

SKATT-PROJEKTET

***Protokol og algoritmer
til optimalt kanalvalg
i 802.15.4-baseret
peer netværk***

White Paper

Steen Krøyer





INDLEDNING/OPSUMMERING.....	3
BAGGRUND.....	3
KOMMUNIKATIONSBEHOV I INDUSTRIELLE KONTROL-APPLIKATIONER.....	3
Figur 1. En generisk kontrol-applikation.....	3
SPECIELLE FORHOLD I TRÅDLØS KOMMUNIKATION.....	4
MINIMERING AF RETRANSMISSIONER OG EFFEKTEEN AF RETRANSMISSIONER.....	4
<i>De oplagte løsninger.....</i>	4
Figur 2. En smart og en mindre smart måde at sende kvitteringer på.....	5
OPTIMALT KANALVALG.....	5
IEEE 802.15.4.....	5
Figur 3. 802.15.4 har 16 kanaler at vælge mellem.....	6
Præmisses.....	7
NET-TOPOLOGI.....	8
Figur 4. Net-topologi.....	8
BESKRIVELSE AF ALGORITME.....	9
CONNECTED STATE – ”NORMAL DRIFT”.....	9
Figur 5. Sekvensdiagram, connected state.....	9
<i>Flow-chart – master.....</i>	12
Figur 6. Flow-chart for master i connected state.....	12
<i>Flow-chart slave.....</i>	13
Figur 7. Flow-chart for slave i connected state.....	13
MASTERSTYRET KANALSKIFTE.....	14
Figur 8. Sekvensdiagram, frekvensskifte.....	14
AUTONOMT (FALL-BACK) KANALSKIFTE.....	15
<i>Master.....</i>	15
<i>Slave.....</i>	15
INITIERING.....	15
RESULTATER.....	16
SIMULERET TESTBÆNK.....	16
SIMULERET TESTOPSTILLING.....	16
Figur 9. Simuleret testopstilling.....	17
Figur 10. To datastrømme multiplexes over samme kommunikationslinie.....	17
SCENARIER.....	18
<i>Scenario 1: Ingen forstyrrelser.....</i>	18
<i>Scenario 2: En enkelt blokeret kanal.....</i>	18
Figur 11. Simuleret scenario 2.....	18
<i>Scenario 3: Periodiske, langvarige blokeringer på 8 samtidige kanaler.....</i>	18
Figur 12. Simuleret scenario 3.....	19
<i>Scenario 4: Periodiske, langvarige blokeringer på alle kanaler samtidigt.....</i>	19
Figur 13. Simuleret scenario 4.....	19
<i>Scenario 5: Periodiske, kortvarige blokeringer på alle kanaler samtidigt.....</i>	19
Figur 14. Simuleret scenario 5.....	20
<i>Scenario 6: Periodiske, kortvarige og hyppige blokeringer på alle kanaler samtidigt.....</i>	20
Figur 15. Simuleret scenario 6.....	20
RESULTATER MÅLT I SIMULATOR.....	21
Tabel 1. Målte resultater.....	22
KOMMENTARER TIL RESULTATER.....	22
<i>Generelt.....</i>	22
<i>Scenario 1: Ingen forstyrrelser.....</i>	22
<i>Scenario 2: En enkelt blokeret kanal.....</i>	22
<i>Scenario 3: Periodiske, langvarige blokeringer på 8 samtidige kanaler.....</i>	22
<i>Scenario 4: Periodiske, langvarige blokeringer på alle kanaler samtidigt.....</i>	22
<i>Scenario 5: Periodiske, kortvarige blokeringer på alle kanaler samtidigt.....</i>	23
<i>Scenario 6: Periodiske, kortvarige og hyppige blokeringer på alle kanaler samtidigt.....</i>	23
KONKLUSION.....	23

Indledning/opssummering

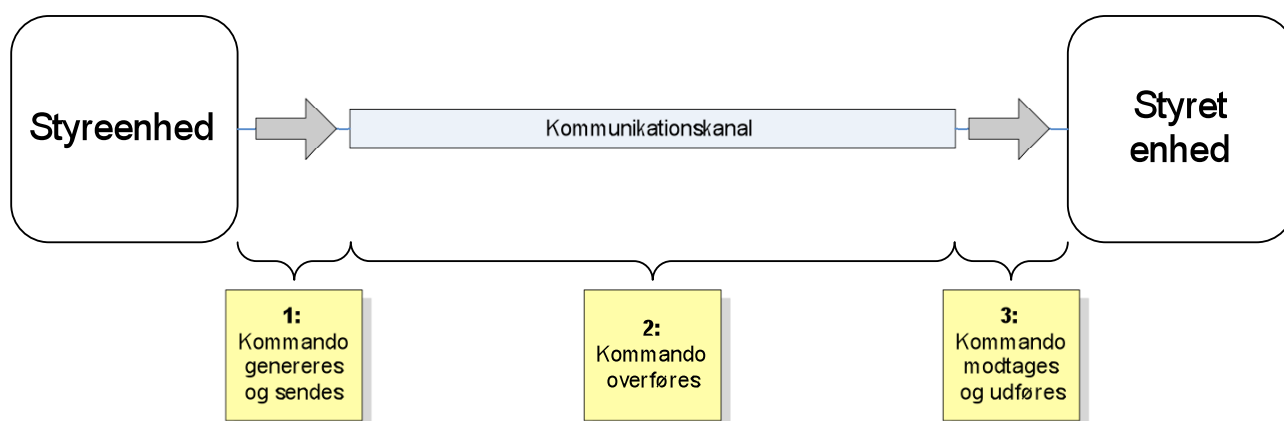
Dette white paper beskriver et forsøg på at maksimere pålideligheden og minimere forsinkelsen (latency) i et trådløst net. Konkret tages der dels afsæt i de ønsker man kan have til kommunikation i industrielle kontrol-applikationer, dels i en bestemt radioteknologi, nemlig IEEE 802.15.4. Den grundlæggende idé er at mindske antallet af retransmissioner (og dermed latency/forsinkelse i udførelsen af kommandoer) ved løbende at prøve at vælge den kanal der er mest fremkommelig.

Der er gennemført en række forsøg i simulerede omgivelser, der belyser den foreslåede løsnings evne til at få et lille netværk til løbende at vælge den mest fremkommelige kanal. Resultaterne indikerer at den foreslåede metode rent faktisk virker, om end den største effekt af et optimalt kanalvalg synes at være en højere pålidelighed snarere end lavere latency.

Baggrund

Kommunikationsbehov i industrielle kontrol-applikationer

Uanset om der benyttes kablet eller trådløs kommunikation, stiller industrielle kontrol-applikationer en række krav til kommunikationen.



Figur 1. En generisk kontrol-applikation.

Afhængigt af en given applikation stilles der bl.a. større eller mindre krav til kommunikationskanalens egenskaber:

- *Pålidelighed*. Hvor sikre kan vi være på at en kommando bliver korrekt overført og efterfølgende udført?
- *Forsinkelse*. Hvor lang tid går der fra det tidspunkt hvor styreenheden sender en kommando og til den er overført?
- *Latency*. Tiden der går fra en kommando sendes fra styreenheden til udførelsen påbegyndes, også kaldet latency, er ofte vigtig. Variationer i kommunikationskanalens forsinkelse af den overførte kommando, fører til variationer i latency.

Hvis vi ser bort de forhold og variationer der knytter sig til den tid det tager at generere og udføre en kommando (og de er jo uafhængige af kommunikationskanalen), viser det sig i praksis ofte at den største forskel på kablet og trådløs kommunikation er den variation der kan være i latency. Kablede kommunikationsløsninger (field busses) til kontrolapplikationer er netop designet til at minimere

forsinkelsen af overførte kommandoer og ikke mindst kunne garantere en øvre grænse for denne (og dermed give en deterministisk, øvre grænse for latency).

Specielle forhold i trådløs kommunikation

Principielt er det de samme problemer der gør sig gældende i trådede og trådløse kommunikationskanaler, såsom refleksion og dæmpning af signalet og støj. Men hvor disse problemer forholdsvis nemt kan bringes under kontrol i et veludviklet, trådet kommunikationssystem, er de i praksis umulige helt at kontrollere når der benyttes trådløs kommunikation.

For at opnå en tilpas høj pålidelighed med trådløs kommunikation, er man derfor nødt til at benytte forskellige teknikker der gør det muligt at overføre information selvom nogle data går tabt. De vigtigste teknikker hertil er omtalt mere detaljeret andetsteds i dette white paper, men her er de vigtigste:

- Fejldetektering. Det skal være muligt for en modtager at se at den overførte information er blevet ”ødelagt”, dvs. ændret på en eller anden måde.
- Fejlkorrigerende koder. Benyttes der fejlkorrigerende koder, kan en modtager ikke blot se at data er blevet ændret, men den kan også (hvis der ikke er for mange fejl) selv beregne hvori fejlen(e) består og rette dem.
- Retransmission. Hvis alt andet svigter, må ødelagte data gendeses en eller flere gange til de kommer intakte frem.

I industrielle kontrol-applikationer er det værd at bemærke disse forskellige teknikker har forskellige nytteværdi.

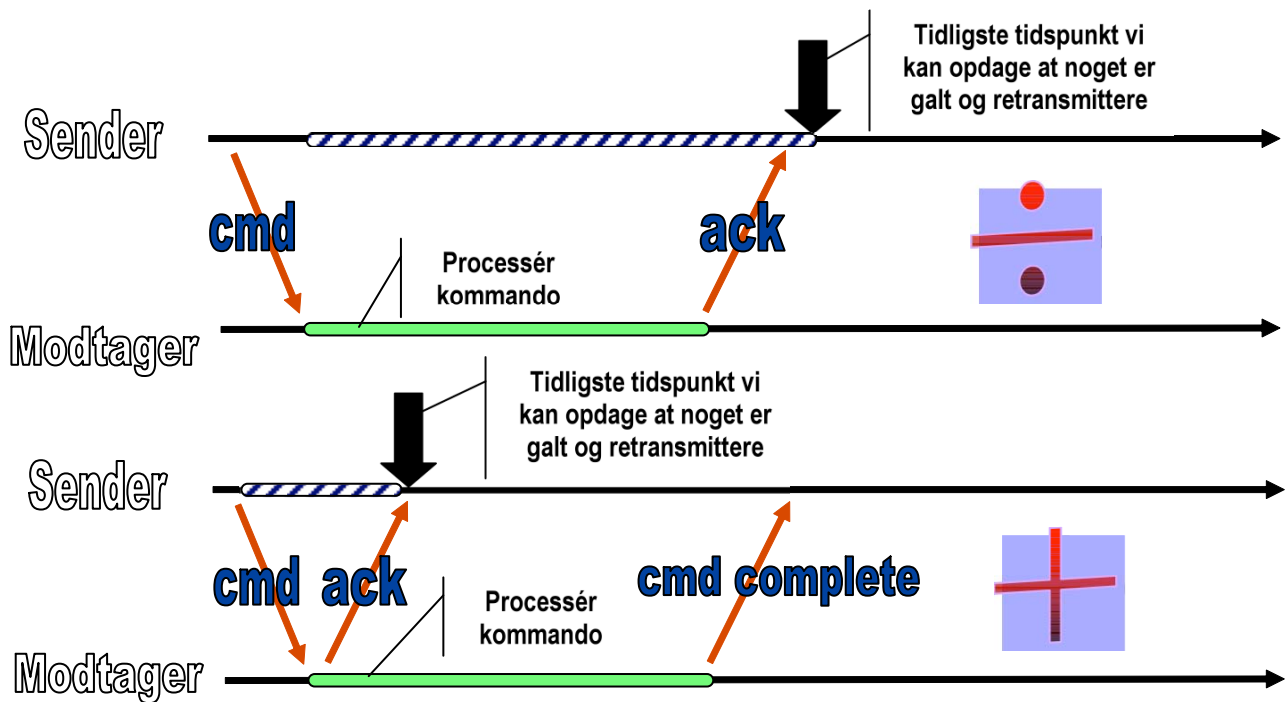
- Fejlkorrigerende koder kræver mere af sender og modtagers dataprocessingsevne, og er derfor ikke velegnede i applikationer hvor der er begrænsede processor- og lagerressourcer. Men fejlkorrigerende koder har den store fordel at de øger pålideligheden uden at øge latency.
- Retransmission er nemt at implementere, men har den ulempe at det ikke bare øger latency men også gør den mere uforudsigelig (mindre deterministisk).

Især det sidste, at retransmission giver både større og mindre deterministisk latency, er stærkt uønskeligt i kontrol-applikationer. Men samtidigt er retransmission et næsten uundværligt værktøj til at højne pålideligheden af trådløs kommunikation. Overordnet er konklusionen at retransmission er et nødvendigt onde der skal holdes på et minimum, når vi snakker trådløs kommunikation til brug i industrielle kontrol-applikationer.

Minimering af retransmissioner og effekten af retransmissioner

De oplagte løsninger

Når man skal minimere antallet af retransmissioner og den effekt de har, skal man indledningsvis starte med at lave fornuftige protokoller ovenpå den trådløse kommunikation. I Figur 2 below illustreres det at man kan kvittere for modtagelsen af en kommando mere eller mindre smart. Hvis man venter med at kvittere for modtagelsen af en kommando til efter den er udført, tager det længere tid før afsenderen (via en udeblevet kvittering) er i stand til at opdage at den skal gense data. Kvitteres der derimod straks efter modtagelsen, kan afsenderen ved udeblevet kvittering, hurtigere gense kommandoen, hvilket samlet set mindsker den gennemsnitlige forsinkelse i udførelsen af kommandoer = mindre latency.



Figur 2. En smart og en mindre smart måde at sende kvitteringer på.

Optimalt kanalvalg

Alle kneb gælder når det handler om at mindske antallet af retransmissioner, f.eks. brugen af fejlkorrigerende koder. Alt i alt er der ikke noget nyt heri, det har designerne af de nyeste trådløse teknologier selvfølgelig tænkt på. Et område der derimod ikke er så velundersøgt endnu, er muligheden for hele tiden at vælge den mest optimale kanal til den trådløse kommunikation.

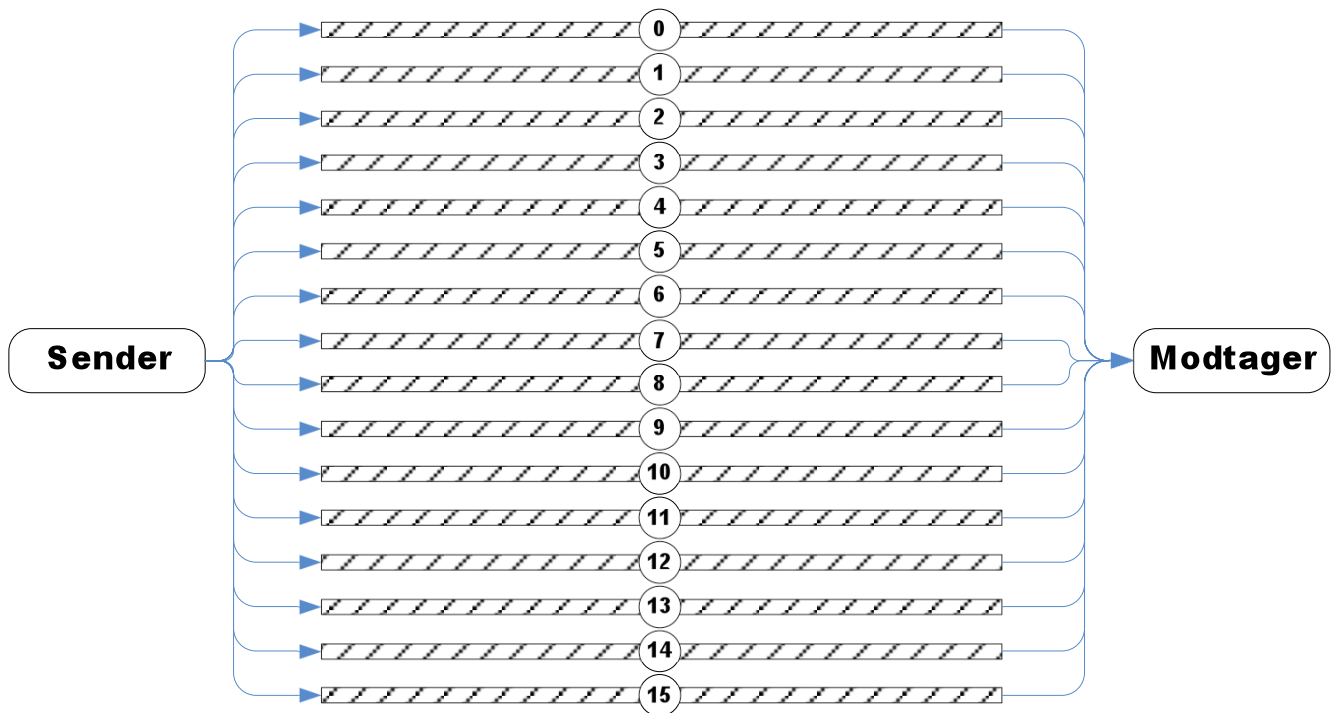
IEEE 802.15.4

Her forlader vi så den generelle diskussion af trådløse teknologier, og kigger nærmere på én bestemt, nemlig IEEE 802.15.4. IEEE 802.15.4 (fremover: 802.15.4) er interessant af flere årsager. Den er designet direkte til kortdistance (WPAN, Wireless Personal Area Networks) low-power kontrol-applikationer, den kan benyttes som en selvstændig protokolstak eller sammen med en overbygning som i f.eks. ZigBee. Som de fleste andre trådløse kommunikationsstandarder kan 802.15.4 benyttes i flere frekvensområder, men fælles for dem alle, er at det er det licensfrie ISM (Industrial, Scientific, Medical) bånd ved 2.4 GHz der er mest interessant.

Uden at gå i dybden med de tekniske detaljer, er her en oversigt over de mest interessante data for brugen af 802.15.4 i 2.4 GHz-båndet:

- Der arbejdes i området 2,400-2,4835 GHz.
- Der benyttes DSSS og O-QPSK modulation med en bitrate på 250 kb/s.
- Der benyttes 16 forskellige kanaler.

Det der i denne sammenhæng er mest interessant er at der benyttes 16 forskellige kanaler. En sender og dens modtagere har altså et valg mellem 1 ud af 16 forskellige kanaler hvorpå de kan kommunikere (nummereret fra 0 til 15):



Figur 3. 802.15.4 har 16 kanaler at vælge mellem.

Fremkommeligheden på de forskellige kanaler vil hele tiden variere, afhængigt af støjforhold, refleksioner af signalet osv. Men hvis netværket hele tiden var i stand til at vælge den kanal der havde den største fremkommelighed, ville antallet af retransmissioner, og dermed latency, blive minimeret. Da forholdene på den enkelte kanal hele tiden ændrer sig, betyder det i praksis at netværket løbende må overvåge alle kanaler for hele tiden at kunne vælge den bedste.

At måle kvaliteten på en given kanal kan ske på to måder:

- Man kan løbende måle på hvor mange pakker man rent faktisk er i stand til at overføre. Hvis en afsender forventer svar/kvittering for alle afsendte beskeder, kan man på afsendersiden måle kvaliteten af forbindelsen som

$$PRTCP = \frac{\# \text{ Svar modtaget} \times 100}{\# \text{ Kommandoer sendt}} \%$$

hvor PRTCP = *Packet Round-Trip Completion Rate*. PRTCP er ikke god til at måle hvor mange kommandoer der er blevet overført til modtageren, idet en pakke godt kan være blevet overført mens svaret/kvitteringen er gået tabt. Så i princippet kan alle pakker fra afsenderen være kommet frem mens alle svarene er gået tabt, hvilket vil resultere i en PRTCP på 0 %. Men det er uden den store betydning for PRTCP som indikator af kanalkvalitet – hvis alle svarene går tabt, går 50 % af pakkerne i nettet tabt uanset hvad og kanalen må vurderes som dårlig. Så måler man en PRTCP på 0 % bør der skiftes kanal, uanset de præcise omstændigheder.

PRTCP er et værktøj til løbende, på afsendersiden, at måle kvaliteten af den kanal man aktuelt anvender. Bemærk i øvrigt at beregning af PRTCP ikke kræver nogen form for særlig support i netværket – den vil kunne beregnes i alle netværk der benytter en request-response protokol.

- Man kan måle aktiviteten/energiindholdet i kanalen over et bestemt tidsrum. Jo mere energi der er, jo mere støj og/eller konkurrerende datatrafik er der, og jo ringere må kanalen vurderes som værende.

I 802.15.4 er der specificeret en mekanisme som gør det muligt at måle energiindholdet på en kanal, kaldet ED (Energy Detection) Scan. I praksis betyder det at man kan få et tal ud i intervallet 0-255, som udtrykker hvor meget energi der er målt i kanalen i et bestemt (kortvarigt) tidsrum.

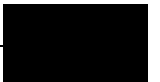
Resten af dette white-paper beskriver en metode til at benytte disse mekanismer (løbende beregning af PRTCP og ED scan) til løbende at vælge den optimale kanal i IEEE 802.15.4 baserede netværk.

Præmisser

Der ligger visse antagelser bagved ideen om at det kan betale sig hele tiden at prøve at vælge den bedste kanal:

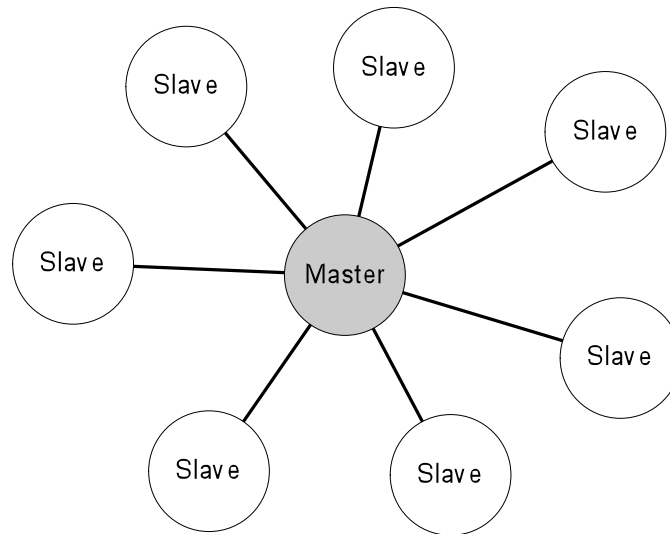
1. *De primære støjkilder genererer tilpas smalbåndet støj.* Hvis de dominerende støjkilder udsender støj der er så bredbåndet at samtlige 16 kanaler, i det 83,5 MHz brede bånd 802.15.4 benytter, ”jammes”, hjælper det ikke at skifte kanal.
2. *Der er et vist mål af korrelation mellem de primære støjkilders temporale og frekvensmæssige opførsel.* Med andre ord, når en støjkilde optræder på en kanal, forbliver den på kanalen i en vis tid. Hvis der ikke er en vis korrelation, sådan at støjkilderne optræder kortvarigt og på skiftende, tilfældige kanal, vil det ikke kunne betale sig at skifte kanal (fordi forstyrrelserne kun er kortvarige inden de flytter til en anden kanal).
3. *Omkostningen ved at overvåge alle kanaler kan negligeres.* En typisk 802.15.4 transceiver enhed kan ikke både overføre payload trafik og samtidigt lave en ED scanning. Så i den tid hvor der laves et ED scan, kan der ikke overføres payload trafik, hvilket betyder at den effektive overførselshastighed for payload mindskes (og latency øges fordi payload trafik må vente til ED scanningen er færdig). Så enten må den tid der netto bruges på ED scanning være så lille at den kan negligeres, eller også må nettogevinsten ved hele tiden at være på den bedste kanal kunne opveje omkostningen ved at gennemføre ED scanninger.

Mht. punkt 2, støjkilder i 2,4 GHz båndet, er der kun publiceret meget få artikler der belyser typiske støjkilder og deres opførsel. Visse typer af ”støjkilder”, som f.eks. WLANs der arbejder i 2,4 GHz båndet, er selvfølgelig kendte, men det er ikke lykkedes at finde undersøgelser der antyder andet end at ganske mange trådløse teknologier kan sameksistere overraskende godt i 2,4 GHz båndet. Det er slet ikke lykkedes at finde publicerede resultater eller data der belyser problemer med støjkilder i industrielle miljøer.



Net-topologi

Den beskrevne protokol og tilhørende algoritmer er designet til at fungere i et IEEE 802.15.4-baseret peer netværk med denne topologi (en stjernetopologi):



Figur 4. Net-topologi.

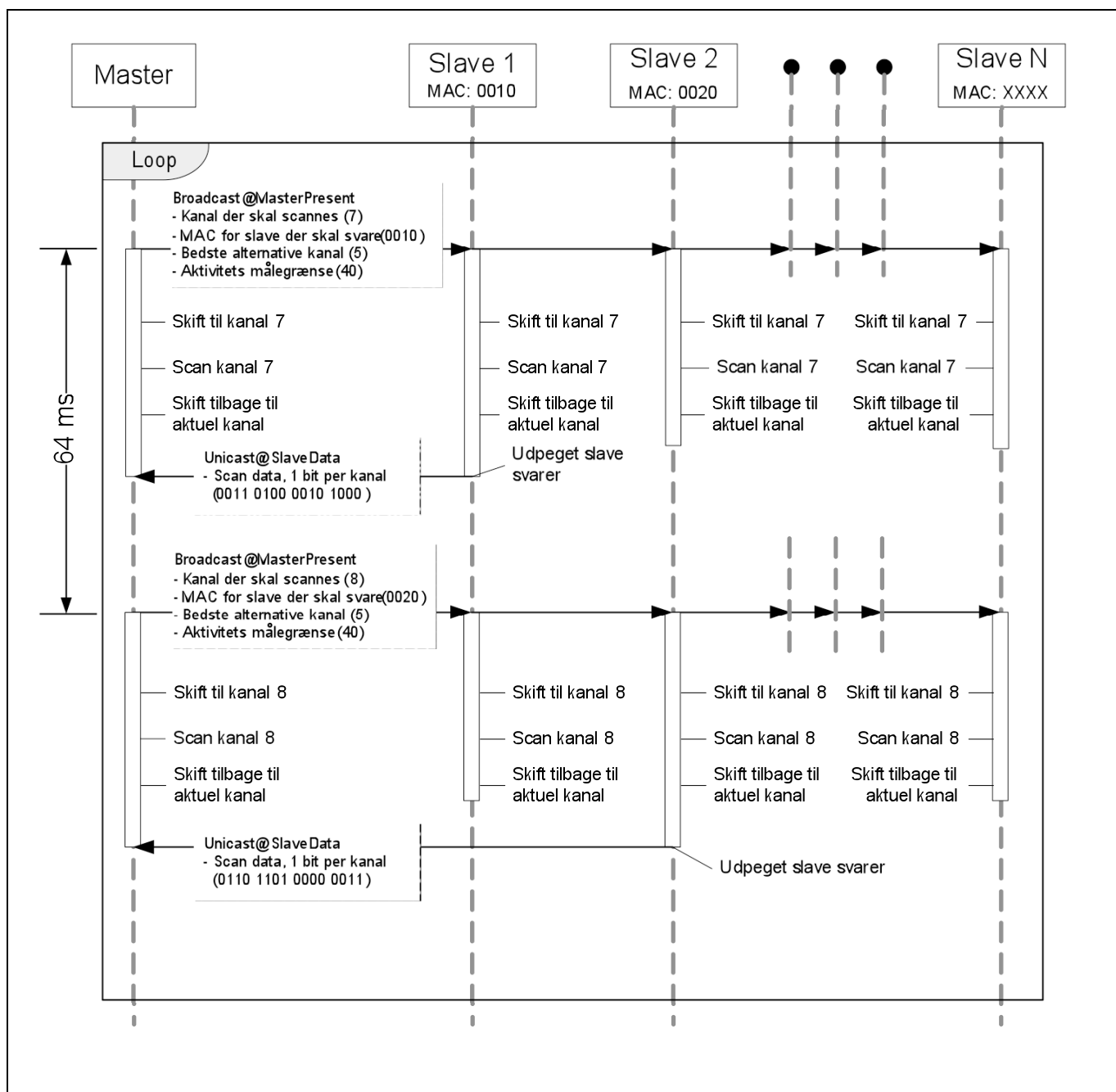
En master kommunikerer med et antal slaver, som hver især kun kommunikerer med masteren. Der benyttes ikke den mulighed der findes i 802.15.4 for at lade en koordinator i nettet styre alle knuders adgang til nettet via et beacon-signal. Der er derfor tale om et peer-netværk, idet alle knuder selv bestemmer hvornår de vil sende og modtage.

Beskrivelse af algoritme

Nedenstående beskrives protokollens og algoritmernes virkemåde på 4 niveauer/i 4 situationer:

1. Når masteren og mindst en slave har fundet hinanden – ”connected state”.
2. Når masteren initierer et styret kanalskifte for hele netværket.
3. Når en slave mister forbindelsen til masteren.
4. Når netværket startes og master og slave skal til at finde hinanden på nettet.

Connected state – ”normal drift”



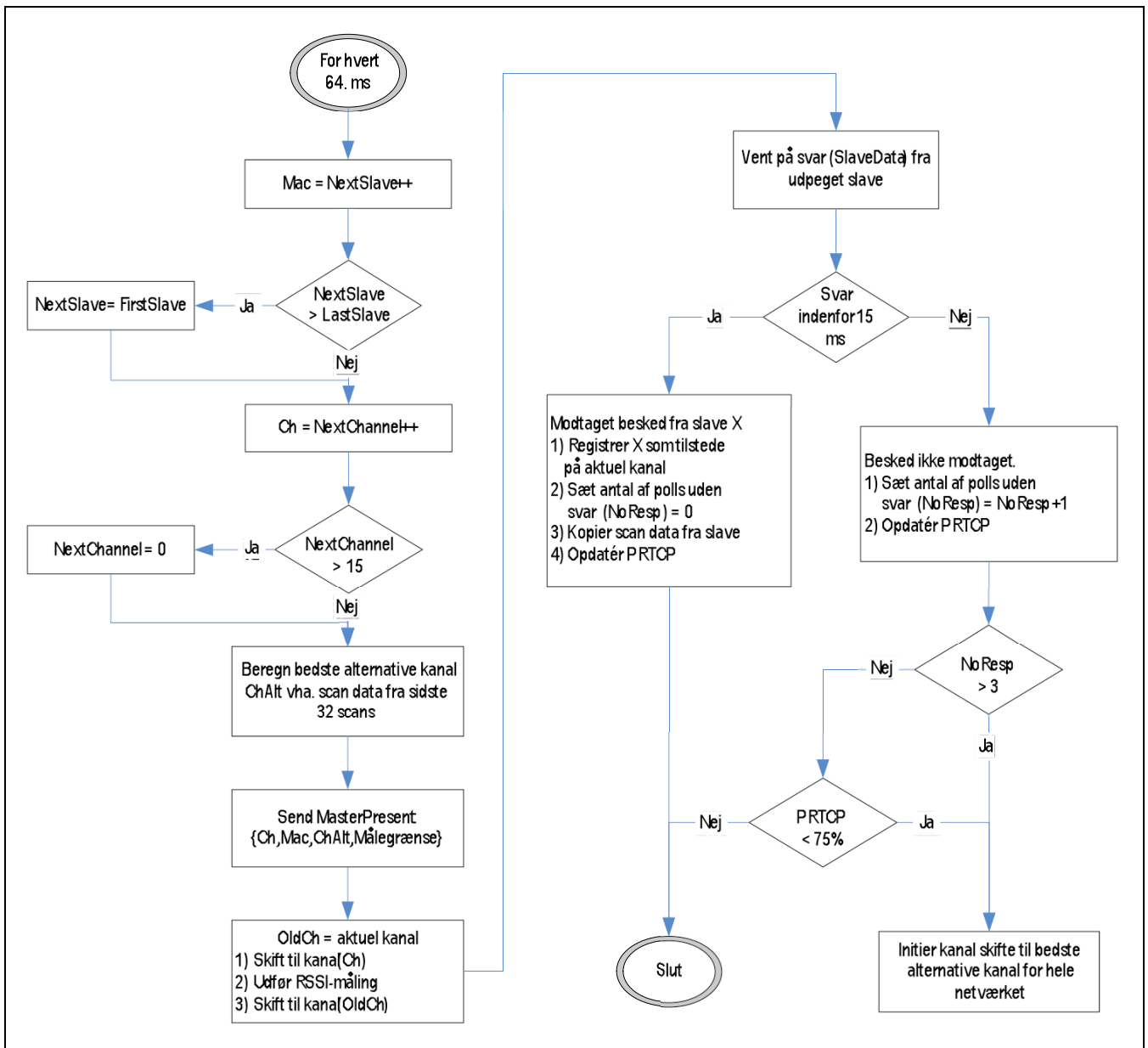
Figur 5. Sekvensdiagram, connected state.

- Med et fast interval på 64 ms broadcaster masteren en MasterPresent besked. I denne besked findes følgende parametre:
 1. Den næste kanal der skal scannes for aktivitet. Dette er en rundløbende værdi der kører igennem værdierne 0-15 og så starter forfra.
 2. MAC adressen på den slave der forventes at returnere data for kanalaktivitet. Dette er en rundløbende værdi som løber alle slavernes MAC-adresse igennem og så starter forfra.
 3. Den p.t. bedste alternative kanal til den nettet aktuelt benytter.
 4. Den værdi der skal bruges som grænse for at afgøre om det energiniveau der måles i en kanal (RSSI-værdi) indikerer at kanalen er optaget af støj eller transmissioner.
- *Rationaler:*
 - *Et MasterPresent broadcast hvert 64 ms betyder at nettet får scannet alle kanaler i løbet af ca. 1 sekund (1024 ms helt præcist).*
 - *Ved at bygge protokollen op om et masterstyret broadcast med helt faste intervaller åbnes mulighed for low-power applikationer hvor knuderne sover ind i mellem hvert broadcast.*
 - *For at minimere trafikken ifbm. hvert MasterPresent broadcast, er det kun en af slaverne som returnerer værdier for scanning af kanaler. Så længe der er færre slaver end kanaler (16), kan masteren nå at polle en slaves scandata for en given kanal hjem inden slaven scanner kanalen igen. Hvis alle slaver skulle svare, ville det øge den tid nettet var utilgængeligt for payload trafik, hvilket ville betyde mere tabt båndbredde eller højere latency for payload trafik.*
 - *Masteren orienterer hele tiden slaverne om hvad den p.t. bedste alternative kanal til den nuværende er. Skulle en slave blive "tabt" kan den bruge denne information som et godt gæt på hvor den kan gå hen og genfinde netværket.*
 - *Masteren fortæller hele tiden slaverne hvor høj en RSSI-værdi der skal måles på en kanal for at denne skal betragtes som optaget (af støj eller andre knuder). Dette åbner mulighed for en adaptiv algoritme/protokol der tager hensyn det generelle støjniveau (noise floor).*
 - *De periodisk MasterPresent beskeder tjener dels til at hjælpe slaver med at finde masteren på nettet, dels som et værktøj til at polle scandata hjem fra slaverne.*
- Så snart MasterPresent beskeden er broadcastet, skifter master og alle slaver der hørte beskeden, til den udpegede kanal og scanner denne for aktivitet. Når en knude har målt en RSSI-værdi, sammenlignes denne med den aktivitetsmålegrænse der stod i MasterPresent-beskeden. Er RSSI-værdien over denne, registreres den målte kanal som "optaget", ellers som "ledig". Hver knude vedligeholder en bitvektor med 16 bits (en for hver kanal), hvor en 1-bit indikerer at knuden har målt den pågældende kanal som "optaget" og en 0-bit indikerer at kanalen er målt som ledig.
 - *Rationale:*
 - *Ved at nøjes med at registrere en kanal som optaget/ikke-optaget kan knuderne nøjes med en bit per kanal (16) til at opbevare resultatet af scanning af alle kanaler, dvs. to bytes eller et 16-bit ord.*
- Herefter skifter alle knuder tilbage til den p.t. aktive kanal. Den slave der er udpeget i MasterPresent-beskeden returnerer sin bitvektor med scandata. Slaven sender sit SlaveData-svar som en unicast besked til masteren, men uden at bede om en kvittering fra masteren (og dermed uden at benytte retransmission ved udeblevet kvittering). Slaverne returnerer scandata for alle kanaler når de polles, ikke kun data for den sidste scanning af en kanal. De øvrige (ikke-udpegede) slaver sender ikke noget, men gemmer bare resultatet af scanningen.
 - *Rationaler:*
 - *Ved at lade alle master og slaver scanne samtidigt, mistes mindst båndbredde, idet der alligevel ikke kan sendes/modtages payload-trafik mellem master og slaver så længe*

- masteren scanner en kanal for aktivitet.*
- *Der er ikke nogen grund til at benytte den sædvanlige 802.15.4 kvitterings-/retransmissionsmekanisme. Det forhold at der ikke kommer noget svar, er nok til at fortælle masteren at enten har den udpegede slave ikke hørt MasterPresent, eller også er SlaveData-svaret gået tabt. I begge tilfælde skal den løbende vurdering af den aktuelle kanals kvalitet nedgraderes.*
 - *Ved ikke at benytte den sædvanlige 802.15.4 kvitterings-/retransmissionsmekanisme, mindskes den tid netværket ikke er til rådighed for payload-trafik, hvilket mindsker tabet af båndbredde og sænker latency for payload-trafik.*
 - Når masteren selv er færdig med at scanne den udpegede kanal, lytter den i et begrænset tidsrum (15 ms) efter et svar (en SlaveData-besked) fra den udpegede slave. Masteren opdaterer en løbende vedligeholdt værdi for PRTCP (Packet Round-Trip Completion Rate), et tal mellem 0 og 100 %, som indikerer hvor mange svar på MasterPresent beskeder den har modtaget. PRTCP beregnes simpelt som:

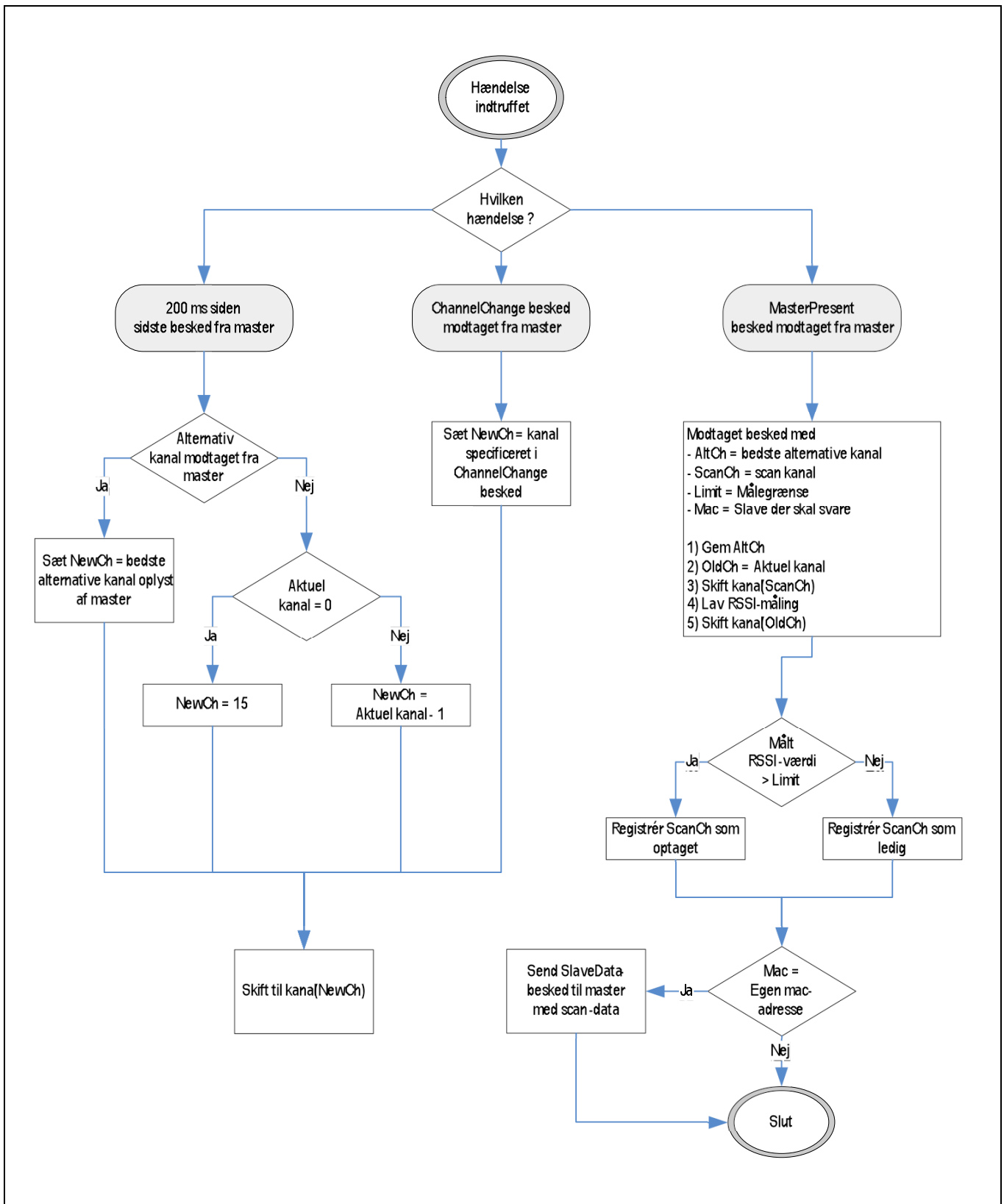
$$\frac{\# \text{SlaveData pakkermodtaget} \times 100}{\# \text{MasterPresent pakkersendt}} \%$$
 - *Rationaler:*
 - *PRTCP udtrykker hvor mange komplette "handshakes" (MasterPresent sendt fra master og modtaget af en slave og SlaveData sendt fra slave og modtaget af master) der er udført. 0 % betyder at alle handshakes fejler, altså at hvert forsøg på at sende en pakke fra master til en slave og tilbage igen fejler fordi der er mistet pakker (hvad enten det er MasterPresent eller SlaveData). 100 % betyder at alle pakker (både MasterPresent og SlaveData kommer igennem).*
 - *PRTCP måles over de sidste 64 handshakes der er forsøgt gennemført, og giver et udtryk for den aktuelle kanals kvalitet målt over en vis tid (2-4 sekunder)*
 - Hvis PRTCP falder under 75 % eller hvis der ikke er kommet svar fra en slave på 3 efter hinanden følgende MasterPresent broadcasts, vil masteren initiere et kanalskifte for netværket.
 - *Rationaler:*
 - *Ved at lade alle master og slaver scanne samtidigt, mistes mindst båndbredde, idet der alligevel ikke kan sendes/modtages payload-trafik mellem master og slaver så længe masteren scanner en kanal for aktivitet.*
 - *PRTCP er et udtryk for den løbende, gennemsnitlige, kvalitet af den aktuelle kanal og kan derfor registrere en gradvis forringelse af kanalens kvalitet.*
 - *Antallet af svar der udebliver i træk bruges til at registrere pludselige, transiente forstyrrelser og agere herpå.*
 - Sammenfattende: Når master og slaver har fundet hinanden på en kanal, står masteren altså, via MasterPresent-beskeder, og beder alle knuder i nettet om at scanne alle kanaler, fra 0 til 15 og forfra igen. Samtidigt benytter masteren MasterPresent-beskederne til rundløbende at polle hver knude for dens lagrede scandata.
 - *Sammenfattende rationaler:*
 - *Ved at benytte broadcasts mindskes den mængde pakker protokollen udveksler. Dels kan alle slaver modtage MasterPresent samtidigt, dels skal slaverne ikke kvittere for modtagelsen.*
 - *Ved at benytte ikke-kvitterede unicast pakker fra slaver til master mindskes den mængde pakker protokollen udveksler.*
 - *Den samlede effekt er at den tid nettet er utilgængeligt for payload-trafik mindskes, hvilket øger den effektive båndbredde og mindsker latency for payload-trafik.*

Flow-chart – master



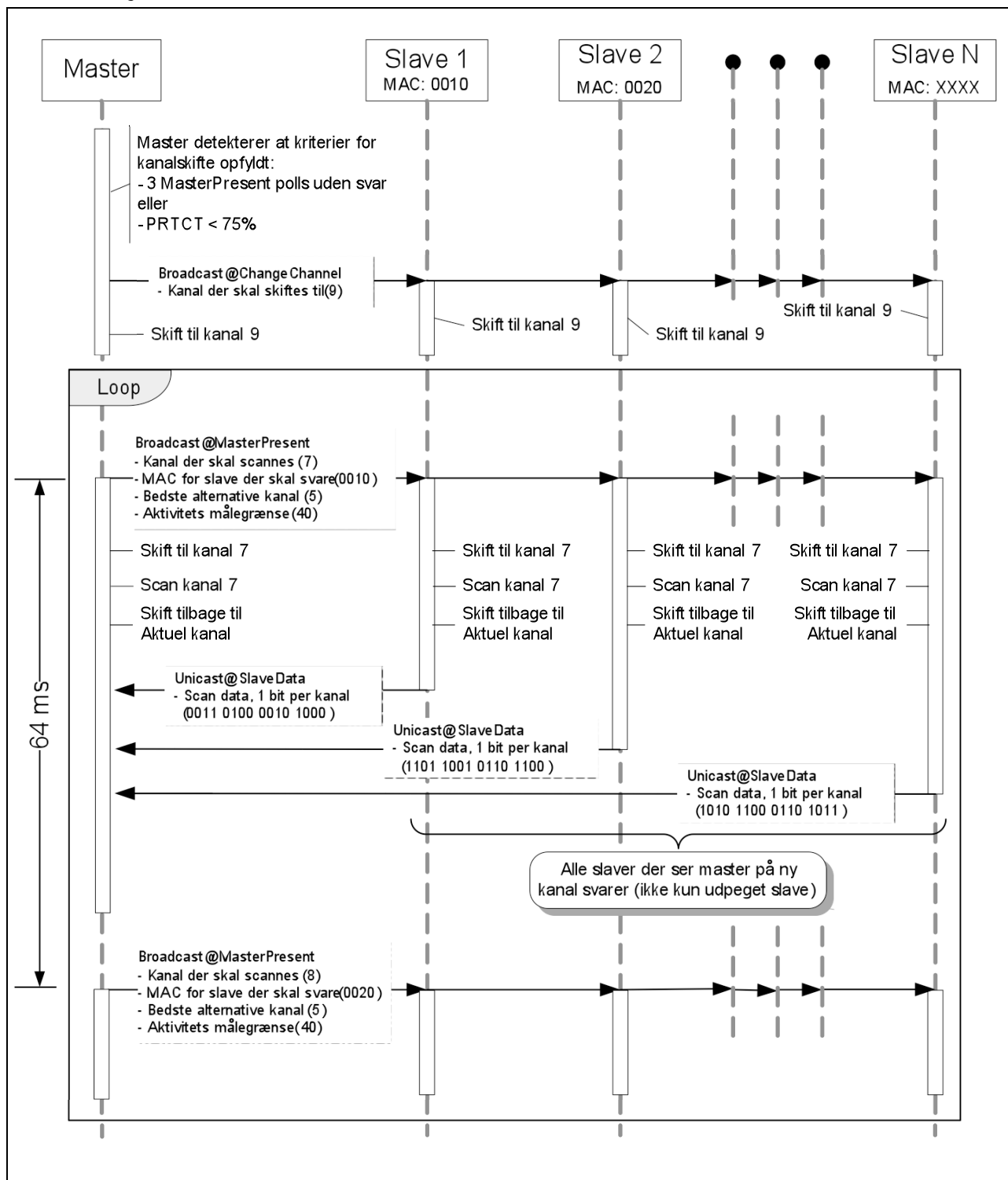
Figur 6. Flow-chart for master i connected state.

Flow-chart slave



Figur 7. Flow-chart for slave i connected state.

Masterstyret kanalskifte



Figur 8. Sekvensdiagram, frekvensskifte.

- Efter udsendelse af hver MasterPresent og tilhørende poll af scandata fra en udpeget slave, opdaterer masteren 2 parametre: PRTCT og antallet af polls uden svar. Hvis PRTCT målt over de sidste 64 polls falder under 75 %, eller hvis der har været 3 på hinanden følgende polls uden svar fra slaver, beslutter masteren at skifte kanal.
- De gemte scandata (masterens egne plus dem der pollet hjem fra slaverne) bruges til at finde den p.t. bedste alternative kanal.
- Masteren broadcaster en ChannelChange-besked med den nye kanal til alle slaver.

- Masteren og alle slaver der hørte ChannelChange-beskedene skifter øjeblikkeligt til den nye kanal.
- Masteren resetter sin 64 ms poll cyklus og udsender straks den næste MasterPresent-besked på den nye kanal og fortsætter herefter sine med de sædvanlige 64 ms mellemrum.
 - *Rationale:*
 - *Ved straks at sende et MasterPresent broadcast på den nye kanal (snarere end at vente op til 64 ms), har master og slaver mulighed for straks at finde hinanden på den nye kanal.*
- Første gang en slave ”opdager” masteren på en ny kanal, sender den altid en SlaveData-besked, uanset om den er udpeget til at svare eller ej.
 - *Rationale:*
 - *Det er vigtigt at alle master og alle slaver hurtigt genfinder hinanden på en ny kanal. For at undgå at masteren først ved om en slave er ”kommet med” når den specifikt polles, svarer alle slaver altid på den første MasterPresent de ser på en ny kanal.*

Autonomt (fall-back) kanalskifte

Master

Masteren kører i en fast 64 ms poll cyklus uanset hvilken kanal den er på, også efter at have skiftet til ny kanal eller lige efter power-on.

1. Dette indebærer at hvis den ikke har hørt fra mindst 1 slave efter 3 polls på en given kanal, skifter den kanal igen.
2. Ved 2 eller flere på hinanden følgende kanalskifte uden at der er hørt fra en slave, bruges en fall-back strategi hvor der efter det første kanalskifte (som blev beregnet på baggrund af scandata) blot skiftes til næste højere kanal (og startes forfra med kanal 0 når der har været pollet på kanal 15).

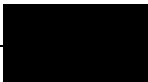
Slave

En slave der er taber forbindelsen til masteren, f.eks. fordi den ikke hører en ChannelChange-besked fra masteren, benytter flg. fall-back strategi:

1. Når den ikke har hørt fra masteren på den nuværende kanal i 200 ms, skifter den kanal.
2. Hvis slaven har lagret en alternativ kanal fra en MasterPresent-besked den tidligere har hørt, vil den først prøve at springe derhen. Så hvis den kanal der var specificeret i en misset ChannelChange-besked er den samme som var anført som bedste alternative kanal i den sidste MasterPresent-besked slaven har hørt, vil slaven direkte springe til den korrekte kanal og begynde at søge der.
3. Hvis slaven ikke har lagret nogen alternativ kanal (f.eks. fordi den lige er blevet tændt) overgår slaven til en strategi hvor den, startende med den kanal den nu tilfældigvis er på, lytter på hver kanal i 200 ms, hvorefter den skifter til næste, lavere kanal (og starter forfra ved kanal 15 når den har lyttet på kanal 0). Dette fortsætter den med indtil den finder masteren.

Initiering

Ved power-on/reset starter master og slaver på kanal 0.



Resultater

Simuleret testbænk

Den beskrevne protokol/algoritme er blevet implementeret og afprøvet i simulerede omgivelser.

Beslutningen om først at udvikle og teste protokol og algoritmer vha. en simulator skyldes flere forhold:

- Det er betydeligt hurtigere at udvikle og afprøve koden på en alm. udviklings-PC. En ændring i koden kræver kun en re-compilering hvorefter ændringen straks kan afprøves, uden at ny kode først skal overføres/flashes til en lille embedded platform. Det er også betydeligt nemmere at finde og rette fejl på en PC med fuld symbolsk debugging, breakpoints, inspektion af variable osv.
- Det er meget nemmere at simulere bestemte støj- og forstyrrelsesmønstre i software end at skulle opsætte en testopstilling hvor en avanceret HF-signalgenerator først skal konfigureres/opsættes. Samtidigt er der ikke problemer med uønsket, indstrålet støj.

Den udviklede simulator simulerer et fuldt 802.15.4 MAC-lag og til dels det fysiske lag, forholdsvis detaljeret:

- Begivenheder på det fysiske lag ("air interfacet") simuleres med en opløsning på 10 μ s, svarende til 20 chips á 500 ns (der går 32 chips på et symbol á 4 databits) eller den tid det tager at sende 2,5 databits på air interfacet.
- Kollisioner simuleres (om end lidt for stringent – hvis der er bare 10 μ s overlap mellem to fysiske datarammer, registreres det som en kollision). I virkelighedens verden ødelægger det ikke en dataramme hvis der f.eks. ryger et par bits i rammens pre-amble.)
- MAC-lagets CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance) algoritme simuleres korrekt.
- Der sendes korrekt acknowledgement rammer og retransmitteres når de udebliver.
- Der benyttes 3 16-bits adresser i hver MAC-frame (source, destination og PAN identifier).
- Der kan simuleres vilkårlige støjmønstre, med signaler af varierende styrke og varighed ned til simulatorens opløsning på 10 μ s.

Fysiske forhold simuleres ikke korrekt:

- Alle enheder hører hinanden lige godt – der tages ikke hensyn til afstanden imellem enheder.
- Al støj under en bestemt grænse ignoreres (ødelægger ikke datarammer), mens støj over grænsen "zapper" evt. datarammer der sendes samtidigt.

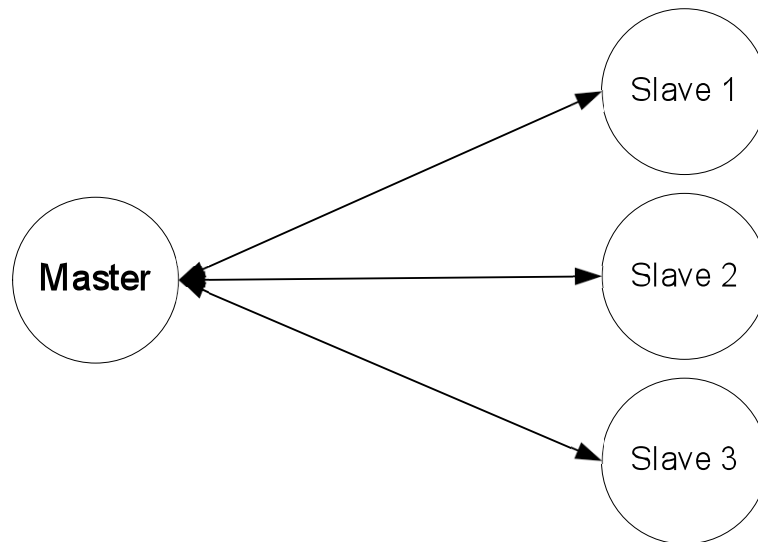
Simulatoren er skrevet i C# og fylder knapt 3800 linier kode. Det viste sig at være en større opgave at simulere 802.15.4 MAC-laget (og til dels PHY-laget) end først antaget, idet det er nødvendigt at få ganske mange detaljer med (som f.eks. korrekt simulering af CSMA-CA algoritmen) for at få brugbare "måledata" i tidsdomænet. Simulatoren blev således skrevet om undervejs i projektet. Den seneste version er, på forfatterens laptop, i stand til at simulere begivenheder med en opløsning på 10 μ s med en hastighed på omkring 60 % af realtid (dvs. det tager ca. 105 s at simulere 60 s netværksaktivitet).

Simuleret testopstilling

De efterfølgende resultater er målt i en testopstilling som vist i Figur 9 below. En master knude styrer et lille netværk med 3 slaver. Masteren gennemfører 2 typer af kommunikation med hver slave:

- Master og slaver implementerer den ovenfor beskrevne protokol og tilhørende algoritmer til optimalt kanalvalg. Så masteren broadcaster periodisk beskeder der får netværket til at måle aktiviteten på en bestemt kanal, og slaverne indrapporterer resultaterne til masteren.
- Der udveksles "payload" trafik mellem master og hver slave. Masteren genererer periodisk en "kommando" til en tilfældig slave, og slaven svarer tilbage når den har udført ordren.

Der måles så, over en periode og i et bestemt scenario, på hvor lang tid det tager at overføre og påbegynde udførelsen af ”kommandoer”, dvs. latency. Der opsamles bl.a. statistik der viser hvad den mindste, største og gennemsnitlige latency er for det givne scenario. Der er gennemført målinger for en række forskellige scenarier der belyser netværkets evne til hele tiden at vælge den optimale kanal under forskellige støjforhold. For hvert scenario sammenlignes netværkets ydelse med og uden protokollen til optimalt kanalvalg.

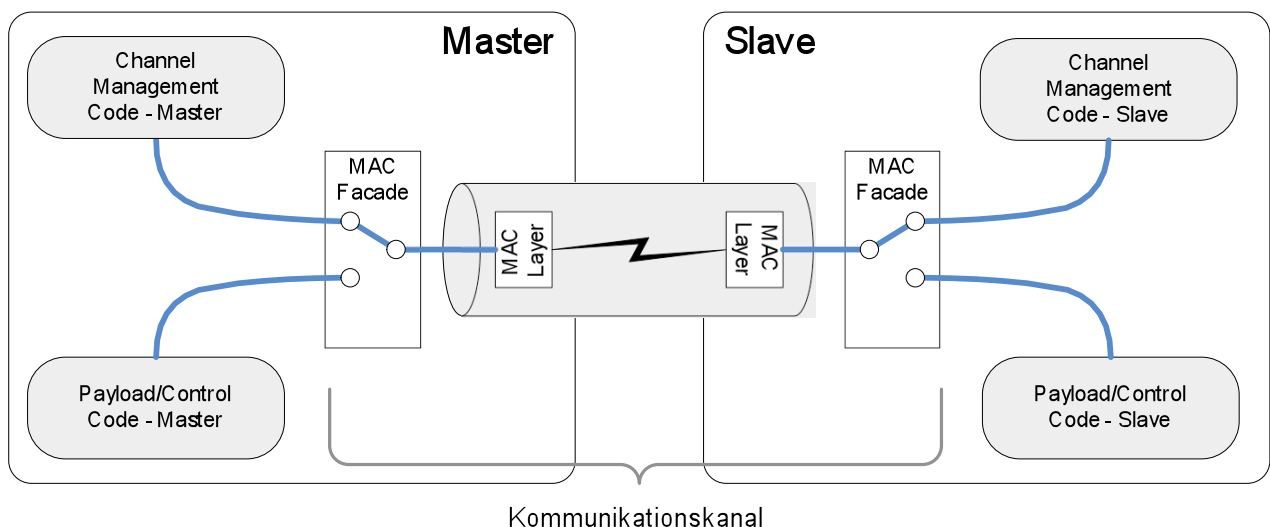


Figur 9. Simuleret testopstilling.

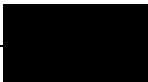
Følgende figur illustrerer testopstillingen på en lidt anden måde. MAC-laget på master og slave er gemt bagved en ”MAC Facade”. MAC-facadens opgave er at multiplekse to forskellige datastrømme over den samme kommunikationslinie:

1. Payload-data, dvs. kommandoer fra masteren og slavens svar herpå.
2. Data der hidrører fra protokollen til optimalt kanalvalg.

I praksis sker der det, at når noget kode ønske at sende en pakke, beder den MAC-facaden om adgang til MAC-laget. Når denne adgang gives, har den pågældende kode fuld eneret til MAC-laget indtil koden slipper MAC-laget igen. Så længe koden til kanalovervågning og -valg (kaldet Channel Management Code på figuren) bruger MAC-laget, kan der ikke overføres payload-trafik (kommandoer og svar) og omvendt.



Figur 10. To datastrømme multiplekseres over samme kommunikationslinie.



Scenarier

Fælles for de nedenfor beskrevne scenarier er:

- Der måles på netværkets opførsel/overførsel af data i ét minut.
- Master og slaver får lov til at finde hinanden (det tager ca. 8 ms) inden netværket udsættes for støj.
- Masteren sender med tilfældige intervaller på 250-500 ms en 4-bytes ”kommando” til en tilfældig slave.
- Der lægges hele tiden en ikke-forstyrrende baggrundsstøj på alle kanaler.

Scenario 1: Ingen forstyrrelser

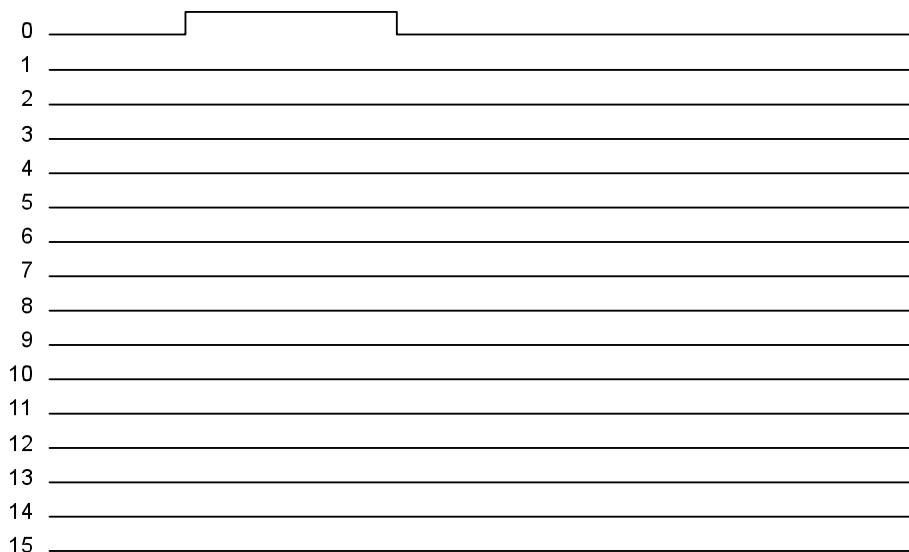
I dette scenario genereres der ikke støj i netværket, som altså blot kan benytte den samme kanal hele tiden. Dette gør det muligt direkte at se det overhead som protokol og algoritmer til optimalt kanalvalg påfører netværket.

1. Netværket starter og knuderne finder hinanden på kanal 0.
2. Master genererer payload-trafik (kommando-svar) til tilfældige slaver.

Scenario 2: En enkelt blokeret kanal

I dette scenario lægges massiv støj ind på den kanal netværket benytter (kanal 0) efter stykke tid. Forventningen er at netværket skifter til en ledig kanal. Når der ikke benyttes protokol til optimalt kanalvalg, er netværket effektivt spærret så længe støjen er til stede.

1. Netværket starter og knuderne finder hinanden på kanal 0.
2. Master begynder at generere payload-trafik (kommando-svar) til tilfældige slaver.
3. Efter 5 sekunder lægges massiv støj på kanal 0 i 20 sekunder.

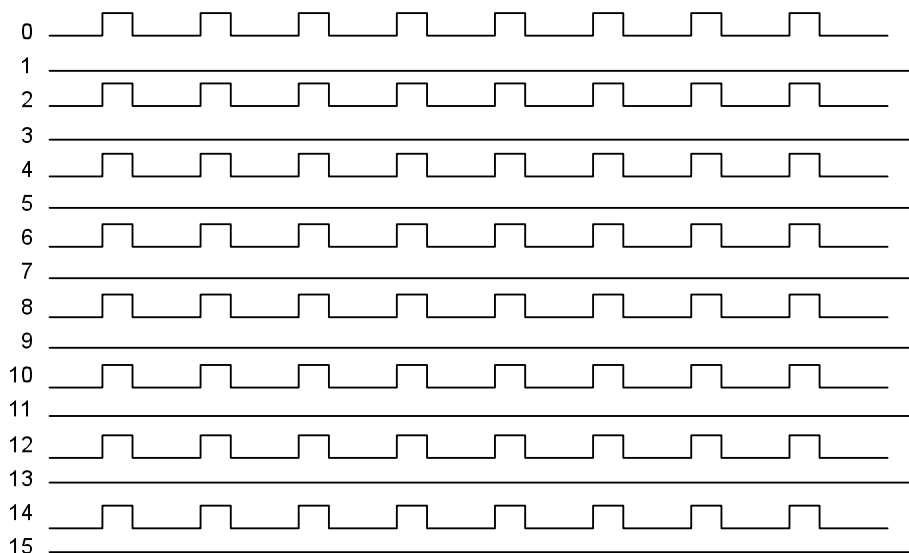


Figur 11. Simuleret scenario 2.

Scenario 3: Periodiske, langvarige blokeringer på 8 samtidige kanaler

I dette scenario lægges massiv støj periodisk ind på halvdelen af alle kanaler.

1. Netværket starter og knuderne finder hinanden på kanal 0.
2. Master begynder at generere payload-trafik (kommando-svar) til tilfældige slaver.
3. Efter 5 sekunder lægges massiv, periodisk støj ind på kanalerne 0,2,4,6,8,10,12,14. Støjen forekommer som 1 sekunders pulser med 5 sekunders mellemrum. På de øvrige kanaler er der kun uforstyrrende baggrundsstøj.

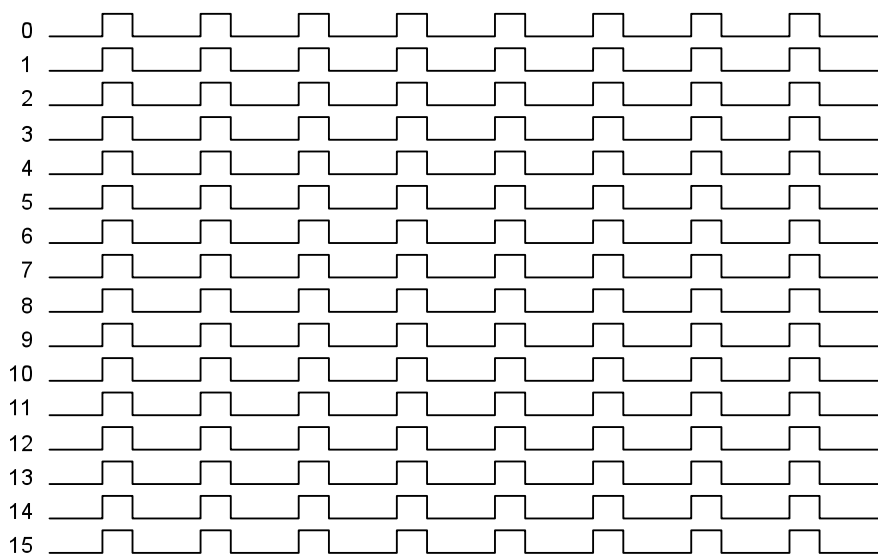


Figur 12. Simuleret scenario 3.

Scenario 4: Periodiske, langvarige blokeringer på alle kanaler samtidigt

I dette scenario lægges massiv støj periodisk ind på alle kanaler.

1. Netværket starter og knuderne finder hinanden på kanal 0.
2. Master begynder at generere payload-trafik (kommando-svar) til tilfældige slaver.
3. Efter 5 sekunder lægges massiv, periodisk støj ind på alle kanaler. Støjen forekommer som 1 sekunders pulser med 5 sekunders mellemrum.

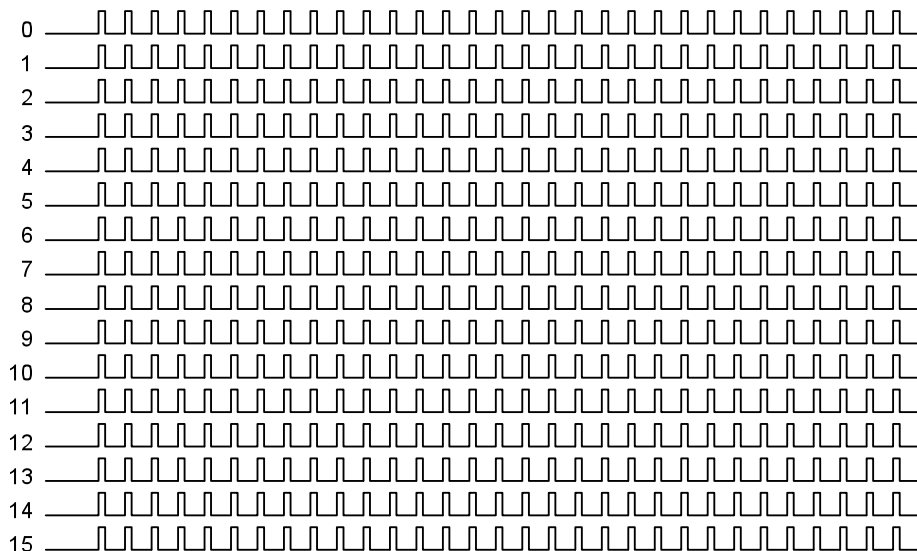


Figur 13. Simuleret scenario 4.

Scenario 5: Periodiske, kortvarige blokeringer på alle kanaler samtidigt

I dette scenario lægges massiv støj periodisk ind på alle kanaler.

1. Netværket starter og knuderne finder hinanden på kanal 0.
2. Master begynder at generere payload-trafik (kommando-svar) til tilfældige slaver.
3. Efter 5 sekunder lægges massiv, periodisk støj ind på alle kanaler. Støjen forekommer som 0.1 sekunders pulser med 0.5 sekunders mellemrum.

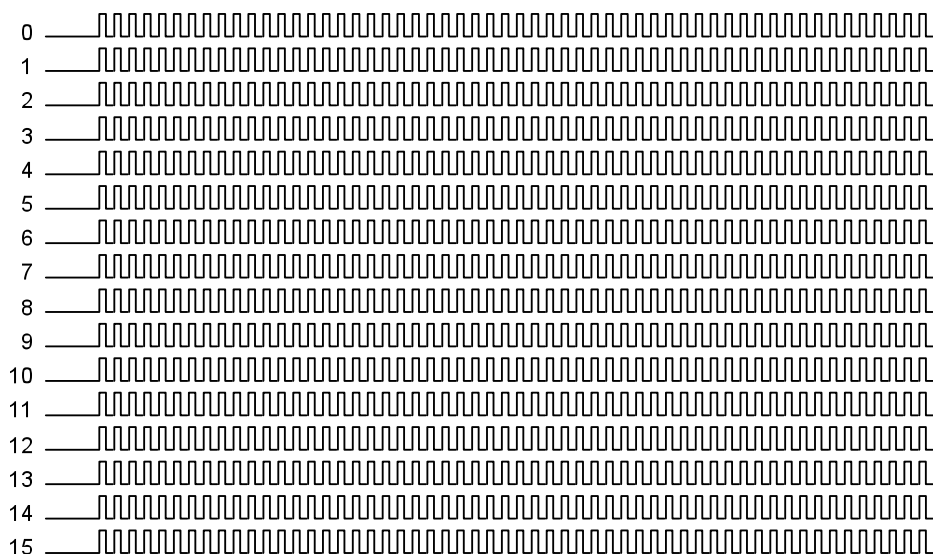


Figur 14. Simuleret scenario 5.

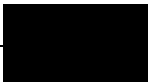
Scenario 6: Periodiske, kortvarige og hyppige blokeringer på alle kanaler samtidigt

I dette scenario lægges massiv støj periodisk ind på alle kanaler.

1. Netværket starter og knuderne finder hinanden på kanal 0.
2. Master begynder at generere payload-trafik (kommando-svar) til tilfældige slaver.
3. Efter 5 sekunder lægges massiv, periodisk støj ind på alle kanaler. Støjen forekommer som 0.1 sekunders pulser med 0.2 sekunders mellemrum.



Figur 15. Simuleret scenario 6.



Resultater målt i simulator

Scenario	Parameter	Protokol/algorithmte til optimalt kanalvalg		
		Ikke anvendt	Anvendt	
1	Nul-scenario			
	Antal payload pakker sendt af master	162	162	
	Antal payload pakker modtaget af slave	162	162	
	Overførselsrate (modtaget x 100)/sendt	100.0 %	100.0 %	
	Antal retransmissioner	0	0	
	Latency (min/max/gennemsnit)	1.04 / 3.28 / 2.26 ms	1.04 / 3.28 / 2.23 ms	
	Antal pakkekollisioner	0	0	
	Antal pakker ødelagt af støj	0	0	
	Pakker ødelagt af støj/kollisioner i alt	0 (0.0 %)	0 (0.0 %)	
Antal kanalskift	-	0		
2	Kanal 0 blokeret i 20 sek			
	Antal payload pakker sendt af master	162	154	
	Antal payload pakker modtaget af slave	122	154	
	Overførselsrate (modtaget x 100)/sendt	75.3 %	100.0 %	
	Antal retransmissioner	117	0	
	Latency (min/max/gennemsnit)	1.04 / 3.28 / 2.14 ms	1.04 / 3.28 / 2.22 ms	
	Antal pakkekollisioner	0	0	
	Antal pakker ødelagt af støj	156	4	
	Pakker ødelagt af støj/kollisioner i alt	156 (24.2 %)	4 (0.2 %)	
Antal kanalskift	-	1		
3	Kanal 0, 2, 4, 6, 8, 10, 12, 14 blokeret af 1 sek pulser med 5 sek mellemrum			
	Antal payload pakker sendt af master	162	154	
	Antal payload pakker modtaget af slave	137	154	
	Overførselsrate (modtaget x 100)/sendt	84.6 %	100.0 %	
	Antal retransmissioner	75	0	
	Latency (min/max/gennemsnit)	1.04 / 3.28 / 2.22 ms	1.04 / 3.28 / 2.22 ms	
	Antal pakkekollisioner	0	0	
	Antal pakker ødelagt af støj	100	4	
	Pakker ødelagt af støj/kollisioner i alt	100 (15.4 %)	4 (0.2 %)	
Antal kanalskift	-	1		
4	Alle kanaler blokeret af 1 sek pulser med 5 sek mellemrum			
	Antal payload pakker sendt af master	162	102	
	Antal payload pakker modtaget af slave	137	100	
	Overførselsrate (modtaget x 100)/sendt	84.6 %	98.0 %	
	Antal retransmissioner	75	6	
	Latency (min/max/gennemsnit)	1.04 / 3.28 / 2.22 ms	1.04 / 3.28 / 2.16	
	Antal pakkekollisioner	0	6	
	Antal pakker ødelagt af støj	100	303	
	Pakker ødelagt af støj/kollisioner i alt	100 (15.4 %)	309 (13.2 %)	
Antal kanalskift	-	147		

5	<i>Alle kanaler blokeret af 0.1 sek pulser med 0.5 sek mellemrum</i>		
	Antal payload pakker sendt af master	162	161
	Antal payload pakker modtaget af slave	139	141
	Overførselsrate (modtaget x 100)/sendt	85.8 %	87.6 %
	Antal retransmissioner	70	67
	Latency (min/max/gennemsnit)	1.04 / 3.28 / 2.16 ms	1.04 / 5.36 / 2.31 ms
	Antal pakkekollisioner	0	0
	Antal pakker ødelagt af støj	101	241
	Pakker ødelagt af støj/kollisioner i alt	101 (15.4 %)	241 (10.1 %)
	Antal kanalskift	-	0
6	<i>Alle kanaler blokeret af 0.1 sek pulser med 0.2 sek mellemrum</i>		
	Antal payload pakker sendt af master	161	126
	Antal payload pakker modtaget af slave	120	98
	Overførselsrate (modtaget x 100)/sendt	74.5 %	77.8 %
	Antal retransmissioner	130	90
	Latency (min/max/gennemsnit)	1.04 / 11.76 / 2.40 ms	1.04 / 8.56 / 2.22 ms
	Antal pakkekollisioner	0	8
	Antal pakker ødelagt af støj	179	486
	Pakker ødelagt af støj/kollisioner i alt	179 (27.3 %)	494 (21.2 %)
	Antal kanalskift	-	76

Tabel 1. Målte resultater.

Kommentarer til resultater

Generelt

Når netværket udsættes for stød, bliver det selvfølgelig sværere at få pakker igennem. Dette ses dog primært på tallene for overførselsraten. Når støjen bliver så generende at pakker ind i mellem slet ikke kan overføres, får det med den valgte målemetode ikke konsekvens for målt latency, idet kun pakker der rent faktisk kommer igennem tæller med i statistikken for latency (pakker der ikke kommer igennem har jo principielt en uendelig forsinkelse og tælles derfor ikke med).

Scenario 1: Ingen forstyrrelser

Tallene viser at når der ikke er nogen forstyrrelser, er der ikke nogen omkostninger for brug af protokol/algorithm til optimalt kanalvalg (i hvert fald hvad angår latency og tabt båndbredde).

Scenario 2: En enkelt blokeret kanal

Når der ikke anvendes protokol/algorithm til optimalt kanalvalg, koster det pålidelighed (lavere overførselsrate) når den kanal der bruges jammes helt. Når optimalt kanalvalg anvendes, finder netværket hurtigt en ledig kanal, hvilket giver en markant højere pålidelighed uden omkostninger i form af højere latency. Den tid der går med at skifte kanal 1 gang, giver et lille båndbreddetab (der overføres lidt mindre payload-trafik).

Scenario 3: Periodiske, langvarige blokeringer på 8 samtidige kanaler

Billedet og konklusionerne er de samme som i scenario 2. Når protokol/algorithm til optimalt kanalvalg anvendes finder nettet hurtigt en alternativ kanal.

Scenario 4: Periodiske, langvarige blokeringer på alle kanaler samtidigt

De ganske langvarige støjpulser er stærkt generende. Når der ikke anvendes protokol/algorithm til optimalt kanalvalg, koster det pålidelighed (lavere overførselsrate) når den kanal der bruges jammes hyppigt og langvarigt – næsten hver 7. pakke tabes. Når optimalt kanalvalg anvendes, udløser de langvarige støjpulser hyppige kanalskift (147). Det koster båndbredde (der sendes kun 102 payload-

pakker mod 162 uden protokol/algoritme). Til gengæld øges den effektive pålidelighed, fordi nettet ikke prøver at sende payload-pakker når støjen blokerer nettet men prøver at finde en ledig kanal så længe støjen varer. Man kan sige at man giver afkald på båndbredde mod at få en bedre pålidelighed.

Scenario 5: Periodiske, kortvarige blokeringer på alle kanaler samtidigt

Når støjpulserne bliver så kortvarige, og med tilpas stor afstand, ligner resultaterne med og uden protokol/algoritme hinanden igen. Det skyldes at støjpulserne ikke varer længe nok, eller ødelægger pakker nok, til at udløse kanalskift. Når protokol/algoritme anvendes fås en lidt højere pålidelighed, mod en lidt højere maksimal latency.

Scenario 6: Periodiske, kortvarige og hyppige blokeringer på alle kanaler samtidigt

Dette scenario ligner resultatmæssigt scenario 4. Når afstanden mellem de korte støjpulser bliver kort nok, ødelægges mange pakker, hvilket udløser en del kanalskift (76). Dette har igen den effekt, at den effektive pålidelighed øges, fordi nettet ikke prøver at sende payload-pakker når støjen blokerer nettet men prøver at finde en ledig kanal så længe støjen varer.

Konklusion

Med afsæt i de ønsker man kan have til kommunikationskanal i industrielle kontrol-applikationer, er der udviklet en protokol og algoritmer, der forsøger at benytte specielle mekanismer i den trådløse standard IEEE 802.15.4, til løbende at lade et netværk vælge en kanal med høj fremkommelighed. Protokol og algoritmer er afprøvet i simulerede omgivelser der simulerer MAC-laget i IEEE 802.15.4 (og til dels PHY-laget) forholdsvis detaljeret. Resultaterne indikerer, at så længe at der er en eller flere kanaler med markant højere fremkommelighed end andre, vil netværket benytte disse. Det primære udbytte er højere pålidelighed. Resultaterne indikerer også, at når alle kanaler samtidigt forstyrres tilstrækkeligt meget, bruger protokol og algoritmer en forholdsvis stor del af tiden på at lede efter en bedre kanal. Da der ikke sendes payload-trafik så længe netværket leder efter en ny kanal, er nettoresultatet en højere pålidelighed (flere af de sendte pakker når frem), men en lavere effektiv båndbredde. Man kunne indvende at det samme kunne opnås meget simplere, ved at simpelthen ikke at sende data når der er generende støj eller forstyrrelser, men det er ikke rigtigt, idet man ikke derved høster gevinsten ved at flytte til en mere fremkommelig kanal hvis en sådan findes.

På dette stade er den største mangel ved resultaterne, udover at der bruges simulerede omgivelser, at det ikke vides om de støj- og forstyrrelsesmønstre der er anvendt her er realistiske. De er dels valgt for at belyse specifikke egenskaber ved protokol/algoritme, dels for at stresser protokol/algoritme. Hvis ”typisk” industriel støj (hvis der findes noget sådant) er tilpas bredbåndet til at jamme alle de 16 kanaler IEEE 802.15.4 benytter i 2.4 GHz, er resultaterne som her beskrevet måske ganske realistiske.

Samlet set er resultaterne lovende, men det er let at pege på en række opfølgende opgaver der kunne afklare om den foreslåede teknik til optimalt kanalvalg reelt kan bruges:

- Protokol og algoritmer skal tunes til kun at forsøge at skifte til en bedre kanal, hvis de løbende kanalscanninger i netværket indikerer at der er en markant bedre, alternativ kanal.
- Der skal simuleres med støj- og forstyrrelsesmønstre som empirisk er påvist som almindelige og realistiske.
- Protokol og algoritmer skal naturligvis afprøves på rigtig radio-hardware, som udsættes for støj og forstyrrelser i realistiske, industrielle miljøer.