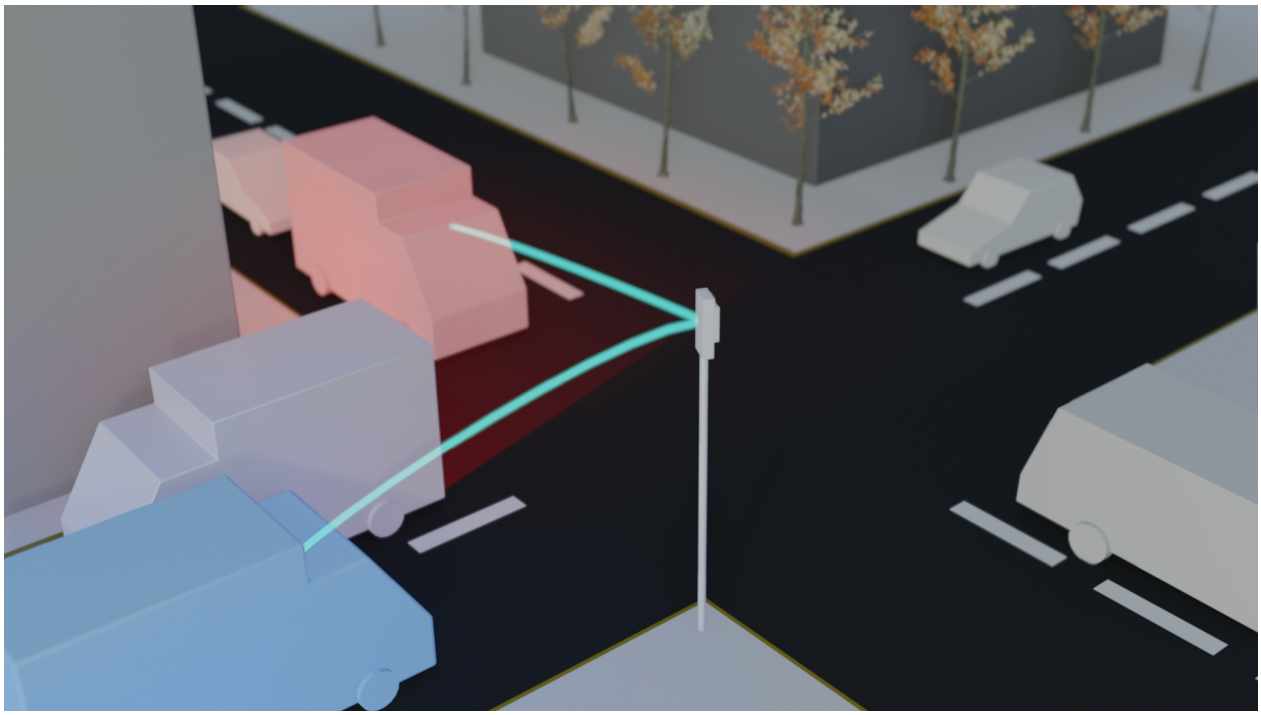
CHALMERS | GÖTEBORGS UNIVERSITET



Automation av ledade kommersiella tunga fordon

Virtuella trafikljus

Ett kandidatarbete vid Instutitionen för Data- och Informationsteknik

Erik Strid, Iman Radjavi, Joar Blom Rydell,
Oskar Nilsson, Viktor Lyster, William Kruse

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2019

Automation av ledade kommersiella tunga fordon

Virtuellt trafikljus

Erik Strid, Iman Radjavi, Joar Blom Rydell, Oskar Nilsson,
Viktor Lyster, William Kruse

Handledare: Elad Michael Schiller, Institutionen för Data- och Informationsteknik

Examinator: Sven Knutsson, Institutionen för Data- och Informationsteknik

Kandidatarbete 2019
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
SE-412 96 Göteborg
Sverige
Telefon +46 31 772 1000

Omslagsbild: Illustration av infrastrukturbaserad kommunikation mellan virtuellt trafikljus och autonoma fordon. Författarnas egen bild.

Göteborg, Sverige 2019

Abstract

With the development and introduction of autonomous vehicles, opportunities for communication and sharing information between agents are introduced. A virtual traffic light, i.e. a connected system with a server and communicating vehicles, enables coordination between vehicles at an intersection and a way to optimize the traffic flow. The system should be fault tolerant in the presence of communication failures, and be scalable and resource-efficient. With the help of given algorithms and a system for intersection simulations, a virtual traffic light could be implemented. The project proposes an implementation of a system for coordinating autonomous vehicles through infrastructure-based communication, which was evaluated through tests and experiments. A realistic network simulator was developed to evaluate the system's fault tolerance in the presence of communication failures. The system is scalable and can be operated on a microcomputer with limited computing resources, but further evaluation of the system's fault tolerance is required.

Keywords: Virtual traffic light, V2I, V2V, fault tolerant, scalable, infrastructure, communication, network simulation

Sammandrag

Med utvecklingen och införandet av autonoma fordon introduceras möjligheter för informationsdelning och kommunikation mellan aktörer i trafiken. Ett virtuellt trafikljus, det vill säga ett uppkopplat system med server och kommunicerande fordon, möjliggör koordination mellan fordon i korsningen och ett sätt att effektivisera trafikflödet. Systemet ska vara feltolerant vid förekomsten av kommunikationsproblem, samt vara skalbart och resurssnålt. Med hjälp av givna algoritmer och ett system för korsningssimuleringar kunde ett virtuellt trafikljus implementeras. Projektet föreslår en implementation av ett system för koordinering av autonoma fordon genom infrastrukturbaserad kommunikation, som evaluerades genom tester och experiment. En verklighetstrogen nätverkssimulator utvecklades för att evaluera systemets feltolerans vid förekomst av kommunikationsproblem. Systemet är skalbart och kan drivas på en mikrodator med begränsade beräkningsresurser, men ytterligare evaluering av systemets feltolerans krävs.

Nyckelord: Virtuellt trafikljus, V2I, V2V, feltolerant, skalbart, infrastruktur, kommunikation, nätverksimulering

Förord

Vi vill tacka alla som under projektets gång hjälpt till med projektet. Elad Michael Schiller för hans kontinuerliga expertis, stöd och motivation. Emelie Ekenstedt, Karl Kangur, Anand Hariharan och Reema Sweeny Pinto för deras studier som möjliggjort detta projekt. Burkin Günke som har varit till stor hjälp med att bygga upp vår förståelse för laborationsutrustningen. Vi vill även tacka Volvo Group Trucks Technology för deras tillkännagivande och motivation kring projektet.

Ett stort tack till Chalmers som bidragit med en laborationssal där vi effektivt kan utveckla och testa våra system.

Författarna - Maj 2019, Göteborg

Nomenklatur

Bandbredd: *Den teoretiska maxgräns på hur mycket data ett nätverk kan skicka och ta emot.*
CPU (Central Processing Unit): *Komponenten i en dator som exekverar (utför) program.*
GulliView: *Visuellt positionssystem som används i den fysiska evalueringsmiljön.*
I2I (Infrastructure-to-Infrastructure): *Kommunikation mellan infrastruktur och infrastruktur.*
MAC-adress (Media access control): *En unik identifierare för varje enskilt nätverkskort.*
Mbps (Megabit per sekund): *Överföringshastigheten på den information som skickas mellan två punkter.*
NS-3: *Diskret nätverkssimulator*
ROS: (Robot Operating System): *Operationssystem som möjliggör kommunikation mellan interna system och andra robotar.*
RViz: *3D-visualiseringsverktyg till ROS.*
Throughput: *Den verkliga mängd data som kan sändas från sändare till mottagare inom ett specifikt tidsintervall.*
UDP (User Datagram Protocol): *Förbindelselöst kommunikationsprotokoll över IP-nätverk.*
V2I (Vehicle-to-Infrastructure): *Kommunikation mellan fordon och infrastruktur.*
V2V (Vehicle-to-Vehicle): *Kommunikation mellan fordon.*
ZooKeeper: *Centraliserad tjänst som möjliggör lagring av information.*

Innehåll

1 Inledning

Enligt *WHO* (World Health Organization) omkommer varje år 1,35 miljoner människor till följd av trafikolyckor [1]. I en statistisk undersökning gjord av Singh [2], framgår det att den mänskliga faktorn var den kritiska orsaken till 94% av de olyckor som skedde mellan år 2005 och 2007 i USA. Genom införande av autonoma fordon skulle den siffran drastiskt kunna minska, hävdar Isidore [3]. Autonoma fordon är ett högaktuellt utvecklings- och forskningsområde och redan idag färdas olika grader av semi-autonoma fordon på vägarna. En aktör i fordonsbranchen förutspår en ökning av autonom körning på drygt 130% från år 2018 till år 2019 [4].

Med utvecklingen och införandet av autonoma fordon introduceras möjligheten för informationsdelning genom kommunikation, vilket i sin tur möjliggör koordination av autonoma fordon. Denna koordination kan implementeras i trafiksituationer som kräver samverkan mellan fordon, exempelvis vid korsningar. Många av dagens autonoma fordon är utrustade med ett antal olika sensorer, som exempelvis kameror och LiDAR-sensorer, för att orientera sig. Kommunikation är en simpel och prestandasnål lösning på koordineringsproblemet, jämfört med sensorer i fordon som enbart kan uppfatta sin egen omgivning och har begränsningar i möjligheten att samverka med andra fordon.

Fysiska trafikljus är i dagsläget en lösning för koordinering av fordon. Däremot har dagens fysiska trafikljus, med sina enklare sensorer och signaleringsmöjligheter, begränsningar när det kommer till att utbyta information med sin omgivning. Dagens trafikljus är därför inte optimala när det kommer till att anpassa sig efter omgivningen för att effektivisera trafikflödet.

För att göra autonoma fordon ännu säkrare, kan man komplettera sensorerna genom att koordinera dem med hjälp av kommunikation. Ett virtuellt trafikljus är ett uppkopplat system som koordinerar autonoma fordon genom kommunikation och informationsdelning. Det kan vara information om fordonens position, hastighet och intentioner. Det ger möjlighet till planering och effektivisering av trafikflöden. Enligt Osborne görs tester redan nu i Pittsburgh [5], där fordonen delar viktiga datapunkter mellan sig.

Kommunikation i allmänhet, och trådlös kommunikation i synnerhet, introducerar risken för paketförluster och nätverksfördröjningar. Ett uppkopplat system måste därför vara feltolerant för att kunna vara tillräckligt robust vid förekomsten av kommunikationsfel. Nästa generations mobila nätverk 5G kommer, med sin låga nätverksfördröjning, möjliggöra att virtuella trafikljus kan prestera snabbare och säkrare. Enligt Ericsson kommer 5G erbjuda ultrahög pålitlighet och tillgänglighet vilket gör kommunikationsproblem extremt sällsynta [6]. Under våren 2019 ska en utbyggnadsplan för 5G presenteras med målet att det ska komma i bruk år 2020, enligt Campanello [7].

Gruppen kommer med vägledning av och i samarbete med Volvo Group Trucks Technology undersöka och utveckla ett feltolerant och robust system där autonoma ledade tunga fordon koordineras genom infrastrukturbaserad kommunikation.

1.1 Syfte

Syftet med projektet är att designa och utveckla ett feltolerant system för koordination av autonoma ledade tunga fordon vid olika trafiksituationer genom infrastrukturbaserad kommunikation. Målet är att effektivisera trafikflödet på ett säkert och kostnadseffektivt sätt. Genom simulering och implementation i en fysisk evalueringsmiljö kan det demonstreras vad autonoma fordon kan uppnå genom kommunikation mellan infrastruktur och autonoma fordon.

1.2 Problem

Det övergripande problemet är att introducera kommunikation mellan aktörer i en trafiksituation. Detta för att skapa ett säkert och effektivt trafikflöde utifrån projektets begränsningar och med feltolerans i åtanke.

Risker uppstår när flera fordon vill utnyttja samma vägyta samtidigt, speciellt när de måste korsa olika vägbanor, som vid korsningar och skarpa kurvor. Enligt *NHTSA* (National Highway Transportation Safety Agency) stod kollisioner i korsningar för 40% av det totala antalet olyckor i trafiken i USA [8].

Flera fordon som samtidigt befinner sig i närheten av en korsning ska genom kommunikation enas om vem som har företräde. Det finns tre olika protokoll som kan användas vid korsningar för detta. Det första protokollet innefattar ett enväldigt realtids virtuellt trafikljus som koordinerar fordonen (V2I). Det andra innefattar att fordonen kommunicerar med varandra och låter den med högst prioritet köra först (V2V). Det tredje innefattar att ett realtids virtuellt trafikljus bestämmer vilka filer som har företräde, men fordonen är tillåtna att kommunicera och ge varandra tillåtelse att bryta mot trafikljusets direktiv.

För att autonoma fordon ska kunna köra bland icke-autonoma fordon bör ett realtids virtuellt trafikljus vara kompatibelt med de nuvarande fysiska trafikljusen. Det virtuella trafikljuset innefattar en medlemsservice där autonoma fordon meddelar trafikljuset att de närmar sig korsningen och vart de planerar att åka. Det virtuella trafikljuset ger direktiv till det fysiska trafikljuset utifrån trafikläget. Utan ett virtuellt trafikljus måste de autonoma fordonen få reda på vad de fysiska trafikljuset visar på ett annat sätt, exempelvis genom kamera.

Kommunikationsfördröjningar kan orsakas av att meddelanden blir korrupta och påverkar systemets förmåga att hantera data i realtid. En kritisk miljö där autonoma fordon kan förekomma är fjärrstyrda flygplatser, som ställer höga krav på kommunikationssäkerhet och -stabilitet. Olyckor i sådana miljöer skulle vara förödande och kan medföra oerhört höga kostnader samt hälsorisker, då framkomlighet för mekaniker och räddningspersonal troligtvis är begränsad.

1.3 Avgränsningar

Projektet är avgränsat till en fysisk evalueringsmiljö samt en korsningssimulator. Den fysiska evalueringsmiljön består endast av autonoma modellastbilar och beskrivs mer detaljerat i kapitel 4 (s. 14). Eftersom det endast finns autonoma fordon i evalueringsmiljöerna, finns inget fysiskt trafikljus och projektet kommer därmed inte ta hänsyn till att göra det virtuella trafikljuset kompatibelt med något fysiskt. Följden av förenklingen är att fordonen inte har några sensorer som känner av omgivningen, vilket leder till att fordonen litar blint på det virtuella trafikljuset.

Vidare begränsas projektet till de två första protokollen (V2I och V2V), som beskrivs i avsnitt 1.2. Förslaget om koordinering av autonoma fordon genom V2V-kommunikation inkluderar algoritmer som är baserade på tidigare studier [9; 10; 11; 12; 13]. Litteraturen inkluderar ett antal förslag för feltolerans och pålitlig trådlös kommunikation som har fordonskommunikation i åtanke [14; 15; 16; 17; 18; 19; 20]. Endast implementationen av det andra protokollet (V2V) kommer integreras i den fysiska evalueringsmiljön. Ett system för det första protokollet (V2I) kommer endast evalueras i en nätverkssimulator. Implementationen för V2I-kommunikation är enklare och anser en feltolerant koordinationsservice. Det finns ett antal exempel på feltolerant service som kan vara basen för koordinationsservicen. Vårt projekt bygger på en implementation av [21] som använder en infrastruktur endast för att ta emot medlemsservicen. Vi använder Apache Zookeeper för vår medlemsservice som vi antar att alla fordon är anslutna till. Projektet kommer att foku-

sera på att evaluera tre- och fyrvägskorsningar, då den fysiska evalueringsmiljön endast består av trevägskorsningar och korsningssimulatorn av en fyrvägskorsning.

1.4 Relaterade arbeten

I en rapport presenterar Lefère et al. [22] en interaktionsmedveten modell för att resonera kring risker vid trafiksituationer. Modellen bedömer risken vid en trafiksituation genom att jämföra ett fordons avsedda manöver med vad fordonet förväntas göra enligt trafikreglerna. Simuleringen som projektet har handahållit använder sig av en modifierad riskbedömning, som bygger på den som Lefère et al. utvecklat, och ska implementeras till den fysiska evalueringsmiljön.

Faraj, Fidan och Gaudet [23] undersöker hur autonoma fordon kan minska tomgångstiden och därmed bränsleförbrukningen vid korsningar med trafikljus genom att fordonen kommunicerar. Fordonen i undersökningen använder sig av V2I-kommunikation för att optimera hastigheten fram till korsningen. Faraj et al. jämför situationer när fordonen kommunicerade med varandra och när de inte gjorde det. Resultatet visar att kooperativ hastighetsreglering ger betydligt mindre tomgångstid, antal stopp samt energiförbrukning. Undersökningen visar att samarbete kan förbättra trafikflödet genom olika trafiksituationer. Vårt projekt använder V2I-kommunikation men inte för att reglera hastigheten fram till korsningen, utan för att optimera trafikflödet genom korsningen.

Santos et al. [24] använder sig av kommunikation mellan fordon (V2V) för att reglera avståndet till fordonet framför. Likt det här projektet använder de NS-3 för att simulera ett trådlöst nätverk 802.11p (i.e. *Wireless Access in Vehicular Environments* (WAVE)) och ROS för att styra fordonen i deras simulering. Skillnaden mellan projekten är att det här projektet använder kommunikation för att koordinera fordon vid en korsning, istället för att reglera avståndet till fordonet framför.

I en rapport av Savic, Schiller och Papatriantafilou [25] föreslås en algoritm för det andra protokollet (V2V) som beskrivs i avsnitt 1.2. Likt det här projektet förutsätter de att fordonen i deras fall är helt autonoma. Skillnaden är att de endast använder sig av V2V kommunikation medan vårt projekt också tillämpar en infrastrukturbaserad koordinator för att bestämma vilket fordon som ska få passera korsningen först.

Morales-Ponce, Schiller och Falcone påpekar att nätverk bestående av rörliga noder med trådlös kommunikation har ett inneboende problem med exempelvis paketförluster [26]. De föreslår ett protokoll för hur kommunikationsproblem kan hanteras. Protokollet lämpar sig till säkerhetskritiska kooperativa fordons applikationer. Likt det här projekt använder de sig av NS-3 för att simulera hur deras protokoll tolererar kommunikationsfel, som exempelvis paketförlust.

Det finns ett flertal lösningar för hur ett virtuella trafikljuset kan gå tillväga för att undvika riskfyllda trafiksituationer i en korsning. Ett tillvägagångssätt som Wuthishuwong et al. föreslår är att implementera lås för noder som är gemensamma för flera körriktningar [27]. Genom att försöka låsa en eller flera noder under en tidsperiod, går det på så sätt att säkerställa om att vara ensam i noden. Detta tillvägagångssätt begränsar antalet bilar i korsningen, vilket kan motverka optimalt trafikflöde då efterföljande bilar ska passera korsningen i samma riktning. Ett tillvägagångssätt som detta projekt inte kommer implementera, men principen har används för att avgöra om rutter är konflikterande.

Ett annat tillvägagångssätt är att det virtuella trafikljuset kommunicerar vilka körfält som är aktiva. Genom att lagra information om fordons planerade rutt, hastighet samt position, kan det virtuella trafikljuset välja ut en mängd aktiva körriktningar. Bento, Parafita och Nunes kombinerar det virtuella trafikljuset med fysiska och kan således användas av autonoma och icke-autonoma fordon [28]. Den kombinationen skulle kunna underlätta i en eventuell övergångsperiod där det

förekommer fordon med mänskliga förare och autonoma fordon i samma korsning. Detta projekt kommer implementera en liknande lösning, med skillnaden att implementeringen introducerar feltolerans och en resurssnål lösning som möjliggör skalbarhet.

2 Metod

Arbetet utfördes i en dedikerad laborationssal på Chalmers anpassad för projektet. Laborationssalen innehåller datorer, kommunikationsutrustning samt projektets fysiska evalueringsmiljö med tillhörande fordon, som kunde användas för tester. Lokalen gav en samlingsplats, som underlättade för möten och idéutbyten mellan de grupper som hade tillgång till salen.

Till en början av projektet lades stor tid på att planera, läsa på och sätta sig in i ämnet. Det lades även tid på att lära sig och/eller förbättra sina färdigheter i programspråket Python, samt att sätta sig in i de system som skulle användas i projektet.

2.1 Agil arbetsmetod

Projektet innefattade en rad delmål som möjliggjorde att en agil arbetsmetod kunde utnyttjas. Genom att arbeta iterativt och inkrementellt, möjliggjordes ständig evaluering och stadig utveckling. Gruppen delades upp i mindre arbetsgrupper och kunde på så sätt vara kortsiktigt oberoende av varandra.

Regelbundna möten har skett veckovis med handledare för att diskutera utveckling och lämpliga steg framåt. Det har varit ett tillfälle att lyfta aktuella problem samt funderingar och på så sätt få input från en väl insatt person i ämnet. Det har även skett regelbundna möten med anställda på Volvo Group Trucks, där projektet diskuterats och utvärderats.

I och med den agila arbetsmetoden med dess iterativa och inkrementella arbetssätt, har versionshantering varit av yttersta vikt. Projektets versionshanteringsverktyg har varit *git* med både *GitHub* och senare *BitBucket* som plattform. På detta sätt har det varit enkelt att observera förändringar i kod som de olika delgrupperna skrivit och arbetat med. Genom att grena ut från huvudgrenen, har delgrupper kunnat arbeta med samma kodstycken. På så sätt arbeta separat på sina deluppgifter, för att sedan sammanfoga grenarna när deluppgifter färdigställts.

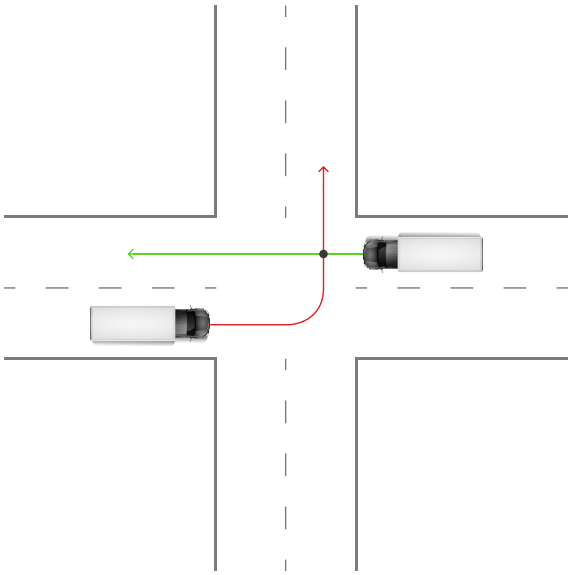
2.2 Befintligt material och system

Projektets grund består av ett system som byggts upp av tidigare kandidat- och masterarbeten. Därför krävdes att gruppen satte sig in i de givna systemen och dess delsystem, med andra ord den fysiska evalueringsmiljön samt korssningssimulatorn med V2V-protokoll, som beskrivs i kapitel 4 (s. 14). Som instuderingsmaterial fanns systembeskrivningar på GitHub, kommentarer i kod, samt tidigare studenter som var insatta i systemen.

Med inspiration av tidigare arbeten och pseudokod, tillhandahållna av handledare och nuvarande masterstudenter, implementerades sedan algoritmer. Dessa algoritmer har bevisats säkra, genom antingen logik eller på statistisk grund. I ett första steg implementerades algoritmerna i korsningssimulatorn, som möjliggjorde kontinuerlig testning och utveckling. En verklighetstrogen nätverkssimulator utvecklades, där avsiktligt skapade paketförluster matades in för att evaluera systemets feltolerans.

3 Teori för infrastrukturbaserad kommunikation

En konflikt i en trafiksituation uppstår när minst två fordon försöker uppta samma vägyta, vilket gör att det finns risk för en kollision om åtgärder inte tas. Det kan handla om att fordonen har korsande vägbanor, exempelvis vid vänstersvängar med motgående trafik, se figur 1. För att göra en sådan manövrering säker krävs antingen att det görs en riskbedömning av situationen, där risken uppskattas till minimal, eller att en överordnad beslutsfattare ger ett fordon rätt till väg. Oavsett vilken metod som väljs behövs information om fordon som befinner sig i närheten, både fordonens position och deras intentioner eller en uppskattning av dem båda.



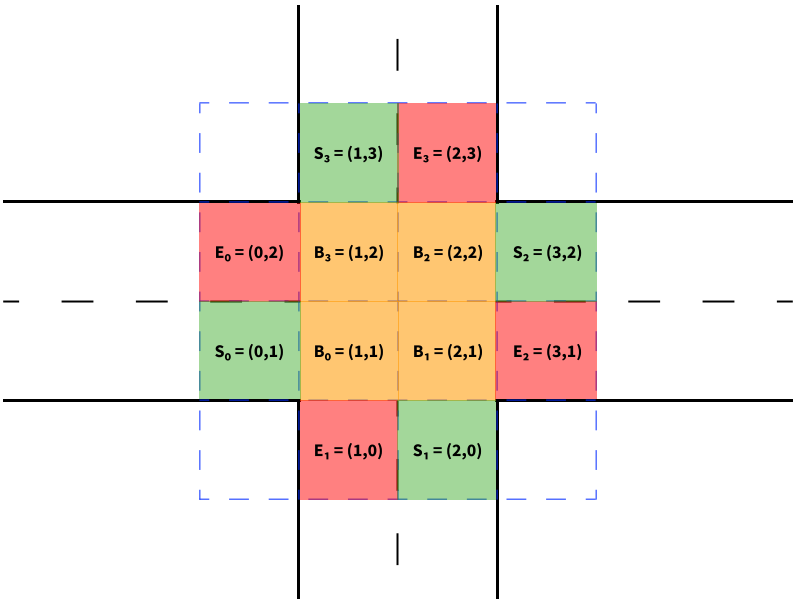
Figur 1: Exempel på riskfylld trafiksituation. *I bilden används element från Freepik.com skapat av macrovector.*

För ett system som innefattar styrning genom infrastrukturbaserad kommunikation (*V2I*), krävs det en rad algoritmer för att säkerställa systemets funktionalitet och säkerhet. Alla fordon i korsningen behöver ha identisk och korrekt information om korsningen, samt interna system för att kunna hantera trafiksituationer på ett säkert sätt.

3.1 Definition av korsning och rutter

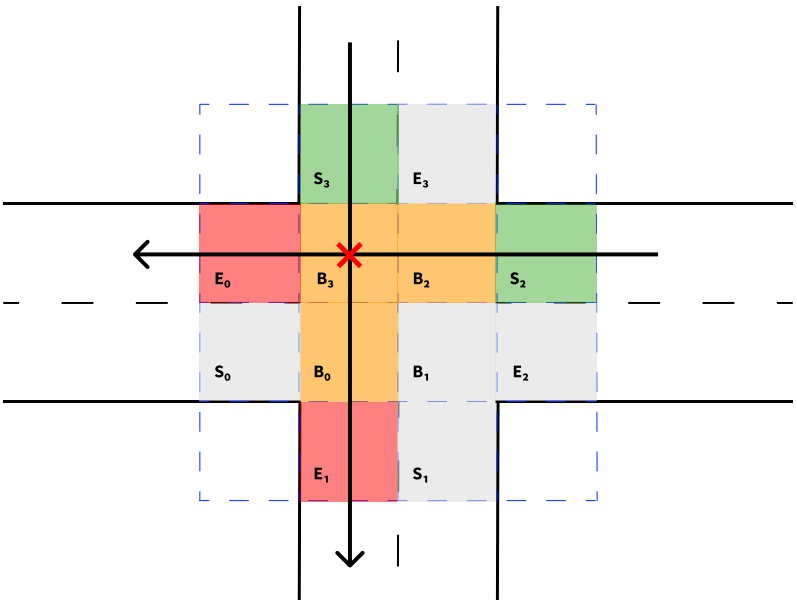
För att möjliggöra koordinering med en centralt styrande korsningshanterare, behövs alla möjliga kombinationer av samtida rutter vars körbanor inte korsar varandra. I en korsning med n förbindelser, där $n \geq 2$ och varje förbindelse består av en in- och utfart, ges att antalet möjliga rutter i korsningen är $n \cdot (n-1)$, givet att u-svängar inte är tillåtna. För en fyrvägskorsning, vilken kan ses i figur 2 (s. 7), blir det tolv möjliga rutter då det vid varje infart går att köra höger, vänster eller rakt fram.

Korsningar definieras och delas upp i ett tvådimensionellt koordinatsystem, där varje in- och utfart får en bestämd koordinat. För att avgöra om rutter hamnar i konflikt, delas även området i korsningen upp i block med koordinater. Detta visualiseras i vår fyrvägskorsning i figur 2 (s. 7), där gröna block representerar infarter, röda block utfarter och orangea block ytor som utnyttjas i korsningen beroende på manöver.



Figur 2: Uppdelning av en fyrvägskorsning.

Vilka block som kommer upptas i korsningen behöver fördefinieras för varje rutt. En raksträcka från S_2 till E_0 upptar block B_3 och B_2. En raksträcka från S_3 till E_1 upptar block B_3 och B_0. Det går då att avgöra om rutter kolliderar med varandra, det vill säga om samma block upptas av flera rutter i korsningen samtidigt. Detta illustreras i figur 3 där block B_3 är gemensam för båda rutterna vilket innebär en konflikt.



Figur 3: Två rutter i konflikt.

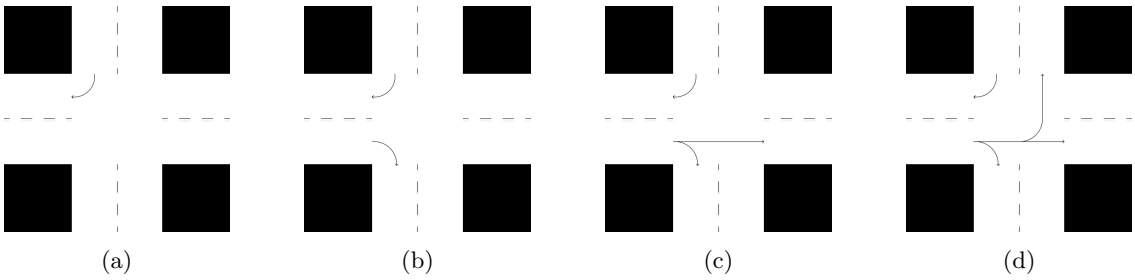
Enligt Dirichlets lådprincip [29] är det omöjligt att ha fler än n aktiva rutter i en generell n-vägskorsning, då det finns n in- och utfarter. Detta gäller eftersom varje rutt går från en infart till en utfart. En fyrvägskorsning kan därmed endast ha som mest fyra aktiva rutter samtidigt. Med

dessa antaganden i åtanke går det därav att beräkna det totala antalet möjliga kombinationer av rutter för en generell n-vägskorsning med ekvation 1. I denna beräkning inkluderas även korsande och kolliderande rutter.

$$\sum_{i=0}^{n-1} \binom{n*(n-1)}{n-i} \tag{1}$$

Mängder med färre än n rutter anses vara triviala och ineffektiva, då korsningen inte utnyttjas till sin fulla grad. För att få en så effektiv korsning som möjligt, väljs de maximala mängderna ut, det vill säga mängder med n rutter, se ekvation 2. Anledningen till detta är att mängder med färre än n rutter är delmängder av de maximala mängderna. Ett exempel illustreras i figur 4. För en fyrvägskorsning blir det 495 mängder enligt ekvation 2.

$$\binom{n*(n-1)}{n} \tag{2}$$



Figur 4: Exempel på delmängder av en mängd med fyra aktiva rutter. Där 4a är en delmängd av 4b, som är en delmängd av 4c, som i sin tur är en delmängd av den maximala mängden i 4d.

3.2 Algoritm för konfliktfria mängder

För att bestämma de rutter och mängder som är konfliktfria måste alla rutter i varje mängd jämföras mot varandra. Rutter som startar i samma punkt kan använda samma block i korsningen och anses vara konfliktfria. För rutter som inte startar i samma position måste de block som upptas av respektive rutt jämföras med varandra.

Algoritm 1 (s. 9) är ett förslag på en lösning för att bestämma samtliga konfliktfria mängder utifrån en given mängd. Algoritmen fungerar även för trevägs-korsningar och korsningar med fler in- och/eller utfarter per tillfart, så länge det har definierats tillåtna körriktningar i förväg. Exempelvis när vänster körfält endast tillåts göra en vänstersväng.

Algorithm 1 Finding non-conflicting sets

Variables:
$sets \leftarrow$ every set of possible paths
$nonConflictingSets \leftarrow$ empty set

Interfaces:
$containsConflictingBlocks(p1, p2)$: checks if two paths occupy the same block
$getFrom()$: get the starting position of a path

```
for set in sets do
    conflict = false
    outerLoop:
    for p1 in set do
        for p2 in set do
            if p1 == p2 or p1.getFrom() == p2.getFrom() then
                continue
            if containsConflictingBlocks(p1, p2) then
                conflict = true
                break outerLoop
    if conflict == false then
        add set to nonConflictingSets
return nonConflictingSets
```

3.3 Fundament för infrastrukturbaserad kommunikation

Idén för ett virtuellt trafikljus, som sköter koordination av autonoma fordon vid en korsning, är att en server kontinuerligt gör beräkningar för att skapa tillståndsvariabler för fordonen. Dessa tillståndsvariabler innehåller information om vilken konfliktfri mängd som ska vara aktiv, hur länge den ska vara aktiv samt en acceptansflagga. Om samtliga fordon i korsningen accepterat tillståndet, sätts flaggan av servern. Tillståndet som beskrivs kallas för en *fas*, se ekvation 3.

$$\texttt{phase} = (\texttt{set} : \mathit{int},\ \texttt{till} : \mathit{time}\ \mathit{or}\ \bot,\ \texttt{agreement} : \mathit{boolean}) \tag{3}$$

En samling av tre faser är det som skapar ett *schema*, se ekvation 4. Dessa tre faser är *nu*, *nästa* och *preliminär*. För att skapa säkerhet i systemet är det bara den preliminära fasens acceptansflagga som kan ändras. På så sätt riskeras det inte att flaggan hos nästkommande fas ändras i sista sekund och att oenighet skulle råda bland fordonen i korsningen.

$$\texttt{schedule} = (\texttt{now} : \mathit{phase},\ \texttt{next} : \mathit{phase},\ \texttt{tentative} : \mathit{phase}) \tag{4}$$

För schema och fas finns det grundtillstånd, se ekvation 5. Dessa används vid fall då kommunikation till servern bryts eller för ett tillstånd där samtliga fordon inte har accepterat tillståndet i den preliminära fasen. När något avvikande inträffar återgås det till detta grundtillstånd som är samma för samtliga fordon. Grundtillståndet gör systemet mer robust och säkerställer att konflikter inte sker vid förekomster av störningar.

$$\begin{aligned} &\texttt{dfltPhase} = (\texttt{0},\ \bot,\ \texttt{False}) \\ &\texttt{dfltSchedule} = (\texttt{dfltPhase},\ \texttt{dfltPhase},\ \texttt{dfltPhase}) \end{aligned} \tag{5}$$

3.4 Styrningsalgoritmer för aktörer i korsning

Det virtuella trafikljuset innefattar en medlemservice som meddelar fordonen om vilken server de ska ansluta sig till. Servern använder sig av två algoritmer för att koordinera fordonen. Fordonen i sin tur använder sig av en algoritm för att kommunicera med servern.

3.4.1 Styrningsalgoritm för server

Servern ska meddela samtliga fordon om rätt schema och lyssna på svar från fordonen och säkerställa att konsensus råder. Ett förslag på lösning kan ses i algoritm 2 (s. 11). Servern får reda på vilka fordon som är i närheten av korsningen via medlemskapet och kan då skicka ut schemat för de tre kommande faserna till de berörda fordonen. Utskicket görs kontinuerligt med ett bestämt intervall för att försäkra att meddelandena kommer fram. När servern sedan får svar från samtliga fordon i korsningen, har fordonen mottagit och accepterat det kommande schemat och har därmed samma ruttmängd. Acceptansflaggan kan då sättas till sann för preliminärfasen, annars förblir flaggan flask. När en fas med en falsk acceptansflagga blir aktuell, bör samtliga fordon agera med försiktighet. Ett exempel på en försiktighetsåtgärd är att stanna.

När nuvarande fas går ut, alltså att dess giltighetstid är mindre än den aktuella tiden, skiftas faserna ett steg. Nuvarande fas ersätts av nästkommande fas och den nästkommande ersätts av den preliminära. En ny preliminär fas måste därför väljas för att kunna kommuniceras ut till fordonen i korsningen.

Algorithm 2 En algoritm för servern

Constants:
T_{period}: the invocation period of the do forever loop.
$dfltPhs = (0, \perp, False)$
$dfltSch = (dfltPhs, dfltPhs, dfltPhs)$

Structures:
$phase$ = (set: int, $till$: time or $\perp$, $agreement$: Boolean) : a phase of a schedule (planned set, expiration time and consensus predicate)
$schedule = (now : phase, next : phase, tentative : phase)$: current schedule state structure

Interfaces:
clock(): get current time
getFuturePhase(state): returns a $phase$
getAgentsIDs(): returns agentID's from the membership service

Variables:
state[n] = [dfltSch, ..., dfltSch]: schedule of all agents in **getAgentsIDs()**, where state[i] is the schedule of coordinator p_i

$StateHandler$:
loop {once in every T_{period} time units}
  if $\neg state[i].(tentative.till \geq next.till \geq now.till)$ **then**
    $state[i] \leftarrow dfltSch$
  if $state[i].now.till = \perp \vee$state[i].now.till $<$ clock() **then**
    $state[i] \leftarrow state[i].(next, tentative, getFuturePhase(state[i]))$
  state[i].tentative.agreement $\leftarrow$ (state[i].tentative.till $\neq \perp$) $\wedge$ $\bigwedge_{k \neq i}$ state[k].tentative.(set, $till$) = $state[i].tentative.(set, till)$
  for $p_k \in$ getAgentIDs( )$\setminus \{p_i\}$ **do**
    send $\langle$state[i]$\rangle$ to p_k

$MessageHandler$:
 upon message $\langle m \rangle$ received from p_j **do** state $[j] \leftarrow m$

3.4.2 Algoritm för preliminär fas

Preliminärfasen väljs bland de konfliktfria mängderna på ett sådant sätt att maximalt antal fordon kan köra samtidigt genom korsningen. Algoritm 3 (s. 12) som används för detta är en *Greedy best-first search*-algoritm [30]. Den utgår endast från det nuvarande lokalt logiska valet och på så sätt bortser från andra val som globalt skulle kunna vara mer effektiva.

För att säkerställa att fordon som fått tillträde till korsningen ska hinna köra igenom, läggs dess id på en global lista så att det i nästa iteration kan kontrollera att dess rutt finns med i den aktiva mängden. När fordonet sedan lämnar korsningen meddelas servern som tar bort dess id från listan och en mängd utan dess rutt kan potentiellt väljas.

Algorithm 3 En algoritm för att bestämma nästa aktiva mängd

Constants:
nonConflictingSets : contains the non-conflicting sets for the intersection.
agentStates : contains the state of every agent in the intersection.

Interfaces:
getPath(agent): get the path of given agent
subsetOf(A, B): checks if A is a subset of B

Variables:
nrOfOccurences : global variable to help choose optimal *nonConflictingSet*
index : used to point at a *nonConflictingSets*
futurePrioTemp : temporary set consisting of agents with active routes

```
for set in nonConflictingSets do
    n = 0
    agentsInSet = empty set
    for agent in agentStates do
        if getPath(agent) = ⊥ then
            futurePrio.remove(agent)
        if getPath(agent) in set then
            agentsInSet.add(agent)
            n = n + 1
    if n > nrOfOccurences and (subsetOf(futurePrio, agentsInSet) or subsetOf(agentsInSet,
    futurePrio) then
        nrOfOccurences = n
        index = i
        futurePrioTemp = agentsInSet
if futurePrio = futurePrioTemp then
    return prevIndex
else
    futurePrio = futurePrioTemp
    prevIndex = index
    return index
```

3.4.3 Styrningsalgoritm för fordon

Fordonens uppgifter är att lyssna efter meddelanden från servern, kontrollera att den nuvarande fasen fortfarande är aktuell och skicka tillbaka schemat till servern. Ett förslag på en lösning kan ses i algoritm 4 (s. 13). Meddelanden som fordonet mottager från servern innehåller ett schema som efter mottagandet läggs över till fordonets lokala schema. Detta lokala schema använder sedan fordonet i sina beslut när det befinner sig i närheten av korsningen. Fordonen byter fas på samma sätt som servern med skillnaden att den preliminära fasen väljs till grundfasen (*dfltPhase*) tills ny preliminär fas mottagits av servern. Efter kontrollen av nuvarande fas, skickas fordonets tillstånd till servern som sedan kontrollerar samtliga fordon.

Algorithm 4 En algoritm för fordon (klienter)

1: **Constants:**
2: T_{period}: the invocation period of the do forever loop.
3: $dfltPhs = (0, \perp, False)$
4: $dfltSch = (dfltPhs, dfltPhs, dfltPhs)$

5: **Structures:**
6: $phase$ = (set: int, $till$: time or $\perp$, $agreement$: Boolean) : a phase of a schedule (planned set, expiration time and consensus predicate)
7: $schedule = (now : phase, next : phase, tentative : phase)$: current schedule state structure

8: **Interfaces:**
9: **clock()**: get current time

10: **Variables:**
11: localState = [dfltSch]: schedule of the client
12: localOutput: (phase, phase) = (dfltPhs, dfltPhs)

13: $StateHandler$:
14: **loop** {once in every T_{period} time units}
15: **if** clock() > $(localState.now.till + T_{period})$ **then**
16: $localState \leftarrow localState.(next, tentative,$ dfltPhs)
17: **write** $(localState.now, localState.next)$ **to** $localOutput$
18: **send** ⟨state⟩ **to** p_i the coordinator

19: $MessageHandler$:
20: **upon** message ⟨m⟩ **received from** p_i the coordinator **do** localState $\leftarrow m$

4 Systemarkitektur

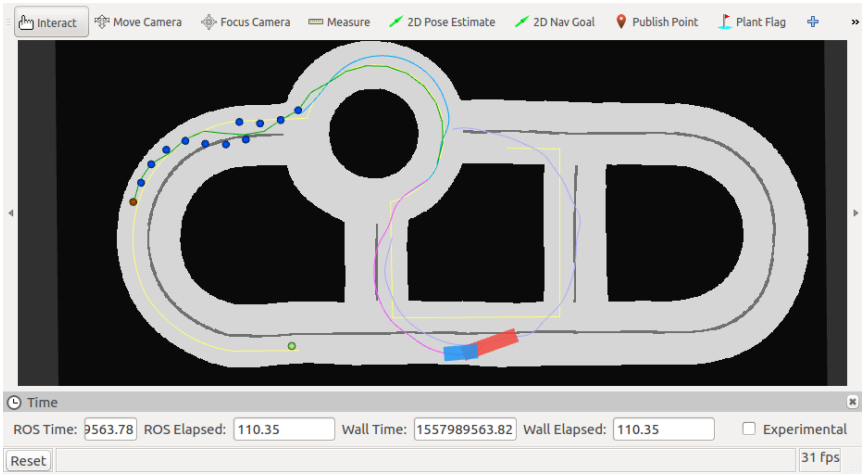
Projektet består av en fysisk evalueringsmiljö, en nätverkssimulator och två stycken korsningssimulatorer. Det finns en korsningssimulator för V2V och en för V2I.

4.1 Fysisk evalueringsmiljö

Laborationslokalen har en fysisk evalueringsmiljö som visas i figur 5a, samt två stycken modellastbilar med trailer som visas i figur 5b. Miljön är anpassad till modellastbilarna och består av tre korsningar, en rondell och två snäva svängar. Ett styrningsprogram för miljön finns även tillgängligt, som visas i figur 5c.

(a) Trafikmiljö

(b) Modellastbil



(c) RViz visualisering

Figur 5: Den fysiska evalueringsmiljön och dess styrningsprogram.

Modellastbilarna som använts är Tamiya Volvo FH med tillhörande trailer. Den styrs av en monterad Raspberry Pi 3 som kör ett Linux-baserat operativsystem. Raspberry Pi 3 är en enkortsdator som i vår miljö används för att kommunicera med andra datorer i nätverket. Enhetsdatorn tar emot meddelanden och skickar styrsignaler till lastbilen.

De programvaror som används i den fysiska evalueringsmiljön är *ROS*, *RViz* samt *GulliView*. ROS är ett robotbaserat operativsystem som sköter kommunikationen mellan de olika delarna i systemet. ROS består av *noder* och *topics* där noder är systemets olika delar och topics är kanaler som noder skickar meddelanden igenom. Noder kan publicera meddelanden och prenumerera på topics. När en nod publicerar ett meddelande på ett topic skickas meddelandet till alla noder som prenumererar på detta topic. En fördel med ROS är att noderna kan köras på olika system vilket möjliggör att enkortsdatorn på lastbilen kan kommunicera med andra enheter på nätverket.

RViz är ett tredimensionellt visualiseringsverktyg för ROS och har anpassats till detta projekt för att visa ett tvådimensionellt perspektiv av den fysiska evalueringsmiljön ovanifrån, vilket kan ses i figur 5c (s.14). Fordonen efterliknas på vägbanan och förflyttar sig genom att följa en planerad rutt. Detta möjliggörs då varje fordon publicerar sin personliga position- och riktningsdata på ett ROS-topic. RViz läser därefter av detta topic för att på så sätt rendera ut fordonet på korrekt position.

För att lokalisera fordonen i den fysiska evalueringsmiljön används GulliView, som är ett visuellt lokaliseringssystem utvecklat av Arkheden, Zaragatzky, Lindhé och Gustafsson [31]. GulliView använder sig av takmonterade kameror för att lokalisera objekt genom att söka efter så kallade "april-taggar", vilka är monterade på fordonen. Gulliview används eftersom GPS inte är tillräckligt exakt för våra applikationer. GPS kan dock vara ett bra alternativ för implementation på riktiga fordon.

4.2 Korsningssimulator

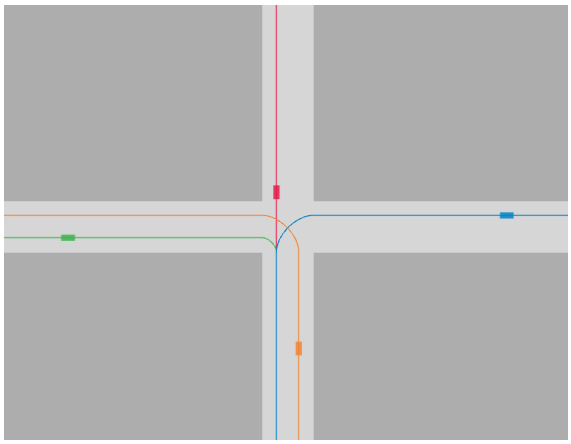
Korsningssimulatorn simulerar en virtuell fyrvägskorsning där ett fordon kan köra från varje riktning, vilket på så sätt tillåter att upp till totalt fyra fordon kan simuleras samtidigt. Simulatorn tillåter användaren att individuellt bestämma varje fordons planerade rutt samt startposition. För att visualisera korsningssimulatorn i realtid används även här ROS-verktyget RViz, se figur 6 (s.16). Beroende på vilket protokoll som körs, kan fordonet antingen kommunicera med ett infrastrukturbaserat virtuellt trafikljus (V2I) eller med de andra fordonen i korsningen (V2V).

4.2.1 Korsningssimulatorn med V2V protokoll

Detta protokoll låter fordonen som befinner sig i korsningen gemensamt bestämma sinsemellan om vem som ska ha företräde. Genom att fordonen publicerar och prenumererar på varandras ROS-topics vet alla fordon var de andra fordonen befinner sig och vart de planerar att åka.

Utöver att fordonen delar information med varandra, kör de även individuellt en modifierad riskestimator-algoritm, som bygger på en lösning av Lefèvre et al [22]. Utifrån publicerad data från samtliga fordon, beräknar och bedömmer riskestimatorn om fordonets planerade rutt kan genomföras utan kollision. Hur säker en planerad rutt exakt är, översätter riskestimatorn till ett numeriskt värde där ett högre värde betecknar en högre risk för kollision.

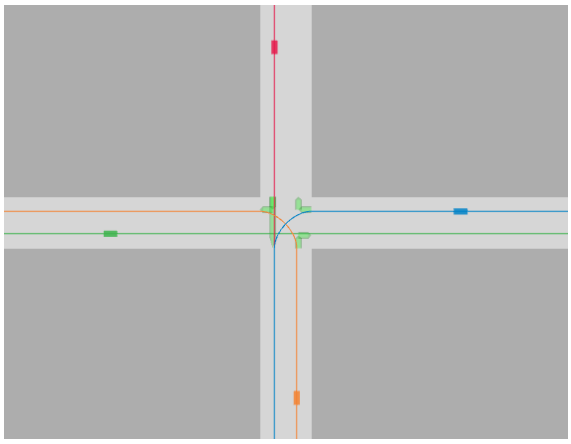
Fordonen kör även en manöverförhandlar-algoritm. Denna algoritm möjliggör för fordonen i korsningen att komma överens och enas om vilket fordon som ska köra först. När ett fordon närmar sig korsningen måste den först be om tillåtelse från de närvarande fordonen innan den kan genomföra sin manöver.



Figur 6: Simulering av V2V-kommunikation vid en korsning.

4.2.2 Korsningssimulatorn med V2I protokoll

Denna korsningssimulatorn bygger endast på infrastrukturbaserad kommunikation med styralgoritmerna från kapitel 3. Varje fordon ansluter till den implementerade koordinatorservern för att utbyta scheman och på så sätt få reda på vilken konfliktfri mängd som gäller i korsningen. När fordonen skickar bekräftelsen att de mottagit ett schema från servern skickar de även med sin planerade rutt. Detta möjliggjorde att servern kunde veta alla fordons planerade rutt för att beräkna nästa aktiva konfliktfria mängd. I simulatorn visualiseras den aktiva konfliktfria mängden med gröna pilar i korsningen, vilket illustreras i figur 7. Detta för att kunna observera vilka fordon som får köra.



Figur 7: Modifierad version för att simulera V2I-kommunikation med konfliktfria mängder.

4.3 Nätverkssimulator

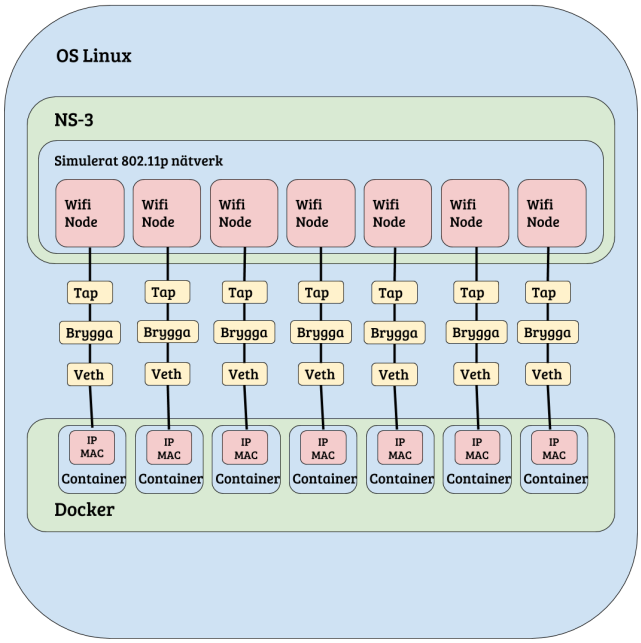
Nätverksimulatorn har implementerats för att testa applikationer och systems feltolerans vid förekomst av kommunikationsproblem. Programvaror som används till nätverkssimulatorn är Docker, NS-3 samt virtuella nätverksverktyg som finns tillgängliga i Linux. Dessa tre programvaror är huvudkomponenterna för nätverkssimuatorn.

Docker används för att lätt kunna skapa instanser av applikationer som sedan lätt kan köras och flyttas mellan system. Detta görs genom en *Docker Image*, vilket essentiellt är en fil uppbyggd av flertalet lager för att kunna exekvera kod. Denna image, eller avbildning, byggs genom instruktioner för att kapsla in en eller flera applikationer. När en avbildning sedan körs, skapas en *container*-instans av den avbildningen som då lätt kan startas, stoppas, flyttas eller tas bort.

Docker-containrar fungerar likt en virtuell maskin. Skillnaden är att en container inte har ett eget operativsystem och behöver därför inte simulera egen hårdvara och drivrutiner. Istället använder den samma operativsystemskärna som värddatorn, vilket gör att Docker är resurssnålt och tar upp lite utrymme på värddatorn. En annan positiv aspekt är att en avbildning byggd i Linux går att köra på en dator med exempelvis operativsystemet Windows. Trots att flera Docker-containrar använder samma operativssystemkärna är de som standard helt isolerade ifrån varandra.

Programvaran NS-3 är en plattform för att bygga verklighetstrogna nätverkssimulatorer. De kan användas för att simulera diskreta nätverk med vanligt förekommande kommunikationsproblem, som exempelvis nätverksfördröjning eller uteslutna paket. Detta möjliggör testning av hur ett system skulle reagera i en specifik nätverksmiljö. Flertalet modeller av verkliga nätverksprotokoll och enheter finns tillgängliga som moduler i NS-3, vilket gör den mycket avancerad och mångsidig.

Det finns tre virtuella nätverksverktyg som är centrala i simuleringsmiljön: nätverkstappar, -bryggor och -kablar. En nätverkstap är ett virtuellt nätverksgränssnitt som kan ta emot och skicka datapaket med hjälp av IP-adresser [32]. En nätverksbrygga kan koppla ihop två eller flera nätverk. Virtuella nätverkskablar fungerar på samma sätt som fysiska kablar.



Figur 8: Nätverkssimulatorns uppbyggnad

Uppbyggnaden av nätverkssimulatorn visas i figur 8 (s. 17). I vårt projekt simulerar NS-3 ett Wi-Fi 802.11p nätverk vilket är ett godkänt protokoll i IEEE 802.11 standarden för trådlösa nätverk, speciellt uttaget för trådlös fordonskommunikation [33]. Nätverket simuleras med inställningar för realistiska nätverkstillstånd, som exempelvis kommunikationsfördröjning och paketförluster. Eftersom det är ett trådlöst nätverk, går det att ställa in vilket frekvensband som ska användas, likväl bandbredd och datahastighet [34]. NS-3 skapar noder som tilldelas var sin IP-adress och blir placerade i en tvådimensionell miljö med fasta avstånd mellan varandra.

För att Docker-containrarna ska kunna kommunicera mellan varandra över NS-3 nätverket måste containrarna länkas till noder i NS-3. Varje nod ansluts till en nätverkstap som i sin tur ansluts till en nätverksbrygga. En virtuell nätverkskabel benämns som ”Veth”i figur 8 (s. 17) och används sedan för att koppla ihop containern med nätverksbryggan. Containern kommer då tilldelas en slumpmässig MAC-adress med motsvarande IP-adress.

5 Resultat

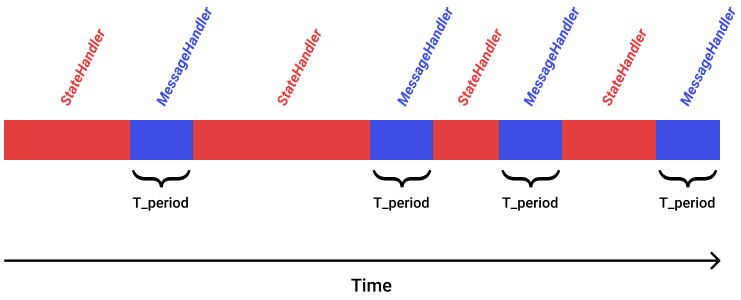
I detta kapitel presenteras resultatet från implementeringen av V2I-protokollet och dess resultat från nätverkssimulatorn. Ett par forskningsfrågor har legat som grund till dessa resultat. Vilken nivå av skalbarhet är möjlig att uppnå för systemet med begränsade datorresurser? Är det möjligt att ha ett adaptivt system som anpassar sig efter trafiken? Är det möjligt att ha ett robust system vid förekomsten av kommunikationsfel? För att besvara dessa frågor har ett antal experiment och tester utförts. Utöver detta presenteras resultatet av implementering av V2V-protokollet i den fysiska evalueringsmiljön.

5.1 V2I-protokollet

Vi önskar i detta stycke att undersöka om det är möjligt att ha ett skalbart och prisförmånligt system för koordination av autonoma fordon, samt undersöka dess kapacitet. Algoritmen är lätt att sätta sig in i och begripligt för människor med begränsade programmeringskunskaper. Den ska även kunna drivas av begränsad datorkapacitet.

5.1.1 Implementation av koordinatoralgoritm

En implementation av algoritmen för servern (algoritm 2 s. 11) utvecklades i Python 3.7. Algoritmen består av två delar, *StateHandler* och *MessageHandler*, som behöver köras parallellt och oberoende av varandra. Ett mål för implementationen var att undvika flera processortrådar för att kunna köras med begränsad datorkraft.



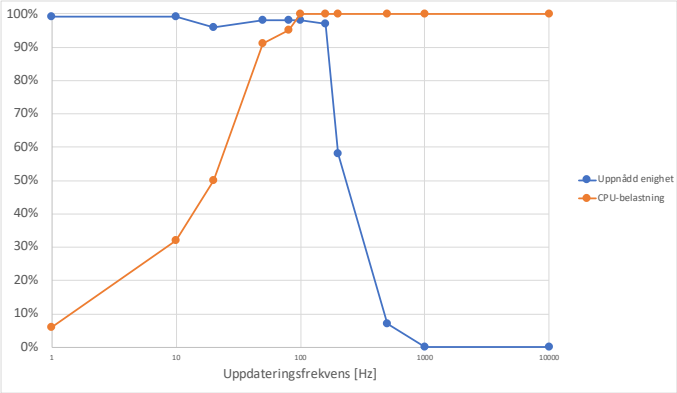
Figur 9: En illustration av hur den asynkrona implementationen exekverar med tiden. När StateHandler vilar T_{period} sekunder, exekveras MessageHandler.

Genom att utveckla en asynkron implementation av algoritmen, kan båda delar av koden köras på endast en tråd, samtidigt parallellism kan uppnås. Asynkron implementation fungerar eftersom StateHandler loopen itererar med en viss uppdateringsfrekvens, vilket i sin tur innebär att loopen vilar med ett visst tidsintervall. Eftersom koden är asynkron, kan denna vilotid utnyttjas för att köra MessageHandler loopen tills det är läge för StateHandler att börja exekvera igen, förloppet illustreras i figur 9. Uppdateringsfrekvensen för StateHandler loopen påverkar möjligheten att ta emot meddelanden i MessageHandler. Då uppdateringsfrekvensen är låg ges mer tid för MessageHandler att köra, vilket betyder att fler meddelanden kan tas emot. Det behövs alltså en balans mellan att kunna köra StateHandler loopen så ofta som möjligt, samtidigt som det går att ta emot en stor mängd meddelanden.

5.1.2 Prestanda och skalbarhetstester

Serverimplementationen testades på en Raspberry Pi 3 B som har en 1.2GHz fyrkärnig processor (Broadcom BCM2837 64bit) samt 1GB primärminne. Denna mikrodator valdes för servern på grund av dess kompakthet, pris, tillgänglighet samt renommé. Testerna nedan kördes på ett lokalt nätverk med klienter anslutna genom Wi-Fi, för att simulera en verklig situation med trådlös kommunikation.

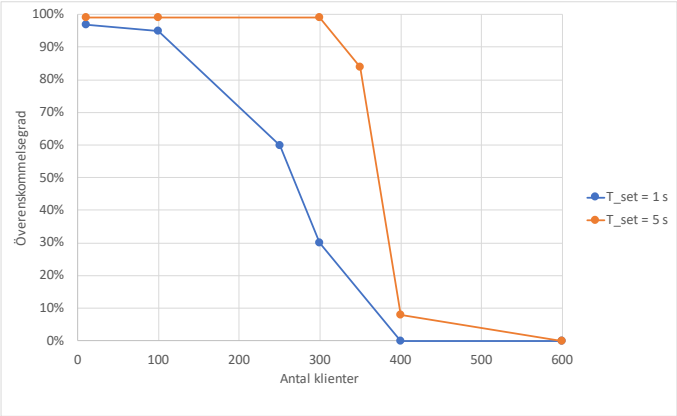
Ett första experiment gjordes för att undersöka serverns uppdateringsfrekvens. Det antogs att en korsning inte innehåller mer än 100 uppkopplade fordon samtidigt. Därför fixerades antalet klienter till 100. Tiden (T_{set}) för en mängd fixerades till 5 sekunder. Detta ansågs vara en rimlig tid tills att ett fasskifte sker, då det ger tillräckligt med tid för klienter att acceptera scheman. Tiden gav oss även möjligheten att simulera en god mängd fasskiften under en rimlig tid. Det simulerades 100 fasskiften vilket innebär att varje test tog ungefär 8 minuter att genomföra.



Figur 10: Processorbelastning och överenskommelser vid olika uppdateringsfrekvenser för servern med 100 klienter. $T_{set} = 5s$

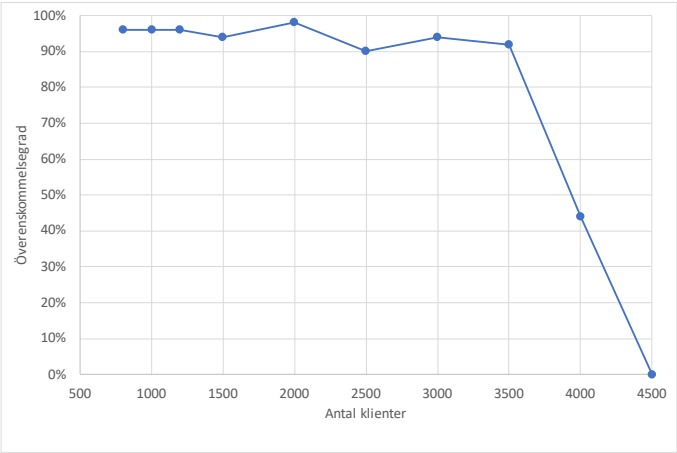
I figur 10 visas att när processorbelastningen når 100% minskar serverns förmåga att hantera meddelanden och andelen överenskommelser minskar. Alla meddelanden hinner inte hanteras av servern när denna flaskhals uppstår. Samtidigt finns möjlighet att uppnå överenskommelse, trots det att servern är överbelastad. Genom att variera variablerna fastiden T_{set} och uppdateringstiden T_{period} går det att utvärdera serverns förmåga att hantera meddelanden och antal klienter.

Kapaciteten av servern evaluerades genom att ansluta olika mängder klienter och se hur servern hanterar en växande mängd trafik. Två olika scenarion undersöktes, när fastiden var 1 sekund samt 5 sekunder, då uppdateringsfrekvens var 10 Hz.



Figur 11: Procent uppnådd grad av överenskommelse vid olika mängder klienter anslutna till servern, med varierande fastid T_{set} och samma uppdateringstid T_{period}

Från resultatet som presenteras i figur 11 framgår det att servern lyckades hantera upp till 300 klienter med $T_{set} = 5$ sekunder. Med $T_{set} = 1$ sekund går det att se att den lyckas hantera upp till 100 klienter utan väsentligt minskad förmåga att uppnå överenskommelse. Processorbelastningen kan antas vara samma för både $T_{set} = 5$ och $T_{set} = 1$, då uppdateringsfrekvensen är lika. Det vi kan se i graferna i figur 11 är att en längre T_{set}-tid ger större chans att lyckas enas om en mängd, trots att alla meddelanden inte alltid hinner hanteras och processorbelastningen är 100%.



Figur 12: Serverkapacitet och uppnådd grad överenskommelse. Periodtid $T_{set} = 30$ sekunder och uppdateringstid $T_{period} = 2$ sekunder.

Ökad T_{period} ger servern mer tid att hantera meddelanden och ökad T_{set}-tid ger fler möjligheter att enas om en mängd. I figur 12 visas att servern lyckas hantera 3500 klienter anslutna med T_{period} på 2 sekunder och T_{set} på 30 sekunder.

5.1.3 Adaptivt trafiksystem

Det fundamentala i algoritm 2 som möjliggör ett adaptivt system, är en funktion som bestämmer nästa konfliktfria mängd som ska gälla. För att bestämma nästa mängd användes algoritm 3.

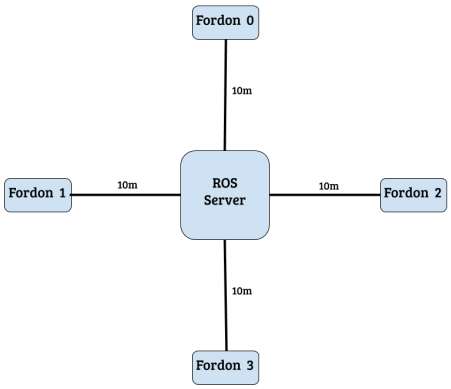
För detta behövdes information om fordons intentioner för att avgöra vilken mängd som ger bästa trafikflödet. Klienternas schema behövde därför modifieras för att kunna skicka med deras intentioner till servern.

I korsningssimulatorn för V2I-kommunikation skickades fordonens rutt till servern. När fordon lämnat korsningen, skickades istället ett betydelselöst värde till servern för att påvisa att denne lämnat korsningen. Detta möjliggjorde att servern kunde veta alla fordons planerade rutt för att matcha bästa konfliktfria mängd för gällande trafiksituation. Efter empiriska observationer i korsningssimulatorn, har det studerats att systemet anpassar sig efter trafikflödet. Aktiv mängd växlar allt eftersom fordon kört genom korsningen, och resultatet av serverns beräkningar för optimal mängd ändras.

5.2 Nätverkssimulering

Nätverksimulatorn består av sju stycken noder: ROS, ZooKeeper, servern samt fyra stycken bilnoder. ROS-noden kör både ROS-mastern och visualiseringen, där ROS-mastern är en av fordonsnoderna som resten av fordonsnoderna ansluter sig till.

Kommunikationsfelen bestäms då NS-3 startas samtidigt som noderna placeras ut i ett 2D-nät. Nodernas placering visas i figur 13. Nivån på kommunikationsfelen går att ställa in i NS-3 med hjälp av tillgängliga moduler för kommunikationsfel. Modulen vi fokuserar på möjliggör att ange den mängd paketförlust som respektive nod i det trådlösa nätverket ska ha. Det betyder att en viss procent data som skickas till noden förkastas och når inte mottagaren. På de centraliserade noderna appliceras varken kommunikationsfördröjningar eller paketförluster. Samtliga paketförluster uppstår istället när fordonsnoderna ska ta emot meddelanden från andra noder. Det sker alltså inga kommunikationsfel när fordonen skickar meddelanden till de centrala noderna, eller när de centrala noderna skickar meddelanden mellan varandra.



Figur 13: Nodernas placering i det simulerade nätverket

För att kontrollera trafiksystemets feltolerans utfördes ett antal olika tester som testar realistiska scenarion. Vetskapen om hur fordon påverkas av kommunikationsfel, mer specifikt paketförluster i vårt fall, vid meddelandesändning är väsentligt, då systemet bygger på trådlös kommunikation. Genom att applicera varierande paketförlust på fordonen, går det att på så vis få en uppfattning om hur pass robust och feltolerant systemet är mot att kunna hantera kommunikationsfel.

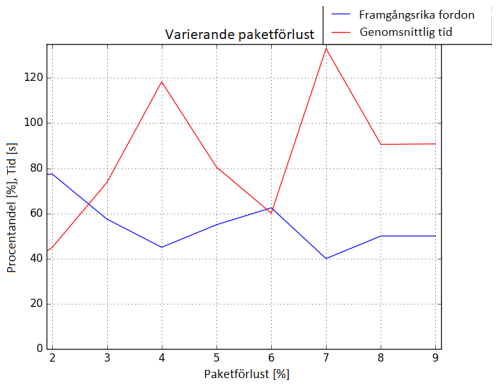
Evalueringen gjordes genom att köra varje testsimulering 10 gånger och sedan ta medelvärdet av körningarna. Vid varje körning gavs varje fordon ett slumpmässigt avstånd från korsningen och

en slumpmässig rutt genom korsningen. Vid varje evaluering ändrades endast en variabel i taget, exempelvis andel paketförluster eller antalet fordon som ska appliceras med paketförlust. Varje körning gjordes på förbestämd tid (90 sekunder) och inleddes med att initiera fordonen, vilket tog 30 sekunder att göra. Alltså återstod 60 sekunder för fordonen att passera korsningen.

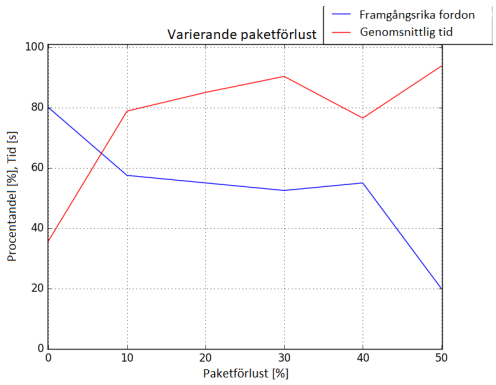
5.2.1 Systemets påverkan av paketförluster

För att avgöra huruvida paketförluster påverkar det virtuella trafikljuset, evaluerades tre olika scenarion. I scenario ett ökar procenten paketförluster successivt från 2-9% över alla bilar samtidigt. I det andra scenariot ökar procenten paketförluster successivt från 0-50% över alla bilar samtidigt. I scenario tre ökar antalet fordon med paketförlust från noll till fyra stycken, med en fixerad paketförlust på 8%. De tre testerna evaluerades med NS-3 inställt att använda standardbandbredden 6 Mbps på det trådlösa nätverket.

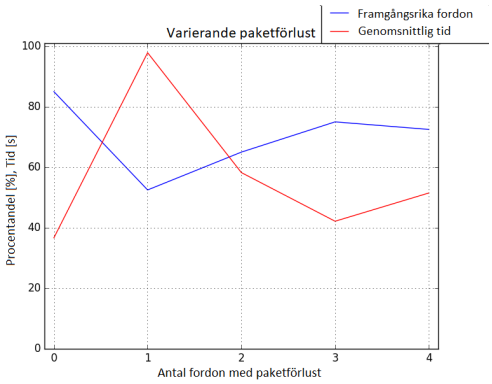
I figur 14a och 14b går det att läsa av hur systemet beter sig vid varierande mängd paketförluster. I figur 14c visas hur systemet beter sig då antalet fordon med paketförlust varieras. Den blå grafen visar hur många fordon som tar sig genom korsningen under körningen. Den röda grafen visar den genomsnittliga tiden det tar för fordonen att ta sig genom korsningen. Fordon som inte lyckas slutföra sin rutt ges maxtiden 60 sekunder.



(a) Varierande paketförlust mellan 2-9%.



(b) Varierande paketförlust mellan 0-50%.



(c) Varierande antal fordon med paketförlust.

Figur 14: Andel fordon som tar sig genom korsningen (blå linje), samt dess genomsnittliga tid att slutföra rutten (röd linje).

I figur 14a ser vi hur antalet fordon som tar sig genom korsningen framgångsrikt minskar, då andelen paketförluster ökar. Generellt i detta resultat går det att åskåda en nedåtgående trend på

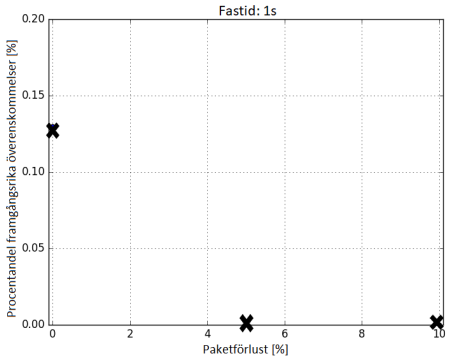
antalet framgångsrika fordon då andelen paketförluster ökar. Likväl kan man tyda att fordonens genomsnittliga tid att ta sig genom korsningen ökar.

Precis som i figur 14a (s. 23) ser vi hur antalet fordon som tar sig genom korsningen framgångsrikt minskar i figur 14b (s. 23), då mängden paketförlust ökar. Skillnaden från föregående test är mängden paketförlust, som istället uppgår till 50%. En liknande trend går att åskåda här. Det sker en nedgång i andelen framgångsrika fordon. Då andelen paketförlust ligger mellan 10 till 40% är resultatet mer eller mindre oförändrad för att sedan vid 50% sjunka drastiskt.

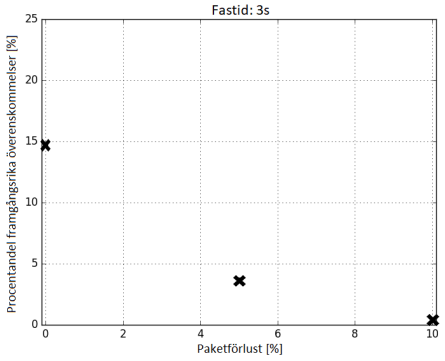
I figur 14c (s. 23) ser vi att antalet framgångsrika fordon är som högst då inga paketförluster sker. Generellt sett kan en negativ trend i andelen framgångsrika fordon skådas, när antalet fordon med inviduella paketförluster introduceras.

5.2.2 Systemets påverkan av serverns uppdateringstid

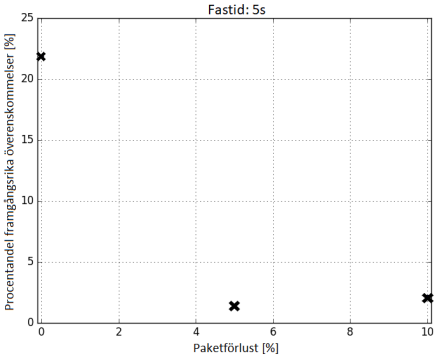
För att avgöra huruvida systemet påverkas av hur snabbt servern byter mängd, evaluerades korsningssimulatorn med olika fastider (T_{set}) som då avgör hur ofta servern ska byta mängd. Under evalueringen hade samtliga fordon samma andel paketförlust där andelen ökades i tre steg från 0% till 10%. Servern hade en uppdateringsfrekvens på 10 Hz ($T_{period} = 0.1$ sekunder) under hela testet, medan fastiden evaluerades vid 1, 3 samt 5 sekunder. NS-3 var inställd på att använda standardbandbredden 6 Mbps på det trådlösa nätverket.



(a) Fastid 1s



(b) Fastid 3s



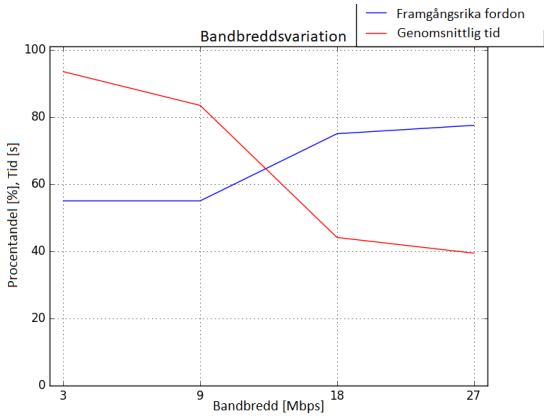
(c) Fastid 5s

Figur 15: Överenskommelsegrad vid olika fastider för systemet

Figurer 15a, 15b och 15c (s. 24) visar graden av överenskommelse mellan fordonen. När fastiden är 1 sekund, blir det ett fåtal överenskommelser vid avsaknad av paketförluster. Vid förekomst av paketförluster, blir det dock inga överenskommelser över huvudtaget. Graden av överenskommelse ökar däremot då fastiden ökat till 3 sekunder. Vi ser att det även sker överenskommelser när det förekommer paketförluster. Ökar fastiden ytterligare till 5 sekunder, ökar också antalet överenskommelser när det inte förekommer några paketförluster. Vid förekomst av paketförluster, är antalet överenskommelser likartade då fastiden är 3 och 5 sekunder.

5.2.3 Bandbreddens påverkan på systemet

För att se huruvida bandbredd har någon påverkan på det trådlösa nätverket, gjordes ett test med att successivt öka den tillgängliga bandbredden mellan samtliga noder. Detta gjordes genom att konfigurera om en modul för bandbredd som finns tillgänglig i NS-3. Genom att göra detta, går det teoretiskt sätt att skicka och ta emot mer data samtidigt mellan respektive noder. Testet kördes med en fixerad mängd paketförlust på 8% på samtliga fordon. Serverns uppdateringsfrekvens var 10 Hertz ($T_{period} = 0.1$ sekunder) och fastiden 5 sekunder.



Figur 16: Varierande bandbredd på det trådlösa nätverket

Grafen i figur 16 är uppbyggda på samma sätt som i figur 14 (s. 23). Vi ser en tydlig ökning av andelen fordon som tar sig genom korsningen framgångsrikt då bandbredden ökar från 9 Mbps till 18 Mbps. Mellan 3 Mbps och 9 Mbps är andelen framgångsrika fordon konstant med resultatet från figur 14a (s. 23) vid 8% paketfölust. Då bandbredden sedan ökar och är mellan 18 Mbps och 27 Mbps, kan det skådas liknande nivåer av andelen framgångsrika fordon som när nivån av paketförlust var 0-2% i figur 14a (s. 23). Vidare kan det tydas att även den genomsnittliga tiden det tar för ett framgångsrikt fordon är relativt hög mellan 3 Mbps och 18 Mbps, för att sedan stadigt sjunka vid 18 Mbps och över.

5.3 Tillämpning av V2V-protokoll i fysisk evalueringsmiljö

V2V-protokollet, utan manöverförhandlare portades först över till den fysiska evalueringsmiljön. Detta betyder att lastbilarna bara stannar om de anser att risken är för stor. För att evaluera riskestimatorns funktionalitet i evalueringsmiljön, planlades två stycken lastbilar med korsande rutter med avsikt att orsaka kollision. När lastbilarna närmade sig korsningen stannade lastbil 1 helt, varvid lastbil 2 fortsatte köra utan större uppehåll. När lastbil 2 tagit sig genom korsningen återupptog lastbil 1 sin rutt genom korsningen.

Vid liknande experiment planlades två icke-korsande rutter, där kollision inte kunde inträffa, detta för att testa att riskestimatorn inte bedömmer en ofarlig trafiksituation som riskfylld. Då lastbilarna närmade sig varandra, observerades det en minimal hastighetsminskning från båda lastbilarna, men de lyckades fortfarande ta sig genom korsningen. Under särskilda förhållanden då lastbilarna tog ut specifika svängar mer, och därmed inkräktade på motgående körbara, kunde kollision dock inträffa.

Efter implementering av manöverförhandlaren stannade lastbilarna med större säkerhet. Vägregler kan även implementeras för att ge visa färdriktningar prioritet för att efterlikna trafikregler som huvudled.

6 Diskussion

Under projektets gång har algoritmer för V2I-kommunikation implementerats och testats i en befintlig korsningssimulator. En nätverksimulator har utvecklats för att evaluera systemets feltolerans och ett protokoll för V2V-kommunikation har portats till en fysisk evalueringsmiljö. Projektets resultat diskuteras mer detaljerat i det här kapitlet.

6.1 V2I-protokollet

Projektets implementation av en koordinationsalgoritm valdes på grund av dess enkelhet att förstå samt dess resurssnålhet. Önskemålet att kunna köra algoritmerna på enklare mikroprocessorer med beräkningsbegränsningar, gjorde att en singeltrådad algoritm och simpla kommunikationsprotokoll implementerades.

6.1.1 Systemets skalbarhet

Serveralgoritmen för koordinatorn (algoritm 2) kan implementeras på en rad olika sätt. Ett naturligt val hade varit att utnyttja flera trådar för ökad parallellism och ge möjlighet för servern att utföra uppgifter parallellt. Att simultant kunna hantera mottagna meddelanden, bestämma mängd samt förmedla beslutet till fordon i korsningen, hade varit en önskvärd egenskap. En algoritm med trådar ställer dock hårdvarukrav på systemet och är därför inte lämpat för inbyggda system som har begränsningar på exempelvis antalet processorkärnor. Trådar introducerar overheadkostnader i form av en trådschemaläggare som tar processorkraft. Algoritmen implementerades istället med en asynkron implementation, som ger trådlika egenskaper i form av uppgiftsdelning utan overheadkostnader associerade till trådar. Systemet anses därför vara lämpat för inbyggda system.

Trots att servern och kommunikationsalgoritmen drevs på en Raspberry Pi, kunde upp till 3500 klienter anslutas i våra tester. Offras snabbhet i systemet går det att uppnå en högre kapacitet. Prioriteras istället snabbhet, begränsas kapaciteten för antalet anslutna klienter. Det gick att uppnå höga nivåer av överenskommelse genom att antingen ha uppdateringsfrekvenser på 10 Hertz och upp till 300 klienter, eller 100 Hertz och 100 klienter, se figur 11 (s. 21). Vi bedömer därmed att systemet är högst skalbart och kan användas med begränsade datorresurser.

6.1.2 Adaptiv respons beroende på trafikflöde

Målet med att ha ett adaptivt system, som anpassas beroende på trafikflödet och fordonens önskade färdriktning, lyckades uppnås när servern implementerades med algoritm 3. Genom att fordon skickade med sin planerade rutt i sitt schema till servern, får den samtliga fordons önskade rutter. Utifrån ruttdatan kan servern beräkna den maximala mängden som bör användas för att alltid låta maximalt antal fordon köra. När en bil kört igenom korsningen skickar den svar till servern som då kan göra om beräkningen för den maximala mängden. På så sätt är det virtuella trafikljuset adaptivt utifrån trafikflödet.

6.2 Tillämpning av V2V-protokollet i fysisk evalueringsmiljö

Utan manöverförhandlare gällde inte några regler vad gäller förkörsrätt. Trots detta lyckades lastbilarna ta sig genom korsningen framgångsrikt vid flera tester. Vid fallet då lastbilarna hade korsande rutter lyckades lastbil 1 bedöma risksituationen som allvarlig och hann därför stanna i tid. Eftersom lastbilarna konstant publicerar positions- och hastighetsdata, bedömde lastbil 2 riskläget minimalt då lastbil 1 stannade och fortsatte själv genom korsningen.

Eftersom riskestimatorn tvingas genomföra beräkningar på en förenklad rutt genom korsningen, kan den inte hantera det läge när lastbilen tar ut en sväng och tvingas korsa motgående vägbana. Ett exempel är när lastbil 1 ska svänga höger och får möte från lastbil 2 som planerar att åka rakt fram genom korsningen. Lastbil 1 tvingas ta ut svängen för att kunna ta sig runt svängen, men eftersom riskestimatorn som körs på lastbil 2 fortfarande tror att lastbil 1 är i sin egna körbana, finns det en risk att den inte beräknar tillräckligt hög risk och stannar för att på så sätt undvika kollision.

Manöverförhandlaren löser inte detta problem men den inför trafikregler till systemet vilket ökar robustheten och säkerställer trafikflödet. Med bara riskestimatorn går det inte att avgöra vem som ska få företräde.

6.3 Nätverkssimulering

När vi evaluerade korsningssimulatorn i nätverkssimulatorn använde datorn vi körde det på 100% av sin processorkapacitet. Det kan vara en anledning till varför det blir få överenskommelser mellan fordonen, då datorn helt enkelt inte hinner exekvera alla instruktioner. Något som också kunde ses i tester på servern. En annan anledning kan vara att fordonen i korsningssimulatorn är för långsamma på att svara servern, eftersom det blir betydligt fler överenskommelser när servern evalueras utanför korsningssimulatorn i ett verkligt nätverk.

I vår simulering är alla noder placerade statiskt i ett 2D-nät men i verkligheten kör fordonen runt och avstånden mellan fordon ändras kontinuerligt. Det hade gått att simulera det i NS-3 att noderna i 2D-nätet uppdaterar sin position i takt med att fordonet kör. Vi valde att inte implementera det då det handlar om ytterst små skillnader i kommunikationsfördröjningen, utan fokus lades på paketförlusterna.

I nuläget sker paketförlusterna endast på mottagarsidan. Paketförlusterna kan också ske vid sändarsidan. Skillnaden mellan paketförlust på sändarsidan gentemot mottagarsidan, är ifall en nod ska skicka meddelanden till flera mottagare, påverkas alla mottagare vid en paketförlust på sändarsidan. Om paketförlusten istället sker på mottagarsidan påverkas endast mottagaren av paketförlusten. Paketförlusterna implementerade i nätverkssimulatorn är således asymmetrisk. En mer realistisk nätverksimulator skulle vara att det sker paketförluster både på mottagarsidan och sändarsidan. I vårt fall skulle det leda till ännu färre överenskommelser och därmed hade systemet inte klarat av lika hög mängd paketförluster.

När vi evaluerade hur systemet påverkas av paketförluster, evaluerades tre scenarion: två när andelen paketförlust ökar och ett när antalet fordon med paketförluster ökar. Det är ganska stor skillnad mellan figur 14a och figur 14c (s. 23) då mängden paketförluster är 8% och antalet fordon med paketförluster är 4. Skillnaden kan bero på att nätverksimulatorn körde 90 gånger vid 8% paketförluster i figur 14a (s. 23), medan den endast kördes 50 gånger i figur 14c (s. 23). En annan anledning kan vara att NS-3-konfigurationsfilen måste kompileras om då antalet bilar med paketförluster ändras, men inte när mängden paketförluster ändras. Detta eftersom testet för 14c (s. 23) skriver om en variabel i konfigurationsfilen för att ändra antalet bilar, något som inte sker

för de två första scenarierna.

Eftersom nätverkssimulatorn påverkar kommunikationen mellan Docker-containrar, är nätverkssimulatorn generell och kan användas för att evaluera en rad olika system. Det enda kravet som finns på vår simulator är att systemet som ska evalueras kan implementeras i en container. Vi kan alltså använda vår nätverkssimulator för att exempelvis evaluera hur de andra protokollen, som nämns i avsnitt 1.2 (s. 2), beter sig vid paketförluster.

6.4 Systemets feltolerans

Från testerna med paketförluster går det att observera ett antal intressanta resultat om hur systemet beter sig. Från figur 14a (s. 23) ser vi att antalet framgångsrika fordon blev högre med 6% paketförlust, jämfört med 4%. Detta resultat kan framstå som förvånande då det intuitivt borde vara tvärtom. Detta kan bero på ett antal faktorer. Då fordonens avstånd till korsningen randomiseras, resulterar det i att det blir de fordon som kommer fram till korsningen först som får åka genom direkt, medan fordonen som måste stanna blir stående så pass länge att de inte hinner genom korsningen innan körningen är slut.

En annan faktor kan vara att det helt enkelt är för liten mängd paketförlust som används för att testet ska vara tillräckligt exakt. Eftersom liknande beteende sker mellan 7-8% paketförlust, för att sedan stabiliseras vid 8%, kan det vara ett tecken på att mängden paketförlust som testas är för liten. Exempelvis om vi tittar på figur 14b (s. 23) som ökar med 10%, är den grafen betydligt stabilare.

Jämförs alla de tre scenarierna med varandra, finns det resultat som borde vara likartade men som i själva fallet inte är det. Exempelvis kan det anses att antalet framgångsrika fordon i figur 14a (s. 23) vid 8% borde vara samma som antalet i figur 14c (s. 23) vid fyra fordon. Men realistiskt sätt fungerar det inte på så sätt att alla fordon får samma mängd paketförlust samtidigt. Fordonen kör olika eller befinner sig på positioner som kan hindra kommunikationen temporärt. Något som testet för figur 14c (s. 23) försöker eftersträva.

När mängdernas fastider ökar ser vi att det sker fler överenskommelser mellan fordonen. Flest överenskommelser blev det när fastiden var 5 sekunder med inga paketförluster, men när packetförlusterna ökade skiljde det endast några procentenheter mellan 3 och 5 sekunders fastider. Resultatet är lite förvånande då det med ökad tid mellan fas- och mängdbyten ger serven mer tid att få in ett klartecken från alla fordonen. Det väntade resultatet vore ifall skillnaden mellan 3 och 5 sekunders uppdateringstid hade varit liknande även när paketförluster förekommer. Anledningen till det oväntade resultatet kan vara att 2 sekunder är för lite tid för att systemet ska hinna hantera paketförlusterna.

Genom att öka bandbredden på det trådlösa nätverket, visade det sig att det hjälpte systemet att få fler framgångsrika fordon igenom korsningen. Resultatet var väntat då ökad bandbredd gör att fler meddelanden kan skickas och tas emot samtidigt, vilket i sin tur gör att meddelanden kommer fram till mottagaren fortare. Jämförs resultaten från figur 14a (s. 23) och 16 (s. 25), går det att se att en bandbredd mellan 18 och 27 Mbps, där mängden paketförluster är på 8%, motsvarar samma antal framgångsrika fordon som vid en bandbredd på 6 Mbps med 2% paketförlust. Detta tyder på att systemet klarar av en högre mängd paketförluster då bandbredden ökar.

Lösningen med virtuellt trafikljus och kooperativ kommunikation mellan fordon som vi presenterat, kommer i verklig skala garanterat kommunicera över mobila nätverk. I dagsläget är det 4G som används men under år 2020 kommer 5G att vara i bruk [7]. Skillnaden mellan 4G-nätet och 5G-nätet, är att 5G-nätet levererar betydligt högre hastigheter samtidigt som det blir kortare svarstider. Förbättrade hastigheter med 5G kommer troligen kunna förbättra möjligheten att

implementera ett virtuellt trafikljus anpassat till att kunna hantera en fullstor skala fordon.

6.5 Samhälle och etik

Allt eftersom fler och fler aktörer ger sig in i marknaden för autonoma och självkörande fordon, blir frågor om ansvar allt mer aktuell. Både vad gäller legalt och moraliskt ansvar.

6.5.1 Etik

De etiska frågorna kring självkörande bilar diskuteras flitigt - hur fordonen programmeras att fatta beslut och vem som ansvaret faller på ifall något händer. Detta är frågor som behöver diskuteras i allmänhet och detta projekt i synnerhet.

Tekniken kan antas vara värdeneutral och att den i sig varken är god eller ond. Eftersom den utvecklas av människor, kan deras egna tankar och värderingar potentiellt föras över till koden. Arbetet med att utveckla ett autonomt fordon görs inte av en person som kan hållas ansvarig för koden. Detta gör ansvarsfrågan diffus, speciellt ifall artificiell intelligens och/eller maskininlärning används.

Sett utifrån ett utilitaristiskt etiskt synsätt, går det att se många fördelar med självkörande bilar. Uppskattningsvis 94% av trafikolyckor sker på grund av den mänskliga faktorn, varav 41% beror på bristande uppmärksamhet enligt Singh [2]. Dessa olyckor skulle helt kunna elimineras med en dator som förare.

Dock finns det situationer som inte är lika enkla att hantera och bör diskuteras. Jämförelser med det klassiska exemplet med det skenande tåget och spårväxeln kan appliceras här. Ponera att en person ser ett skenande tåg som rör sig mot en grupp av fem människor liggandes på spåret: ska personen då göra valet att ändra spårväxeln så att tåget istället byter till sidospåret där en person ligger? Vad är mest etiskt att göra?

Även om autonoma fordon minskar antalet dödsolyckor i trafiken, är det osannolikt att de som trots allt kommer omkomma i olyckor med autonoma fordon även skulle ha dött med mänskligt framförda fordon. Folk som inte skulle ha omkommit om *status qou* rådde och fordon framförs av människor, kan då komma att omkomma om man introducerar denna nya tekniken.

Frågor kring datorers uppskattning av människoliv och hur den ska väga mellan dem är problematiska. Olyckor är oundvikliga och datorer kommer då att ställas inför val utan rätt eller fel. Datorer kan däremot, till skillnad från människor, vara helt rationella genom att endast påverkas av signaler från dess sensorer, inte genom känslomässiga intryck. Detta kan ge fördelar i beslutsfattningen.

6.5.2 Ansvarsfrågan

Hevelke och Nida-Rumelin skriver att det mest uppenbara lösningen på ansvarsfrågan är att lägga ansvaret på fordonstillverkaren, som trots allt är den som är ansvarig för den slutgiltiga produkten [35]. Detta skulle dock potentiellt hindra utvecklingen, då ökad ansvarsbörda minskar eller eliminerar fordonstillverkares incitament att skapa marginella förbättringar av säkerheten för att slippa ansvar enligt Merchant och Lindor [36]. Självkörande fordons fulla kapacitet kommer inte till att utnyttjas, om det är så att man skapar incitament att inte utveckla och istället nöja sig med det acceptabel, men inte utmärkt, produkt.

Precis som med vanliga fordon, så är autonoma fordon inte skonade från olyckor och framförallt inte i det tidiga skede dem befinner sig i nu. Exempel på det kan vara en dödsolycka i Arizona, USA med en av Ubers självkörande bilar som såg och upptäckte en fotgängare som korsade vägen, men lät bli att bromsa. Uber friskrevs från allt ansvar [37] och syndabocken blev istället föraren, som riskerar att åtalas för *vehicle manslaughter* då hon inte varit uppmärksam nog vid olyckan [38].

I det tidiga utvecklingsstadiet som självkörande fordon befinner sig i nu, är det kanske rimligt att ha en överseende människa med ansvar som ytterst ansvarig. Men skulle detta bli *modus operandi* i framtiden, kan man hävda att fördelarna och meningen med självkörande fordon försvinner, om inte helt, i alla fall delvis.

6.5.3 Samhälleliga aspekter

Självkörande fordon i olika former kan möjliggöra mobilitet hos personer i samhället som inte kan eller får framföra ett fordon, på grund av exempelvis låg ålder eller bristande fysiska förmåga. Genom att introducera ett transportmedel som inte kräver en mänsklig förare, kan dessa människors rörlighet i samhället ökas och på så sätt ge dem tillgång till platser de inte kunnat åtnjuta tidigare. Man kan då hävda att den sociala hållbarheten i samhället skulle öka, genom att göra platser i samhället tillgängligt för fler.

Ett förenklat resande skulle kunna öka benägenheten för människor att öka sitt resande eller för företag att skicka varor. En rapport skriven av *World Economic Forum* och *The Boston Consulting Group*[39] visar att antalet fordon kommer bli färre men att mängden resande kommer öka . Det blir lättare att nyttja tjänsten och därför kommer man utnyttja den mer. Denna ökning i resande kan då påverka miljön negativt och framförallt den redan ansträngda miljön i städer. Enligt en rapport från U.S Energy Information Administration, kan autonoma fordon dock köras mer effektivt med *eco-driving, platooning* och göras effektivare genom att bara ha tillräcklig kapacitet[40], snarare än överkapacitet som många fordon har nu. Nettoeffekten blir därför svår att predicera.

En annan aspekt som man bör ta i beaktning hur självkörande bilar inkräktar på individers integritet och hur den data som genereras av självkörande bilar behandlas och utnyttjas. Dorothy J. Glancy lyfter fram hur autonoma fordon påverkar individers integritet, genom att autonoma fordon tar kontroll över hur människor rör sig mellan platser, en viktig aspekt i människors liv [41]. Hon nämner också att det kommer väcka bekymmer angående integriteten rörande personlig information, när autonoma fordon generarar personlig data om användarna [41]. Denna data kan användas att spåra användare och deras resvanor och använda den till saker som inskränker på användarens integritet på olika sätt.

6.5.4 Socioekonomiska aspekter

Den autonoma fordonsindustrin kan bidra både positivt och negativt när det kommer till arbetsmarknaden och jobbtillfällen. I USA har 13,3 miljoner människor, motsvarande 9,1% av arbetskraften, sysselsättning inom transportsektorn där 3,6 miljoner av dem har förare som yrkestitel [42]. Antalet människor som kommer få sysselsättning på grund av självkörande bilar kan antas vara betydligt färre och kan inte mäta sig med vare sig människor som kommer mista sin sysselsättning eller den kortsiktiga nettoeffekten på arbetslösheten. Därför kan det tänkas bli en negativ trend om självkörande fordon slår igenom på bred front i samhället.

Joseph A. Schumpeter skrev om kapitalismen och drivkraften bakom den kapitalistiska motorn, vilka han hävdade är nya konsumenter, varor, tjänster, nya former av industriella organisationer och produktions- och transporteringmetoder som företag skapar[43]. Vidare pratar han om nya

marknader och hur dessa nya mutationer kan revolutionera en ekonomisk struktur från insidan och förstöra den gamla för att skapa en ny. Processen benämnde han som *kreativ förstörelsen* och det är en essentiell del av kapitalismen[43]. Resurser frigörs i och med nya effektivare processer och dessa kan strömma till områden där de gör mer nytta. Denna resursfördelning blir förödande för den gamla industrin, men för samhället i stort innebär det möjligheter att använda resursen till annat och på så sätt skapa mer total nytta. Som Alm och Cox tar upp, för att samhället ska få dra nytta av den *kreativa förstörelsen*, måste det accepteras att individer och aktörer kommer få det sämre. Det kanske inte bara i det korta loppet, utan möjligtvis för evigt [44]. Men en effektivare användning av resurser innebär större potentiell nytta.

Den totala nyttan för samhället kan bli större av en produkt, som exempelvis autonoma fordon. Hur denna nytta visar sig är dock inte lika enkel att visa. Skulle nyttan spridas ut över hela samhället, att alla individer får ta del av den, är frågan knappast problematisk. Genom att titta historiskt, går det att se att det i regel inte är att hela samhället får ta del av nyttan, utan snarare enstaka individer eller grupper. Att konkurrera ut mänskliga förare och skapa arbetslöshet hos en stor grupp människor med lägre utbildning, till gagn för en grupp utbildade ingenjörer och utvecklare, kan ses som problematiskt.

6.5.5 Projektets aspekter

Vårt projekt sker i en kontrollerad och avskärmad miljö, utan människor och verkliga trafiksituationer. Våra idéer och lösningar kan komma att spridas utanför vår avgränsade evalueringsmiljö och involvera autonoma fordon i verklig trafik. Vikt måste därför läggas på att skapa ett system där säkerhet och undvikandet av olyckor har högsta prioritet. Feltolerans och att säkerställa robusthet i systemet har haft yttersta prioritet sedan start. Med inspiration av tidigare arbeten och med hjälp från handledare och masterstudenter har algoritmer som implementerats bevisad säkerhet.

6.6 Reflektion och framtida forskning

Gränsande områden där liknande tankar och system kan tänkas tillämpas är andra former av transportsätt. Det förekommer korsande farleder till havs, ofta i form av farleder till hamn- eller älvinlopp längs kuststräckor. Många båtar är dessutom redan utrustade med AIS (*Automatic Identification System*) som ger en rad datapunkter, exempelvis båtens position, kurs och hastighet.

Ett annat område där en priseffektiv lösning utan dyr sensorteknik kan tänkas vara eftertraktat är olika lagersystem. Miljön i ett lager är förutsägbart, då förändringar av omgivningen påverkas av interna aktörer som kan meddela systemet om dessa. Är lasten eller det som förvaras på lagret farligt, kan ett robust system för självkörning med kommunikation och sensorer vara en säker lösning för att minimera risken för människor. Ett exempel på en sådan miljö skulle kunna vara vid lagring av kärnkraftsavfall.

Ett framtida projekt kan vara att anpassa NS-3 för att simulera ett 4G-nätverk, eller ett 5G-nätverk när väl det är lanserat och NS-3 har lagt till en modul för 5G. Det hade gjort att systemet kan evalueras i nätverksimulatorn på ett ännu mer realistiskt sätt. Om virtuella trafikljus utvecklas till den grad att en server kan styra flera korsningar, kommer kommunikationen ske över ett längre avstånd än vad ett Wi-Fi nätverk kan klara av.

Faraj, Fidan och Gaudet använde kooperativ kommunikation mellan fordon för att optimera hastigheten fram till en korsning [23]. Ett framtida projekt kan vara att kombinera deras lösning med vårt virtuella trafikljus. Fordon kan på så sätt optimera sin hastighet fram till korsningen ifall

deras körbana inte är aktiv. En sådan lösning hade både optimerat trafikflödet i korsningen och bränsleförbrukningen hos fordonet.

Eftersom korsningssimulatorn med V2V-protokoll implementerats i den fysiska evalueringsmiljön och den med V2I-protokoll byggs på samma byggstenar borde inte implementationen vara allt för krävande. Dock var det inget som hanns med i det här projektet.

7 Slutsats

Detta kandidatarbete har arbetat med att utveckla en lösning för en korsningshanterare baserad på kommunikation. En kommunikationsalgoritm för server och kommunicerande autonoma fordon i korsningar har tagits fram. Algoritmen är skalbar och kan med begränsade datorresurser hantera en större mängd klienter. Den är också adaptiv och kan hantera ett skiftande trafikflöde. En nätverkssimulator med möjlighet att simulera kommunikationsfel, har utvecklats för att kunna evaluera olika protokoll och testa olika systems feltolerans. Ett V2V-system har implementerats i den fysiska evalueringsmiljö och bevisat sig positivt.

Litteraturförteckning

[1] "Global status report on road safety 2018," World Health Organization, Genève, Schweiz, 2018.

[2] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," National Highway Traffic Safety Administration, Washington DC, USA, DOT HS 812 115, 2015. [Online]. Tillgänglig: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115, hämtad: 2019-02-05.

[3] C. Isidore, "Self-driving cars are already really safe," *CNN Money*, Mar. 2018, [Online]. Tillgänglig: https://money.cnn.com/2018/03/21/technology/self-driving-car-safety/, hämtad: 2019-02-05.

[4] L. Fridman, "Tesla Vehicle Deliveries and Autopilot Mileage Statistics," 2019, [Online]. Tillgänglig: "https://hcai.mit.edu/tesla-autopilot-miles/, hämtad: 2019-02-05.

[5] C. Osborne, "Virtual replacement for traffic lights given green light for pilot tests," *Between the Lines*, Juli 2018, [Online]. Tillgänglig https://www.zdnet.com/article/virtual-replacement-for-traffic-lights-given-green-light-for-pilot-tests/, hämtad: 2019-02-05.

[6] Ericsson AB, "5g radio access," *Ericsson White Paper*, April 2016, [Online]. Tillgänglig: https://www.ericsson.com/assets/local/publications/white-papers/wp-5g.pdf, hämtad: 2019-05-12.

[7] S. Campanello, "Nu tar sverige klivet mot 5g – så ska frekvenserna fördelas," *NyTeknik*, Dec. 2018, [Online]. Tillgänglig: https://www.nyteknik.se/premium/nu-tar-sverige-klivet-mot-5g-sa-ska-frekvenserna-fordelas-6942201, hämtad: 2019-02-13.

[8] "Crash factors in intersection-related crashes: An on-scene perspective," National Highway Traffic Safety Administration, Washington DC, USA, 2010, DOT HS 811 366, 2010. [Online]. Tillgänglig: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811366, hämtad: 2019-03-13.

[9] V. Savic, E. M. Schiller, and M. Papatriantafilou, "Distributed algorithm for collision avoidance at road intersections in the presence of communication failures," *arXiv preprint arXiv:1701.02641*, 2017.

[10] ——, "Aiding autonomous vehicles with fault-tolerant v2v communication," *arXiv preprint arXiv:1710.00692*, 2017.

[11] O. Morales-Ponce, E. M. Schiller, and P. Falcone, "How to stop disagreeing and start cooperating in the presence of asymmetric packet loss." *Sensors (14248220)*, vol. 18, no. 4, 2018.

[12] A. Casimiro, J. Kaiser, E. M. Schiller, P. Costa, J. Parizi, R. Johansson, and R. Librino, "The karyon project: Predictable and safe coordination in cooperative vehicular systems," in *Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on*. IEEE, 2013, pp. 1–12.

[13] O. Morales-Ponce, E. M. Schiller, and P. Falcone, "Cooperation with disagreement correction in the presence of communication failures," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 1105–1110.

[14] M. Mustafa, M. Papatriantafilou, E. M. Schiller, A. Tohidi, and P. Tsigas, "Autonomous tdma alignment for vanets," in *Vehicular Technology Conference (VTC Fall), 2012 IEEE*. IEEE, 2012, pp. 1–5.

[15] S. Dolev, A. Hanemann, E. M. Schiller, and S. Sharma, "Self-stabilizing end-to-end communication in (bounded capacity, omitting, duplicating and non-fifo) dynamic networks," in *Symposium on Self-Stabilizing Systems*. Springer Berlin Heidelberg, 2012, pp. 133–147.

[16] P. Leone and E. M. Schiller, "Self-stabilizing tdma algorithms for dynamic wireless ad hoc networks," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

[17] C. Berger, E. Dahlgren, J. Grunden, D. Gunnarsson, N. Holtryd, A. Khazal, M. Mustafa, M. Papatriantafilou, E. M. Schiller, C. Steup *et al.*, "Bridging physical and digital traffic system simulations with the gulliver test-bed," in *International Workshop on Communication Technologies for Vehicles*. Springer Berlin Heidelberg, 2013, pp. 169–184.

[18] O. Landsiedel, T. Petig, and E. M. Schiller, "Dectdma: A decentralized-tdma," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer International Publishing, 2016, pp. 231–247.

[19] M. Elad, P. Leone, and E. M. Schiller, "Self-stabilizing tdma algorithms for dynamic wireless ad hoc networks."

[20] T. Petig, E. M. Schiller, and P. Tsigas, "Self-stabilizing tdma algorithms for wireless ad-hoc networks without external reference," in *Stabilization, Safety, and Security of Distributed Systems*. Springer International Publishing, 2013, pp. 354–356.

[21] E. Ekenstedt, A. Casimiro, and E. M. Schiller, "Membership-based maneuver negotiation in safety-critical vehicular systems," *Forthcoming Publication*, 2019.

[22] S. Lefèvre, C. Laugier, and J. Ibañez-Gusmàn, "Intention-aware risk estimation for general traffic situations, and application to intersection saftey," INRIA, Paris, Frankrike, RR-8379, 2013.

[23] M. Faraj, B. Fidan, and V. Gaudet, "Cooperative autonomous vehicle speed optimization near signalized intersections," Faculty of Engineering, University of Waterloo, Waterloo, Canada, 2018.

[24] T. C. d. Santos, A. E. Gómez, C. M. Filho, D. Gomes, J. C. Perafan, D. F. Wolf, F. Osório, and L. A. Rosero, "A simulation framework for multi-vehicle communication," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, 2015, pp. 301–308.

[25] V. Savic, E. M. Schiller och M. Papatriantafilou, "Distributed algorithm for collision avoidance at road intersections in the presence of communication failures," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, USA, 2017, ss. 1005-1012. [Online] Tillgänglig: https://doi.org/10.1109/IVS.2017.7995846, hämtad: 2019-02-13.

[26] O. Morales-Ponce, E. M. Schiller och P. Falcone, "How to stop disagreeing and start cooperatingin the presence of asymmetric packet loss," *Sensors*, vol. 18, no. 4, 2018, [Online] Tillgänglig: http://www.mdpi.com/1424-8220/18/4/1287, hämtad: 2019-03-13.

[27] C. Wuthishuwong, A. Traechtler, and T. Bruns, "Safe trajectory planning for autonomousintersection management by using vehicle toinfrastructure communication," *EURASIP Journal on Wireless Communicationsand Networking*, vol. 2015, no. 33, Feb 2015, doi: 10.1186/s13638-015-0243-3, [Online]. Tillgänglig: https://jwcn-eurasipjournals.springeropen.com/track/pdf/10.1186/s13638-015-0243-3, hämtad: 2019-04-02.

[28] L. C. Bento, R. Parafita, and U. Nunes, "Intelligent traffic management at intersections supported by v2v and v2i communications," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, Anchorage, AK, USA, Sep. 2012, pp. 1495–1502, [Online]. Tillgänglig: https://ieeexplore.ieee.org/document/6338766, hämtad: 2019-04-18.

[29] Nationalencyklopedin, "Dirichlets lådprincip," [Online]. Tillgänglig: http://www.ne.se/uppslagsverk/encyklopedi/lång/dirichlets-lådprincip, hämtad: 2019-04-30.

[30] C. Wilt, J. Thayer, and W. Ruml, "A comparison of greedy search algorithms," 2010, "[Online]. Tillgänglig: https://www.aaai.org/ocs/index.php/SOCS/SOCS10/paper/viewFile/2101/2515, hämtad: 2019-05-08".

[31] A. Arkheden, R. Zaragatzky, A. Lindhé, and R. Gustafsson, "Ett prisvärt alternativ för global visuell lokalisering och styrning av autonoma fordon," 2016, 78.

[32] [Online]. Tillgänglig: https://www.fir3net.com/Networking/Terms-and-Concepts/virtual-networking-devices-tun-tap-and-veth-pairs-explained.html, hämtad: 2019-03-18.

[33] D. Jiang and L. Delgrossi, "Ieee 802.11p: Towards an international standard for wireless access in vehicular environments," *VTC Spring 2008 - IEEE Vehicular Technology Conference*, May 2008.

[34] Nsnam, "Design documentation," 2019. [Online]. Available: https://www.nsnam.org/docs/models/html/wifi-design.htmldesign-details

[35] A. Hevelke och J. Nida-Rumelin, "Responsibility for crashes of autonomous vehicles: An ethical analysis," *SCIENCE AND ENGINEERING ETHICS*, vol. 21, no. 3, pp. 619–630, 2015, [Online] Tillgänglig: https://link.springer.com/content/pdf/10.1007%2Fs11948-014-9565-5.pdf, hämtad: 2019-03-13.

[36] G. E. Marchant och R. A. Lindor, "The coming collision between autonomous vehicles and the liability system," *Santa Clara Law Review*, vol. 52, no. 4, 2012, [Online] Tillgänglig: https://digitalcommons.law.scu.edu/lawreview/vol52/iss4/6, hämtad: 2019-03-13.

[37] C. Shu, "Prosecutors find uber not criminally liable in 2018 arizona self-driving crash that killed a pedestrian," 2018, [Online]. Tillgänglig: https://techcrunch.com/2019/03/05/prosecutors-find-uber-not-criminally-liable-in-2018-arizona-self-driving-crash-that-killed-a-pedestrian, hämtad: 2019-05-16.

[38] H. Somerville and D. Shepardson, "Uber driver was streaming hulu show just before self-driving car crash - police report," June 2018, [Online]. Tillgänglig: https://www.reuters.com/article/uber-selfdriving-crash/uber-driver-was-streaming-hulu-show-just-before-self-driving-car-crash-police-report-idUSL1N1TO05R, hämtad: 2019-05-16.

[39] J. Moavenzadeh and N. S. Lang, "Reshaping urban mobility with autonomous vehicles: Lessons from the city of boston," June 2018, "[Online]. Tillgänglig: http://www3.weforum.org/docs/WEF_Reshaping_Urban_Mobility_with_Autonomous_Vehicles_2018.pdf, hämtad: 2019-05-08".

[40] "Autonomous vehicles: Uncertainties and energy implications," May 2018, [Online]. Tillgänglig: https://www.eia.gov/outlooks/aeo/pdf/AV.pdf, hämtad: 2019-05-16.

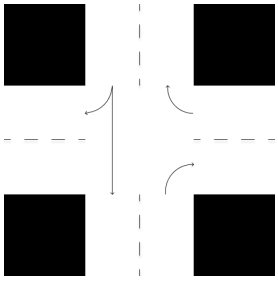
[41] D. J. Glancy, "Privacy in autonomous vehicles," *Santa Clara Law Review*, vol. 52, no. 4, pp. 1171–1239, 12 2012, [Online]. Tillgänglig: https://www.eia.gov/outlooks/aeo/pdf/AV.pdf, hämtad: 2019-05-16.

[42] "Transportation employment," United States Department of Transportation, 2018, [Online]. Tillgänglig: https://www.bts.gov/transportation-economic-trends/tet-2018-chapter-4-employment, hämtad: 2019-05-16.

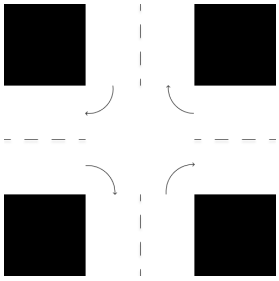
[43] J. A. Schumpeter, *Capitalism, Socialism and Democracy.* New York: Harper, 1942, pp. 82-95.

[44] R. Alm and W. M. Cox, "Creative destruction," 2018, [Online]. Tillgänglig: https://www.econlib.org/library/Enc/CreativeDestruction.html, hämtad: 2019-05-16.

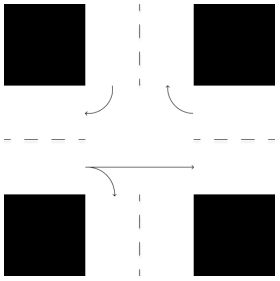
Bilaga Elva konfilkfria mängder



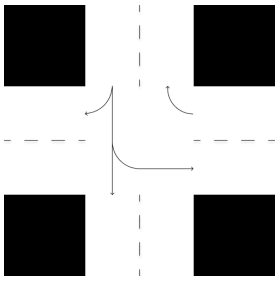
(a) Mängd 1



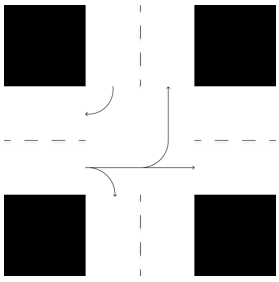
(b) Mängd 2



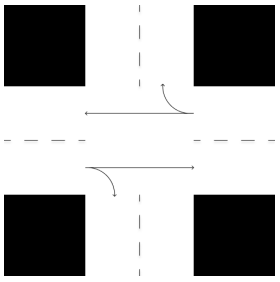
(c) Mängd 3



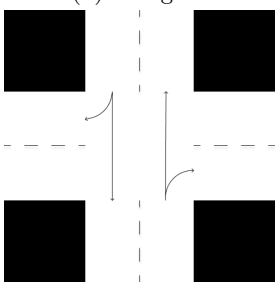
(d) Mängd 4



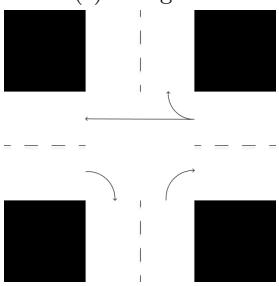
(e) Mängd 5



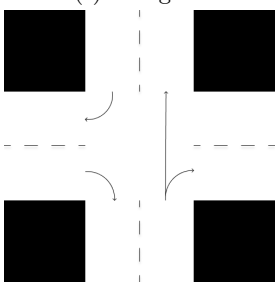
(f) Mängd 6



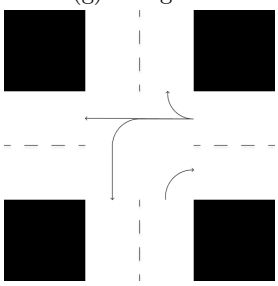
(g) Mängd 7



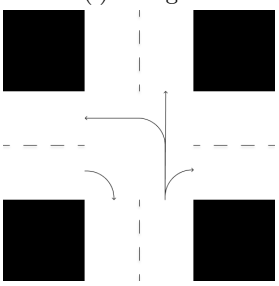
(h) Mängd 8



(i) Mängd 9



(j) Mängd 10



(k) Mängd 11