ms)	Maksimum ende til ende forsinkelse (ms)	Minimum ende til ende forsinkelse (ms)	Gj. snittlig gjennom-strømning (bps)
31	12	12	0	73	308,8	906,1	199	1588
37	12	12	0	73	317,9	914,8	207,7	1588
42	12	12	0	73	325,4	922,1	215,3	1588

Tabell 26. Resultater i forbindelse med simuleringen av strømforbruket for de ulike pakkeformatene.

Ende til ende forsinkelsen varierte veldig, fra minimum 199, 207,7 og 215,3 ms avhengig om pakkestørrelsen er på 31 bytes, 37 eller 42 bytes, til maksimum 906,1, 914,8, 922,1 ms. Dette viser at desto større en datapakke blir, desto høyere blir forsinkelsen. Dette har som sagt med at det tar lengre tid å sende en pakke ettersom størrelsen på pakken øker. En annen ting som

påvirker ende til ende forsinkelsen er at datapakkene rutes via multihopp, noe som fører til høyere forsinkelse ettersom hver sensornode som operer som ruter må motta, for så å videresende hver pakke. Dessuten ville nok forsinkelsen vært høyere hvis simuleringen hadde tatt hensyn til kryptering og dekryptering. Dette fordi det ville oppstått en del forsinkelse hos hver sensornode avhengig om det kun hadde vært ende nodene som foretar kryptering og dekryptering, eller om hver enkelt mellomliggende sensornode også ville foretatt kryptering, samt dekryptering. Avhengig av hvilken krypteringsalgoritme som benyttes, og hvor rask algoritmen er, så vil det oppstå en høyere forsinkelse. Gjennomstrømningen i denne simuleringen lå konstant på 1588. Det har med at pakkene som sendes fra applikasjonslaget var relativt like, samt at tilfellene var relativt like, det vil si at pakketapet og antall kollisjoner var like, samt varigheten til simuleringene.

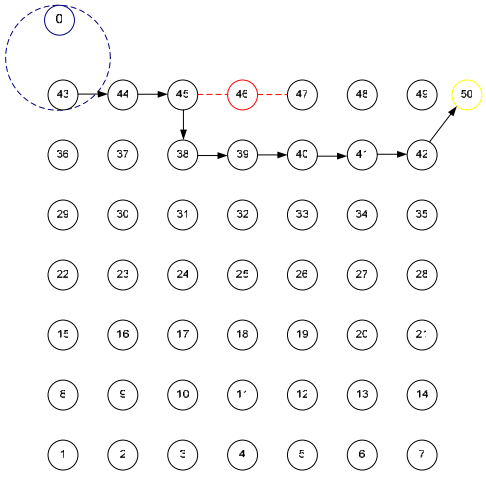
Protokollene som NS-2 simulatoren benytter er hovedsakelig beregnet for 802.11 og ad hoc nettverk. Bruken av disse protokollene førte til at levetiden til sensornodene ble litt lavere, ettersom kontrollpakkene som disse protokollene bruker er større enn de udp pakkene som den detekterende sensornodene sendte mot sinken, samt større enn kontrollpakker som burde brukes i et sensornettverk. Men uansett hvilke protokoller som benyttes, så vil uansett sensornodene ha ulik levetid avhengig av sikkerhetsoverheaden og pakkestørrelsen. Ut ifra disse simuleringene, så ser man at sensornoder som ligger mellom sensornoden som observerer fenomenet, og sinken, opplever høyest strømtap. Dette er logisk, siden sensornoden som detekterer fenomenet, kun sender ut udp pakker, og sinken kun mottar udp pakker, mens alle sensornodene som ligger i ruten mellom sensornoden som detekterte hendelsen, og sinken, må operere som rutere, ved at de både må imot pakker, for så å videresende dem mot sinken. En av grunnene til at såpass få pakker gikk tapt, var blant annet fordi trafikkintensiteten var relativt lav (2 udp pakker hvert 10ende sekund), samt at pakkene som ble sendt, var relativt små. Det å slå av RTS-CTS sparte sensornettverket for mye overhead, samt at det økte gjennomstrømningen. Ved bruk av RTS-CTS, så transmitterer først sender en forespørsel om å få lov til å transmittere en pakke til en sensornode, for så å vente til den får klarsignal til å transmittere. Det hadde også vært ønskelig å prøve ut en rutingsprotokoll som baserer seg på kringkasting, slik som mange av rutingsprotokollene i et sensornettverk. I simulering ble det altså benyttet en rutingsprotokoll som baserer seg på faste ruter, mens de fleste rutingsprotokollene i et sensornettverk baserer seg på kringkasting. I simuleringen opplevde vi at pakkene benyttet samme rute i sensornettverket for alle pakker som ble sendt, ettersom AODV sender ut rutingsinformasjon før forsendelse av en udp pakke. Dette ble veldig forutsigbart. Mens i et sensornettverk som benytter ruting basert på kringkasting, så vil en udp pakke bli sendt til alle naboene til en sensornode. Forskjellen er at ved bruk av AODV, så blir kun sensornettverket påvirket når RREQ sendes i begynnelsen av kommunikasjonen. Deretter foregår all kommunikasjon ved hjelp av unikast. Det vil si at den samme ruten i sensornettverket brukes for hver pakke, og på denne måten så er det kun sensornodene langs ruten som blir påvirket av trafikken. Dessuten vil det ikke forekomme flere kopier av samme melding i sensornettverket, ettersom AODV kun sender ut en melding direkte til nærmeste nabo. Hvis for eksempel vi hadde brukt en rutingsprotokoll som baserer seg på kringkasting, så ville større deler av sensornettverket blitt rammet, ettersom alle sensornoder kringkaster en pakke de mottar. Dessuten ville dette også ført til at alle sensornodene i sensornettverket ville opplevd høyere strømforbruk, spesielt de sensornodene som måtte befinne seg nærmest sinken. Ved bruk av en rutingsprotokoll som baserer seg på kringkasting, så kan man garantere at pakkene kommer frem, men det vil forekomme mange kopier av samme melding i sensornettverket, noe som vil føre til høyere belastning, samt høyere strømforbruk i hele sensornettverket. Det at det forekommer mange kopier av samme

melding i sensornettverket vil føre til at sensornodene nærmest sinken vil måtte motta flere kopier av samme melding, noe som vil være veldig ressurskrevende.

Det var dessuten ønskelig å laste ned en AODV rutingsprotokoll som baserer seg på multicast AODV. Men det viste seg at denne utvidelsen ikke var spesielt beregnet for NS-2 simulatoren [23] på nåværende tidspunkt, samt at den inneholdt mye feil.

6.2.3.2 Levetiden til en rute

Ut ifra simuleringene i 6.2.3.1, kan man også se på levetiden til en rute i sensornettverket som går fra en sensornode og videre mot sinken. En rute har like lang levetid som sitt svakeste ledd, det vil i denne sammenhengen være den sensornoden med lavest strømkapasitet. Avhengig av antall pakker en sensornode sender eller mottar, så vil levetiden til sensornodene degraderes deretter. Rutingen i denne topologien var statisk, men når en av sensornodene dør, så vil AODV måtte opprette en ny rute ettersom den tidligere ruten ikke er operativ lengre. Dette indikerer at et sensornettverk er avhengig av en rutingsprotokoll som er veldig strømeffektiv. Hvis sensornettverket bruker den samme ruten til å sende pakkene hele tiden, så vil dette føre til at visse sensornoder dør fortere ut. Hvis et sensornettverk bruker en slik type rutingsprotokoll som AODV, så hadde det vært en fordel at strømforbruket ble fordelt utover sensornettverket, slik at ikke bare visse sensornoder benyttes som rutere. Men dette ville derimot krevd flere kontrollpakker for å realiseres. Dette kan heller ikke demonstreres i denne sammenhengen, ettersom det ikke eksisterer slike protokoller med denne egenskapen. Det er ønskelig at rett før en sensornode dør, så sender sensornoden ut en melding til naboene sine om at den har liten batterilevetid igjen. Hvis det sendes pakker til sensornoden etter at den har dødd, så vil pakkene droppes. Neste pakke vil måtte rutes i en ny retning. I simuleringen som den først var satt opp, så viste det seg at når en sensornode døde, så stoppet den døde sensornoden å videresende pakker, mens sensornoden som brukte den døde sensornoden som rute, bare fortsatte å bruke den, uten å opprette en ny rute. Dette førte til at all trafikk hos den døde sensornoden stoppet opp. Dette hadde med at AODV beholder en rute så lenge det sendes data periodevis over linken. Altså så lenge det ikke gikk en periode på mer enn 10 sekunder uten trafikk, så beholdes samme rute. Det den døde sensornoden skulle ha gjort, var å gi tilbakemelding til de andre sensornodene om at den snart var tom for batteri, og at ruten mot den snart vil slutte å eksistere. På denne måten ville de andre sensornodene blitt klar over dette, og kunne da foretatt seg noe, som blant annet å danne en ny rute. Vi kunne løst problemet på 3 måter, enten ved å få hendelsene til å genereres for eksempel hver 11. sekund, eller endre verdien i aodv.h som styrer hvor lenge en rute kan være inaktiv. En siste mulighet ville være å implisitt sette sensornode 46 til en strømkapasitet lik null allerede ved oppstart. I alle tilfellene dannet AODV følgende rutingstopologi:



Figur 38. Ruteendring når en sensornode dør ut.

Som sagt, så ville det vært ønskelig at når en sensornode dør, så vil de andre sensornodene danne en ny rute rundt den døde sensornoden kanskje allerede før den døde ut. Rutingsprotokoller som utvikles for et sensornettverk bør være utstyrt med egenskaper som gjør at sensornoden kan velge ruter som er strømeffektive. Det vil si at kostnaden ved å sende en pakke via strømeffektive ruter er lavere enn ved andre ruter. En annen viktig egenskap rutingsprotokollene må ha, er en slags tilbakemeldingsmekanisme slik at sensornoden med lav strømkapasitet kan gi de andre sensornodene i sensornettverket beskjed om at den snart er tom for strøm, og at den ikke kan mottatt flere pakker. Sensornoden kan for eksempel sende ut en melding om at den snart er tom for strøm når gjenværende strømkapasitet nærmere seg en viss terskel. På denne måten kan de andre sensornodene få beskjed om dette, og på denne måten foreta seg noe.

For at vi skulle få AODV til å danne en ny rute rundt sensornode 46 under simuleringen, så endret vi 2 verdier i aodv.h. Disse verdiene het MY_ROUTE_TIME og ACTIVE_ROUTE_TIME, og vi satte dem til å være 5 sekunder. Det betyr at en rute i AODV er aktiv så lenge det forekommer trafikk i løpet av 5 sekunder. Etter at vi endret disse verdiene, så sendte AODV ut rutingsforespørsler for hver pakke som skal sendes hvis det ikke forekommer trafikk i løpet av 5 sekunder. På denne måten vil den finne ut om en rute er nede. Dette var ikke tilfellet i forrige simulering, ettersom trafikk ble generert hvert 10ende sekund, noe som akkurat var innenfor tiden slik at AODV ikke trengte å sende ut AODV pakker for hver hendelse for oppstår. Derfor benyttet AODV den samme ruten hele tiden, uten å sende forespørsel om ny rute. I dette tilfellet så vil sensornettverket oppleve mye mer trafikk ettersom AODV nå vil sende ut RREQ og RREP for hver udp pakke som skal sendes. Vi benyttet en simuleringstid på 202 sekunder, samt at hver hendelse ble generert hvert 10ende sekund. Dessuten satte vi strømkapasiteten til sensornode 46 til å være mye lavere enn de andre sensornodene, noe som førte til at sensornode 46 døde ut etter at simuleringen hadde vart i cirka 60 sekunder. Dette førte til at 1 udp pakke gikk tapt av totalt 20 som ble sendt. Dette pakketapet oppsto da sensornode 46 gikk tom for strøm, og på grunn av dette måtte trafikken rutes om sensornode 46. Men som sagt, så hadde det vært en fordel om at sensornode 46 hadde gitt tilbakemelding til sensornode 45 om at den snart er tom for strøm, og at en ny rute måtte kartlegges. Dette ville muligens ført til at ingen udp pakker ville gått tapt på grunn av at sensornode 46 døde, fordi at hvis sensornode 46 kunne gitt beskjed til sensornode 45 om at den snart var tom for strøm, og da kunne sensornode 45 rutet trafikken via sensornode 38 i det hele tatt før sensornode 46 døde ut. Da ville sannsynligvis alle de 20 pakkene kommet frem til sinken.

6.2.4 Strømforbruket ved ulik nodetetthet

Denne simuleringen har som hensikt å illustrere hvordan strømforbruket til sensornodene i sensornettverket endres ettersom nodetettheten øker og distansen mellom sensornodene reduseres. Dette kan illustreres ved å ta strømbruket ved å sende over lengre distanse (høyeste sendeeffekt), og sammenligne det mot strømforbruket ved å sende over kortere distanse (lavere sendeeffekt), og da over flere hopp. Vi kan da forestille oss at det er 3 forskjellige sensornoder, som hver har forskjellig sendeeffekt. En sensornode kan sende på 3 mWatt, noe som tilsvarer et strømforbruk på 32,2 mAh, en annen kan sende på 1 mWatt, som tilsvarer 23,3 mAh, og den siste sensornoden kan sende med en effekt på 0,01 mWatt, som tilsvarer 14,9 mAh. Hvis disse sensornodene skal operere innenfor samme område, og hvis det er lik avstand mellom sendernoden, og sinken, så er sensornodene avhengig av at nodetettheten er tilfredsstillende slik at pakkene kan rutes fra sendernoden til sinken ved bruk av den gitt sendeeffekten. I disse tilfellene sender sensornodene med høyest, middels, og laveste sendeeffekt. Denne sendeeffekten er spesifisert i manualen til radioen [46], samt tabell 13.

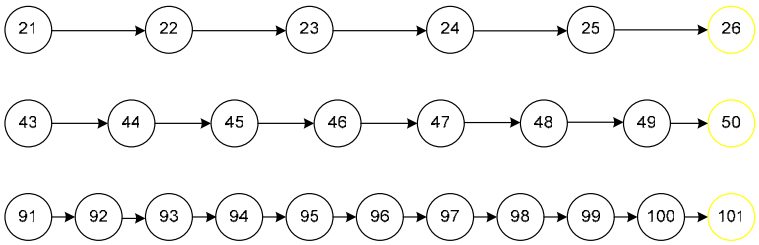
Det ble benyttet variert antall sensornoder i simuleringen for å få økt nodetettheten tilstrekkelig slik at man også kan transmittere over lavere sendeeffekt. Totalt 25 noder, 49 og 100 noder ble benyttet i hver av simuleringene. Det ble da foretatt antakelser om at strømforbruket var 32,2 mAh for å transittere når man benytter 25 noder med en avstand på 100 meter mellom sensornodene, mens strømforbruket var 23,3 mAh ved å transmittere en pakke når 49 noder benyttes med en avstand på 75 meter mellom sensornodene, mens strømforbruket var 14,9 mAh når 100 noder benyttes med en avstand på 50 meter mellom sensornodene. Dette vil skje over samme område, så derfor var det viktig at nodetettheten økte ettersom lavere sendeeffekt ble benyttet. I simuleringene ble det benyttet følgende parametere:

Verdier	Simulering 4	Simulering 5	Simulering 6
Avstand mellom noder Horisontalt/Vertikalt (meter)	99/99	74/63	49/44
Sendeeffekt (mWatt)	3,0	1,0	0,01
RX/CS sensitivitet (meter)	100	75	50
Datarate (bps)	38400	38400	38400
Antall sensornoder	25	49	100
Pulsraten (pr sekund)	10	10	10
Transmisjonshyppighet (pr sekund)	2	2	2
Frekvens (MHz)	868	868	868
TX power (mWatt)	96,6	69,9	44,7
RX power (mWatt)	47,7	47,7	47,7
SensePower (uWatt)	3,0	3,0	3,0
Strømkapasitet (J/s)	27000	27000	27000
Pakkestørrelse (bytes)	31	31	31
RTS-CTS	AV	AV	AV

Tabell 27. Oversikt over data som ble brukt ved simulering av nodetettheten.

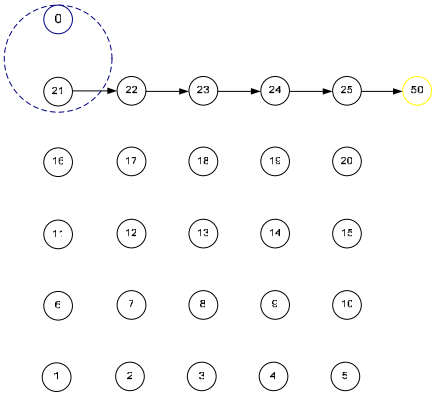
6.2.4.1 Resultatet av simuleringen av strømforbruket ved ulik nodetetthet

Det er strømforbruket som er viktig i denne sammenhengen, så ved å simulere med forskjellige nodetetthet, så kan man også benytte lavere sendeeffekter, noe som vil gi et svar på hvilken betydning forskjellene i sendeeffekt har på strømforbruket i sensornettverket. Figur 39 illustrerer tankegangen, ved at over samme område, så er det nødvendig med høyere sendeeffekt desto lavere nodetetthet som eksisterer, mens ved høyere nodetetthet, så kan trafikken rutes over kortere avstand, og heller via flere hopp, noe som vil redusere strømforbruket.

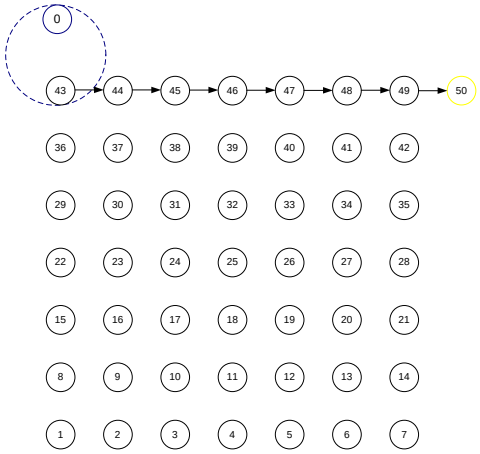


Figur 39. Lengre avstand, færre hopp. Kortere avstand, flere hopp.

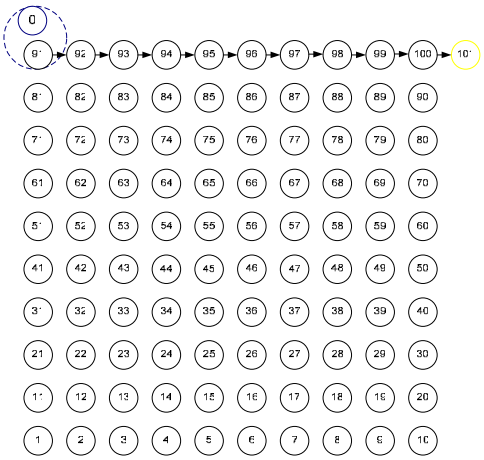
Det som også er viktig her, er å ta hensyn til hvor lang avstand det er mellom sender og mottaker ved 1 enkelt hopp, samt hvor mange noder det er ved multihopp, og hvor lang avstand det er mellom hver enkelt av sensornodene. Meningen er som sagt å illustrere at inni et område som er like stort for alle tilfellene, og ettersom nodetettheten øker, så kan sensornodene benytte lavere sendeeffekt, og heller foreta flere hopp for å nå sinken, og på denne måten bevare strømkapasiteten lengre. Derfor er det viktig at sensornodene er i stand til å foreta en effektiv effektkontroll, slik at den selv klarer å kalkulere avstanden til sine naboer, og på denne måten kunne kalkulere den optimale sendeeffekten. AODV dannet følgende rutingstopologi:



Figur 40. Sensornettverk bestående av 25 sensornoder.



Figur 41. Sensornettverk bestående av 49 sensornoder.



Figur 42. Sensornettverk bestående av 100 sensornoder.

Figurene 40, 41 og 42 viser en illustrasjon over hvordan sensornettverket blir seende ut ettersom nodetettheten øker. Avstandene mellom sensornodene i de forskjellige tilfellene er 100, 75 og 50 meter. Det vil si at sensornodene ut ifra nodetettheten, må rute pakker gjennom 4, 6, og 9 sensornoder før pakken når sinken. For at sensornodene skal kunne nå sine nærmeste naboer over disse avstandene, så er det antatt at sendeeffektene må være på henholdsvis 3mW, 1mW og 0,01 mW. Altså desto lengre avstand, desto høyere sendeeffekt er nødvendig. Sensitiviteten på mottakeren er den samme for alle tilfellene. Ettersom nodetettheten øker, så er det ikke nødvendig lengre å måtte benytte så høy transmisjonseffekt, noe som resulterer i at kostnadene ved å transmittere en pakke vil være mindre, og at pakken heller må rutes via flere sensornoder før den når sinken. Hvis sensornodene hadde vært utstyrt med en intelligent effektkontroll, så kunne sensornodene beregnet optimal sendeeffekt, og på den måten fordelt strømkostnadene utover i sensornettverket gjennom flere hopp.

Simuleringene gav oss følgende resultater:

Node nr (31 bytes)	Strømforbruk pr minutt (joule)	Strømforbruk pr time (joule)	Simulert levetid (ant. dager)
21	0,036831	2,20986	364
22	0,054577	3,27462	233
23	0,055591	3,33546	222
24	0,055835	3,3501	223
25	0,045837	2,75022	264

Tabell 28. Strømforbruket ved simulering av en nodetetthet på 25 sensornoder.

Node nr (31 bytes)	Strømforbruk pr minutt (joule)	Strømforbruk pr time (joule)	Simulert levetid (ant. dager)
43	0,029947	1,79682	437
44	0,045452	2,72712	286
45	0,047695	2,8617	259
46	0,048108	2,88648	256
47	0,048789	2,92734	256
48	0,049748	2,98488	253
49	0,038706	2,32236	314

Tabell 29. Strømforbruket ved simulering av en nodetetthet på 49 sensornoder.

Node nr (31 bytes)	Strømforbruk pr minutt (joule)	Strømforbruk pr time (joule)	Simulert levetid (ant. dager)
91	0,024881	1,49286	539
92	0,037766	2,26596	345
93	0,041154	2,46924	306
94	0,041572	2,49432	303
95	0,041155	2,4693	306
96	0,040828	2,44968	308
97	0,040935	2,4561	307
98	0,041566	2,49396	303
99	0,040988	2,45928	307
100	0,03089	1,8534	394

Tabell 30. Strømforbruket ved simulering av en nodetetthet på 100 sensornoder.

Tabellene 28, 29 og 30, viser helt klart at levetiden til hver enkelt sensornode i sensornettverket blir forlenget ettersom nodetettheten øker. Når nodetettheten øker, så vil sensornodene kunne transmittere med mindre effekt, ettersom avstanden til nærmeste nabo blir mindre ved høyere tetthet. Dette gjør at en pakke må rutes via flere hopp desto høyere tettheten er, men det vil igjen føre til at strømforbruket blir mer fordelt utover sensornettverket. Når avstanden er 100 meter, så har den sensornoden med lavest strømkapasitet kun en levetid på 222 dager, mot 253 og 303 dager i de andre tilfellene. Gjennomsnittelig levetid i hvert av tilfellene er 261, 295, og 341 dager. Men desto høyere

trafikkintensiteten er, desto lavere vil levetiden til sensornodene som opplever denne trafikkintensiteten være. Dette skal vi se på i neste kapitel.

Videre i simuleringen kartla vi også resultater som antall tapte, sendte og mottatte pakker på applikasjonslaget. I tillegg kartla vi maksimum, minimum, og gjennomsnittlig ende til ende forsinkelse for pakkene, samt gjennomsnittlig gjennomstrømning.

Nodetetthet (ant. noder)	Ant. Sendte pakker (udp)	Ant. Mottatte pakker (udp)	Ant. droppede pakker (udp)	Antall kollisjoner	Gj. Snittlig ende til ende forsinkelse (ms)	Maksimum ende til ende forsinkelse (ms)	Minimum ende til ende forsinkelse (ms)	Gj. snittlig gjennom-strømning (bps)
25	12	12	0	30	218,5	662,5	140,3	1590
49	12	12	0	73	308,8	906,1	199	1588
100	12	12	0	146	441,4	1257,1	288,2	1586

Tabell 31. Ulike resultater fra simuleringen av strømforbruket ved ulik nodetetthet.

Tabellen viser at ingen udp pakker gikk tapt, men ettersom nodetettheten økte, så forekom det en variert mengde kollisjoner, noe som skyldes at trafikkintensiteten i hele sensornettverk økte. I første tilfelle så oppsto det 30 kollisjoner, mens i de andre tilfellene oppsto det 73, og 146 kollisjoner. Dette gjelder for hele sensornettverket, og er for det meste pakker som AODV kontrollpakker. Det vil si forespørsel om en rute, og slike typer pakker. Ettersom tettheten økte, så sendte AODV flere rutingsforespørsler gjennom sensornettverket, noe som igjen førte til at pakketapet økte. Dette merkes spesielt hos de mellomliggende sensornoden, hvor man kan se at strømforbruket synker drastisk. Ellers så ser man ut ifra tabell 31 at ende til ende forsinkelsen øker ettersom nodetettheten øker. Dette har med forsinkelsen hos hver sensornode å gjøre. Desto flere sensornoder en pakke må rutes via, og desto flere kollisjoner som oppstår, desto høyere forsinkelse fører det til. Men denne simuleringen viser også at desto høyere nodetettheten er, desto lengre blir levetiden til hver enkelt sensornode i sensornettverket. Derfor er det helt klart en balanse mellom lengre levetid, og høyere forsinkelse, hvor den første helt klart er viktigst. Den gjennomsnittlige gjennomstrømningen forble ganske lik i disse simuleringene også. 1590, 1588 og 1586 bps ble den gjennomsnittlig gjennomstrømningen for hvert av pakkeformatene.

En annen ting vi kan merke oss, er at ettersom en pakke må rutes gjennom flere sensornoder, så vil dette føre til høyere belastning i sensornettverk, dette fordi hver sensornode må videresende hver pakke den mottar. Dette kan føre til mye trafikk, spesielt i et sensornettverk som baserer rutingen på kringkasting. Forsinkelsen vil gå opp, men strømforbruket vil gå ned. Det er det viktigste.

6.2.5 Simulering av 1 fenomen, samt 4 og 9 detekterende sensornoder

I simuleringene ble det benyttet samme verdier som i første simulering, bortsett fra at carrier sensing- og mottaker- radiusen ble økt til 100 meter. På denne måten kan man illustrere hvordan sensornettverket oppfører seg ved høy trafikkintensitet. Konsekvensen for at radiusen har økt til 100 meter, er at hver sensornode kan sende og mottatt pakker fra alle sine 8 naboer. Tabell 32 oppsummerer parameterne som er blitt benyttet i disse simuleringene.

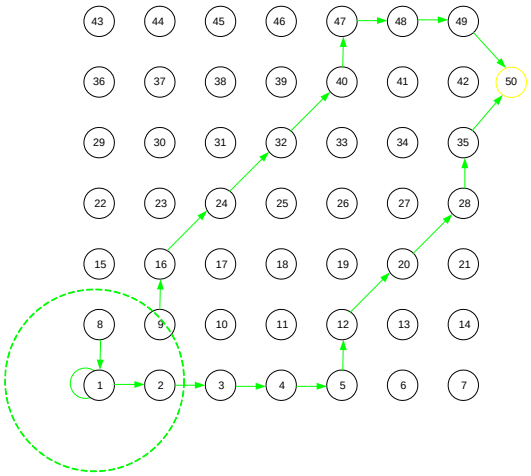
Verdier	Simulering 7	Simulering 8
Avstand mellom noder Horisontalt/Vertikalt (meter)	74/63	74/63
Sendeeffekt (mWatt)	3,0	3,0
RX/CS sensitivitet (meter)	100	100
Datarate (bps)	38400	38400
Antall sensornoder	49	49
Pulsraten (pr sekund)	10	10
Transmisjonshyppighet (pr sekund)	2	2
Frekvens (MHz)	868	868
TX power (mWatt)	96,6	96,6
RX power (mWatt)	47,7	47,7
SensePower (uWatt)	3,0	3,0
Strømkapasitet (J/s)	27000	27000
Pakkestørrelse (bytes)	31	31
RTS-CTS	AV	AV

Tabell 32. Oversikt over parametere som ble brukt i simuleringene av trafikkintensiteten.

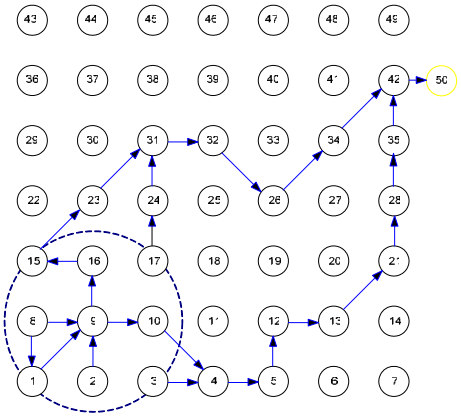
6.2.5.1 Resultatet av simuleringene med 4 og 9 detekterende sensornoder

Denne simuleringen tok først og fremst for seg trafikkintensiteten, men vi skal også se på hvordan strømforbruket endres. Alt som er markert med grønt i rutingstopologien (figur 43) tilhører simulering med 4 detekterende sensornoder, mens alt som er blått (figur 44), tilhører simuleringen foretatt med 9 detekterende sensornoder.

AODV dannet følgende rutingstopologi for hver av simuleringene:



Figur 43. Nettverkstopologi til 4 detekterende sensornoder, og 1 fenomen.



Figur 44. Nettverkstopologi til 9 detekterende sensornoder, og 1 fenomen.

I simuleringene prøvde vi ut med at 4 og 9 sensornoder detekterte 1 fenomen. Tabellene 33, 34 og 35 viser resultatene fra simuleringene. Det ble observert at desto flere sensornoder som oppfattet det samme fenomenet, desto flere pakker gikk tapt. Dette forekom fordi når et vist antall sensornoder detekterer samme fenomen samtidig, så fører det til at alle disse sensornodene vil prøve å kapre linken ved tilnærmet samme tidspunkt, slik at de kan sende meldinger tilbake til sinken. Hver enkelt av sensornodene sendte ut 12 udp pakker. Ved bruk av 4 detekterende sensornoder, så oppsto det ingen pakketap av de 48 udp pakkene, men 991 kollisjoner. Ved å øke detekterende sensornoder til 9, så resulterte det i at det forekom 28 tapte udp pakker blant 108 sendte udp pakker, noe som skyldtes at trafikkintensiteten gikk opp spesielt rundt fenomennoden. Fenomennoden hadde i dette tilfellet samme koordinater som sensornode 9, noe som gjorde at den også kunne detekteres av nabonodene til sensornode 9. Ende til ende forsinkelsen endret seg også drastisk i disse simuleringene, hvor gjennomsnittlig ende til ende forsinkelse i hvert av tilfellene var på 693 ms ved 4 detekterende sensornoder, mens gjennomsnittlig ende til ende forsinkelsen var så høy som 2 sekunder ved 9 detekterende sensornoder. Denne høye forsinkelsen har også med at CSMA/CA ikke gir en rettferdig deling av mediet. I flere tilfeller så har visse sensornoder vært nødt til å vente lengre enn andre for å kunne sende en datapakke ettersom mediet har vært opptatt. En annen ting var at den gjennomsnittlige gjennomstrømningen var mye høyere ved 9 detekterende sensornoder, noe som skyldes at trafikkintensiteten i hele sensornettverket var høyere under de 50,2 sekundene pakkene ble sendt. Tabell 33 nedenfor viser en oversikt over antall sendte, mottatte, droppede, samt antall kollisjoner. I tillegg viser tabellen også gjennomsnittlig, maksimum og minimum ende til ende forsinkelse, samt gjennomsnittlig gjennomstrømning.

Nodetetthet på 49 noder (ant. detekt. noder)	Ant. sendte pakker (udp)	Ant. mottatte pakker (udp)	Ant. droppede pakker (udp)	Antall kollisjoner	Gj. snittlig ende til ende forsinkelse (s)	Maksimum ende til ende forsinkelse (s)	Minimum ende til ende forsinkelse (s)	Gj. snittlig gjennom-strømning (bps)
4	48	48	0	991	0,6933	6,3041	0,1556	6339
9	108	80	28	2242	2,04	12,08	0,137	10367

Tabell 33. Resultater av simuleringen av 9 detekterende sensornoder, og 1 fenomen.

All denne trafikken hadde stor innvirkning på strømforbruket, noe tabell 34 og 35 viser:

Node nr (31 bytes)	Strømforbruk pr minutt (Joule)	Strømforbruk pr time (Joule)	Simulert levetid (ant. dager)
1	0,2417391	14,504346	78
2	0,2916864	17,501184	64
3	0,2913759	17,482554	64
4	0,2900637	17,403822	65
8	0,2215269	13,291614	85
9	0,2502144	15,012864	75
12	0,26988795	16,193277	69
16	0,20180475	12,108285	93
20	0,26741745	16,045047	70
24	0,1764801	10,588806	106
28	0,2482434	14,894604	76
32	0,15551595	9,330957	121
35	0,21923325	13,153995	86
40	0,1512189	9,073134	124
47	0,11696265	7,017759	160
48	0,1463967	8,783802	128
49	0,11889855	7,133913	158

Tabell 34. Simulert strømforbruk ved 3 detekterende sensornoder.

Node nr (31 bytes)	Strømforbruk pr minutt (Joule)	Strømforbruk pr time (Joule)	Simulert levetid (ant. dager)
1	0,339093	20,34558	55
2	0,38244555	22,946733	49
3	0,3506409	21,038454	53
4	0,3504654	21,027924	54
5	0,28559655	17,135793	66
8	0,40773915	24,464349	46
9	0,5029938	30,179628	37
10	0,39382875	23,629725	48
12	0,28756755	17,254053	65
13	0,2954232	17,725392	63
15	0,57975345	34,785207	32
16	0,52498935	31,499361	36
17	0,35919045	21,551427	52
21	0,27620325	16,572195	68
23	0,54989685	32,993811	34
24	0,3934332	23,605992	48
26	0,53614035	32,168421	35
28	0,3106458	18,638748	60
31	0,5360418	32,162508	35
32	0,50550615	30,330369	37
34	0,49983345	29,990007	38
35	0,34645725	20,787435	54
42	0,4037634	24,225804	46

Tabell 35. Simulert strømforbruk ved 9 detekterende sensornoder.

Figurere 43 og 44 viser hvordan trafikken rutes ettersom trafikkintensiteten øker. Hver enkelt sensornode får kortere lengre levetid ettersom trafikkintensiteten øker. Når 4 sensornoder detekterer samme fenomen i sensornettverket, så resulterer det i at sensornoden med laveste levetid, altså de sensornodene som har opplevd mest belastning, ligger på 64 dager mot 32 i det andre tilfellet hvor 9 detekterende sensornoder eksisterer. Maksimal levetid for en sensornode i disse tilfellene er 160 mot 68 dager i det andre tilfellet, men dette er sensornodene som har opplevd minst belastning av de sensornodene som har mottatt pakker i sensornettverk.

Sensornodene fikk noe ujevn levetid, men vi kan se i tabell 35 at levetiden pr sensornode er mye lavere enn i tabell 34. I tabell 34 ser vi at sensornodene 35 og 49 begge leverer pakker til sink, noe som gjør at de fordeler strømforbruket seg imellom. Spesielt sensornode 35 opplever høy belastning, noe som gjør at den er en av sensornodene nærmest sink som har lavest forventet levetid. Dette skyldes som sagt trafikkintensiteten, samt hvilke rute pakkene sendes via. I tabell 35 er det flere sensornoder som må rute data fra mange andre sensornoder. Dette gjelder spesielt sensornodene 4, 9, 31 og 42. Avhengig av trafikkintensiteten, så vil disse sensornodene måtte forbruk mye strøm på å videresende mange pakker fra flere sensornoder. Sensornode 42 opplever at all trafikk rutes via denne før trafikken når sinken. Dette er som sagt ressurskrevende, noe som fører til at levetiden er på 46 dager.

I hver av disse simuleringene så er det henholdsvis 4 og 9 sensornoder som detekterer hendelser. Disse sensornodene detekterer hendelser fra det samme fenomenet, noe som gjør

det er totalt unødvendig for alle sensornodene å måtte rapportere samme hendelse tilbake til sinken. Derfor kan det være veldig nyttig om sensornodene først tar å utveksle data seg imellom, slik at ikke samme hendelse rapporteres av henholdsvis 4 og 9 sensornoder i sensornettverket, noe som bare vil føre til økt trafikk. Det mest ideelle ville være om sensornoden først kunne konsultert seg imellom, før data sendes mot sinken, eller om en sensornode som mottar en melding, ikke videresender meldingen hvis den allerede har videresendt samme fenomen tidligere. SensorSim er dessverre ikke utstyrt med en slik mekanisme, men i et ordentlig sensornettverk ville dette vært særdeles nyttig.

6.3 Drøfting av simuleringen versus estimering

Tidligere i rapporten regnet vi ut den maksimale tiden det tar før en IV på 3 bytes gjentar seg ved forsendelse av 1 pakke i minuttet. Resultatet viste at det ville ta maksimalt 11597 dager før IV garantert gjentaes ved forsendelse av 1 pakke i minuttet. I simuleringene, så har hver sensornode sendt 12 pakker i minuttet, noe som vil føre til at det vil ta maksimalt 966 dager før IV gjentaes. Ut ifra alle simuleringene, så er det ingen sensornoder som vil ha en så lang levetid, noe som gjør at det er med stor sannsynlighet at IV ikke vil gjentaes i løpet av levetiden til noen av disse sensornodene. Derfor kan vi konkludere med at en IV på størrelse med 3 bytes fint kan benyttes i vårt tilfelle. MAC verdien på 8 bytes er mer enn tilfredsstillende, ettersom en sensornode med tilsvarende strømkapasitet som Mica2 mote, aldri vil være i stand til å motta så mange pakker slik at en MAC på 8 bytes vil kunne forfalskes.

Vi estimerte at strømforbruket ville øke drastisk ved å innføre sikkerhet i et sensornettverk. I kapitel 5 regnet vi ut at pakkeformatene på 31, 37 og 42 ville ha en økning på henholdsvis 11,7 % og 21,5 % i forhold til et pakkeformat på 31 bytes uten sikkerhet. Dette strømforbruket inkluderte også startsymbolet og synkroniseringsbytene, men det gjaldt kun i forhold til en enkelt pakke. I simuleringene vi har foretatt oss, så sendes det mye mer data en selve udp pakkene, noe som førte til at resultatene i simuleringen ble lavere. Simuleringene viste at ved forsendelse av 12 udp pakker, samt en del AODV og MAC kontrollpakker, så ble dette jevnet ut til 2,7 % til 4,1 % ved innføring av autentisering og integritet, samt 5,3 % til 7,2 % ved innføring av konfidensialitet. Dette er altså ikke en drastisk nedgang, og denne type sikkerhet kan fint innføres i et sensornettverk. Denne reduseringen ved simulering i forhold til estimert redusering, har med at det ble sendt mange AODV pakker, samt at MAC laget sendte ACK tilbake for hver pakke som ble sendt. Dessuten forekom det en del kollisjoner som førte til at ACK måtte sendes på nytte. Spesielt mange AODV pakker ble droppet i sensornettverket fordi det oppsto kollisjoner, samt at bufre ble fulle.

Vi tok den gjennomsnittlige gjennomstrømningen for simuleringen, noe som viste at gjennomstrømningen er relativt lav. Dette er forventet siden et sensornettverk har faste driftssykluser, og hvor disse driftssyklusene er periodiske, med lang sovetid. Estimert gjennomstrømning var mye høyere selvfølgelig, og den gjaldt kun når 2 sensornoder kommuniserer. Simuleringen viste altså at den gjennomsnittlige gjennomstrømningen er lav, og dette er et sannsynlig tilfelle i et sensornettverk som opererer med lav driftssyklus. Et sensornettverk vil etter all sannsynlighet ikke ha høy gjennomsnittlig gjennomstrømning ettersom sensornettverket kun vil operere i perioder, og da ikke kontinuerlig.

I kapitel 6.1 tok vi for oss forskjellige scenarioer som kan oppstå i et sensornettverk, og hvor vi estimerte hvordan strømforbruket ville utarte seg avhengig av hvilken funksjon sensornoden foretok seg. For eksempel om sensornoden kun detekterte et fenomen, for så å

rapportere tilbake til sinken, eller om sensornoden var en mellomliggende node som må motta data fra en annen sensornoden, for så å videresende data til neste sensornode. Eller om sensornoden fungerer som en aggregeringsnode, hvor sensornoden mottar fra flere, for så å videresende data. I simuleringen antok vi at aggregeringsnoden ikke har mulighet til å sammenslå data, men kun videresende datapakkene. Som forventet så førte disse scenarioene til økt strømforbruk hos sensornodene. De sensornodene som opplevde høyest strømforbruk var de mellomliggende sensornodene som fungerte som rutere, samt dataaggregeringsnoden. Den detekterte sensornoden hadde lavest strømforbruk, og det hadde som sagt med at den detekterende sensornoden kun sendte ut data, mens de mellomliggende og aggregeringsnodene måtte motta, for så å videresende data mot sinken. Dette strømforbruket avhenger av hvilket pakkeformat som brukes, hvor stor trafikkintensiteten er, altså hvor mange pakker en sensornode mottar og sender ut, og hvor høy sendeeffekten er, noe som avhenger av distansen mellom sensornodene. I simuleringene tok vi ikke hensyn til ledig tid, men strømforbruket foregikk kun i de periodene der sensornodene foretok aktivitet.

6.4 Del konklusjon

Kapitelet tok for seg mange ulike scenarioer som kan oppstå i et sensornettverk, både estimerte og simulerte scenarioer. Disse scenarioene viste at det er flere ting som fører til økt strømforbruk i et sensornettverk. Hovedsakelig oppstår strømforbruket på grunn av antall pakker som sendes og antall pakker som mottaes, samt hyppigheten og varigheten til driftssyklusen og hvilke tilstand sensornodene befinner seg i. Måten sensoren operer på har også en stor betydning på levetiden til sensornodene. I et ordentlig sensornettverk vil også faktorer som overhøring av andres kommunikasjon ha stor betydning for strømforbruket ettersom sensornodene vil forbruke strøm på å overhøre andres kommunikasjon. Dette løses i SMAC ved at sensornodene kun foretar lytting i perioder, for så å sove resten. I tillegg vil det også forekomme et lite strømforbruk assosiert med ledigtid. Ut ifra resultatene, så ser vi hvor viktig ressurs batteriet er for sensornodene. Derfor er det nødvendig at sensornodene sover mesteparten av tiden, og kun vekkes opp i perioder. Simuleringene viser blant annet at økt overhead i datapakker fører til høyere strømforbruk, samt økt transmisjonstid, og økt forsinkelse. I simuleringene forble gjennomsnittlig gjennomstrømning relativt lavt, men dette skyldes at gjennomstrømningen ble kalkulert over hele simuleringen, og at trafikkintensiteten var lav i de fleste simuleringen. Lav trafikkintensitet er noe man kan forvente i et sensornettverk ettersom kostnadene assosiert med transmisjon av data er såpass høye. Det at sensornettverket benytter multihopp fører til økt forsinkelsen i sensornettverk ettersom datapakkene må rutes via flere sensornoder for å nå sink. Det er også forventet at når et ordentlig sensornettverk benytter dataaggregering, så vil dette også føre til høyere forsinkelse, og lavere strømforbruk, men det vil i tillegg føre til at aggregeringsnoden sender ut mindre data enn den mottar ettersom den sammenslår data. En annen ting som er ønskelig er å forhindre flere sensornoder fra å rapportere samme hendelse tilbake til sinken. Uten en slik mekanisme, så vil det bli høy trafikkintensitet, noe som bare vil føre til at sensornodene får redusert levetid.

7 Hovedkonklusjon og fremtidig arbeid

Hovedbidraget i denne rapporten har vært å kartlegge hvordan innføring av sikkerhet i et sensornettverk påvirker sensornodenes strømforbruk, funksjonalitet og levetid.
For å kartlegge dette, har vi sett på forskjellige aspekter ved sensornettverket, samt sett på ulike sikkerhetsteknikker som kan være passende for et sensornettverk. Ut ifra sensornettverkets begrensninger, så har vi foreslått 2 passende pakkeformat, 1 som tar for seg autentisering og integritet, og 1 som tar for seg konfidensialitet. Disse pakkeformatene benyttet vi videre for å kartlegge strømforbruket som oppstår i et sensornettverk ettersom økt sikkerhetsoverhead innføres i datapakkene.

Resultatene viste at en begrenset form for sikkerhet ikke vil ha så enorm betydning for levetiden til sensornodene. Denne levetiden degraderes raskere desto mer overhead som innføres. Siden det er ved kommunikasjon at størsteparten av strømforbruket oppstår, så er det viktig å være nøye med hvor mye ekstra overhead som innføres i datapakkene. Hovedmålet ved design av et sensornettverk bør være å forlenge levetiden til sensornodene lengst mulig, slik at sensornodene kan utføre oppgavene over lengst mulig tid. Hvis en sensornode dør ut relativt tidlig, så kan det i verste fall føre til at målingene blir ufullstendig, noe som igjen kan føre til at sensornettverkets formål ikke blir realisert. Derfor må valg av sikkerhet foretaes nøye avhengig av hva sensornettverket skal brukes til. Resultatene viste blant annet at autentisering og integritet ikke ville føre til en ekstrem degradering av levetiden til sensornodene. Konfidensialitet førte til litt større redusering i levetiden, men kan fint brukes i et sensornettverk. Men hvis det ikke er nødvendig å holde data hemmelig, så kan det være greit å droppe konfidensialitet i et sensornettverk for å bevare levetiden til sensornodene. Men dette må bestemmes ut ifra viktigheten til data, samt hva slags applikasjon sensornettverket skal utføre.

For å kunne realisere et sensornettverk må man overvinne begrensningene som et sensornettverk innfører. Dette er begrensninger i form av liten prosessorkraft, lite lagring, lav transmisjonsrate, samt liten strømkapasitet. For at et sensornettverk skal kunne optimalisere levetiden, så må strømforbruket i alle komponentene til sensornoden være minimalt. Spesielt strømeffektive protokoller må utarbeides, slik at sensornodene kan ta beslutninger som er strømeffektive. Det kan innebære at ettersom et sensornettverk har høy nodetetthet, så kan for eksempel sensornodene redusere transmisjonseffekten, og heller rute data over flere hopp. En av simuleringene viste at flere hopp og lavere transmisjonseffekt vil redusere strømforbruket i hver enkelt sensornode. Vi har også sett på flere ulike scenarioer, både estimerte og simulerte scenarioer, samt beregnet strømforbruket i alle disse scenarioene. Disse simuleringene viste at ettersom overhead øker, så vil dette føre til lengre transmisjonstid, høyere gjennomstrømning, høyere forsinkelse, samt høyere strømforbruk.

Innføring av en slik sikkerhet som vi har tatt for oss kan fint realiseres, men det vil som rapporten har bekreftet, påvirke strømforbruket i sensornettverket. Det er enda mye arbeid som gjenstår for å effektivisere og realisere et sensornettverk. Det er spesielt nødvendig å utvikle mer effektive protokoller ut ifra kravene som sensornettverket stiller. Rutingsprotokollene må effektiviseres slik at den mest lønnsomme ruten alltid velges, samt at data rutes på en effektiv måte. Protokollene må være effektive, pålitelig, og rutene må velges basert på lavest mulig strømforbruk, samt så bør forskjellige ruter velges, slik at man ikke tapper en enkelt sensornode for strøm. Effektive protokoller som kan håndtere mobilitet vil være neste steg. De fleste protokollene som er foreslått for et sensornettverk, er designet for

statiske sensornettverk. Derfor bør protokoller utvikles for mobilitet, slik at sensornettverket kan være mer fleksible. Det viktigste er at alle protokollene utvikles for å være mest mulig strømeffektive. Dessuten bør det utvikles en protokoll som gjør det mulig for sensornodene å utveksle data seg imellom slik at ikke flere av de samme sensornodene rapporterer samme hendelse tilbake til sink. Hvis sensornodene utstyres med en slik mekanisme, så vil dette redusere trafikkbelastningen i sensornettverket, noe som igjen vil føre til lengre levetid for sensornodene. Ellers så er det viktig at medium aksess protokollen effektiviseres, slik at effektive driftssykluser kan realiseres. Med dette menes at sensornodene må ha mulighet til å detektere hendelser på en effektiv måte, for så å raskt å prosessere hendelsen, samt transmittere data nødvendig. Dette må skje på en effektiv og strømeffektiv måte, slik at sensornodene ikke opplever for mye ledigtid.

Referanser

[1] I.F. Akyildiz, E. Cayirci, W. Su, Y. Sankarasubramaniam. A Survey on Sensor Networks. IEEE 2002.
[2] A. Hac. Wireless Sensor Network Designs. Wiley. 2003.
[3] Rockwell, http://wins.rsc.rockwell.com
[4] W. Stallings. Network Security Essentials. Prentice Hall. Second edition. 2003.
[5] L. Yuan, G. Qu. Design Space Exploration for Energy-Efficient Secure Sensor Network. IEEE. 2002.
[6] W. Ye, J. Heidemann, D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. IEEE. 2002.
[7] M. Chen, W. Cui, V. Wen, A. Woo. Security and Deployment Issues in a Sensor Network. 2000.
[8] A. Perrig, H. Chan, D. Song. Random Key Predistribution Schemes for Sensor Networks. IEEE 2003.
[9] E. Callaway. Wireless Sensor Networks – Architectures and Protocols. Auerbach Publications 2004.
[10] Sensoria. http://www.sensoria.com/
[11] M. Ilyas. The Handbook of Ad Hoc Wireless Networks. CRC Press 2003.
[12] A. Perrig, R. Szewczyk, V. Wen, D. Culler. SPINS: Securtiy Protocols for Sensor Networks. 2002.
[13] P. Ganesan, R. Venugopalan, A. Dean, F. Mueller. Analyzing and Modelling Encryption Overhead for Sensor Network Nodes.
[14] D. Malan, M. Welsh, S. Moulton. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care.
[15] Nonin. www.nonin.com.
[16] Numed. www.**numed**-direct.co.uk.
[17] Radianse. www.radianse.com.
[18] N. Borisov, I. Goldberg, D. Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. ACM. 2001.
[19] The Network Simulator – NS-2. http://www.isi.edu/nsnam/ns/
[20] S. Park, A. Savvides, M. B. Srivastava. SensorSim: A simulation Framework for Sensor Networks. University of California. ACM. 2000.
[21] NRL`s Sensor Network Extension to NS-2. http://nrlsensorsim.pf.itd.nrl.navy.mil/
[22] Ian Downard. Simulation Sensor Networks in NS-2.
[23] MAODV. http://www.isr.umd.edu/CSHCN/research/maodv/MAODV-UMD.html
[24] S. Slijepcevic, M. Potkonjak, S. Zimbeck, V. Tsiatsis. On Communication Security in Wireless Ad-Hoc Sensor Networks. IEEE. 2002.
[25] T. Ylonene. SSH – Secure login connections over the internet. In Proceedings of the Sixth USENIX Security Symposium. 1996.
[26] AODV. http://moment.cs.ucsb.edu/AODV/aodv.html
[27] Energizer. http://www.energizer.com/learning/
[28] N.R. Potlapally, S. Ravi, A. Raghunathan, N.K. Jha. Analyzing the energy Consumption of Security Protocols. ISLPED 2003.
[29] ChipCon Radio datasheet. www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf.
[30] J. Esslinger. Security Aspects of Sensor Networks. Project presentation.
[31] A. Perrig, J. Stankovic, D. Wagner. Security in Wireless Sensor Networks.
[32] C. Karlof, D. Wagner. Secure Routing in Wireless Sensor Networks: Attacks and countermeasures.
[33] J. L. Hill. System Architecture for Wireless Sensor Networks. Masterthesis 2003.

[34] RSA. www.rsasecurity.com
[35] K. Sohrabi, J. Goa, G. Pottie. Protocols for Self-Organization of a Wireless Sensor Network. IEEE. 2000.
[36] C. Karlof, N. Sastry, D. Wagner. TinySec: A Link Security Architecture for Wireless Sensor Networks. SenSys. 2004.
[37] Codeblue. http://www.eecs.harvard.edu/~mdw/proj/vitaldust/
[38] Telemedicine, An application of sensor network and its security. Presentation.
[39] J. Polastre, R. Szewczyk, D. Culler. Telos: Enabling Ultra-Low Power Wireless Research. IPSN/SPOTS 2005.
[40] Info om TinySec. http://www.cs.berkeley.edu/~nks/tinysec/
[41] W. Stallings. Data & Computer Communications. Prentice Hall, Sixth edition. 2000.
[42] Q. Xue. Runtime Security Composition for Sensor Networks (SecureSense).
[43] TinyOS. http://www.tinyos.net/.
[44] Skipjack. http://www.tropsoft.com/strongenc/skipjack.htm
[45] RC5. http://www.faqs.org/rfcs/rfc2040.html
[46] Xbow. www.xbow.com.
[47] Z. Shelby, J. Haapola, C. Pomalaza-Raez. Energy Optimization In Multihop Wireless Embedded And Sensor Networks.
[48] V. Raghunathan, C. Schurgers, S. Park, M. Srivastava. Energy Aware Wireless Sensor Networks.
[49] G. Anastasi, A. Falchi, A. Passarella, E. Gregori, M. Conti. Performance Measurements of Motes Sensor Networks.
[50] TinyOS. www.cse.wustl.edu/~lu/cs537s/Slides/**tinyos**.**pdf**.
[51] Security for Sensor Networks. SenSec design. http://www.i2r.a-star.edu.sg/icsd/SecureSensor/papers/SenSec-TR-I2R.pdf
[52] D. Culler, D. Estrin, M. Srivastava. Overview of Sensor Network. IEEE. 2004.
[53] S. Park, A. Savvides, M. Srivastava. Simulating Networks of Wireless Sensors.
[54] V. Shankar, A. Natarajan, S. K. S. Gupta. Energy-Efficient Protocols for Wireless Communication in Biosensor Networks. IEEE 2001.
[55] A. D. Wood, J.A Stankovic. Denial of Service in Sensor Networks. IEEE. 2002.
[56] R. Rivest. The RC5 Encryption Algorithm. http://citeseer.ist.psu.edu/rivest95rc.html.
[57] A. Woo, D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. ACM. 2001.
[58] J. Zhao, R. Govindan. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks.
[59] J. Hill, D. Culler. A wireless embedded sensor architecture for system-level optimization.
[60] A. Dunkels, J. Alonso, T. Voigt. Making TCP/IP Viable for Wireless Sensor Networks.
[61] K. Lorincz, D. J. Malan, V. Shnayder, S. Moulton. Sensor Networks for Emergency Response: Callenges and Oppertunities. IEEE 2004.
[62] Y. W. Law, J. Doumen, P. Hartel. Benchmarking Block Ciphers for Wireless Sensor Networks.
[63] N. Sastry, D. Wagner. Security Considerations for IEEE 802.15.4 Networks. Wise 2004.
[64] M. Saraogi. Security in Wireless Sensor Networks.
[65] J. E. Håkegård. wsLAN, wireless sensor Local Area Network. 2005.
[66] R. L. Rivest. The RC5 Encryption Algorithm.
[67] P. J. M. Havinga. System Architecture Specification. 2002.
[68] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister. System Architecture Directions for Networked Sensors. ACM 2000.
[69] W. Du, J. Deng, Y. S. Han, S. Chen, P. K. Varshney. A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge.

[70] J. Borst, B. Preneel, J. Vandewalle. Linear Cryptoanalysis of RC5 and RC6.
[71] J.N. Al-Karaki, A.E. Kamal. Routing Techniques in Wireless Sensor Networks: A Survey. IEEE 2004.
[72] A. Chandrakasan, R. Min, M. Bhardwaj, S-H. Cho, A. Wang. Power Aware Wireless Microsensor Systems.
[73] A. Sinha, A. Chandrakasan. Dynamic Power Management in Wireless Sensor Networks. IEEE 2001.

Kildekode

```
#############################################################################
# Programmerer: Ole Kristian Bergseth
# Beskrivelse:
# Simuleringen vil illustrere hvordan strømforbruket i et sensornettverk
# reduseres over tid. Alle de andre simuleringene har benyttet samme
# struktur, og det er kun byttet ut parametere. Det som er annerledes i de
# andre simuleringene er spesifisert i kapitel 6 under hver av
# simuleringsdelene. Dette gjelder verdier som transmisjonseffekt,
# strømforbruk, antall sensornoder, plasseringen til sensornoden.
#############################################################################
#
# Bruker egne verdier for strømforbruket(burde gjøre om fra mAh til mWatt)
#       -rxPower 0.0966 mWatt (32.2 mA ved full sendeeffekt)
#       -txPower 0.0468 mWatt (15.6 mA ved maksimal sensitivitet)
#       -sensePower 0.000003 (1 mikroA)
#       -initialEnergy 7.5 mWatt (2500mAh)
#
# Disse verdiene er hentet ut i fra beregninger som er gjort i forbindelse
# med masteroppgaven.
#
#############################################################################
#
#
# ======================================================================
# Definerte verdier
# ======================================================================
set val(prop)           Propagation/TwoRayGround    ;# radiopropagasjon modell
set val(netif)          Phy/WirelessPhy             ;# Type nettverksgrensesnitt
set val(mac)            Mac/802_11                  ;# MAC type
set val(PHENOMmac)      Mac                         ;# MAC type for fenomenet
set val(ifq)            Queue/DropTail/PriQueue     ;# Kø type
set val(ll)             LL                          ;# link layer type
set val(ant)            Antenna/OmniAntenna         ;# antenne type
set val(ifqlen)         50                          ;# Maks pakker i køen
set val(nn)             50                          ;# Antall sensornoder
set val(rp)             AODV                        ;# rutingsprotokoll
set val(x)              550                         ;# grid Bredde
set val(y)              550                         ;# grid Høyde
set val(engmodel)       EnergyModel                 ;# energimodell
set val(txPower)        0.0966                      ;# Tranmisjonskostnad (Watt)
set val(rxPower)        0.0477                      ;# Mottakkostand (watt)
set val(sensePower)     0.00003                     ;# Sensingkostnad (Watt)
set val(idlePower)      0.0                         ;# Kostnad ved ledigmodus (Watt)
set val(initeng)        27000                       ;# Strømkapasitet ved start (J/s)
set val(initempty)      1
set val(tr_f)           resultater/energy.tr        ;# tracefile
set val(tr_n)           resultater/energy.nam       ;# nam file

# Antenna settes til å være sentrert i sensornoden, og 1 meter over den
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.0
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1
puts "Starter simuleringen av et sensornettverk."
```

```
# Definerer datarate, frekvens og transmisjonseffekt
# Bruker Two-Ray ground reflection model
Mac/802_11 set PLCPDataRate_   38400          ;# Raten på startsymbolet
Mac/802_11 set dataRate_       38400          ;# dataraten til 802_11
Mac/802_11 set basicRate_      38400          ;# RTS-CTS datarate
Mac/802_11 set RTSThreshold_   3000           ;# Slår av RTS-CTS
Phy/WirelessPhy set freq_      868e+6         ;# frekvensen
Phy/WirelessPhy set Pt_        0.003          ;# transmisjonseffekt
Phy/WirelessPhy set CPThresh_  10.0
Phy/WirelessPhy set CSThresh_  9.4e-11
Phy/WirelessPhy set RXThresh_  9.4e-11
Phy/WirelessPhy set L_         1.0             ;#

# ======================================================================
# Hovedprogram
# ======================================================================

set ns_             [new Simulator]
set tracefd [open resultater/energy.tr w]
$ns_ trace-all $tracefd
set namtrace [open resultater/energy.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

$ns_ use-newtrace

set topo      [new Topography]
$topo load_flatgrid $val(x) $val(y)
set god_ [create-god 51]
$god_ off
$god_ allow_to_stop
$god_ num_data_types 1

#Sette opp en kanal for sensornodene, og en for fenomennoden
set chan_1_ [new Channel/WirelessChannel]
set chan_2_ [new Channel/WirelessChannel]

# Konfigurerer fenomennoden
$ns_ node-config \
   -adhocRouting PHENOM \
   -llType $val(ll) \
   -macType $val(PHENOMmac) \
   -ifqType $val(ifq) \
   -ifqLen $val(ifqlen) \
   -antType $val(ant) \
   -propType $val(prop) \
   -phyType $val(netif) \
   -channel $chan_1_ \
   -topoInstance $topo \
   -agentTrace ON \
   -routerTrace ON \
   -macTrace ON \
   -movementTrace ON

for {set i 0} {$i < 1} {incr i} {
   set node_($i) [$ns_ node $i]
   $node_($i) random-motion 0
   $god_ new_node $node_($i)
   $node_($i) namattach $namtrace
   $ns_ initial_node_pos $node_($i) 25
```

```
   [$node_($i) set ragent_] pulserate 10.
   [$node_($i) set ragent_] phenomenon CO
}

# Konfigurerer sensornodene
$ns_ node-config \
   -adhocRouting $val(rp) \
   -channel $chan_2_ \
   -macType $val(mac) \
   -PHENOMmacType $val(PHENOMmac) \
   -PHENOMchannel $chan_1_ \
   -energyModel $val(engmodel) \
   -rxPower $val(rxPower) \
   -txPower $val(txPower) \
   -sensePower $val(sensePower) \
   -idlePower $val(idlePower) \
   -initialEnergy $val(initeng)

for {set i 1} {$i < 50 } {incr i} {
   set node_($i) [$ns_ node]
   $node_($i) random-motion 1
   $god_ new_node $node_($i)
   $node_($i) namattach $namtrace
}

# Konfigurerer sinken
$ns_ node-config \
   -adhocRouting $val(rp) \
   -rxPower 0 \
   -txPower 0 \
   -idlePower $val(idlePower) \
   -initialEnergy $val(initeng) \
   -channel $chan_2_ \
   -PHENOMchannel "off"

for {set i 50} {$i < 51 } {incr i} {
   set node_($i) [$ns_ node]
   $node_($i) random-motion 1
   $god_ new_node $node_($i)
   $node_($i) namattach $namtrace
}

#
# Koordinatene til sensornodene, sinken og fenomenet
#
$node_(1) set X_ 1
$node_(1) set Y_ 1
$node_(2) set X_ 75
$node_(2) set Y_ 1
$node_(3) set X_ 149
$node_(3) set Y_ 1
$node_(4) set X_ 223
$node_(4) set Y_ 1
$node_(5) set X_ 297
$node_(5) set Y_ 1
$node_(6) set X_ 371
$node_(6) set Y_ 1
$node_(7) set X_ 445
$node_(7) set Y_ 1
$node_(8) set X_ 1
```

```
$node_(8) set Y_ 64
$node_(9) set X_ 75
$node_(9) set Y_ 64
$node_(10) set X_ 149
$node_(10) set Y_ 64
$node_(11) set X_ 223
$node_(11) set Y_ 64
$node_(12) set X_ 297
$node_(12) set Y_ 64
$node_(13) set X_ 371
$node_(13) set Y_ 64
$node_(14) set X_ 445
$node_(14) set Y_ 64
$node_(15) set X_ 1
$node_(15) set Y_ 127
$node_(16) set X_ 75
$node_(16) set Y_ 127
$node_(17) set X_ 149
$node_(17) set Y_ 127
$node_(18) set X_ 223
$node_(18) set Y_ 127
$node_(19) set X_ 297
$node_(19) set Y_ 127
$node_(20) set X_ 371
$node_(20) set Y_ 127
$node_(21) set X_ 445
$node_(21) set Y_ 127
$node_(22) set X_ 1
$node_(22) set Y_ 190
$node_(23) set X_ 75
$node_(23) set Y_ 190
$node_(24) set X_ 149
$node_(24) set Y_ 190
$node_(25) set X_ 223
$node_(25) set Y_ 190
$node_(26) set X_ 297
$node_(26) set Y_ 190
$node_(27) set X_ 371
$node_(27) set Y_ 190
$node_(28) set X_ 445
$node_(28) set Y_ 190
$node_(29) set X_ 1
$node_(29) set Y_ 253
$node_(30) set X_ 75
$node_(30) set Y_ 253
$node_(31) set X_ 149
$node_(31) set Y_ 253
$node_(32) set X_ 223
$node_(32) set Y_ 253
$node_(33) set X_ 297
$node_(33) set Y_ 253
$node_(34) set X_ 371
$node_(34) set Y_ 253
$node_(35) set X_ 445
$node_(35) set Y_ 253
$node_(36) set X_ 1
$node_(36) set Y_ 316
$node_(37) set X_ 75
$node_(37) set Y_ 316
$node_(38) set X_ 149
```

```
$node_(38) set Y_ 316
$node_(39) set X_ 223
$node_(39) set Y_ 316
$node_(40) set X_ 297
$node_(40) set Y_ 316
$node_(41) set X_ 371
$node_(41) set Y_ 316
$node_(42) set X_ 445
$node_(42) set Y_ 316
$node_(43) set X_ 1
$node_(43) set Y_ 379
$node_(44) set X_ 75
$node_(44) set Y_ 379
$node_(45) set X_ 149
$node_(45) set Y_ 379
$node_(46) set X_ 223
$node_(46) set Y_ 379
$node_(47) set X_ 297
$node_(47) set Y_ 379
$node_(48) set X_ 371
$node_(48) set Y_ 379
$node_(49) set X_ 445
$node_(49) set Y_ 379

#Sette sensornodene til å være statiske
$ns_ at 0.01 "$node_(1) setdest 1 1 50.0"
$ns_ at 0.01 "$node_(2) setdest 75 1 50.0"
$ns_ at 0.01 "$node_(3) setdest 149 1 50.0"
$ns_ at 0.01 "$node_(4) setdest 223 1 50.0"
$ns_ at 0.01 "$node_(5) setdest 297 1 50.0"
$ns_ at 0.01 "$node_(6) setdest 371 1 50.0"
$ns_ at 0.01 "$node_(7) setdest 445 1 50.0"
$ns_ at 0.01 "$node_(8) setdest 1 64 50.0"
$ns_ at 0.01 "$node_(9) setdest 75 64 50.0"
$ns_ at 0.01 "$node_(10) setdest 149 64 50.0"
$ns_ at 0.01 "$node_(11) setdest 223 64 50.0"
$ns_ at 0.01 "$node_(12) setdest 297 64 50.0"
$ns_ at 0.01 "$node_(13) setdest 371 64 50.0"
$ns_ at 0.01 "$node_(14) setdest 445 64 50.0"
$ns_ at 0.01 "$node_(15) setdest 1 127 50.0"
$ns_ at 0.01 "$node_(16) setdest 75 127 50.0"
$ns_ at 0.01 "$node_(17) setdest 149 127 50.0"
$ns_ at 0.01 "$node_(18) setdest 223 127 50.0"
$ns_ at 0.01 "$node_(19) setdest 297 127 50.0"
$ns_ at 0.01 "$node_(20) setdest 371 127 50.0"
$ns_ at 0.01 "$node_(21) setdest 445 127 50.0"
$ns_ at 0.01 "$node_(22) setdest 1 190 50.0"
$ns_ at 0.01 "$node_(23) setdest 75 190 50.0"
$ns_ at 0.01 "$node_(24) setdest 149 190 50.0"
$ns_ at 0.01 "$node_(25) setdest 223 190 50.0"
$ns_ at 0.01 "$node_(26) setdest 297 190 50.0"
$ns_ at 0.01 "$node_(27) setdest 371 190 50.0"
$ns_ at 0.01 "$node_(28) setdest 445 190 50.0"
$ns_ at 0.01 "$node_(29) setdest 1 253 50.0"
$ns_ at 0.01 "$node_(30) setdest 75 253 50.0"
$ns_ at 0.01 "$node_(31) setdest 149 253 50.0"
$ns_ at 0.01 "$node_(32) setdest 223 253 50.0"
$ns_ at 0.01 "$node_(33) setdest 297 253 50.0"
$ns_ at 0.01 "$node_(34) setdest 371 253 50.0"
$ns_ at 0.01 "$node_(35) setdest 445 253 50.0"
```

```
$ns_ at 0.01 "$node_(36) setdest 1 316 50.0"
$ns_ at 0.01 "$node_(37) setdest 75 316 50.0"
$ns_ at 0.01 "$node_(38) setdest 149 316 50.0"
$ns_ at 0.01 "$node_(39) setdest 223 316 50.0"
$ns_ at 0.01 "$node_(40) setdest 297 316 50.0"
$ns_ at 0.01 "$node_(41) setdest 371 316 50.0"
$ns_ at 0.01 "$node_(42) setdest 445 316 50.0"
$ns_ at 0.01 "$node_(43) setdest 1 379 50.0"
$ns_ at 0.01 "$node_(44) setdest 75 379 50.0"
$ns_ at 0.01 "$node_(45) setdest 149 379 50.0"
$ns_ at 0.01 "$node_(46) setdest 223 379 50.0"
$ns_ at 0.01 "$node_(47) setdest 297 379 50.0"
$ns_ at 0.01 "$node_(48) setdest 371 379 50.0"
$ns_ at 0.01 "$node_(49) setdest 445 379 50.0"

$node_(0) set X_ 1.0
$node_(0) set Y_ 444.0
$ns_ at 0.01 "$node_(0) setdest 1.0 444.0 50.0"
$ns_ at 0.01 "$node_(0) color blue"

$node_(50) set X_ 480.0
$node_(50) set Y_ 379.0
$ns_ at 0.01 "$node_(50) setdest 480.0 379.0 50.0"
$ns_ at 0.01 "$node_(50) color yellow"

#############################################################################
# Fester en sensoragent til hver sensornode

# attach a Sensor Agent (i.e. sensor agent) to sensor node
for {set i 1} {$i < 50 } {incr i} {
   set sensor_($i) [new Agent/SensorAgent]
   $ns_ attach-agent $node_($i) $sensor_($i)

   [$node_($i) set ll_(1)] up-target $sensor_($i)
}

#############################################################################
# Setter opp en UDP forbindelse mot sinken, og fester en sensorapplikasjon til den
set sink_0 [new Agent/UDP]
$ns_ attach-agent $node_(50) $sink_0
for {set i 1} {$i < 50} {incr i} {
   set src0_($i) [new Agent/UDP]
   $ns_ attach-agent $node_($i) $src0_($i)
   $ns_ connect $src0_($i) $sink_0
   set app0_($i) [new Application/SensorApp]
   $app0_($i) attach-agent $src0_($i)
   $ns_ at 0.5 "$app0_($i) start $sensor_($i)"
}

# Sier ifra til nodene når simuleringen er over
#
for {set i 0} {$i < 51 } {incr i} {
   $ns_ at 60.2 "$node_($i) reset";
}
$ns_ at 60.2 "stop"
$ns_ at 60.2 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {
   global ns_ tracefd namtrace
   $ns_ flush-trace
```

```
    close $tracefd
    close $namtrace
}


puts "Starter simuleringen av et sensornettverk"
$ns_ run
```