Ontwerp en evaluatie van een zelfconfigurerende draadloze mesh-architectuur

Design and Evaluation of an Auto-Configuring Wireless Mesh Network Architecture

Stefan Bouckaert

Promotor: prof. dr. ir. I. Moerman
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Elektrotechniek

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. D. De Zutter
Faculteit Ingenieurswetenschappen
Academiejaar 2009 - 2010

UNIVERSITEIT GENT

ISBN 978-90-8578-359-6
NUR 986
Wettelijk depot: D/2010/10.500/35

Universiteit Gent
Faculteit Ingenieurswetenschappen
Vakgroep Informatietechnologie

Ontwerp en evaluatie van een zelfconfigurerende draadloze mesh-architectuur

Design and Evaluation of a Self-Configuring Wireless Mesh Network Architecture

Stefan Bouckaert

Proefschrift tot het bekomen van de graad van
Doctor in de Ingenieurswetenschappen:
Elektrotechniek
Academiejaar 2009-2010

Universiteit Gent
Faculteit Ingenieurswetenschappen
Vakgroep Informatietechnologie

Promotor: prof. dr. ir. Ingrid Moerman

Universiteit Gent
Faculteit Ingenieurswetenschappen

Vakgroep Informatietechnologie
Gaston Crommenlaan 8 bus 201, B-9050 Gent, België

Tel.: +32-9-331.49.00
Fax.: +32-9-331.48.99

Dit werk kwam tot stand in het kader van een
specialisatiebeurs van het IWT-Vlaanderen
(Instituut voor de aanmoediging van Innovatie door
Wetenschap en Technologie in Vlaanderen)

Proefschrift tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen:
Elektrotechniek
Academiejaar 2009-2010

Dankwoord

Het dankwoord. Een bijzonder stukje tekst: het laatste punt op de to-do lijst alvorens dit boek naar de drukker kan vertrekken. Het sluitstuk van een goede vier jaar wetenschappelijk onderzoek. Vermoedelijk ook het deel dat door zowat iedereen die dit boek ter hand neemt als eerste zal gelezen worden.

Onder het mom van *related work* studie bladerde ik voor het schrijven van onderstaande tekst nog even door de doctoraatsboeken van (ex-) collega's, op zoek naar hun wijze (dank)woorden. De exacte referenties zal ik u besparen, maar zowat allen zijn ze het er over eens: een doctoraat schrijven doe je niet alleen. Minstens even belangrijk als de wetenschappelijke output zijn de mensen die je onderweg ontmoet. Het zal u weinig verbazen dat ook ik me bij deze conclusies aansluit: de afgelopen jaren zijn gekenmerkt door onvergetelijke momenten en boeiende ontmoetingen. Collega's worden ex-collega's, maar heel wat vrienden-collega's blijken vrienden te blijven. En dat is ongetwijfeld één van de belangrijke meerwaarden van de afgelopen periode als doctoraatsstudent. Daarom, alvast aan allen die de afgelopen jaren op directe of indirecte manier hebben bijgedragen tot de totstandkoming van dit boek, al was het maar door een babbeltje in de gang of aan de koffieautomaat: een welgemeende dankjewel!

Hoewel het onmogelijk is om iedereen bij naam te noemen in dit dankwoord, kunnen een aantal personen en organisaties toch niet anoniem blijven. Vooreerst wens ik de vakgroepvoorzitter prof. Daniël De Zutter en zijn voorganger, prof. Paul Lagasse, te bedanken voor de blijvende inspanningen die van INTEC een solide onderzoeksomgeving maken. Ik bedank in deze context eveneens prof. Piet Demeester voor de dagelijkse sturing van de IBCN onderzoeksgroep: het is een luxe om binnen IBCN onderzoek te kunnen verrichten in een aangename en bijzonder goed uitgeruste omgeving. Uiteraard gaat mijn dank ook uit naar mijn promotor, prof. Ingrid Moerman, voor het vertrouwen, de interessante discussies en de steeds waardevolle feedback. Ik dank de drijvende krachten achter het IWT en het IBBT, respectievelijk om vier jaar lang dit onderzoek te financieren, en voor het opzetten van waardevolle projecten in samenwerking met de industrie en de daaruit voortvloeiende contacten.

Voor het werk binnen het Eye-Sense project bedank ik in het bijzonder Nicolas 'Nico' Letor. Verder dank ik iedereen die actief was binnen het GeoBIPS project voor de bijzonder leerrijke en uiterst aangename samenwerking. Een speciale vermelding gaat hier uit naar Dries Naudts en Johan Bergs. Ik zal niet snel vergeten hoe we voor dag en dauw richting Ranst vertrokken om er in de gietende regen een persdemo voor te bereiden, of hoe er met persluchtflessen op de rug op de meest uiteenlopende plaatsen testen werden uitgevoerd. Ik dank eveneens alle academische en industriële partners voor de samenwerking binnen het DEUS project, in het bijzonder Kristof Willemyns, (alweer) Nico en Peter De Cleyn om samen de technische kant van 'WP4' tot een goed einde te brengen.

Waar tijdens de projecten detailkennis of gespecialiseerde apparatuur ontbrak voor metingen op de fysische laag, kon er steeds gerekend worden op de mensen van de WiCa groep voor assistentie. Bedankt Leen, Wout, Emmeric en David.

Het aantal (ex-)medewerkers binnen de mobile groep van IBCN is ondertussen zo groot geworden dat ik me niet waag aan een volledige opsomming. Ik bedank iedereen waarmee ik samenwerkte of die ooit tijdens een vergadering, in de wandelgangen of (ver) daarbuiten voor informatie of inspiratie zorgde!

Omdat er nu eenmaal heel wat tijd 'op bureau' wordt doorgebracht, is een aangename werkomgeving bijzonder belangrijk. Ik voelde me onmiddellijk thuis in bureau 3.12 samen met Jan, Philippe, Jeroen, Benoît, Dries, Tom, Andy en Frederic. Door de tijd heen zochten anciens nieuwe oorden op, de luxe-bureauplaatsen werden ingepikt en de lege plaatsen opgevuld met nieuwe collega's. In volgorde van verschijnen: eveneens bedankt aan Nicolas, Irina, David, Willem, Tom, Sammy en Eric. In welke configuratie van het bureau dan ook: de bureauactiviteiten zoals karting en trappistenavondjes (in die chronologische volgorde) werden zeer gesmaakt!

Tijdens de bureauherschikking begin dit jaar was de finale doctoraatsrush al ingezet en door alle drukte zijn ongetwijfeld een aantal bureaudiscussies aan mij voorbijgegaan. Desalniettemin had ik vanaf dag één het gevoel dat we in onze nieuwe deel-van-de-mobile-bureau-3.15 met een heel enthousiaste groep samen zitten. Niet alleen voor de goede sfeer, maar ook voor alle interessante al dan niet mobile gerelateerde (bureau-)discussies en, desgevallend, samenwerking voor papers: bedankt Pieter B, Eli, Pieter DM, Bart J, Peter, Evy, Lieven, Frank, Jono en Kristof. Ik kan dit paragraafje toch niet afsluiten zonder Bart J, bij wie ik keer voor keer onaangekondigd terecht kon met vragen over het w-iLab.t, nog eens extra te bedanken.

Dankzij het admin-team bleef de serverinfrastructuur draaien en werd de nood aan extra hardware steeds snel ingelost: bedankt Wouter, Bert DK, Bert DV, Pascal, Serge, Brecht en Jonathan. Ik dank ook Andy voor de ondersteuning en automatiseringen bij het opzetten en organiseren de practica. Met dank aan het secretariaat en de finances kon ik het niet-technische papierwerk tot een minimum beperken, werden administratieve vraagjes snel opgelost, hotelreserveringen en vluchten vlotjes geboekt, en onkosten correct afgerekend. Bedankt aan Martine, Davinia, de Ilses, Karien, Bernadette, Marleen, Dalila, en iedereen die verder achter de administratieve firewall verborgen zit!

In de marge van het werk was steeds een plaatsje gereserveerd voor sport en ontspanning. Of het nu om de korte 'carrière' in vervlogen tijden binnen het voetbal of het badminton gaat, dan wel om de langere geschiedenis van de volleybalwedstrijden: het was en is nog steeds een plezier om samen te sporten. Verder zorgden de quizzen op maandagavond met onze quizploeg 'Vet Smaakt Slecht' jaren lang voor amusement, genoten we van spelletjesavonden, verjaardagen, in-

stuiven, barbecues, (mini-) weekendjes en zelfs van een zomerreis. Bedankt Andy, Bart D, Bart DV, Bart L, Benoît, Bert DV, Bruno, Cedric, Chris, David, Delphine, Dieter, Elke, Ellen, Eric, Erik, Filip, Ine, Ingrid, Jan VO, Jeroen DW, Jurgen, Koert, Kristof L, Kristof W, Lisa, Mario, Nele, Niels, Olivier, Pieter B, Pieter S, Pieter T, Philip, Peter, Sarah, Sofie D, Stefanie, Steven, Stijn E, Tom B, Tom VL en alle gastsporters en -quizzers!

Soms was er tijdens een conferentie even de tijd om vooraf, nadien, of tussendoor de omgeving te verkennen. Deze tussendoortjes werden mee ingekleurd dankzij Johan, Nico, Bart B, Evy, Koen, Sofie V, Steven en Stijn V, waarvoor dank!

Ook buiten het werk was er gelukkig ruimte voor ontspanning. Voor het weekendvoetbal, de occasionele quizzen, de ardennenweekends, de bijhorende fietstochtjes en meer van dat moois: bedankt Kobe en Miet, Matthias en An, Maarten en Anne, Thijs en Katrien, Tijs en Saar. Minder talrijk maar daarom niet minder gewaardeerd zijn de ontmoetingen met ‘de weekendbende’. Dank aan allen, uiteraard voor de weekends maar ook voor alle ontmoetingen tussendoor! In de categorie bendes is nog een laatste ereplaatsje voor ‘de BE-bende’: wie zien elkaar niet zo vaak maar steeds opnieuw is het voor mij op het BE-weekend een blij terugzien. Bedankt aan Philippe voor de avonturen in Schotland en Noorwegen en aan alle anderen die niet in deze lijst geraakt zijn maar er steeds opnieuw voor zorgen dat de agenda ook in het weekend gevuld blijft!

Uiteraard is in dit dankwoord ook een plaatsje voorzien voor mijn familie en in het bijzonder voor mijn moeder: bedankt voor alle steun en de goede zorgen, niet enkel tijdens de doctoraatsjaren maar ook alle jaren lang daarvoor.

De geroutineerde dankwoordenlezer onder u voelt ongetwijfeld het einde van dit dankwoord naderen en weet dat hier het ‘save the best for last’ principe wordt gehanteerd. In mijn geval luistert ‘the best’ naar de naam Sofie. Sofie, bedankt voor je naleeswerk en bovenal bedankt voor al het moois gedurende de afgelopen jaren. Nu de laatste zin die voor dit boek geschreven wordt bijna op papier staat, kijk ik alvast uit naar een iets minder hectische periode samen met jou!

Gent, mei 2010
Stefan Bouckaert

Table of Contents

List of Figures

List of Tables

List of Acronyms

A

ACK	Acknowledgement
AES	Advanced Encryption Standard
AODV	Ad-Hoc On Demand Distance Vector
APE	Ad-Hoc Protocol Evaluation
ARP	Address Resolution Protocol

C

CBR	Constant Bit Rate
COTS	Commercial-Off-The-Shelf
CQP	Channel Quality Parameter
CQV	Channel Quality Vector
CRT	Cathode Ray Tube
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance

D

DCF	Distributed Coordination Function
DHCP	Dynamic Host Configuration Protocol
DIFS	Distributed coordination function Interframe Space
DoS	Denial of Service
DSR	Dynamic Source Routing Protocol
DUT	Device Under Test

E

EE	Environment Emulator

F

FDM	Frequency-Division Multiplexing
FEC	Forward Error Correction
FRESME	Frequency Selection based on Message Exchange

G

GUI	Graphical User Interface

I

IBCN	Intec Broadband Communication Networks
iDB	Information Database
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
I/O	Input / Output
ILP	Integer Linear Programming
IP	Internet Protocol
ISM	International Scientific and Medical (frequency band)

L

LLC	Link Layer Control

M

MAC	Medium Access Control (layer)
MAN	Metropolitan Area Network

MANET	Mobile Ad-Hoc Network
MIMO	Multiple-Input, Multiple-Output
MTU	Maximum Transmission Unit
MUX/DEM	Multiplexer / Demultiplexer

N

NDB	Neighbor Discovery Beacon
NIC	Network Interface Card
NFS	Network File Share

O

OLSR	Optimized Link State Routing
OSI	Open System Interconnection (Reference Model)

P

PCI	Peripheral Component Interconnect
PDA	Personal Digital Assistant
PHY	Physical (layer)
PIN	Personal Identification Number
PoE	Power over Ethernet
PXE	Preboot Execution Environment

Q

QoE	Quality of Experience
QoS	Quality of Service

R

RF	Radio Frequency

RSSI	Received Signal Strength Indication
RT	Reconnaissance Team
RTS/CTS	Request To Send / Clear To Send

S

SDM	Space-Division Multiplexing
SHA	Secure Hash Algorithm
SNR	Signal to Noise Ratio

T

TBRPF	Topology Dissemination Based on Reverse-Path Forwarding
TCP	Transmission Control Protocol
TETRA	Terrestrial Trunked Radio

U

UDP	User Datagram Protocol
UPNP	Universal Plug aNd Play
USB	Universal Serial Bus

V

VPAN	Virtual Private Ad-Hoc Network
VoIP	Voice over IP

W

WEP	Wired Equivalent Privacy
WINES	Wireless Network Emulation System
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WPA	Wi-Fi Protected Access
WPAN	Wireless Personal Area Network

Samenvatting –Summary in Dutch–

In onze hedendaagse maatschappij worden we steeds meer afhankelijk van communicatienetwerken. Hoewel het Internet pas midden jaren '90 bekend werd bij het grote publiek, is het vandaag voor velen ondenkbaar om een werkdag zonder het Internet door te brengen. Ook in de vrije tijd speelt digitale communicatie een grote rol: de krant lezen, een filmticket reserveren, e-mailen met vrienden, sociale netwerksites raadplegen- de mogelijkheden zijn eindeloos. Tegelijkertijd komen er dankzij de miniaturisatie en prijsverlaging van de elektronica steeds meer mobiele toestellen zoals laptops en smartphones in omloop. De beschikbaarheid van deze toestellen in combinatie met de stijgende vraag naar netwerkconnectiviteit leidt er toe dat steeds meer informatie op een draadloze manier wordt geraadpleegd. Mobiele telefoons met Internettoegang zijn allang geen uitzondering meer, en draadloze toegangspunten gebaseerd op Wi-Fi technologie zijn alomtegenwoordig. De snelheid waarmee deze draadloze technologieën de markt veroveren is indrukwekkend.

Als alternatief voor de bovenstaande en soortgelijke technologieën die in essentie enkel in de laatste stap van de verbinding de klassieke telefoon- of netwerkkabel vervangen door een draadloze verbinding, wordt al sinds begin de jaren '70, oorspronkelijk onder impuls van het Amerikaanse leger, onderzoek gedaan naar netwerken die volledig onafhankelijk van bekabelde infrastructuur kunnen werken. Deze zogenaamde draadloze multi-hop ad-hoc netwerktechnologie stelt eindgebruikertoestellen of knopen in staat om rechtstreeks met elkaar te communiceren over een draadloze verbinding. Indien twee knopen zich te ver uit elkaar bevinden om rechtstreeks te kunnen communiceren, zullen tussenliggende knopen zich gedragen als routers om zo de informatie van bron naar bestemming over te brengen via meerdere draadloos verbonden knopen als tussenstap.

Ook buiten een militaire context zijn heel wat toepassingen voor ad-hoc netwerken denkbaar: als na een natuurramp de bestaande communicatie-infrastructuur is vernietigd, zijn hulpdiensten in staat om snel een netwerk op te zetten door onderling draadloze netwerken op te bouwen. Tijdens een vergadering kunnen bestanden rechtstreeks tussen computers worden uitgewisseld zonder dat een bedraad toegangspunt nodig is. Kortom: voor elke situatie waar een (tijdelijke) netwerkinfrastructuur nodig is en het installeren van netwerkkabels te duur, te omslachtig of onmogelijk is, kunnen ad-hoc netwerken een uitkomst bieden. Bovendien zijn heel wat draadloze toestellen zoals laptops en smartphones vandaag al uitgerust met draadloze technologieën die theoretisch in staat zijn deel te nemen aan deze draadloze ad-hoc netwerken.

Door de vele nuttige toepassingen en het wijdverspreide karakter van de hardware die ad-hoc netwerken mogelijk maakt, kende het onderzoeksdomein de laatste decennia een enorm succes. Desondanks worden draadloze ad-hoc netwerken in ons dagelijkse leven zo goed als nooit gebruikt.

De kernvraag waarrond het eerste deel van dit boek is opgebouwd is dan ook hoe deze opmerkelijke tegenstelling te verklaren valt. Daartoe belicht een inleidend hoofdstuk de domeinen van draadloze digitale communicatie in het algemeen en draadloze ad-hoc netwerken in het bijzonder. Een overzicht van toepassingen, technologieën en onderzoeksdomeinen kadert het onderzoek in een bredere maatschappelijke en theoretische context.

Het tweede hoofdstuk graaft vervolgens naar de oorsprong van de geringe populariteit van draadloze ad-hoc netwerken. Meerdere oorzaken worden geïdentificeerd. De belangrijkste vaststelling is dat meerdere theoretische beloftes van ad-hoc netwerken niet altijd worden waargemaakt: waar in de literatuur vaak gesproken wordt over de gebruiksvriendelijkheid, stabiliteit, schaalbaarheid en het zelforganiserend en zelfherstellend karakter van ad-hoc netwerken, blijkt bij praktische validatie vaak dat deze claims niet worden waargemaakt. De oorzaak van deze afwijking wordt gezocht in het feit dat traditioneel heel wat onderzoek naar ad-hoc netwerken enkel gebeurt door middel van analytische berekeningen en simulaties. Door middel van experimenten wordt aangetoond dat verschillende basisveronderstellingen gemaakt bij dergelijk onderzoek fout of onnauwkeurig zijn, wat tot fundamentele problemen kan leiden bij de praktische realisatie van theoretisch succesvolle algoritmes.

Rekening houdende met de vastgestelde praktische beperkingen wordt daarna een hiërarchische draadloze meshnetwerkarchitectuur uitgebouwd, waarbij bouwblokken gedefinieerd worden die het mogelijk maken om niet alleen een theoretisch performant draadloos netwerk uit te werken, maar het ook praktisch te kunnen realiseren. Hiertoe wordt ook een knooparchitectuur opgesteld, die losse koppeling van de verschillende bouwblokken over de lagen heen toelaat. De draadloze mesh knopen zijn krachtiger dan traditionele draadloze eindgebruikertoestellen, en vormen een draadloos netwerk dat de eindgebruikers verbindt met elkaar en eventueel met externe diensten. De eindgebruikertoestellen zelf zijn zo niet langer verantwoordelijk voor het doorsturen van informatie van andere eindgebruikers.

In een derde en vierde hoofdstuk van het boek worden vervolgens belangrijke bouwblokken van de architectuur ontwikkeld. Het derde hoofdstuk bestudeert hoe draadloze mesh knopen die zijn uitgerust met meerdere draadloze netwerkkaarten efficiënt gebruik kunnen maken van het beschikbare draadloze spectrum. Meer specifiek wordt eerst een protocol voor kanaalselectie ontwikkeld, met als concrete toepassing het configureren van de kanalen van mesh knopen die gemonteerd zijn op voertuigen van interventieteams, zoals de brandweer of civiele bescherming. In een dergelijk dynamisch scenario kunnen knopen op elk moment worden toegevoegd en weer verdwijnen, en is een snelle kanaalconfiguratie met zo weinig mogelijk controleverkeer wenselijk. Aangezien in een dergelijk dynamisch netwerk geen informatie beschikbaar is over het gedrag op lange termijn, wordt een kanaal voor datacommunicatie pas gereserveerd op het moment dat informatie wordt uitgewisseld. De kanaalreservering gebeurt in onderlinge afspraak tussen de twee knopen waartussen een draadloze verbinding wordt opgezet. Het protocol wordt ontwikkeld, en voor het specifieke geval van draadloze IEEE 802.11g compatibele netwerken gesimuleerd en geïmplementeerd op een testbed. Resultaten tonen de haalbaarheid en snelheid van de aanpak aan. Bij simulaties in een roostertopologie met 100 mesh knopen zorgt het gedistribueerde protocol gemiddeld in 86% van de gevallen voor een optimale verdeling van het dataverkeer over de beschikbare kanalen voor datatransport.

Vervolgens wordt een basisconcept ontwikkeld voor het schatten van de maximale bandbreedte van bron tot bestemming in draadloze mesh netwerken met meerdere interfaces. De methode schat de maximale bandbreedte voor elke link afzonderlijk, en verspreidt deze informatie daarna over het netwerk. Doordat elke knoop in het mesh netwerk de beschikbare informatie over de topologie en kanaalselectie kent, kan voor elk mogelijk pad in het netwerk een schatting gemaakt worden van de te verwachten maximale bandbreedte. Ook dit protocol wordt gerealiseerd, en in een testopstelling wordt de geschatte capaciteit van een verbinding door middel van het ontwikkelde protocol vergeleken met een referentiemeting. De resultaten tonen aan dat de schattingsmethode de referentiemeting sterk benadert, terwijl ze hiervoor maar een heel beperkte hoeveelheid controleverkeer aan het netwerk moet toevoegen.

Hoewel de ontwikkelde protocollen, eens geïnstalleerd, volledig automatisch werken, worden in het derde hoofdstuk instellingen zoals adresconfiguratie of de basisconfiguratie van de interfaces niet behandeld, en achter de schermen manueel uitgevoerd. Nochtans is het doorvoeren van een correcte configuratie van draadloze systemen niet triviaal, zeker niet voor mensen met een beperkte of helemaal geen kennis in verband met draadloze netwerken.

Daarom focust hoofdstuk vier op het ontwikkelen van bouwblokken die toelaten een draadloos mesh netwerk automatisch te installeren, uit te breiden en te beheren, zonder manuele configuratie te vereisen. De ontwikkelde methode laat toe om nieuwe draadloze knopen op een veilige manier in een netwerk te integreren, zonder vooraf een fysieke verbinding met de node te moeten maken. Nieuwe knopen kunnen rechtstreeks uit de verpakking aan een bestaand netwerk worden toegevoegd door ze op een willekeurige plaats binnen het bereik van het bestaande draadloze netwerk te plaatsen. De hele procedure neemt bovendien maar enkele seconden in beslag. De protocollen worden bovendien niet enkel theoretisch uitgewerkt: alle subsystemen worden geïmplementeerd zodat experimenten op een Wi-Fi testbed mogelijk worden. Deze testen bewijzen de haalbaarheid van de aanpak: in enkele minuten wordt een beveiligd draadloos mesh netwerk met meer dan 20 nodes probleemloos uitgerold.

Zowel bij het vaststellen van de problemen in de huidige generatie ad-hoc netwerken, als bij de ontwikkeling van de protocollen werd veelvuldig gebruik gemaakt van implementaties op hardwareplatformen die vandaag de dag beschikbaar zijn. Deze aanpak heeft er ongetwijfeld toe geleid dat de ontwikkelde protocollen veel beter overweg kunnen met talloze problemen die zich voordoen in een draadloze omgeving dan het geval was geweest indien enkel een theoretisch pad werd bewandeld. Het succes van deze aanpak is bovendien duidelijk, aangezien de resultaten in dit boek niet enkel op papier bestaan maar experimenteel kunnen geverifieerd worden. Nochtans is het gebruik van experimentele methodes voor het ontwikkelen van draadloze netwerken niet vanzelfsprekend. Net zoals het ondoordacht gebruik van simulatoren tot foute conclusies kan leiden, kan het gebruik van implementaties bij de ontwikkeling van draadloze protocollen voor veel frustraties zorgen: het implementeren van een oplossing vraag veel tijd, die niet noodzakelijk beloond wordt met gunstige resultaten.

Steunend op de ervaring die werd opgedaan tijdens de totstandkoming van bovenstaande realisaties wordt daarom in hoofdstuk 5 een generieke methodologie ontwikkeld voor het ontwerpen, ontwikkelen en eventueel tot uitvoering brengen van draadloze netwerkprotocollen. Door het volgen van deze methodologie kunnen onderzoekers die draadloze netwerken in de ruime zin van het woord bestuderen heel wat basisfouten vermijden, sneller tot meer betrouwbare protocollen en resultaten komen, en de hoeveelheid en kwaliteit van de wetenschappelijke output verhogen. Dit hoofdstuk behandelt bijgevolg veeleer basisaspecten van onderzoek in draadloze netwerken dan het zorgt voor verdere uitdieping van een specifiek aspect. Een dergelijke methodologie is evenwel van groot belang om de kwaliteit en bruikbaarheid van toekomstig onderzoek naar draadloze ad-hoc netwerken te verzekeren.

Om de methodologie te illustreren wordt tijdens dit hoofdstuk bovendien een extra uitbreiding toegevoegd aan de oplossing voor het automatisch installeren van een draadloos mesh netwerk. Gebaseerd op metingen die via de draadloze netwerkchip worden verkregen, wordt een oplossing ontwikkeld die toelaat te detecteren of een bepaalde geografische locatie al dan niet geschikt is om een nieuwe mesh knoop te plaatsen. Hoewel deze ontwikkeling hoofdzakelijk illustratief wordt gebruikt, is ze eveneens ruimer toepasbaar en draagt ze bij tot het vereenvoudigen van de installatieprocedure en verbeteren van de draadloze netwerkkwaliteit.

Een laatste hoofdstuk plaatst de conclusies uit dit werk nogmaals op een rijtje, en werpt een blik op de toekomst van draadloze mesh en draadloze ad-hoc netwerken. Door tijdens het onderzoek naar draadloze ad-hoc netwerken te vertrekken vanuit realistische veronderstellingen, en mede dankzij de realisaties uit dit boek, hoeft een toekomst voor draadloze ad-hoc netwerken niet eens zo veraf te liggen.

Summary

Our current society is depending more and more on communication networks. Even though the Internet was not known by the general public until the mid-nineties, many people today depend on a permanent Internet connection in order to efficiently organize their professional lives. Furthermore, the Internet has proved to be an indispensable tool for catching up with the latest news, making movie reservations, emailing friends, or visiting one of many social networking sites. At the same time, due to the miniaturization of electronic components and continuous price drops, a massive number of powerful mobile devices, such as laptops or smartphones, have entered the market. The need for information combined with the high availability of mobile devices causes an increasing amount of information to be consulted in a mobile way. Mobile phones with Internet access are no longer an exception and wireless access points based on Wi-Fi technology are widespread. The rate at which these technologies are adopted is impressive.

Cellular based techniques and Wi-Fi access points essentially replace only the last part of the traditional telephone cable or network cable by a wireless connection. As a fully wireless alternative to the above and other similar technologies which rely on the availability of a wired infrastructure, wireless ad-hoc networks have been studied since the seventies under the initial impulse of the American Department of Defense. These wireless ad-hoc networks enable end-user devices or nodes to communicate to each other over a direct wireless link. In case two nodes are no longer within each other's transmission range, intermediate ad-hoc nodes will behave as routers, forwarding packets over multiple wireless hops until the destination is reached.

Plenty of applications for wireless ad-hoc networks exist outside a military context: if communication infrastructure is destroyed after a nature disaster such as an earthquake, emergency services may quickly organize their own replacement communication infrastructure by configuring a wireless ad-hoc network. During meetings, files can be exchanged over a direct wireless connection without the need for an access point. Put shortly, for any situation in which a (temporary) network infrastructure is needed and cable installment is too expensive, too laborious or simply unwanted, ad-hoc networks may be a solution. Moreover, many of today's devices are already equipped with wireless technologies that are theoretically able to support these networks.

Because the enabling hardware is widespread and applications are promising, there has been a massive amount of international research related to ad-hoc networks during the last decades. Nevertheless, wireless ad-hoc networks are hardly used in our everyday environment.

In a first part of this book, an explanation to this remarkable observation is sought. To this end, a first chapter provides an introduction to digital wireless communication and wireless ad-hoc networks. An overview of applications, supporting technologies and research topics is given in order to position this work within a larger research context.

In the second chapter, different causes for the limited success of ad-hoc networks are revealed. Most importantly, it is observed that countless works in literature praise the robustness, self-organizing and self-recovering nature of wireless ad-hoc networks. However, these claims can often not be validated when the solutions are deployed in reality. The root of this discrepancy is sought in the fact that traditionally, much of the research on ad-hoc networks is performed solely using mathematical models or simulations. Through a series of experiments based on a selection of current generation Wi-Fi compatible devices, it is shown that several basic assumptions used during theoretical research, such as 'increasing the transmission power leads to a better link' or 'the transmission range of a node has a circular shape' may be inaccurate or even wrong. These assumptions may eventually lead to fundamental issues when deploying wireless ad-hoc networks under real-life conditions; for example, using a selection of Wi-Fi based devices, it is shown that the simultaneous use of multiple interfaces in a single integrated device, even when the interfaces are operating on theoretically orthogonal frequencies, is less optimal than would be expected from a theoretical point of view.

Next, keeping the observed practical limitations in mind, a hierarchical wireless mesh network is defined. The building blocks of this architecture provide functionalities that enable the deployment of high quality wireless mesh networks, not only in theory, but also in practice. To this end, a node architecture is defined that allows loose interconnection of the different building blocks across the different layers of the OSI stack. In comparison with the end-user devices that are used in traditional wireless networks, mesh nodes are more powerful in terms of processing power and memory capacity, and may be equipped with multiple wireless interfaces. The wireless mesh nodes automatically form a wireless backbone network. Next, end-user clients can connect to this backbone through wireless access points which are connected to the mesh nodes. As such, the end-user devices are no longer responsible for forwarding the traffic of other end-users. As such, several requirements of traditional ad-hoc networks may be relaxed, while still retaining the fully wireless character of the network.

In the third and fourth chapter of this book, several building blocks of the mesh architecture are developed. The third chapter studies how multi-interface mesh networks can efficiently use the available wireless spectrum. More specifically, first, a channel selection protocol is developed. As a use case, the deployment of mesh routers on top of emergency vehicles of intervention teams is considered. In this dynamic scenario, network nodes may be added at any time in the network and disappear again after a few minutes. As such, a fast channel configuration with as few control overhead as possible is desirable. Moreover, in such dynamic network where random traffic links are possible, one cannot rely on long-term traf-

fic profiles. Therefore, a channel for data communication is reserved on demand, only when data traffic is exchanged. The specific link to data channel mapping is performed in mutual agreement between the two nodes forming the link. The protocol is designed, simulated, and implemented on a testbed for the specific case of wireless IEEE 802.11g compatible networks. Results prove the feasibility and quick response of the channel selection approach. In simulations using a 10 x 10 raster topology, the distributed channel selection protocol is shown to provide an optimal mapping between the wireless links and channels that are available for data transport.

Next, a basic concept for the estimation of the end-to-end throughput capacity of wireless multi-hop, multi-channel paths is developed. The technique estimates the links capacity of each individual link in the network, and disseminates the information through the network by piggybacking the estimations onto the routing protocol messages. Based on these estimations and information on the network topology and channel configuration, every node in the network is capable of calculating the throughput capacity of any end-to-end path in the network. Again, the protocol is implemented. In a test set-up, the capacity estimations of the developed protocol are compared with a reference capacity measurement, determined by flooding the network. Results indicate that the followed estimation approach approximates the reference results, while only causing a fraction of the overhead.

Even though these first two protocols perform their task in a fully automated way, the mesh nodes were implicitly assumed to have been manually configured prior to installation. For example, in order to be able to participate in a network, among other things, a correct IP address, security parameters and a wireless network interface configuration is required. However, configuring the settings of a wireless network is not a trivial task for someone with limited or zero knowledge of wireless networks.

Therefore, the focus of chapter four is on the development of an integrated solution enabling automatic deployment, expansion and management of wireless mesh networks. The developed mechanism allows wireless nodes to be added to a wireless mesh network in a secure way, without requiring physical access to the device. New nodes can be added to an existing mesh backbone directly from the box, by placing them anywhere within the coverage of the already available network and passing through a few steps on a single configuration interface. The entire configuration procedure is completed within seconds, allowing to install a secure wireless mesh network with minimal efforts. The different protocols used for the deployment technique do not only exist in theory, all subsystems are implemented on top of IEEE 802.11 technology, as such enabling experimental verification. The experiments prove the feasibility of the followed approach: in just minutes, a secure wireless mesh network with over twenty nodes is deployed effortlessly.

Both in the process of determining the issues with current generation wireless ad-hoc networks as while developing the architectural building blocks of the proposed mesh network, experimentally driven research complemented theoretical considerations and results obtained from simulation. This approach resulted in more robust wireless networking protocols that are able to cope with a wide range of issues caused by the unpredictable wireless environment. A proof of the successfulness of the approach is that the results in this book do not only exist on paper but can also be verified experimentally. However, performing experimentally driven research in wireless networks was found to be a complex task: ill-considered experimental research just as easily leads to wrong conclusions as thoughtless interpretation of simulation results. As a result, experimentally driven research may be a time-consuming and frustrating experience which is not necessarily rewarded with scientific output.

Therefore, based on the experience that was gathered during the realizations of the wireless mesh protocols in this book, a generic methodology for the design and deployment of wireless networking protocols is developed in a fifth chapter. The methodology is aimed at providing wireless network researchers with a guideline that will help them to avoid mistakes, produce more reliable protocols and performance analysis results with less effort, and increase the volume and quality of scientific output. As such, rather than studying a single aspect in depth, this chapter tackles a more basic aspect of wireless networking research. However, the development of such methodology is considered to be of great importance to guarantee the quality and feasibility of future wireless ad-hoc networking research.

Moreover, in order to illustrate the designed methodology, an extension to the wireless mesh auto-configuration solution is designed and implemented in this chapter. The extension allows to determine whether a certain geographical location is suitable for deploying of a new mesh node, by retrieving physical layer measurements from the wireless driver. Even though the solution is mainly intended as an illustration, it further simplifies the installation of a wireless mesh backbone and guarantees that high quality wireless links will be available.

In a final chapter, the most important conclusions of this work are summarized, and an outlook on the future of wireless mesh and wireless ad-hoc networks is given. By basing wireless ad-hoc research on realistic assumptions, and thanks to the realizations in this book, a bright future for wireless ad-hoc networks should not necessarily be far away.

1 Introduction

1.1 Digital wireless communication

Less than three decades ago, computer communication was only used by a small group of professionals and computer enthusiasts. Personal computers were relatively expensive and not as widespread as they are today. Under the impulse of improved electronics manufacturing processes, computer devices and peripherals became smaller, cheaper, and more powerful. Today, computers and electronic devices are omnipresent. With the global expansion of the Internet, broadband communication is playing an important role in the lives of a large and increasing number of people worldwide [1].

While many use the Internet only for recreational and social purposes such as playing games, watching videos or staying in contact with friends, others have developed a professional dependence on connectivity. For example, video conferences connect office rooms across the planet, calendars are exchanged and stored online, a wide range of databases are consulted remotely, business communication is organized through e-mail, the financial market receives orders through the Internet, and online trade platforms are organized to buy or sell goods.

The interplay between dependence on connectivity, device miniaturization and a combination of complex socio-economic factors leads to an increased demand for anytime anywhere connectivity: at work, at home, or on the road. As such, the recent success of wireless communication technologies does not come as a surprise. The most obvious example is wireless voice communication. While

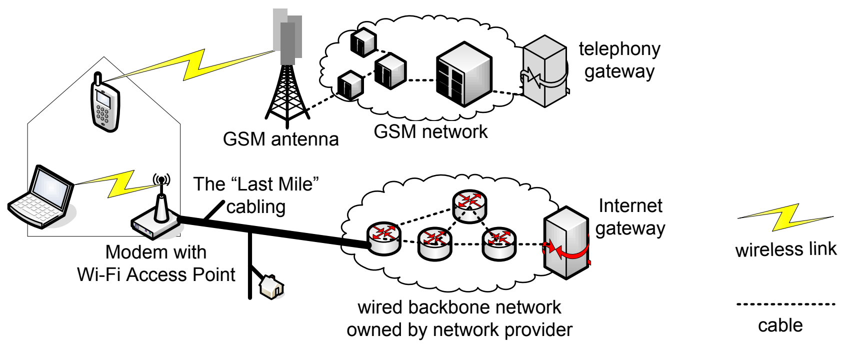


Figure 1.1: Conceptual overview of last-hop wireless links as used by GSM and Wi-Fi access point.

the first generation analog cellular telephony was introduced in Japan in 1979, discussions on the second generation digital GSM standard were only started in the eighties and led to the first commercial deployment in Finland in 1991. According to [2], there are over four billion GSM subscriptions today.

A second example is the popularity of wireless access points using Wi-Fi technology [3] to wirelessly connect a smartphone or laptop to the broadband connection at home or the office. While Wi-Fi chipset certification only started in 2000, over 475 million Wi-Fi chipsets were shipped in the year 2009 [4].

Figure 1.1 depicts how in these two examples, a single wireless connection is used to cover the last hop of the communication path to the end-user device: in the case of GSM, the wireless part of the communication is terminated at the receiving antenna. From that point on, the voice call is transferred over wires via a chain of control systems inside the GSM network core of the provider and will possibly continue through a gateway to another network provider until it reaches its destination. In the event that the receiving phone number is a mobile phone as well, the phone signal stays on the cable until it reaches an antenna close to the receiver: only the first and last hop in the communication path are wireless. Similarly, in case an access point is used to connect a Wi-Fi device such as a laptop to the Internet, only the link between the access point and the laptop is wireless.

In both network types, the end-user client entirely relies on the coordination capabilities existing within the wired part of the network: the wireless end-user device has to negotiate with the GSM infrastructure or the Wi-Fi access point respectively to access the network. In neither of these cases, communication between wireless end-devices is possible without support from the network infrastructure. Two cell phones never transmit directly to each other, and two Wi-Fi laptops always need to go through the access point in order to communicate with each other.

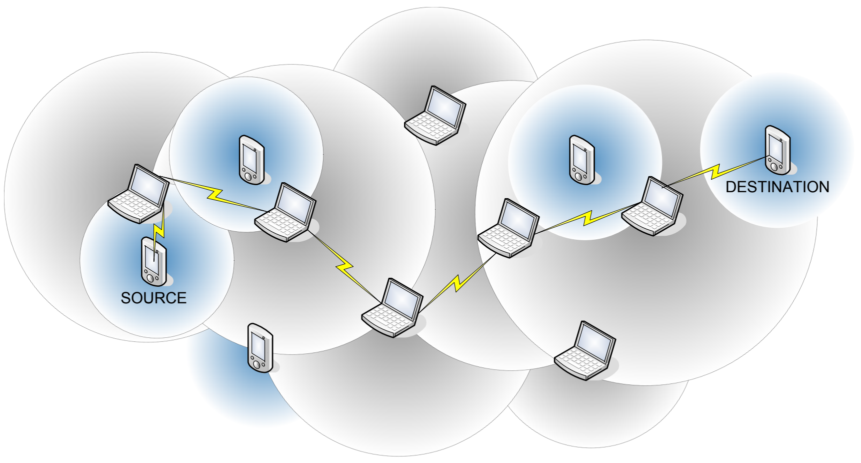


Figure 1.2: Wireless multi-hop ad-hoc network. Network nodes can act as clients and as routers.

1.2 Wireless ad-hoc networks

The above two examples indicate how conventional networks have been extended to support mobility by adding a wireless link to a previously available wired network infrastructure. In this dissertation, the family of wireless ad-hoc networks is studied. Wireless ad-hoc networks are networks between wireless devices or *nodes* that do not rely on any previously existing infrastructure [5]. Whenever two or more devices are within each other's transmission range, they detect each other's presence and automatically form a network. If two nodes are placed too far apart such that direct communication is not possible, intermediate nodes may act as a router and automatically form *multi-hop* paths to forward the messages towards their destination (cf. Fig. 1.2), thus forming multi-hop ad-hoc networks. Network nodes may be mobile and move around freely inside the network, triggering multiple changes in the network topology and requiring the network to constantly re-organize itself. While wireless multi-hop ad-hoc networks can exist in isolation, they may also be connected to other networks, such as the Internet. While 'wireless ad-hoc networking' does not necessarily imply a multi-hop character, the terms 'wireless multi-hop ad-hoc', 'wireless ad-hoc' and 'ad-hoc' will be used interchangeably in this dissertation for simplicity reasons.

Because multi-hop ad-hoc networks are able to operate in the absence of network infrastructure by relaying digital information over multiple wireless hops inside the network, the characteristics of these networks differ considerably from infrastructure wireless networks such as used in GSM or in Wi-Fi wireless local area networks (WLANs). In an idealized, dense wireless multi-hop ad-hoc network, networks between devices would simply ‘exist’, without the need for manual intervention. When new devices join the network, the network coverage would automatically be extended. There are many advantages to wireless ad-hoc networks: as no infrastructure such as expensive routers or cables is needed, and the digging and installation of cables is avoided, the costs that are associated with the roll-out are limited to the devices themselves. Furthermore, single points of failure in the network infrastructure are avoided. The self-organizing capabilities of the devices enable fast and flexible network installation and assure recovery from link breaks in the event a node is switched off, destroyed or moves to a new location. Since each ad-hoc node can act as a client or a router and helps to deliver information to nodes that are out of the communication range of a sending node, the required transmission range of a single node in dense ad-hoc networks is generally low. As such, several technologies operating in unlicensed spectrum such as Wi-Fi or Bluetooth are suited to support ad-hoc networks, avoiding the use of expensive radio licenses. Note that, while this enables end-users to install ad-hoc networks at home or at the office without needing to buy spectrum licenses, using unlicensed spectrum may not be the ideal approach for network operators; in return for the license cost, an operator has better control over the quality and use of wireless spectrum to which he has bought the exclusive rights, which may give him a large advantage over competitors [6].

1.2.1 Application scenarios and ad-hoc subtypes

The initial research on wireless ad-hoc networks was performed in 1972 through the Packet Radio Networks project, sponsored by the United States Department of Defense [7]. The goal of this research project was to enable packet switched networking in battlefield environments without the support of any infrastructure in a hostile environment by building a wireless network between the soldiers, vehicles and aircrafts. When laptop computers became popular in the 1990s, the interest in the ad-hoc networking concept for non-military applications grew stronger and resulted in the creation of the Mobile Ad-Hoc Networking (MANET) working group [8] within the Internet Engineering Task Force (IETF, [9]).

Ever since, wireless ad-hoc networks have gained increased attention from the international research community, and their applicability to diverse scenarios has been investigated:

- **Home, office, factory and commercial environments.** Ad-hoc networks may be used to exchange files between laptops during meetings or conferences, support multi player games, print documents using a wireless connection, collect environmental data such as temperature measurements for home automation or factory process automation, or enable wireless payments.

- **Emergency situations.** Efficient communication during emergency situations may save lives. Unfortunately, communication networks are not always available under emergency situations. For example, an earthquake or fire may have destroyed existing infrastructure. Under these circumstances, the auto-configuring and rapid deployment characteristics of ad-hoc networks might be used to quickly establish new communication possibilities.

- **Temporary networks.** When network connectivity is required during events such as city festivals or public fairs, network infrastructure might not be available. For example, suppose mobile electronic information signs are distributed across a town during an event, cables between these signs can be avoided using ad-hoc networks. Public safety service personnel such as police offices might also become part of the network, in order to allow them to communicate or adjust messages on the information signs.

- **Rural areas and third world countries.** Whenever no infrastructure is available in a certain area, ad-hoc networks might be used to provide interconnectivity. For example, the ‘One Laptop Per Child’ project [10] strives towards providing children in third world countries with a simple and robust laptop for educational purposes, which is equipped with a Wi-Fi interface configured in ad-hoc mode, to allow digital file distribution inside the classroom.

- **Network coverage extension.** In areas that are already serviced by other wired or wireless networks, ad-hoc networks can be used to extend network coverage in a cheap way. In addition, services may be added to a network. For example, if a single host in the ad-hoc network is connected to the Internet, this host might enable Internet service in the entire ad-hoc network.

Over the years, the diverse scenarios caused several ad-hoc subtypes to emerge. While all these network types are essentially wireless ad-hoc networks in the sense that they use wireless links and automatically detect neighbors and form a dynamic, self-healing network, a distinction is made based on the capabilities or role of the network nodes. As these subtypes grew organically over time, the definitions found in literature are not always uniform. Therefore, a selection of the most important wireless ad-hoc network sub-types and the definition as used in this dissertation follows:

Mobile ad-hoc network. The ‘mobile’ prefix indicates that strong mobility and frequent topology changes are expected. The term ‘Mobile ad-hoc network’ or MANET is often used as a synonym for wireless ad-hoc network.

Network type	Description	Typical research focus
mobile ad-hoc	Stress on node mobility.	organize and maintain connectivity between nodes.
wireless mesh	Hierarchic network. Mesh routers are powerful and have low or no mobility.	reliability, high performance.
wireless sensor	Small-size low-power devices, usually used in great numbers. Collect data from their environment.	energy efficiency, scalability.
vehicular	Networks between vehicles and road infrastructure.	high mobility, reliability and low delay.

Table 1.1: Wireless multi-hop ad-hoc network subtypes

Wireless mesh networks. Hierarchical ad-hoc network. *Mesh routers* are specialized and powerful ad-hoc nodes in terms of processing power and memory capacity. They have more powerful and possibly multiple wireless network interfaces, and have less energy restrictions as they are connected to an external power source or host system with few power restrictions such as a truck. As such, mesh routers, also often referred to as *mesh nodes*, have limited or zero mobility. Mesh routers are dedicated to the routing task and try to provide a robust wireless network backbone for the *mesh clients* [11]. Some mesh routers may function as gateway node and provide additional services such as Internet connectivity to the mesh clients. The coverage of a mesh backbone may be extended via an ad-hoc network between the mesh clients, thus forming a *hybrid wireless mesh network*, depicted in Figure 1.3.

Wireless sensor network. Wireless ad-hoc network, usually between a large number of low-cost, small-scale, low-power nodes with sensing abilities [12]. Sensor nodes typically collect data from their environment such as temperature, pressure, humidity or movement. The measured data is collected and used for further processing such as activation of cooling installations, monitoring nature preserves or road infrastructure. In *wireless sensor and actuator networks*, the sensor network is extended with actuator nodes in order to decrease the dependence on a processing infrastructure. *Body area networks* are networks formed by (bio-) medical sensors attached to the human body. Example applications are monitoring the health of a hospital patient or an athlete.

Vehicular networks. Wireless ad-hoc networks, where the nodes are mounted on vehicles or belong to the road infrastructure [13]. Example applications include

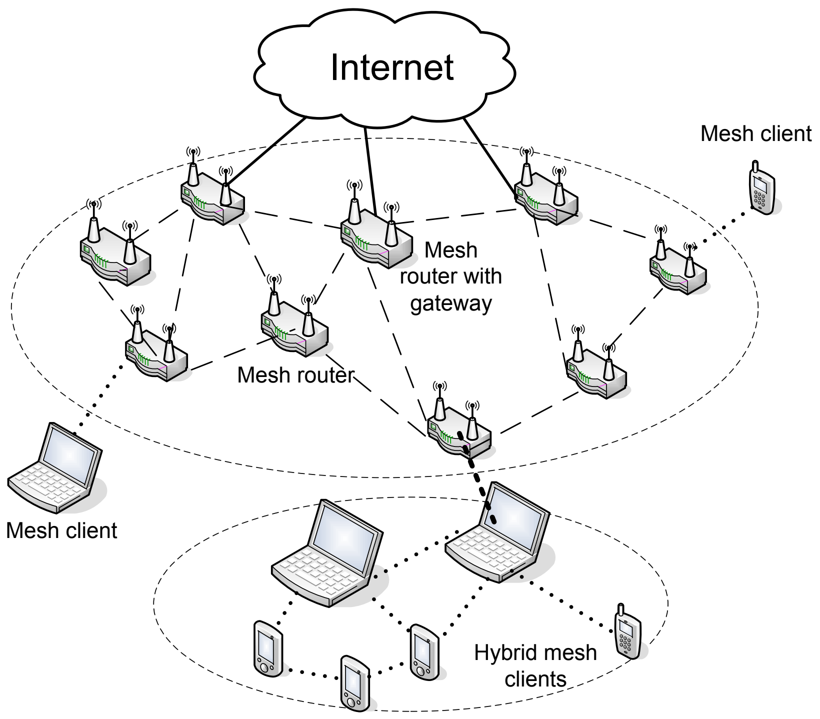


Figure 1.3: Logical overview of a hybrid wireless mesh network.

the monitoring of road condition, dynamic traffic light optimization, producing and disseminating warnings in case of road obstructions or dangerous road conditions, or sending warning messages to following vehicles and their passengers in case of sudden break manoeuvres or obstacles on the road. Vehicular networks face high node mobility. Highly reliable, low-delay networks are required.

These different network types and their most important characteristics are summarized in table 1.1.

1.2.2 Enabling technologies

Wireless ad-hoc networks can be built on top of any wireless technology supporting point to point links. It is not uncommon to investigate a networking or application layer aspect of ad-hoc networks without referring to a specific technology. However, because of government regulations on radio spectrum use, most researchers resort to technologies operating in the industrial, scientific and med-

Technology	Max. data rate	Typ. range	Typ. field of application
IEEE 802.11b	11 Mbps	50 - 100 m	wireless LAN
IEEE 802.11g	54 Mbps	50 - 100 m	wireless LAN
IEEE 802.11a	54 Mbps	15 - 30 m	wireless LAN
IEEE 802.11n	600 Mbps	70 - 150 m	wireless LAN
Bluetooth v2.1	3 Mbps	10 - 100 m	cable replacement
IEEE 802.15.4	250 kbps	10 - 100 m	low power, low data rate

Table 1.2: Supporting technologies for ad-hoc networks operating in ISM bands for WLANs and PANs (non-exhaustive list).

ical (ISM) radio bands. Provided transmission power stays below certain region dependent limits, the ISM band allows unlicensed radio transmissions. The ISM band includes the frequency ranges 2.400 - $2.500\,GHz$ and 5.725 - $5.875\,GHz$. While these frequencies are sufficiently high to enable relatively fast digital data communication, the signal attenuation is relatively limited compared to higher frequencies, providing a balance between range and speed. Unfortunately, the 2.400–$2.500\,GHz$ range is also used by many other products such as cordless phones, baby monitors or microwave ovens, degrading the channel quality by causing interference.

Table 1.2 gives an overview of technologies operating in ISM bands that are currently commonly used for ad-hoc networking research in WLANs and WPANs (Wireless Personal Area Networks). WLANs typically cover a single house or a part of an office building, while WPANs provide short range wireless links, for example replacing cables when sending a picture directly from a digital camera to a printer device. The maximum data rate is specified at physical layer and is not entirely available to upper layer applications. Typical ranges are approximate indoor ranges obtained using omnidirectional antennas, and depend on the physical data rate and output transmission power in use and on the characteristics of the environment.

Many communication standards have been developed by working groups within the Institute of Electrical and Electronics Engineers (IEEE). These communication standards are in constant evolution, and new amendments are frequently specified. As a first example, the IEEE 802.11p draft amendment [14] is currently approaching its approval as a standard for providing wireless access in vehicular environments. Second, the IEEE is planning a new IEEE 802.11ac [15] standard to be studied in 2012. IEEE 802.11ac aims at supporting gigabit wireless links; few details are currently available as to the specific strategy that will be followed. An interesting overview of the IEEE 802.11 family standards is found in [16].

Component	Research topic
Wireless medium	- regulatory aspects related to the RF spectrum
Antenna	- antenna design, antenna arrays - antenna diversity
Hardware and Radio chip	- chip design - low-energy technologies - energy harvesting - adaptive modulation and coding - advanced signal processing - cognitive radio
PHY / MAC	- automatic transmission power schemes - forward error correction (FEC) schemes - rate adaptation strategies - fair channel access, TDMA, priority MAC - collision avoidance - multi-interface, multi-channel networks - link aggregation - packet aggregation - link layer retransmission strategies - neighbor detection
Routing	- neighbor detection - proactive, reactive, hybrid routing protocols - advanced routing metrics - cluster based routing, geographic routing - cooperative routing - mobility support - self configuration
Transport	- end-to-end retransmission strategies - flow control - protocol boosters
Application	- usability - service discovery - middleware for wireless application developers - location based services
Other / stackwide	- quality of service - network architectures - simulators and testbeds - cross-layer protocols - security - techno-economical aspects

Table 1.3: Overview of wireless ad-hoc networking research topics.

1.2.3 Technical challenges and research topics

Whichever application or subtype is studied, developing wireless ad-hoc network protocols is a challenging task. Protocols that were developed for wired networks can often not be reused unmodified because of several reasons.

Most importantly, the transition from a cabled solution to a radio channel has a severe impact on the reliability and quality of data transmissions: *(i)* wireless communication links are characterized by a large bit error rate (BER) compared to wired solutions; *(ii)* only a limited bandwidth is available; *(iii)* because the wireless medium is shared by multiple nodes within a single collision domain, the available bandwidth per node is further reduced; *(iv)* a single radio cannot transmit and receive at the same time, therefore unexpected collisions frequently occur. These collisions may be falsely interpreted by upper layer protocols as congestion; *(v)* traditional protocols might not be able to cope with frequent link breaks caused by failing transmissions, failing nodes or node mobility; *(vi)* rogue or faulty nodes may generate a large amount of traffic, blocking access to the wireless medium for all other nodes. As such, no hard guarantees can be given on the quality of service (QoS). Precautions are needed as to avoid misbehaving nodes from disturbing the network, for example by injecting illegal packets into the network, or by acting as a black hole instead of forwarding packets.

Secondly, because ad-hoc networks do not rely on any centralized infrastructure, distributed protocols are needed whenever possible. A special coordinating role may dynamically be attributed to one or multiple nodes in the network in order to provide functionalities that are traditionally implemented in a centralized way. For example, many cluster algorithms for ad-hoc networks exist [17] in which certain nodes are elected as cluster heads and are responsible for synchronization and configuration of their child nodes. Even in these cases, network nodes should be able to switch roles in case cluster heads fail or leave the network.

In order to deal with these complexities, researchers have tackled issues at all layers of the OSI (Open System Interconnection, [18]) stack. An non-exhaustive overview of research topics, structured based on the different communication layers and components between ad-hoc sender and receiver is presented in Table 1.3. As for the wireless medium, research is ongoing on which RF (Radio Frequency) spectrum sharing policies are best suited in order to support current and future wireless network environments [19]. Hardware designers are continuously improving manufacturing processes and chip designs. These advances eventually result in improved wireless communication technologies and wireless ad-hoc nodes, leading to new opportunities such as energy harvesting circuits like [20] or cognitive radio strategies [21] that may help to support future ad-hoc networks.

Researchers investigating PHY (physical) and MAC (medium access control) layer functionality design specifications, algorithms and protocols which drive the hardware components in such way that transmission and reception of data is enabled or optimized. The development of ad-hoc routing protocols is a research topic traditionally receiving much attention. This research eventually led to two proactive and two reactive routing protocols to be published in IETF request for comment (RFC) protocol description documents. At the cost of a constant signaling overhead, the proactive routing protocols, OLSR [22] and TBRPF [23], actively maintain an up-to-date overview of the routes available in the network. They are best fit for use in wireless mesh networks, having a relatively large available bandwidth. The reactive routing protocols AODV [24] and DSR [25] discover routes in the network only when a specific network path is needed, hereby reducing overhead at the cost of communication delays. Notwithstanding the IETF standardization efforts, researchers are still actively developing new routing protocols or modifying the above protocols to suit their specific needs.

At the transport layer, researchers mainly seek to provide more reliable communication to the upper layer protocols through studying the TCP (Transmission Control Protocol, [26]) behavior on top of wireless links [27] and proposing modifications. At the application layer, end-user and supporting applications are designed and usability research of those applications is performed.

Finally, other stackwide aspects such as wireless ad-hoc network architectures, wireless simulators [28] and wireless testbeds are studied, as well as techno-economic aspects [29] of wireless ad-hoc networks.

1.3 Outline and research contributions

In this introduction, it was stated that wireless ad-hoc networks have been the topic of international research for over thirty years. Furthermore, electronic devices such as laptops, PDAs and smartphones are widely available and are often equipped with one or multiple wireless interfaces with the enabling technologies. A wide range of useful applications that could apply to people around the world can be imagined.

Despite these observations and the plethora of advantages attributed to wireless ad-hoc networks networks –cheap and automatic deployment, resistance to network failures, ability to cope with network dynamics– wireless multi-hop ad-hoc networks are hardly ever used in our daily lives. Although through the years, the state of the art in wireless ad-hoc networks has incrementally advanced, the large amount of time and money invested by the research community currently did not lead to a proportional benefit.

In this work, the factors contributing to this discrepancy are analyzed. Furthermore, solutions are proposed and evaluated in order to make the use of ad-hoc networks more attractive. The three main goals of this work are summarized as follows;

(i) To evaluate the validity of common wireless ad-hoc networking assumptions on small-scale testbeds built using a selection of commercially available hardware and Wi-Fi interfaces. To identify and overcome practical issues when building ad-hoc networks on top of this hardware.

(ii) To develop and evaluate practically feasible wireless ad-hoc networking protocols for the use of multi-channel communication and end-to-end throughput capacity estimation on top of commercially available hardware and Wi-Fi interfaces. To design and build an integrated solution for the deployment and auto-configuration of wireless mesh networks.

(iii) To support future researchers in wireless networks by presenting an overview of techniques for the evaluation of wireless networking protocols. To share hints, techniques and tools which may help researchers to learn from mistakes that were made and from experiences that were gathered during the realization of this dissertation.

To this end, in a first step of Chapter 2, the validity of assumptions that are traditionally used in many ad-hoc research papers is verified through small-scale experimental test set-ups using a selection of Wi-Fi based equipment. Theoretical assumptions concerning wireless signal propagation, transmission power and the use of multiple wireless network interfaces in a single device show to be false during these selected experiments. Moreover, wireless hardware of different vendors is shown to produce heterogeneous results. Furthermore, several non-technical issues are revealed. The observations then lead to the definition of a wireless mesh network architecture with cross-layer optimizations.

In a next step, several building blocks of the architecture are designed and evaluated, both using simulations and by Wi-Fi based implementations. First, in Chapter 3 a protocol enabling on-demand distributed channel selection in wireless multi channel, multi interface mesh networks is designed, programmed, simulated and tested in a small-scale IEEE 802.11g testbed. In contrast with other channel selection approaches at the time of implementation, the protocol does not make use of long-term profiles nor short term measurements, but is based on the exchange of channel status messages between both nodes participating in a wireless link.

A second protocol is then constructed and evaluated which allows end-to-end throughput capacity estimation of a wireless link in a multi-channel environment in a distributed way. The design and evaluation of this technique was done in close collaboration with Dries Naudts, who also took care of the implementation in the scope of the IBBT-GeoBips project. The technique is evaluated using custom-built portable Wi-Fi based multi-interface devices.

Next, in order to enable full automatic configuration of wireless mesh networks, an integrated wireless mesh solution which enables the deployment, expansion and management of a secure wireless mesh network with minimal user intervention is developed in Chapter 4. The focus of this chapter is on the design and development of bootstrapping protocols needed to integrate new mesh nodes directly from the box at any location within an existing mesh network, without requiring a direct wired or wireless link to the new device. This way, the coverage of the network can gradually be extended. All subsystems of the solution are implemented, resulting in a fully functional mesh system which is then evaluated on a Wi-Fi testbed. While new subsystems such as a modular node initialization approach, single-hop and multi-hop node initialization and profile exchange strategies, reliability mechanisms and a GUI (graphical user interface) are designed from scratch, other subsystems of the implementation are based on (auto-configuration of wireless ad-hoc interface, neighbor discovery) or reused from (certificate exchange after node initialization, layer 3 packet encryption, auto-address generation) the VPAN framework developed by Jeroen Hoebeke [30]. The graphical user interface and interaction with the mesh nodes was realized in collaboration with Kristof Willemyns. The routing core and client supporting functions of the mesh network are not discussed in this dissertation, as they were developed in cooperation with, and implemented by Nicolas Letor (UA-PATS) in the scope of the IBBT-DEUS project.

Then, in Chapter 5, based on the experiences gathered during the realization of the wireless mesh protocols, a generic methodology for the design and deployment of wireless networking protocols is developed and presented. An overview of evaluation techniques for wireless networking protocols is given to help wireless protocol researchers in finding the most suited performance analysis technique for their research goal. For those researchers considering experimentally driven research, the methodology helps to obtain more results of better quality in a faster way. The methodology is illustrated by implementing an extension to the previously developed auto-configuration approach. By retrieving wireless signal quality parameters from the wireless driver, the extension allows to estimate the suitability of mesh node deployment locations with respect to obtaining robust links in the wireless backbone, as new wireless mesh nodes are added to a network.

While the contributions in this dissertation are essentially applicable to wireless ad-hoc network devices operating on multiple ad-hoc enabling technologies, application examples and implementations primarily focus on networks based on the IEEE 802.11a/b/g standard. The reason for this choice is the wide availability and relative maturity of devices with IEEE 802.11a/b/g based Wi-Fi interfaces. As such, Wi-Fi based products are an interesting choice for integrators, as they may hit the market in a relatively short time frame. Moreover, the data rates supported by Wi-Fi technologies are adequate to enable a wide range of services. As such, in this dissertation, IEEE 802.11a/b/g is considered to be an ideal technology to get ad-hoc networks one step closer to an everyday reality.

1.4 Publications

1.4.1 A1: Publications indexed by the ISI Web of Science 'Science Citation Index'

[1] **Stefan Bouckaert**, Eli De Poorter, Benôit Latré, Jeroen Hoebeke, Ingrid Moerman, Piet Demeester, "*Strategies and challenges for interconnecting wireless mesh and wireless sensor networks.*", Wireless Personal Communications, Springer Netherlands,53(3):443-463, March 2010.

[2] **Stefan Bouckaert**, Jeroen Hoebeke, Kristof Willemyns, Ingrid Moerman, Piet Demeester, "*An Easy Deployable Wireless Mesh Network.*", in submission phase.

[3] Eli De Poorter, **Stefan Bouckaert**, Ingrid Moerman, Piet Demeester, "*Non-intrusive Aggregation in Wireless Sensor Networks*", Under review in Ad Hoc Networks Journal (Elsevier)

[4] **Stefan Bouckaert**, Ingrid Moerman, Piet Demeester, "*A methodology for experimentally driven wireless networking research.*", in submission phase.

1.4.2 A2: Other International Journal Publications

[1] **Stefan Bouckaert**, Dries Naudts, Ingrid Moerman, Piet Demeester., "*Making Ad Hoc Networking a Reality: Problems and Solutions.*", Journal of Telecommunications and Information Technology, pages 3-11, Vol 2008/1.

1.4.3 P1: Publications indexed by the ISI Web of Science 'Conference Proceedings Citation Index'

[1] Dries Naudts, **Stefan Bouckaert**, Johan Bergs, Abram Schoutteet, Ingrid Moerman, Piet Demeester, "*A wireless mesh monitoring and planning tool for emergency services*", E2EMON'07: Fifth IEEE/IFIP Workshop on End-to-end Monitoring Techniques and Services, pages 31-36, Munich, May 2007.

[2] **Stefan Bouckaert**, Nicolas Letor, Chris Blondia, Ingrid Moerman, Piet Demeester, "*Distributed on demand channel selection in multi channel, multi interface wireless mesh networks.*", 50th IEEE Globecom conference, pages 5086-5091, Washington DC, USA, November 2007.

[3] Nicolas Letor, **Stefan Bouckaert**, Chris Blondia, Ingrid Moerman, Piet Demeester, "*A cluster driven channel assignment mechanism for wireless mesh networks.*", Proc. Second IEEE International Workshop on Enabling Technologies and Standards for Wireless Mesh Networking (MeshTech 2008) in

conjunction with IEEE MASS 2008, pp.659-665, Atlanta, GA, USA, September 29 - October 2 2008.

[4] **Stefan Bouckaert**, Eli De Poorter, Pieter De Mil, Ingrid Moerman, Piet Demeester, "*Interconnecting Wireless Sensor and Wireless Mesh Networks: Challenges and Strategies.*", IEEE Globecom 2009, Honolulu, HI, USA, November 30 - December 4 2009.

1.4.4 Other publications in international and national conferences

[1] **Stefan Bouckaert**, Johan Bergs, Dries Naudts, Erik De Kegel, John Baekelmans, Erik De Kegel, Chris Blondia, Ingrid Moerman, Piet Demeester "*A mobile crisis management system for emergency services: from concept to field test.*", Proceedings of the WiMeshNets06 workshop, the First International Workshop on "Wireless mesh: moving towards applications", part of the third International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine2006), Ontario, Canada, August 2006.

[2] **Stefan Bouckaert**, Ingrid Moerman, Piet Demeester, "*Cross-layer architecture and optimizations in hybrid wireless mesh networks.*", Proceedings of the 7th UGent PhD symposium, Gent, Belgium, November 2006.

[3] **Stefan Bouckaert**, Dries Naudts, Ingrid Moerman, Piet Demeester. "*Problems when realizing ad hoc networks: how a hierarchical architecture can help.*", International Multiconference on Computer Science and Information Technology, Wisla, Poland, October 2007.

[4] Eli De Poorter, **Stefan Bouckaert**, Ingrid Moerman, Piet Demeester, "*Broadening the concept of aggregation in wireless sensor networks.*", the 2nd International Conference on Sensor Technologies and Applications, pages 419-428, Cap Esterel, France, August 2008.

[5] **Stefan Bouckaert**, Eli De Poorter, Benoit Latré, Jeroen Hoebeke, Ingrid Moerman, Piet Demeester, "*Global routing in heterogeneous wired/wireless ecosystems.*", Strategic Workshop On Wireless Innovation for InterDynamic TecHnology (WIDTH) (SW), Rebild, Denmark, 15-17 May 2009.

[6] **Stefan Bouckaert**, Bart Jooris, Ingrid Moerman, Piet Demeester, "*Topology Control and Manipulation in a Large Scale Wireless Network Testbed.*" Proceedings of the 7th UGent PhD symposium, Gent Belgium, November 2009.

[7] **Stefan Bouckaert**, Wim Vandenberghe, Bart Jooris, Ingrid Moerman, Piet Demeester, "*The w-iLab.t testbed*", Tridentcom '10 conference. Accepted

for publication in the Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Springer, 2010.

[8] David Plets, Wout Joseph, Kris Vanhecke, Emmeric Tanghe, Luc Martens, **Stefan Bouckaert**, Ingrid Moerman, Piet Demeester, "*Validation of Path Loss by Heuristic Prediction Tool with Physical-Layer and Link-Layer Measurements*", submitted to APS 2010.

References

[1] OECD. *OECD Broadband Portal*. http://www.oecd.org/sti/ict/broadband.

[2] GSM Association. *GSM Association*. http://www.gsmworld.com/.

[3] WiFi Alliance. *Home Page*. http://www.wi-fi.org/.

[4] In-Stat. *Home Page*. http://www.instat.com/.

[5] Chai-Keong Toh. *Ad hoc mobile wireless networks: protocols and systems*. Prentice Hall PTR, Upper Saddle River, New Jersey, USA, 2002.

[6] Lehr, W. H. & Pupillo, L. M. *The Role of Unlicensed in Spectrum Reform*, page 169. 2009.

[7] Ram Ramanathan and Jason Redi. *A brief overview of ad hoc networks: challenges and directions*. IEEE Communications, 40(5):20–5422, 2002.

[8] MANET Working Group. http://www.ietf.org/html.charters/manet-charter.html.

[9] IETF. *Home Page*. http://www.ietf.org/.

[10] One Laptop Per Child project. *Home Page*. http://laptop.org/en/.

[11] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. *Wireless mesh networks: a survey*. Computer Networks, 47(4):445–487, 2005.

[12] Ian F. Akyildiz, Tommaso Melodia, and Kaushik R. Chowdhury. *A survey on wireless multimedia sensor networks*. Computer Networks, 51(4):921 – 960, 2007.

[13] Hannes Hartenstein and Kenneth P. Laberteaux. *A tutorial survey on vehicular ad hoc networks*. Communications Magazine, IEEE, 46(6):164 –171, June 2008.

[14] *IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments*, May 2008.

[15] IEEE Task Group AD. *Meeting update*. http://www.ieee802.org/11/Reports/tgad_update.htm.

[16] Guitdo Hiertz, Dee Denteneer, Lothar Stibor, Yunpeng Zang, and Bernhard Walke. *The IEEE 802.11 Universe*. IEEE Communication Magazine, pages 62–70, Jan 2010.

[17] Bo Han and Weijia Jia. *Clustering wireless ad hoc networks with weakly connected dominating set*. J. Parallel Distrib. Comput., 67(6):727–737, 2007.

[18] *Blue book*. Recommendations. IUT, Geneva, 1989. 9th CCITT Plenary Assembly, Melbourne, Australia, 14 Nov 1988.

[19] Jon M. Peha. *Emerging Technology and Spectrum Policy Reform*. In Proceedings of United Nations International Telecommunication Union (ITU) Workshop on Market Mechanisms for Spectrum Management, January 2007.

[20] Bin Yang, Chengkuo Lee, Wenfeng Xiang, Jin Xie, Johnny Han He, Rama Krishna Kotlanka, Siew Ping Low, and Hanhua Feng. *Electromagnetic energy harvesting from vibrations of multiple frequencies*. Journal of Micromechanics and Microengineering, 19(3):035001–+, March 2009.

[21] Bruce Fette. *Cognitive Radio Technology*. Elsevier, Linacre House, Jordan Hill, Oxford, UK, 2006.

[22] Thomas Clausen, Christopher Dearlove, Philippe Jacquet, The OLSRv2 Design Team, and The MANET Working Group. *Optimized Link State Routing Protocol (OLSR) version 2*. Ietf draft, Internet Engineering Task Force, April 2010.

[23] Richard Ogier, Fred Templin, and Mark Lewis. *Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)*. RFC Experimental 3684, Internet Engineering Task Force, February 2004.

[24] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC Experimental 3561, Internet Engineering Task Force, July 2003.

[25] David B. Johnson, David A. Maltz, and Yih-Chun Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*. RFC Experimental 4728, Internet Engineering Task Force, February 2007.

[26] Information Sciences Institute. *Transmission Control Protocol*. Ietf rfc 3174, IETF, 1981.

[27] Karthikeyan Sundaresan, Vaidyanathan Anantharaman, Hung-Yun Hsieh, and Raghupathy Sivakumar. *ATP: a reliable transport protocol for ad-hoc networks*. In MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing, pages 64–75, New York, NY, USA, 2003. ACM.

[28] William Kasch, Jon Ward, and Julia Andrusenko. *Wireless network modeling and simulation tools for designers and developers*. Communications Magazine, IEEE, 47(3):120–127, March 2009.

[29] Mark Birchler, Peter Smyth, Georges Martinez, and Mike Baker. *Future of Mobile and Wireless Communications*. BT Technology Journal, 21(3):11–21, 2003.

[30] Jeroen Hoebeke. *Adaptive Ad Hoc Routing and Its Application to Virtual Private Ad Hoc Networks*. PhD in Computer Engineering Science, Ghent University, IBCN - INTEC, B-9050 Ghent, Belgium, 2007.

2 Wireless Ad-Hoc Networking Issues

2.1 Introduction

Despite of the wide range of application scenarios, few wireless ad-hoc network deployments are currently in use. In this chapter, an analysis is made of the factors contributing to this limited success. The analysis encompasses both technical and non-technical aspects: while several issues arise from misinterpretations essentially related to the wireless medium as a communication channel, other problems are a consequence from voids in the current research spectrum. Part of these voids are explained by the hype that was built around the wireless ad-hoc networking concept, which caused initial design goals to be confused with available features.

Traditionally, most wireless networking protocols have been developed with the OSI layer model in mind. The OSI model reduces the complexity of a network stack by grouping functionalities in *communication layers* and standardizing interfaces between them. However, a strictly layered design where adjustments are made to one layer without knowing the inner functioning of other layers may not be the best approach when developing wireless networking protocols.

The major reason is that wireless networks perform less in terms of throughput, delay and reliability compared to wired networks. As such, the choices made at a higher layer of the OSI stack may have a relatively higher influence on lower layer performance. For example, since each IEEE 802.11 packet suffers from considerable MAC overhead, an application generating a large number of small size packets such as a VoIP application may fully occupy the wireless medium notwith-

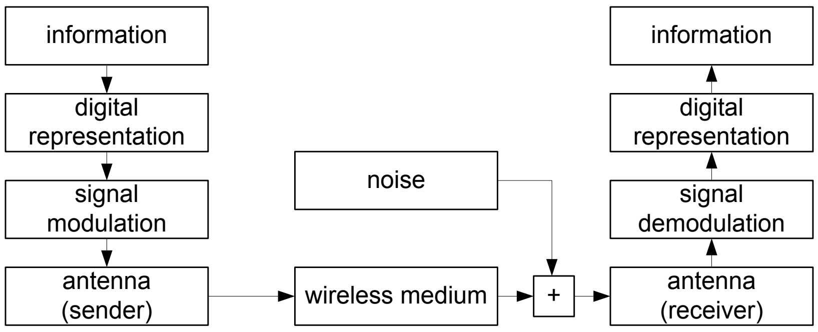


Figure 2.1: Block scheme of a digital wireless communication system.

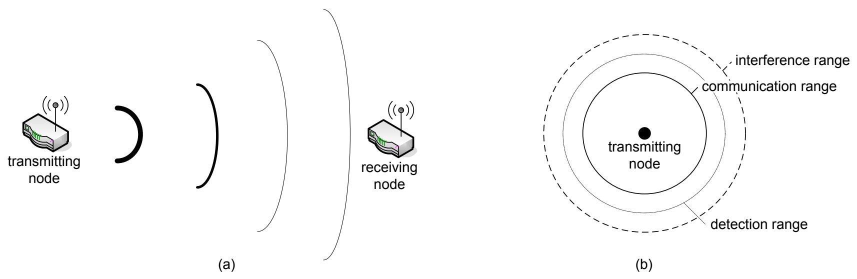


Figure 2.2: (a) Conceptual illustration of radio wave propagation. (b) Illustration in top view.

standing an essentially low application layer data rate [1].

Several authors indicate that performance gains can be obtained by adopting a so called *cross-layer* design [2–5]. In cross-layer design, the interdependence between different layers is recognized and the layered design is intentionally violated by letting parameters at one layer influence the behavior of other layers.

In order to understand the impact of specific choices at one layer of the network stack to the behavior of the entire system in general, at least a basic theoretical and practical understanding of wireless signal propagation and terminology is advisable.

2.2 Wireless signal propagation

2.2.1 General

Figure 2.1 shows a typical block scheme of a digital wireless communication system. Whenever information is transported from source to destination, it is converted to a digital representation in the form of a bit stream. The digital bit stream is then used to modulate a carrier signal, which is broadcast by an antenna. As the signal propagates through the wireless medium, the perceptible signal strength is reduced and noise from other radio sources is added. Thus, the signal arriving at the receiving antenna differs from the signal that was sent. If the signal is sufficiently strong and recognizable at the receiving side, it is decoded and the information is received.

In 1946, Friis derived the relation between the power available at the transmitting and receiving antennas under idealized conditions in unobstructed free-space [6]. When the receiving and sending antenna are ideally aligned and polarized, the relation between the transmitted power P_t and received power P_r equals:

$$P_r = P_t \cdot G_r \cdot G_t \cdot \left(\frac{\lambda}{4\pi R}\right)^2$$

In this equation, G_r and G_t represent the *antenna gains* at the receiver and the transmitter and are used to account for antenna directivity, λ is the wavelength of the wireless signal and R the distance between sender and receiver. The inverse of the factor in parenthesis, $\left(\frac{4\pi R}{\lambda}\right)^2$, is called the *free-space path loss* and shows that under these idealized conditions, the power of a wireless signal degrades proportionally to the square distance between sender and receiver. This is illustrated in Figure 2.2a. Note that the path loss is easily rewritten as $\left(\frac{4\pi R f_c}{c}\right)^2$, with c the speed of light, indicating that the path loss is higher for higher communication frequencies f_c.

A receiving node is able to decode a wireless signal if the ratio between the received signal power and noise signals SNR_r exceeds a threshold signal to noise ratio SNR_{com}. The latter value is called the communication signal to noise ratio (SNR) threshold and varies depending on the specific radio technology and physical layer modulation scheme (thus, data rate) in use. However, even if a signal is too weak and cannot longer be decoded because the sender is too far away, the signal still causes interference at a receiving node if the received power exceeds an interference threshold P_{IF}. As a result, two ranges are typically attributed to sending nodes: communication is possible to all nodes within *communication range*, and the radio transmission is causing interference at all nodes within *interference range*. Some authors define a third radio range [7], called *detection range*. The detection range is defined as the range within which the received power is still large enough to distinguish the signal from background noise, but communication is no longer possible due to a high error rate. In the remainder of this dissertation, this refinement is not considered. The different ranges are illustrated in Figure 2.2b.

In order for a sending and receiving node to be able to decode each other's messages and distinguish them from background noise, they need to agree on the digital format of the encoded message. Furthermore, a common protocol is needed to define how to access the wireless medium without disturbing messages sent by other nodes. The family of IEEE 802.11 standards contains such specifications; since 1999, IEEE 802.11 based technology is being certified by the Wi-Fi alliance [8] as to guarantee interoperability between devices of different vendors. Today, Wi-Fi certified products have become the de facto choice to organize WLANs. Although technically not entirely accurate, the terms 'IEEE 802.11' and 'Wi-Fi' are used as synonyms by many people.

2.2.2 IEEE 802.11

As previously stated, the IEEE 802.11 based Wi-Fi technologies are used by many researchers worldwide and in this work as the basic technology for the development of wireless ad-hoc research. Having been around for several years, the IEEE 802.11a/b/g versions have grown to be the most popular subversions.

While it is not within the scope of this chapter to provide a full description, some basic concepts help to provide a better understanding on the origin of certain issues that are observed in current generation ad-hoc networks. For all technical details on the IEEE 802.11 medium access control and physical layer specifications, the reader is referred to [9].

As mentioned in the introduction, a radio cannot receive and transmit at the same time. As such, a wireless node cannot verify if the wireless medium is free while sending a packet and thus cannot detect collisions. Therefore, the MAC layer of the IEEE 802.11 standards implements a Carrier Sense Multiple Access with Collision Avoidance mechanism (CSMA/CA) through the *distributed coordination function* (DCF). Before a packet is transmitted, the radio senses the wireless medium for activity. If the node detects the medium to be busy because it is within the interference range of another node, the transmission is deferred to a later time; in this case, the packet will only be transmitted if the medium was sensed to be available longer than a period of time randomly generated within the limits of a *contention window* (CW). While this mechanism works well under low traffic load conditions, it cannot detect collisions that are caused when two nodes, which are located out of each other's interference range, transmit a packet to a common third node. This problem is known as the hidden node problem. A similar issue, called the exposed node problem, exists in which packets are delayed unnecessary. Both problems are described in [10].

To overcome the hidden node problem, the optional RTS/CTS (Request To Send/Clear To Send) mechanism was added to the IEEE 802.11 standards. When the mechanism is activated, prior to transmitting a packet, a node will send a broadcast RTS message, informing every node within its transmission range of an upcoming transmission to the destination node. In its turn, the destination node answers this message with a CTS message, which is received by all its neighbors

Technology	PHY layer data rates (Mbps)	Frequency range
IEEE 802.11b	1, 2, 5.5, 11	$2.400\,GHz$
IEEE 802.11g	1, 2, 6, 9, 12, 18, 24, 36, 48, 54	$2.400\,GHz$
IEEE 802.11a	6, 9, 12, 18, 24, 36, 48, 54	$5.000\,GHz$
IEEE 802.11n	*(i)*: 6, 13, 19.5, 26, 39, 52, 58.5, 65 *(ii)*: 13, 26, 39, 52, 78, 104, 117, 130 *(iii)*: 13.5, 27, 40.5, 54, 81, 108, 121.5, 135	$2.400\,GHz$ and $5.000\,GHz$

Table 2.1: Supporting technologies for ad-hoc networks operating in ISM bands. Rates of 802.11n when using guard intervals of 80 nanoseconds and equal modulation for each stream. (i) $20\,MHz$ *channel width, basic rates. (ii)* $20\,MHz$*, 2 streams. (iii)* $40\,MHz$ *channel width, single stream. A maximal IEEE 802.11n PHY rate of up to 600Mbps is possible in some configurations. [12]*

within transmission range. Nodes receiving an RTS and/or CTS message should refrain from sending data for a period of time included in the RTS and CTS messages. This time period is chosen in such way that the wireless medium is available for a sufficiently long period of time as to increase the chances of error-free transmission of the packet that triggered the RTS request. The RTS/CTS mechanism has been criticized for being less efficient in achieving its goals than would appear, failing to solve the hidden station problem in some situations while unnecessary blocking transmissions that could take place under standard CSMA/CA operation [11]. Therefore, and in order not the increase the analysis complexity, the use of RTS/CTS is not considered in the remainder of this dissertation, and it is never enabled during the evaluation of protocols.

Because a successful transmission does not automatically result in a successful reception of a packet, a MAC layer acknowledgment (ACK) and retransmission system is incorporated into the standard: on successful reception of a unicast frame, a receiving node immediately confirms the packet via an ACK-packet. If the sending node does not receive such packet shortly after transmission, the transmission is repeated by the MAC layer for a predefined number of times or until a corresponding ACK packet is received.

The IEEE 802.11a/b/g/n standards support multiple physical layer data rates and allow communication using different center frequencies. An overview is found in Table 2.1. The $2.400\,GHz$ ISM range used by the IEEE 802.11b/g protocols is divided into 13 (14 in Japan) channels with a width of $22\,MHz$, spaced $5\,MHz$ apart. The center frequency of channel 1 and channel 13 are $2.412\,GHz$ and $2.472\,GHz$ respectively. As a result, only 3 channels (e.g. 1, 6 and 11) can be operated simultaneously without any theoretical channel overlap.

The $5\,GHz$ frequency band used by IEEE 802.11a is wider, leading to 13 non-overlapping channels to be available in most countries of the world. Unfortunately, the higher communication frequency leads to higher signal attenuation, making IEEE 802.11a less suited for non-line-of-sight communication or communication at greater distances.

Finally, the IEEE 802.11n standard may operate in both the $2.400\,GHz$ and $5.000\,GHz$ ranges, either using channel widths of $20\,MHz$ or $40\,MHz$. By using multiple antennas and advanced decoding strategies, several simultaneous wireless streams may be operated at the same time by the same nodes.

2.3 Multi-Hop Communication

2.3.1 Single-interface wireless networks

Since wireless multi-hop ad-hoc networks do not rely on existing infrastructure, packets are forwarded over multiple wireless hops in between the different network nodes. Nodes can only communicate if they are tuned to the same wireless channel during the transmission. Thus, if the nodes in the network have only a single network interface, all interfaces should be configured to the same channel in order to enable communication. Although the requirement of using a single channel may be overcome by implementing channel hopping solutions such as [13], this approach is non-standard in Wi-Fi based networks and is not further considered in this dissertation.

Within a single collision domain, only one radio may transmit at any given time. If two nodes are transmitting simultaneously, the radio of the receiver observes the added signals, usually leading to a signal which cannot be demodulated. Consider the example of Figure 2.3 where two packets are sent from source node to destination node over a three hop path with IEEE 802.11g compatible radios. Assume that all nodes are within each other's interference range, and each node is able to communicate to its neighboring node at the highest possible communication rate such that the link bandwidth B_{max} is maximized. To get a single packet from source to destination in the absence of transmission errors, three packet transmissions and three ACK transmissions (not shown in the figure) are required. These transmissions are sequential and the medium is shared between all N active nodes within a single collision domain. As such, if the medium is divided fairly across the different nodes in the network, a perceived bandwidth of $\frac{B_{max}}{N}$ is achieved at best. In the example with one transmitting and two forwarding nodes, the available bandwidth equals $B_{max}/3$ per node. Furthermore, each additional hop in the network causes additional end-to-end delay. Since the transmission sequence is varied due to the CSMA/CA mechanism, delay variations are observed.

As the available bandwidth per node reduces with the number of nodes in the collision domain, scalability of wireless ad-hoc networks quickly becomes an issue

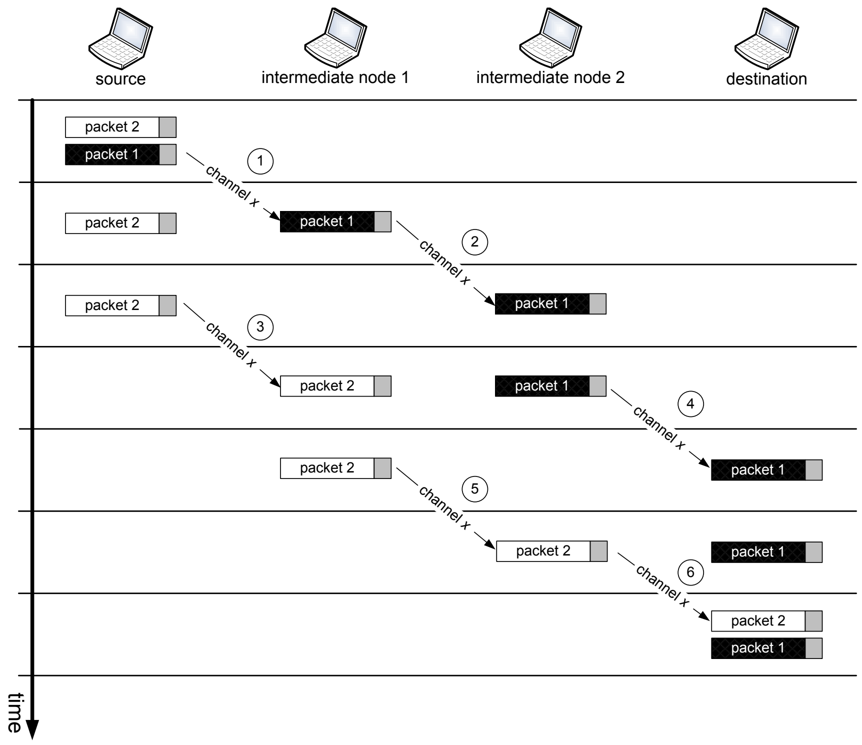


Figure 2.3: Multi-hop propagation in single-interfaced ad-hoc network. The packet sending order is indicated with circled numbers.

in dense networks. In addition to reduced bandwidth, wireless networks suffer from fairness problems: because of variations in packet sizes and transmission power used by the different nodes, certain streams in the network consume a larger than fair share of the available bandwidth [14]. Some fairness issues are solved through the IEEE 802.11e amendment, included in [9], defining modifications to the IEEE 802.11 MAC layer in order to provide better fairness and Quality of Service (QoS).

2.3.2 Multi-interface wireless networks

If multiple wireless interfaces are available per node, optimizations to the transmission scheme are possible. Consider the example of Figure 2.4, where each ad-hoc node in the network has two wireless interfaces. By configuring the three consecutive hops in the network to three non-overlapping frequencies x, y and

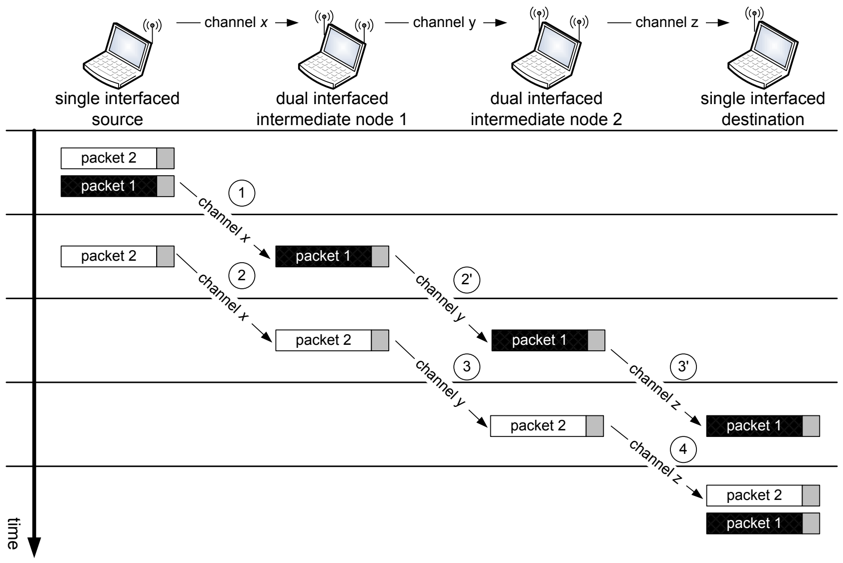

channel x
channel y
channel z
single interfaced source
dual interfaced intermediate node 1
dual interfaced intermediate node 2
single interfaced destination
packet 1
packet 2
1
2
2'
3
3'
4
time

Figure 2.4: Multi-hop propagation in multi-interfaced ad-hoc network. The packet sending order is indicated with circled numbers.

z, transmissions can theoretically occur simultaneously. As such, the end-to-end delay is reduced compared to the single-interface networks. Moreover, the maximum available bandwidth increases with the number of available non-overlapping channels.

However, the connectivity in the network is still limited by the number of available interfaces: in the example configuration, communication between the source node and second intermediate node is no longer possible, even at low data rates, since the source node has no interface configured to channel y or z, and the intermediate node cannot detect packets sent on channel x. Thus, a fully flexible channel use is only possible when every node has as many interfaces available as there are non-overlapping communication channels.

2.4 Theory and reality

In an attempt to answer the question of why wireless ad-hoc networks are not often used, the authors of [15] suggest that research is too often dedicated to finding solutions which are mainly developed for specialized target audiences such as the military or public services.

While this is true and might partially explain the lack of ad-hoc deployments, the problem is more complex. In the previous sections, it was stated that the performance of ad-hoc network technologies is lower compared to wired solutions, that a layered approach is suboptimal when designing wireless networks, and that scalability in traditional single-interfaced networks is a fundamental problem. Moreover, due to the popularity of –and marketing behind– the ad-hoc networking research topic, many of the *desired* features such as automatic deployment, easy management, ability to cope with network failures have since long been promoted as *available* features.

Although recently the limited appreciation of the wireless research community towards using experimental methods in wireless research is changing, and although it has been shown that theoretical breakthroughs are not always as effective as hoped for when solutions are deployed in reality [16], wireless networking research is still often centered around theoretical studies and simulations. This is not surprising, since it is difficult and expensive to perform reproducible large-scale ad-hoc experiments. As a result, wireless networking protocols have traditionally been designed and are still often being designed using a "top-down" approach, in which implementation is the last step in the development process. Decisions at design time are therefore mostly based on theoretical assumptions. Unfortunately, if these design assumptions prove to be wrong or incomplete, unforeseen implementation issues may cause months of theoretical research to become unfeasible due to practical restrictions. For example, *(i)* routing protocols that are relying on the availability of symmetric links may fail or operate suboptimal under real-life wireless conditions that are burdened by frequent asymmetric links [17]. *(ii)* The target hardware on which multi-interface protocols are to be implemented may not be able to simultaneously support multiple traffic streams, as will be demonstrated in Section 2.4.3 and was also indicated by [18]. *(iii)* While fine grained transmission power control solutions can be designed theoretically, the authors of [19] show that implementation of such scheme on currently available hardware platforms may not be feasible and is even unnecessary for a typical indoor WLAN environment.

Therefore, in the next sections, theoretical assumptions and simplified theoretical models are verified by experiment using a selection IEEE 802.11a/b/g compatible hardware. While some findings are trivial to people who are familiar with the physical layer of wireless communication systems, they are unknown to many researchers designing wireless networking protocols above the physical layer. Therefore, these examples provide an illustrative overview of hidden wireless networking issues that wireless network protocol designers may encounter while realizing their solutions on real-life test set-ups. While the measurements and observations in the next sections are strictly spoken only valid for the specific hardware/software combination in use during the specific test on the specific locations, they do at least indicate that theoretic assumptions are invalid for those Wi-Fi based test set-ups under consideration. The observations are then used to determine feasible wireless ad-hoc network protocols and architectures, and result in theoretical models with increased accuracy for the considered current generation Wi-Fi compatible hardware on which the protocols in this dissertation are developed.

2.4.1 Communication range

Based on the Friis equation for signal propagation in free-space, the communication and interference ranges of a wireless node were previously represented by concentric circles with the transmitting node at the center point. However, as also indicated by [20], the world is not flat: objects blocking the signal path result in additional path loss and cause reflections and diffractions, dramatically affecting signal propagation. In indoor environments, signals are blocked and reflected by static objects such as walls, floors and ceilings. People moving through a building, moving furniture and device mobility result in ever changing device communication ranges. In outdoor environments, signals are blocked by trees, hills, buildings and vehicles. Furthermore, devices do not transmit the same amount of energy to all directions. As such, the idea of circular transmission ranges is wrong and results in several misconceptions impacting the design choices of wireless protocols. These misconceptions are described below.

Multi-rate communication. First, the expectation of on/off wireless links is created, where two wireless nodes are either within each others' transmission range, or they are not. The reality is more complex: as previously stated, the physical layer is able to operate at different data rates. The lower the path loss between sender and receiver, the higher data rates may be achieved. Nevertheless, in many simulations related to wireless ad-hoc networks performed with popular network simulators, models are used with only a single fixed transmission and interference range [21]. While ray-tracing techniques exist in order to perfectly model wireless signal reception, these models are too complex to be used in large simulations, and are not illustrative. In [22], the influence of simulation model complexity on the reliability of ad-hoc networks is determined. The authors conclude that while very detailed simulations may be able to accurately predict the performance of a protocol at a given time, they may not be able to predict minor variations and are not flexible enough to quickly explore alternatives.

In order to overcome the limitations of the simplest models without largely increasing the model complexity, measurements between two Compex SAG-54 wireless network interfaces were performed to determine a realistic relation between path loss and corresponding achievable data rate. To this end, one node continuously transmitting packets in the $2.4 GHz$ range and one receiving node were mounted in RF shielded boxes, illustrated in Figure 2.5. The external anten-

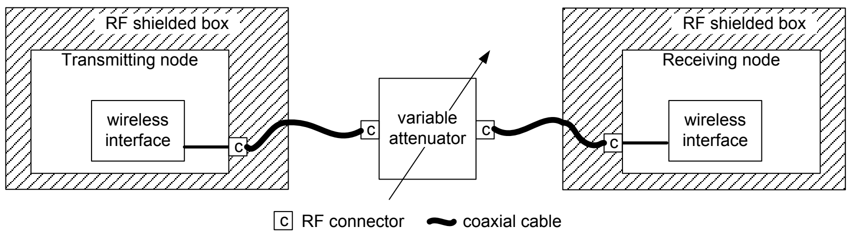


Figure 2.5: Measurement set-up for determining relation between received signal strength and maximum achievable data rate under controlled circumstances. Two wireless interfaces are connected via a variable attenuator over coaxial cables.

nas were disconnected and replaced by coaxial cables guiding the wireless signal from sender to receiver through a variable attenuator. At the receiver side, the *packet loss* and the *received signal strength indicator* (RSSI) value are recorded. The packet loss metric is defined as the ratio between the number of packets *not* arriving at the receiving node and the number of packets that were sent by the transmitter. The RSSI value is measured by the receiving radio and may be interpreted as the received power in dB above the noise floor, as such, it is an inverse measure for the path loss.

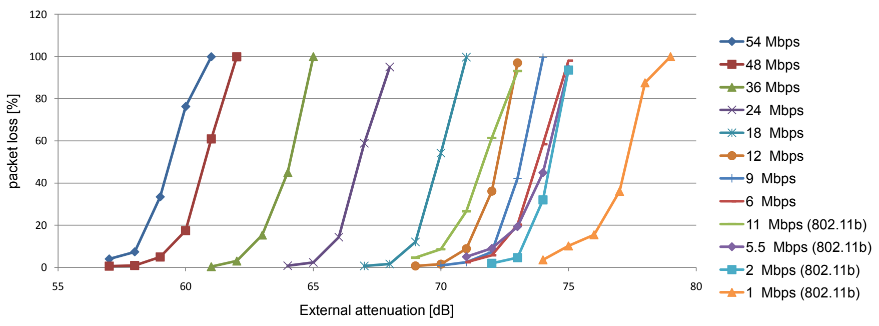


Figure 2.6: Measurement of the relation of the variable external attenuation between two shielded nodes connected over coax and the observed packet loss for different physical layer data rates.

For each data rate, the external attenuation was varied as to discover the attenuation range where the packet loss on the link changed from 0% (no loss, perfect signal quality) to 100% (signal quality too weak to decode packets at the data rate in use). The relation between the configured external attenuation and the packet loss is plotted in Figure 2.6. As expected, the measurements show how higher signal rates can be achieved when the external attenuation is reduced, thus, when the path loss is limited. Furthermore, it is interesting to note that the measurements show no discrete steps: to cross the border from full packet loss to no packet loss at all, the required received signal power differs by $4\,dB$ on average.

In order to give a meaning to this $4\,dB$ gap, consider the illustrative set-up of Figure 2.7. This set-up is used to calculate the difference in free-space path loss at distances R_1 and R_2 from a transmitting node. Given PL_1 the path loss at location 1, and PL_2 the path loss at location 2, what is the relation between the distances R_1 and R_2 in order to observe a difference in path loss of $4\,dB$? Using the previously determined expression for the path loss, now in decibels, this is easily calculated as follows:

$$\begin{aligned}
4dB &= PL_2 - PL_1 = 10log\left(\frac{4\pi R_2}{\lambda}\right)^2 - 10log\left(\frac{4\pi R_1}{\lambda}\right)^2 \\
&= 20log\left(\frac{4\pi}{\lambda}\right) + 20log(R_2) - 20log\left(\frac{4\pi}{\lambda}\right) - 20log(R_1) \\
&= 20log\left(\frac{R_2}{R_1}\right)
\end{aligned}$$

This last equation leads to $R_2 \approx 1.58R_1$. As such, under idealized free-space loss conditions, if a node moves away from an antenna using a constant physical layer data rate, packet loss may go from zero to full packet loss if its distance to the transmitter is increased by a factor 1.58. For example, a node positioned at 10 meter of the a sending node may not be able to communicate at the same data rate by moving an additional 5.8 meter away from the transmitter. Similarly, a node positioned at e.g. 50 meter from a transmitter, suffering high packet loss, may need to decrease its distance from the transmitter to $50/1.58 = 31.65$ meter before the suffered packet loss almost disappears. As such, at a constant data rate, there are important geographical zones in a wireless network where packet loss varies between 0% and 100%.

Finally, Table 2.2 shows the measured minimally required RSSI at the receiving Compex SAG-54 wireless interface packet to enable communication at a data rate with a packet loss of approximately 10% – a packet loss value commonly used when performing physical layer measurement, such as in [23]. Even in combination with a simple signal propagation model such as the free-space path loss model, this table can be used as input for developing a simple yet more reliable multi-rate model: as the RSSI values in this table may be interpreted as SNR values, a distance maps to a SNR value at the receiver, which in its turn determines the highest possible physical bit rate.

Distance vs. signal strength. Unfortunately, the unpredictability of signal propagation causes the relation between distance and signal strength to differ from the

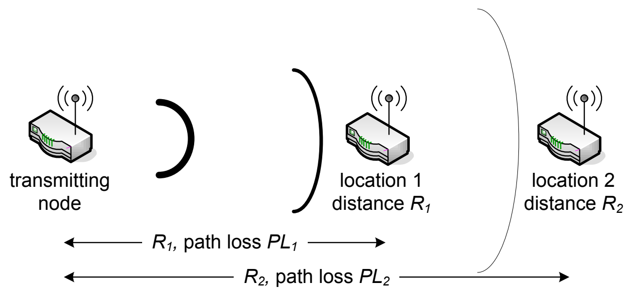


Figure 2.7: Illustrative set-up, used to map a $4\,dB$ signal strength gap to a relation between the distances R_1 and R_2 from the sending node under idealized free-space conditions.

Bitrate [Mbps]	Packet error rate [%]	RSSI at receiver
54	7.34	20
48	4.89	19
36	15.4	14
24	14.3	12
18	12.01	8
12	8.83	6
9	7.25	5
6	5.76	5
11	8.61	7
5.5	9.04	6
2	4.66	5
1	10.17	2

Table 2.2: Measured required received signal strength at the receiving Compex SAG-54 wireless interface for achieving different physical data rates when the packet error rate approximates 10%.

idealized inverse exponential graph. Figure 2.8 shows a partial map of the third floor of the IBCN research group building. A single node T, located in the bottom right corner of the figure is broadcasting packets with a size of $100\,bytes$ at a rate of 10 packets per second, a physical data rate of $1\,Mbps$ and transmission power set to $0\,dBm$ $(1\,mW)$ to all other nodes, represented by black dots. The alphabetical order of the node names corresponds with the order of the shortest geometrical distance to the transmitting node. The numbers on the figure are the RSSI values recorded at the receiving nodes, averaged over multiple test runs, each time sending 9000 packets. Wooden walls are represented as thin lines, concrete walls as thick lines. As expected, the general trend shows a greater path loss for nodes located further away from the transmitter. It also does not come as a sur-

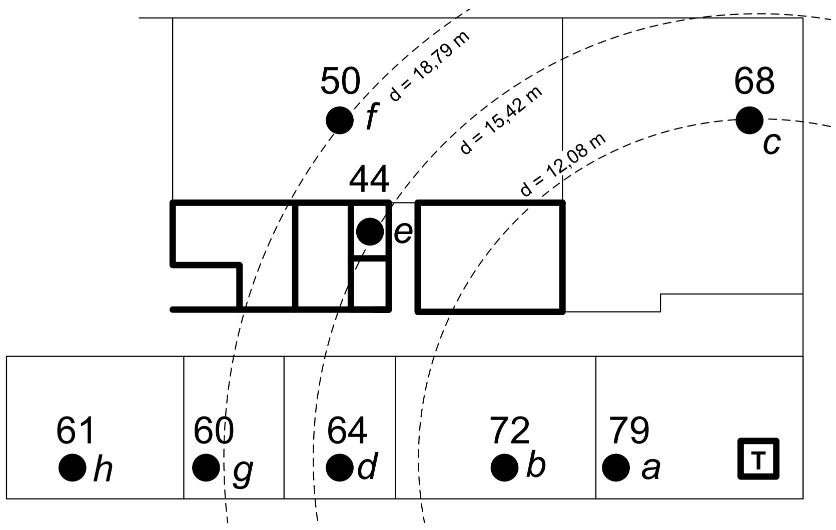


Figure 2.8: Average RSSI measured at receiving nodes with transmitter in bottom right corner. There is no strict relation between distance and RSSI.

prise that node e, surrounded by concrete walls, and node f, which has its direct line-of-sight to the transmitter obstructed by a concrete wall observe a lower signal strength compared to nodes behind wooden walls at the same distance of the sender. However, it is remarkable that the RSSI value observed at the bottom left node h is higher than the RSSI value observed at node g.

In the above scenario, this relative difference in RSSI between the two nodes can never be explained through a simple propagation model. However, in order to verify whether the general propagation trend can be fitted to a model, the measured values were compared to a *shadowing model* of the office building. A shadowing model is an improved version of the free-space path loss model taking into account direct and indirect wireless signals, resulting in constructive and destructive interference. The path loss in dB at a distance d from a wireless transmitter is calculated at follows:

$$PL = PL_0 + 10 \cdot \eta \cdot log\left(\frac{d}{d_0}\right) + \chi$$

In this equation, PL_0 is the path loss in dB at a nearby reference distance $d_0\,[m]$ from the transmitter, η the *path loss exponent* and $\chi\,[dB]$ a term added to account for statistical variations. The path loss exponent η is environment dependent and is determined empirically. For the IBCN offices, η was determined to be 2.85 during a measurement campaign conducted by the Ghent University - INTEC WiCa (Wireless and Cable) research group [24].

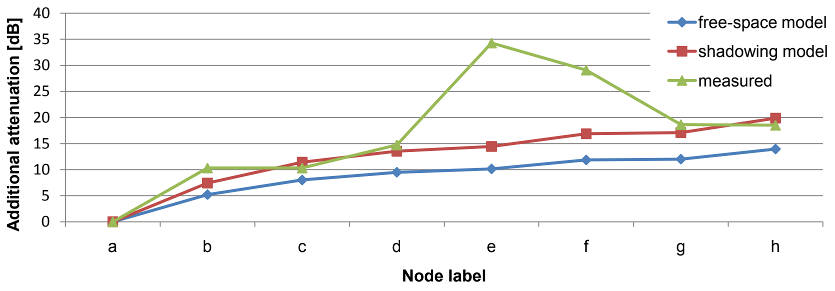


Figure 2.9: Additional path loss observed at the different nodes compared to the path loss observed in node a, comparing two models with the values measured at the nodes.

With node a used as reference node, the path loss through the shadowing model is determined for $\chi = 0$. The free-space path loss in dB is also calculated for a center communication frequency of $2.412\,GHz$ (channel 1) as $10 \cdot log\left(\frac{4\pi R}{\lambda}\right)^2$. The calculated free-space path loss in node a is chosen as the reference value. Finally, the additional path loss compared to the path loss observed in node a is calculated by determining the difference between the measured RSSI values at nodes b to h with the RSSI value measured in node a. The resulting relative additional path losses are depicted in Figure 2.9. While the free-space path loss model underestimates the path loss, the graph also shows that the additional attenuation calculated with the shadowing model are reasonably close to the measured values for all nodes not behind concrete walls. For these nodes –b,c,d,g and h– the average absolute difference and maximal difference between model and measurement are $1.6dB$ and $2.9dB$ respectively. As expected, the difference between model and measurement is large for node e and node f.

From the above observations, following intermediate conclusions can be drawn:

- A wireless link cannot be modeled simply by an on or off state based on distance to a transmitting node, as there is a complex relation between received signal strength, packet loss and physical layer data rates.

- Even though frequently used, free-space path loss models obviously do not accurately model signal propagation in non free-space environments.

- A (fitted) shadowing model produces a reasonable estimation of the path loss observed at a certain point in space, however, it is not built to model the effects of specific construction elements of a building. As such, results obtained through mathematical models or simplified simulators should be interpreted with care and cannot be used to make absolute claims on the performance of developed wireless networking protocols in a realistic environment.

2.4.2 Transmission power

Output transmission power modifications of the wireless network nodes were intentionally left out of the above discussions. However, the transmission power parameter has a profound influence on the operation of a wireless network. As can be seen from the Friis equation (see Section 2.2.1), in a static idealized situation, increasing the transmitted power at a sending antenna with a certain factor increases the received power with the same factor at the receiving antenna. As such, in general, increasing transmission power at a sending node results in nodes located further away being able to decode the transmitted packets. This operation comes at the cost of increased power consumption, which is an unwanted side effect, especially in battery-supplied systems [25]. Furthermore, the interference range of the transmitting node increases, leading to more collisions, thus reducing the bandwidth in the network. When different power levels are used inside a network, asymmetric links are created if a node is within the transmission range of an other node, but not the other way around. In this case, packets sent by a first node can be decoded at the second node, but the corresponding ACK sent by the second node never reaches the first node.

Protocols for power control have been designed to modify the power use of an individual network node, or of the entire network [26], in order to achieve different goals such as maximizing node or network lifetime [27], influencing the number of wireless links, or maximizing throughput in the network [28, 29]. Again, most of this research is theoretical and based on simplified propagation models.

Although some solutions are very promising theoretically, the implementation of these protocols on current generation IEEE 802.11 compatible hardware remains an issue. In [30], the authors report the main issues to be lack of support for power control features in the current hardware and drivers. Furthermore, hardware heterogeneity makes it hard to re-use a single power adaptation protocol with hardware from different vendors.

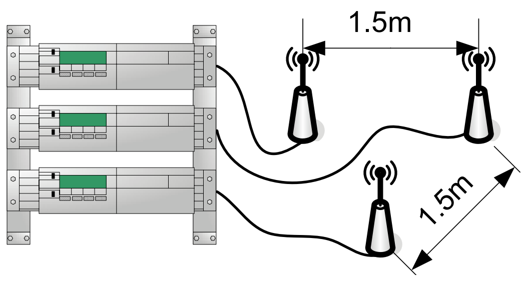


Figure 2.10: Simple power experiment using three rack mount devices with external antenna in a triangle position with sides of 1.5 meters.

Moreover, experiments with the small-scale set-up of Figure 2.10 show that even the simplest wireless ad-hoc set-ups may produce contra-intuitive results. Consider the following experiment: three rack mount computers are equipped with identical IEEE 802.11g compatible devices (DLink AWL-520) operating in ad-hoc mode. As such, transmission between any two nodes is enabled. A CBR (constant bit rate) UDP (User Datagram Protocol) stream is sequentially set up between any two devices in the network, while varying the power setting. While one might expect that in this isolated scenario the wireless link between two nodes will be of better signal quality if the transmission power is increased, measurements indicate otherwise: when a relatively high output transmission power of $100\,mW$ is used, great throughput instabilities are observed: throughput may suddenly drop, and reversing the traffic flow results in different throughput. However, when transmission power is configured to $10\,mW$, links are stable and are continuously able to support the theoretical maximum throughput to any node in any direction.

This experiment was repeated in a shielded environment similar to the set-up of Figure 2.5, confirming the observation that *decreasing* the output signal power at short distances leads to increased wireless link quality. Although not verified through measurements at hardware level, this effect is most likely caused by the fact that the radio receiver cannot handle the high incoming signal power. As such, especially in small-scale set-ups on a limited surface, more signal power does not always lead to better signal quality. Many scenarios exist in which multiple devices are close to each other, such as when exchanging files during a meeting or during wireless synchronization of a PDA with a laptop, making this unexpected effect an important issue.

2.4.3 Multi-channel model

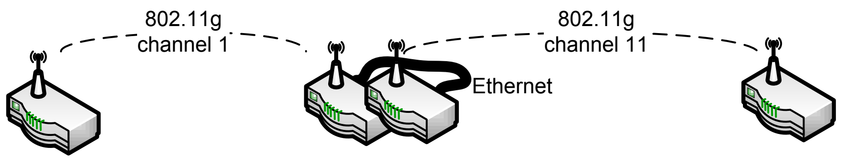


Figure 2.11: 2-hop IEEE 802.11g experiment using the theoretically non-overlapping channels 1 and 11. Two stacked Ethernet-linked nodes are used to represent a dual interface node.

In Section 2.3.2, the theoretical possibility of simultaneously using non-overlapping channels in order to decrease interference and consequently increase the capacity of the wireless network was already discussed. This approach is followed by works such as [31] and [32]. These solutions are based on the assumption that the capacity of wireless networks can be increased dramatically by creating integrated multi-interfaced nodes where the interfaces are configured to operate on theoretically non-interfering channels. While this assumption seems obvious from a theoretical point of view, a simple experiment invalidates this claim;

Consider the test set-up from Figure 2.11. In this experiment, a two hop wireless IEEE 802.11g network is constructed using Linksys WRT54GL Wi-Fi routers in ad-hoc mode. Two of these routers are stacked and connected over an Ethernet interface, representing a multi-interfaced device. The first link is configured to channel 1, the second link to channel 11, and each of the individual links is able to support a UDP stream of over 30 $Mbps$. The middle (stacked) node is configured to forward the traffic received on the first interface directly to the third node. Despite the fact that the routers forming the middle node have sufficient processing power to forward and receive traffic simultaneously on the wired and wireless interfaces, the throughput of the combined stream drops to 16 $Mbps$ - the same speed as achieved when configuring both links on the same channel. Thus, in contrast with the theoretical model, this practical set-up shows that the capacity of the network does not rise with the addition of a secondary interface to the middle node. Other researchers have observed similar effects, which are generally attributed to self-generated interference between the different interfaces [16, 33] caused by interfering signals transmitted on another channel than the one the interface is tuned to itself. This phenomenon is called *adjacent channel interference* [34].

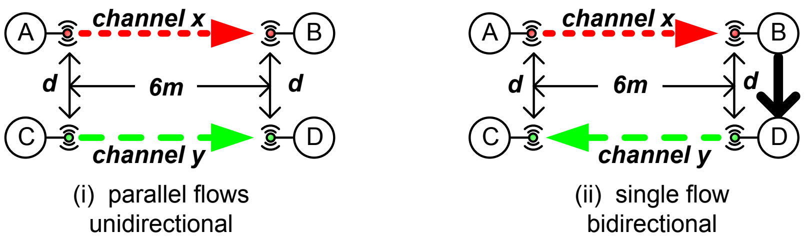

A
B
C
D
channel x
channel y
d
6m
d
(i) parallel flows unidirectional
A
B
C
D
channel x
channel y
d
6m
d
(ii) single flow bidirectional

Figure 2.12: Interference characterization set-ups. Nodes A,C and B,D represent a logical two-interface node with antenna distance d.

In order to further analyze the practical possibilities of multi-channel communication with commercial off-the-shelf (COTS) wireless equipment, consider the test set-up of Figure 2.12i. Similar to the previous experiment, two sets of Ethernet coupled single-interface Linksys WRT54GL routers AC and BD are used to represent a single interface node. This time the devices are not stacked but separated, in order to set the antenna distance d between the interfaces to 1.5 meters. Two wireless links are configured: link $A - B$ is set to a fixed channel $x = 1$, and the channel of link $C - D$ is sequentially modified from channel 1 to channel 11.

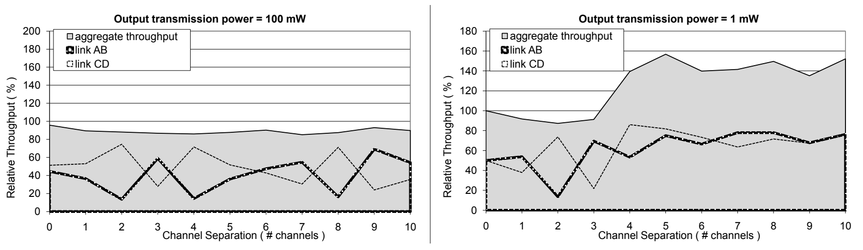


Figure 2.13: Aggregate throughput of two simultaneous streams (cf. Fig. 2.12i) relative to the throughput of a single, non-interfered stream for an output transmission power of 100 mW and 1 mW, and varying communication channel separation.

In an isolated situation, each individual link is able to support an application layer CBR UDP stream with a packet size of $1470\,bytes$ of approximately $24\,Mbps$. Next, two CBR UDP streams are activated simultaneously over links $A - B$ and $C - D$. While changing channel y, the application layer throughput of each individual stream is recorded using the default transmission output power of $100\,mW$. The experiment is repeated with an output transmission power of $1\,mW$. Figure 2.13 shows the measured aggregate application layer throughput of the two simultaneous streams, relative to the maximal application layer throughput of a single, non-interfered stream, for a varying communication channel separation. These measurements show that, at the default output power level of $100\,mW$, even with an antenna separation of 1.5 meters and channel separation of 10 channels (channel $x = 1$, channel $y = 11$), the combined throughput of the two parallel streams does not rise above the throughput that can be achieved with a single-interfaced node. However, when the output power is reduced to $1\,mW$, a capacity gain of nearly 60% is achieved for a channel separation of 5 channels (channel x=1, channel y=6). While still not equal to the theoretical gain of 100%, this experiment shows that capacity *can* be increased with a multi-channel approach on top of the used COTS hardware, but only on the condition that the transmission power is set sufficiently low and antenna separation sufficiently high. However, an antenna separation of 1.5 meters severely limits the practical applicability of such technique.

Therefore, a final set of tests was conducted to verify the impact of antenna separation on the aggregate throughput, using the previous set-up as well as the set-up from Figure 2.12ii, with a transmission power of $1mW$ and channels separation equal to 10. In the latter set-up, a different traffic flow is used, representing a typical multi-hop scenario: instead of using two parallel data streams $A - B$ and $C - D$, the traffic arriving to B from A is now transferred over an Ethernet link to D, and sent back to C. For antenna distances d equal to $1\,cm, 5\,cm$ and $10\,cm + k\,cm$, with $k = 0..6$, two lines showing the aggregate throughput of streams $A-B$ and $C-D$ are plotted in Figure 2.14: the line with diamond ticks is used as a reference scenario and shows the aggregate throughput for non-interfered sequential streams in scenario (i). The line indicated with square ticks shows the aggregate throughput of the same streams when activated simultaneously, also for scenario (i). Finally, the graph with triangle ticks shows the throughput of the single flow $A-B-C-D$ for scenario (ii).

Since the average throughput of a non-interfering stream in this set-up is approximately $25\,Mbps$, it can be concluded that, using the WRT54GL Wi-Fi routers, multi-interface communication at low transmission power remains feasible in scenario (i) when the antenna separation is larger than $55\,cm$, with an aggregate throughput that is 30% lower than the reference scenario but 44% higher than could be expected from a single-interface solution. For scenario (ii), multi-interface communication proves to be efficient for antenna distances larger than $70\,cm$. For closer antenna distances, the throughput degrades to what could be expected from a single interface solution or even worse. Moreover, in contrast with scenario A, a very unstable end-to-end throughput is observed which is very sensitive to minor variations in antenna placement. As such, the graph displays the throughput values maximally achieved as well as the minimum values using error bars. No communication at reasonable throughputs was possible when the antenna distance was further decreased beyond approximately $40\,cm$. At this distance, the radio interference caused by the transmitting antenna of node D at the receiving antenna of node B attempting to receive the relatively weak signals from node A grows to such a degree that communication is no longer possible.

From the above observations with multi-interface set-ups, it can be concluded that the effects of self-generated interference can be reduced by limiting the transmission power at the different interfaces and providing sufficient antenna separation. However, using the current generation of low-cost COTS hardware available during the tests, the theoretical achievable capacity increase using multi-interfaced nodes is hard to achieve. Even worse, multi-interface solutions can lead to unstable network links. The observations also raise questions about the use of multiple interfaces in integrated devices, where antenna separation is hard to achieve: successfully simulated protocols will most likely not provide the same results when deployed in reality. Moreover, since performance is heavily impacted by antenna distance, performance results obtained from testbeds using rack mount computers and external antennas might not be representative for the performance of the target end-user device which might, for example, require a small-size integrated design.

It was previously stated that lowering transmission power is often considered as a measure of freedom in order to decrease interference in the network or increase the lifetime of battery powered devices. The above experiments indicate that when using current COTS hardware as a base for multi-interface nodes, power adaptation is not a measure of freedom but a necessity in order to maintain connectivity in a

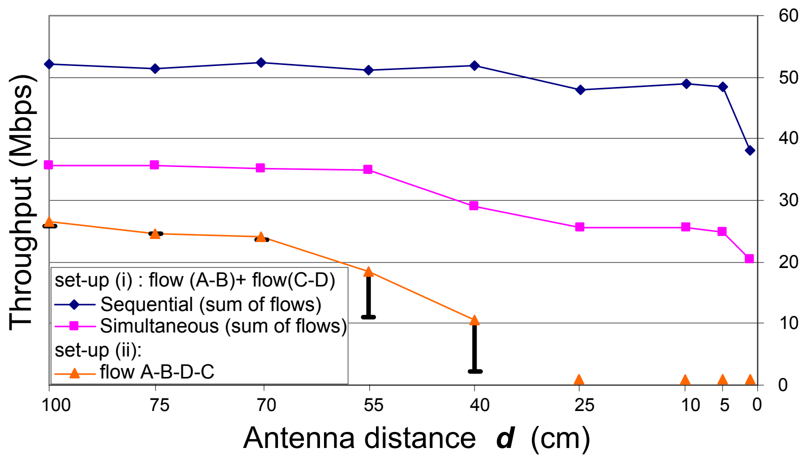


Figure 2.14: Throughput measurement for the test set-ups of Figure 2.12 for varying distance between the antennas. Transmission power is set to 1 mW. Channel x=1, channel y=11.

multi-hop network.

While COTS hardware may exist or may be developed in the future that causes less out-of-band interference or is more resistant to out-of-band signals, in an open environment with heterogeneous hardware, devices of lower quality will always be used. Furthermore, there are physical limits to the dynamic range of receiver circuits. Therefore, in the near future, it seems unlikely that palm-size devices will be able to take full advantage of multiple interfaces operating at medium to high transmission power.

Note that in the future, the relevance of using multiple individual interfaces within a single device to enable the use of multiple frequencies simultaneously may partly be reduced as MIMO (Multiple-Input Multiple-Output, [35]) technologies conquer the market. While the use of multiple individual interfaces is an FDM (frequency-division multiplexing) strategy, MIMO techniques are based on *spatial multiplexing* or SDM (Space-Division Multiplexing) [36]. The success of using multiple independent interfaces simultaneously depends on the degree by which the different FDM channels are isolated in the frequency spectrum from each other. For the Wi-Fi hardware used in the above experiments, it was shown that this isolation is in practice not always as expected. In contrast, the SDM based MIMO techniques such as used by IEEE 802.11n are especially designed to drive multiple antennas with spatial orthogonal RF signals with a channel width of up to $40\,MHz$ from a single control circuit. Furthermore, when MIMO signals are received at the different antennas, advanced signal is applied at PHY level, combining RF signals before demodulation [37, 38]. This contrasts with the use of individual interfaces at the receiver side, where each individual receiver is only attempting to receive the signal sent through its own communication channel. Finally, the use of different individual wireless interfaces in a single device with the goal of enabling simultaneous transmission and reception of packets at the node and thus increase the throughput, should not be confused with diversity techniques such as presented in [39], in which multiple interfaces are used at the receiver side and combined at MAC level with the goal of reducing packet loss in the network.

2.4.4 Hardware issues

It seems reasonable to assume that if an algorithm works on the Wi-Fi hardware of one vendor, it will also work on the IEEE 802.11 Wi-Fi hardware of a different vendor. However, during a measurement campaign, it was found that different Wi-Fi devices produce heterogeneous performance results.

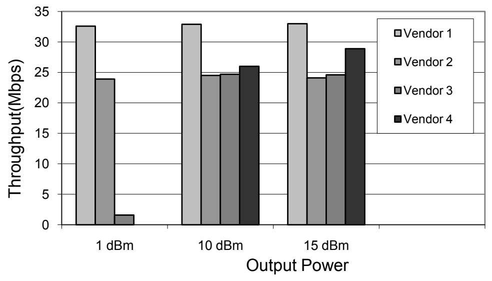


Figure 2.15: Maximal stable throughput between two identical IEEE 802.11a interfaces, each time from a different vendor, in an identical test set-up, using three different output power settings.

As an example, consider the results of throughput measurements between two identical wireless IEEE 802.11a interfaces of different vendors in Figure 2.15. During this experiment, two Alix1C1 [40] embedded devices were separated by 20 meters in an indoor environment, and configured to operate in the $5\,GHz$ range as to avoid interference with other test set-ups. Except for the wireless interface, identical nodes, omnidirectional antennas and the generic Madwifi open source driver (version 0.9.3) for Atheros based radio interfaces [41] were used during all tests. The graph shows the maximal stable throughput that could be achieved between the two nodes with the transmission power successively set to $1\,dBm$, $10\,dBm$ and $15\,dBm$. Although there is hardly a difference in device cost, there are considerable performance differences.

In a different test, spectral measurements of two IEEE 802.11a cards operat-

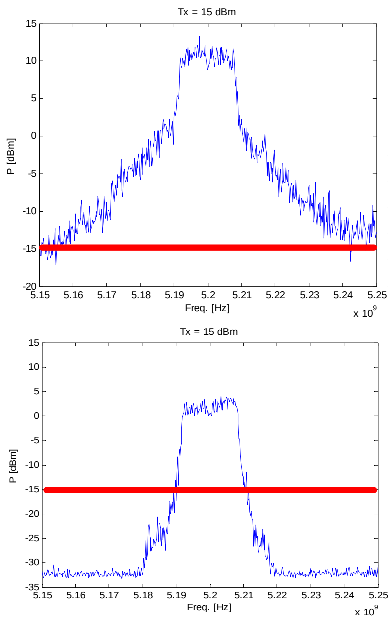


Figure 2.16: Spectrum measurements of two different IEEE 802.11a compatible mini-PCI cards operating at same Txpower (15 dBm), both using channel 40.

ing at the same output power of $15\,dBm$ $(31.62\,mW)$ on the same channel 40 were performed. To this end, the Wi-Fi cards were plugged into a 4G Mesh Cube, marketed at the time of the experiment as a mesh development platform by 4G systems [42]. A Mesh Cube is an integrated system consisting of a basic I/O and CPU, with multiple wireless 802.11 a/b/g interfaces stacked on top. It is small-sized (default 7cmx5cmx7cm when used with two wireless interfaces and without antennas), has low power consumption and runs a modifiable Linux Nylon distribution.

A mesh cube is put inside the previously used shielded boxes of Figure 2.5. With the node continuously sending broadcast traffic, the RF output connector of the cards is directly connected to a spectrum analyzer which is set to hold the maximum values. Two spectral measurements are presented in Figure 2.16. The spectra show that the first card suffers more from frequency leakage than the second card; knowing that the channel width of and IEEE 802.11a channel is $20\,MHz$ and the center frequency of channel 40 is $5.2\ GHz$, all signals below $5.19\,GHz$ and above $5.21\,GHz$ are considered to be out-of-band signals. While the out-of-band signals for the second card are at power levels lower than $-15\,dBm$, the first card produces out-of-band signals higher than $-15\,dBm$ over a frequency width of nearly $100\,MHz$. As a result, adjacent channel interference will be larger in networks built using devices of the first tested type than when using devices of the second tested type. Moreover, integration of the power of the signal over the observed window shows that the total output power of the cards to be $24.95\,dBm$ and $16.15\,dBm$ respectively, suggesting that the first card does not respond well to modification requests of the output transmission power.

During additional tests with the same two cards in IEEE 802.11b mode at a basic rate of $1\,MBps$, the differences in output spectra between the two cards were smaller, although the first card consistently produced more output power than requested: $5.06\,dBm$ instead of the requested $1\,dBm$, $9.61\,dBm$ instead of $5\,dBm$, $13.65\,dBm$ instead of $10\,dBm$, while producing $14.45\,dBm$ when requesting $15\,dBm$. The second card did produce the output power as requested for all settings.

Although errors or unexpected behavior caused by faulty or low-quality hardware can be solved by replacing hardware with hardware from a different type or vendor, ad-hoc network protocols can only become successful if a large group of end-users is able to use the protocols instantly without requiring adjustments to the protocol parameters, regardless of their choice of vendor. For example, if a centralized power adaptation protocol for Wi-Fi devices is developed which determines the power setting for an interface based on the coordinates of a node and an RF propagation model, the optimal power setting for each individual device in the two-node single hop scenario of Figure 2.15 would be the same regardless of the network interface card type, as the two nodes are each time separated by 20 meter. However, while the card of vendor 1 is able to support high-bandwidth communication at a power setting of $1\,dBm$, the card of vendor 3 or 4 would require a higher transmission power to result in a good wireless connection.

While this does not mean that the power adaptation protocol does not have its value, the real-life test results suggest that such protocol cannot be used unmodified in combination with all possible Wi-Fi devices on the market. To be able to apply the protocol under a wide range of environments with a wide range of heterogeneous hardware, a solution would be required that is able to dynamically adapt to the specific hardware types in use, or, the characteristics of different wireless interfaces would have to be known in advance. As a result, in an ad-hoc network where heterogeneous nodes are free to join the network and contribute in the routing process, it cannot be assumed that all nodes will react identically to a specific algorithm's action.

2.4.5 Laboratory environment

The test results from the previous paragraphs show that if algorithms and protocols for wireless ad-hoc networks are designed solely by using simplified models or simulations based on simplified models, they may fail to work as expected when deployed in real-life situations due to unexpected behavior of the RF channel or because of hardware heterogeneity.

Unfortunately, creating a reliable wireless test environment which allows reproducible tests is not a trivial task. A testbed may require a lot of hardware and space, and experiments are time consuming. Furthermore, there is a risk that algorithms are unintentionally tuned to work perfectly in the testing environment but fail to work in an other environment. As such, while real-life experiments may indicate issues with developed ad-hoc protocols, a single successful real-life test in a single environment using a single type of hardware does not automatically mean that the solution will work unmodified in a different networking environment. Factors contributing to this complexity are *(i)* the fact that a different environment comes with different propagation characteristics and background noise [43], and *(ii)* in a new environment, network topologies or traffic patterns may be used that had not been tested before.

2.5 Towards a feasible ad-hoc architecture

2.5.1 Solving issues

In the previous sections, an analysis was made of issues contributing to the slow adoption of ad-hoc networks. To summarize, several assumptions that seem obvious from a theoretical point of view, prove to be false when solutions are deployed on a selection of COTS hardware. Furthermore, during the selected small-scale experiments, it was experienced that the theoretical ease of use and simple network deployment, though often considered as a basic property of wireless ad-hoc networks, does not necessarily exist in reality: in every test set-up, repeated manual configuration of driver parameters and antenna positions was needed in order to achieve stable wireless links.

User-friendly commercial solutions based on wireless ad-hoc networks are seldom found, making ad-hoc networks unknown by the larger public. Even for technically skilled persons, setting up and maintaining ad-hoc networks is a challenge. As such, ad-hoc networks are often overlooked as a possible solution for connectivity. The lack of popularity and deployments hinders the availability of commodity applications, which, in its turn, reduces the incentive for commercial initiatives to fill the missing research gaps.

Other issues are of a more fundamental technical nature such as the limited performance compared to wired solutions, the instability of wireless links caused by variations in the wireless environment or by hardware issues of the participating devices, or the limited amount of spectrum available leading to scalability issues in a multi-hop environment.

Because wireless ad-hoc research has mostly been approached from a theoretical point of view based on oversimplified models, the above issues are still unknown to many researchers. In the remaining chapters of this work, new and improved protocols and architectures for wireless ad-hoc networks are developed, using a bottom-up problem-oriented approach that takes into account the technical and non-technical issues that were identified in this chapter.

First, a wireless mesh network architecture is defined, that limits certain degrees of freedom of the backbone nodes with respect to node mobility and hardware quality. As such, some of the assumptions that were shown to be invalid for a random population of heterogeneous ad-hoc nodes become valid, enabling the use of advanced techniques such as multi-channel communication without the inherent risks of creating network instabilities.

Second, the protocols developed for the devices of this architecture take into account the complexities and unreliability introduced by the wireless environment. Instead of relying on the purely theoretical models, the propagation and interference characteristics observed in this chapter are kept in mind.

Third, the developed solutions are tested on testbeds and during field tests, this way demonstrating their feasibility or revealing additional issues.

Finally, the implementation and experimentation experience collected during the development of protocols is used to define a wireless network experimentation methodology.

The developed wireless mesh network architecture and algorithms are considered as a first, feasible step in bringing ad-hoc network technology to our everyday lives. Once wireless ad-hoc network technology gains momentum through the deployment of mesh networks, and, helped by technological evolutions that will bring better RF communication technologies and more powerful end-user devices, the degrees of freedom that were limited by the mesh architecture may eventually be released.

2.5.2 A heterogeneous hierarchical wireless mesh architecture

In the introductory chapter, a wireless mesh network was already introduced as a hierarchical ad-hoc network where relatively powerful mesh routers in terms of processing power, memory capacity and network interfaces form a network in order to provide connectivity and additional services to mesh clients residing at a lower hierarchical level. These heterogeneous networks have already been described in the literature [44] and are commonly praised for their (theoretical) ease of set-up and network extension capabilities. However, based on the above observations, there are more reasons why hierarchical mesh networks can help to realize robust wireless ad-hoc networks;

- Multi-channel multi-interface protocols theoretically allow more efficient use of the wireless spectrum. However, if an ad-hoc network entirely relies on setting up wireless links between small and mobile end-user devices such as PDAs or smart phones, it might not be feasible to use multi-interface protocols in real-life deployments, as many end-user devices will probably only be able to operate a single high speed wireless interface in the unlicensed band simultaneously, since adding interfaces is suboptimal due to the described interference problems and other limitations such as processor capabilities, power consumption and cost. In contrast, in a mesh network architecture, many cases exist in which the mesh routers are mounted to large host systems such as buildings or trucks, making the size of the mesh router less important. As a result, bigger devices may be built, making it easier to create multi-interface devices with sufficient antenna separation.

- The mesh routers may be operated by companies or network operators, as opposed to ad-hoc networks where all nodes are end-user devices. Because companies and operators have incentives for creating high quality networks such as increasing the productivity of their employees, decreasing maintenance cost, or attracting as much customers as possible, they are more willing to invest in high quality mesh routers. As such, mesh routers could form a wireless part of the company or operator's backbone. Operators might also choose to use licensed technologies such as WiMAX [45] for (a part of) the network backbone, further increasing the networking quality.

- In an ad-hoc network where heterogeneous end-user devices are allowed to join freely, a faulty or low quality node may join the network, resulting in decreased performance and satisfaction for other users in the network: even a user with high quality hardware is not assured of a good connection if he is connected through a low quality node. By allowing direct connections of ad-hoc clients to a powerful mesh backbone, a better and more stable service quality can be guaranteed.

- An ad-hoc network that is fully supported by end-user devices relies on the presence of many nodes with compatible networking protocols at the same location. Since ad-hoc networks are not commonly used at the moment, potential users lack the incentive to install the required protocols, thereby further slowing down the adoption of ad-hoc networks. Furthermore, as previously shown, hardware of different vendors may react differently on the instructions of a protocol. Hence, mass deployment of optimized ad-hoc protocols is not feasible in the short term. Designing protocols for the ad-hoc deployment of a mesh backbone generates tangible results more quickly for three reasons: *(i)* only the mesh routers need to be adjusted, which means that less devices should be modified, *(ii)* the devices within a single mesh backbone belong to a single administrator domain, requiring only a single contact to install ad-hoc protocols. In this mesh backbone, the administrator may install non-standard protocols: as long as the end-user compatibility is assured by offering standardized access to the client nodes, there is no risk for compatibility issues within the considered mesh network deployment. *(iii)* While not a requirement, there is a good chance that all wireless mesh backbone devices are of the same type and vendor, considerably reducing the time needed to analyze the behavior of the mesh solution.

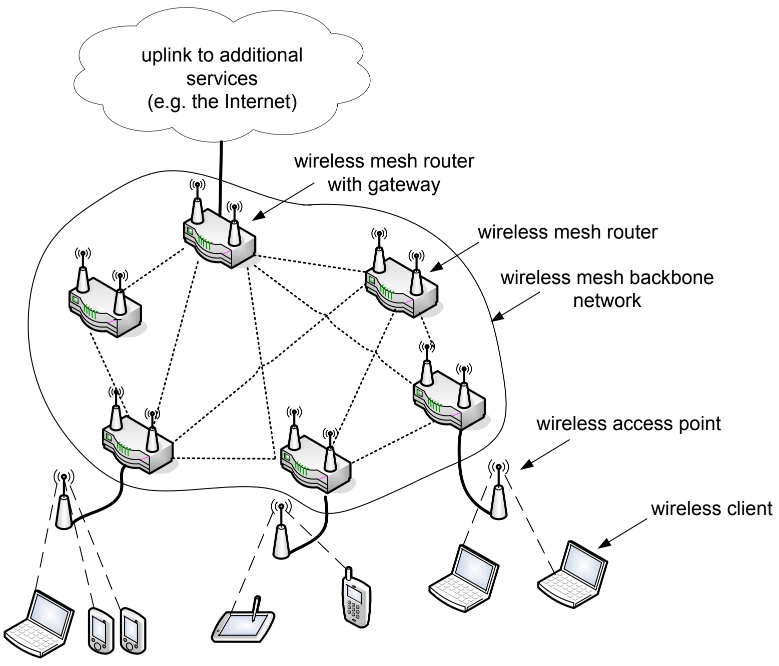

uplink to additional services (e.g. the Internet)
wireless mesh router with gateway
wireless mesh router
wireless mesh backbone network
wireless access point
wireless client

Figure 2.17: Logical view on the hierarchical wireless mesh network architecture considered in this work. Wireless mesh routers form a backbone network, to which access points are connected. Clients connect to the backbone through an access point.

Figure 2.17 shows a logical view on the hierarchical wireless mesh network architecture that is considered in this work, similar to the mesh architecture in the introduction. The relatively powerful wireless mesh network routers in the backbone form a high quality meshed ad-hoc network. Access points (AP) are then connected to this wireless backbone. They are used by client devices to connect to the wireless mesh backbone. Some mesh nodes might be connected to external networks and function as a gateway, providing additional services to the client nodes. For example, wireless clients may be presented with Internet access.

In September 2003, the IEEE started efforts towards creating a standard to allow interoperability between IEEE 802.11 based wireless mesh devices, under the name of IEEE 802.11s. Ever since, the IEEE task group has been optimizing draft standards. At the time of writing, a fifth draft standard is in preparation [46]. The high-level IEEE 802.11s mesh architecture closely matches the high-level architecture considered in this dissertation depicted in 2.17: mesh routers are used to create a stable, meshed backbone network and provide access to non-mesh nodes through access points. Differences in terminology exist [47]: in IEEE 802.11s, mesh routers are referred to as 'mesh points', client devices are called 'stations' and mesh access points providing access to these clients are called 'mesh portals'. The standard includes a description of how to join or start a network, how to access the wireless medium, an IEEE 802.11i [9] based security mechanism, how to discover neighbors, and sets the requirement of supporting the Hybrid Wireless Mesh Protocol [48, 49]. The IEEE 802.11s standard is still not finished. The authors of [50] indicate that many issues are yet to be solved, an observation which is confirmed in the number of open issues listed on the taskgroup's website. There is no question that standardization efforts will help to promote the use of wireless mesh networks. While the progress of the IEEE 802.11s task group is interesting to follow, in this dissertation, no attempt is made at designing algorithms which fit directly into the evolving draft standards in order not to limit the protocol design options.

Similarly, within the IEEE 802.15 WPAN standardization branch, the fifth standardization task group targets the provision of mesh capabilities on top of IEEE 802.15.x MAC and PHY layer specifications. Their efforts have been bundled in the IEEE 802.15.5 recommended practice document. The recommended practice includes specifications on addressing, routing and energy saving functionalities [51]. This WPAN standardization effort is not further considered in this dissertation.

2.5.3 Characteristics and example scenario

The access points that are added to the mesh backbone can either be implemented as standalone devices connected to the mesh routers over an Ethernet connection, or by adding an additional wireless interface and antenna to the mesh router for client connections specifically. By configuring the access point interface to behave as a traditional access point, backwards compatibility with existing client devices is guaranteed. Table 2.3 summarizes the differences in characteristics of the wireless mesh router and wireless mesh client devices that are considered in this dissertation.

Among other things, the table shows that while the node mobility of the wire-

characteristic	mesh router	client device
device type	known type, often homogeneous within a single application	heterogeneous COTS
owner	single administrator	private owners
wireless interface	single or multiple	single
wireless technology	unlicensed or licensed	unlicensed
power	from host system (e.g. vehicle, mains powered), high capacity battery, energy harvesting (e.g. solar energy, wind energy)	limited (low capacity battery)
mobility	low or limited	high
processing power	high	low to high
node memory	high	low to high
network dynamics	high	high
node size	up to tens of centimeters; antennas may be separated by meters.	relatively small
number of links	multiple mesh routers as neighbors	single connection to AP

Table 2.3: Typical characteristics of wireless mesh routers and wireless mesh clients.

less mesh routers may be limited, the mesh backbone network is still highly dynamic. This is clarified through the example of Figure 2.18. Assume an earthquake strikes an area, and most communication infrastructure is destructed. Multi-interface mesh routers could then be mounted to the vehicles of the different rescue teams. The trucks and vehicles are equipped with electric power generators, as such providing the necessary energy. Since trucks are relatively large, multiple interfaces and antennas can easily be mounted. In the beginning, only a few trucks will be on site. After the trucks have reached their location, they rest immobile for a prolonged period of time, resulting in relative immobility of the mesh routers. However, as new trucks arrive at the scene and other trucks leave or move, new wireless links need to be set up to replace failing links, requiring dynamic adjustments inside the network. Furthermore, the network environment may continuously evolve as e.g. an aftershock can tear down additional buildings.

Rescue workers and field hospitals connect to the mesh backbone through the

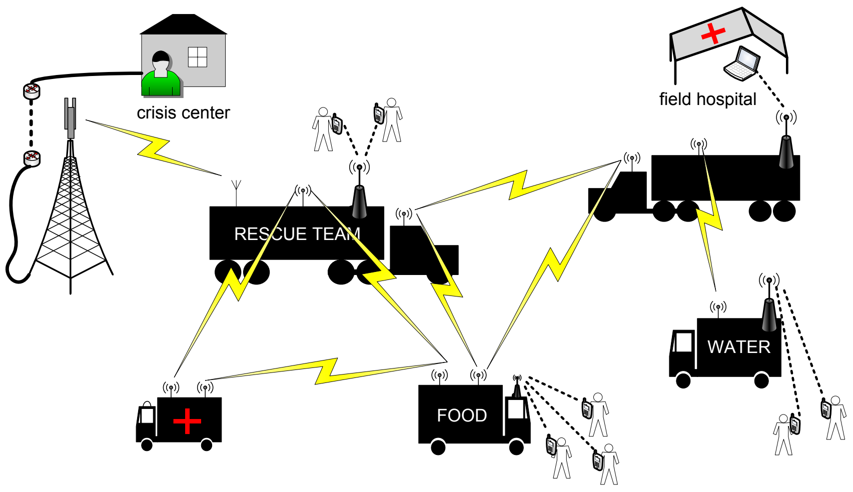


Figure 2.18: Example use case of a disaster area. Mesh routers are attached to rescue vehicles and provide network coverage after an earthquake.

access points mounted on the trucks. Some trucks may use one of their interfaces to create a link with back-end infrastructure such as a remote rescue center. Since the limited devices of the rescue workers make a direct single-hop connection with the mesh backbone, the scalability issues caused by traditional multi-hop ad-hoc forwarding are avoided. Furthermore, since the end-user devices do not need to forward packets sent by other nodes, energy is saved by the end-user devices.

2.5.4 Node architecture: a cross-layer approach

Wireless mesh routers and mesh clients will only be able to fully overcome the issues revealed in this chapter in case each node is capable to locally adapt to the global dynamics of the changing network environment. In essence, such adaptations are only possible if protocols and algorithms are available for gathering and processing information, as well as for changing the node behavior to compensate for the observed effects. Since different nodes may have heterogeneous capabilities, not all measurement and adaptation mechanisms are needed or possible at all nodes. Based on the observations that were gathered from the different small-scale testbeds in this chapter, following architectural components are considered to be important in order avoid practical issues when implementing mesh solutions;

A first requirement for the nodes is to gather **context information**. Algorithms running on top of wireless mesh clients and wireless mesh routers need to be aware of their capabilities. These *device capabilities*, such as number and type of interfaces, memory capacity, processing capabilities and remaining battery power are used by various protocols to determine whether a specific node may be used as a mesh router, or is limited to operate as a client. Furthermore, a list of the *available services* makes it possible for dynamic algorithms to know whether a certain node may function as a gateway to the Internet, provide a lookup service, or has access to other (local) information that can be useful for other devices in the network. Especially for mesh clients, a *user profile* assures that protocols operate within the expectations of the user. For example, some users might prefer low cost over best connection; in this case, when multiple interfaces are available, a cheap connection (e.g. Bluetooth connection to a nearby desktop computer) may be preferred over a better yet more expensive connection (e.g. WiMAX signal over nearby antenna).

Second, **network parameters** should be determined and mechanisms are required to resolve issues such as failing links or to compensate for the effects of a change in networking environment. A *traffic, interface and channel monitoring agent* enables protocols to gather information on the observed medium occupation at different interfaces and/or channels, both caused by the node itself and by surrounding nodes. As such, if additional traffic streams are required, the least occupied channel or interface may be used. Choosing an optimal interface and channel is performed by an *interface and channel selection* agent. For each individual link, a *link quality agent* could monitor the state of current and possible future connections, for example, in terms of packet loss, RSSI or coding scheme at physical layer. This information could then be used in order to notify routing protocols about impending disconnections of a link such that new routes may be chosen well before the connection is lost, or, transmission output power could be raised in the event a node becomes isolated. For this modification and other power optimizations, a *power control agent* is used. Since many end-user client devices already implement some form of power control, developing new power control algorithms may be only feasible in the mesh backbone. Changing transmission power results in a different transmission range and thus a different topology, leading to complex optimization problems [28].

Third, other optimized subsystems are needed to complete the node architecture. The functionality that needs to be provided by these subsystems includes but is not limited to: neighbor discovery, routing metric calculation, addressing, gateway discovery, mobility support, device configuration and a packet scheduler, allowing to prioritize traffic.

As suggested in the beginning of this chapter, optimal design of wireless protocols may not be possible if nodes are developed according to a strictly layered approach depicted in Figure 2.19a, because of the inability to share useful information collected at one layer with another layer. If one layer is not aware of the information available at a different layer, optimization opportunities where the information from one layer can be used to support the decisions at another layer may be missed out on, or, functionality may unnecessarily be duplicated at multiple layers. For example, an application layer video streaming codec may dynamically adjust its forward error connection mechanisms, in case frequent packet loss is detected over an end-to-end network path. At the same time, an adaptive forward error correction strategy at the MAC/PHY layer of IEEE 802.11a/g nodes over which the video is streamed, may also increase the number of bytes used for error detection when MAC layer retransmissions are detected. This combined reaction might decrease the useful video bandwidth more than would strictly be required.

An example of cross-layer interactions used above suggests the use of physical layer parameters to predict impending changes in the routing table. To this end, information should be passed on from the physical layer to the routing layer in one way or another. From a programming point of view, there is no real challenge in sharing information between layers: using some proprietary hack of the network stack, information is easily forwarded from one layer to another. However, from an architectural point of view, thoughtless cross-layer implementations may lead to complex protocols that are very hard to understand and maintain in the long run [52]. Since cross-layer elements are to be included in our node architecture, there is need for an organized way of exchanging cross-layer parameters.

2.5.4.1 Related work

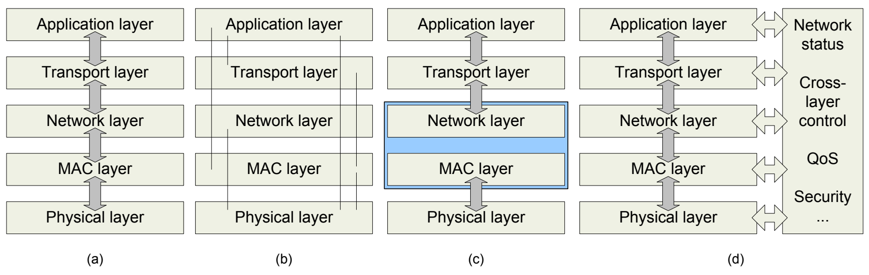


Figure 2.19: Default networking stack (a) and different strategies for enabling the exchange of cross-layer parameters (b–d).

Cross-layer protocols are implemented in many ways. In early cross-layer work, the layered design is largely left unmodified, but triggers are defined that enable signaling of discrete parameters across the network stack (Figure 2.19b). This approach is found in works like [53], where the use of an explicit congestion notification mechanism [54] is described that allows the TCP transport layer protocol to distinguish between packet loss caused by congestion or by the time varying nature of the wireless channel. In this case, discrete information on packet losses is propagated from the MAC layer to the transport layer under the form of a bit which can be set in the TCP header. The lines depicted in Figure 2.19b non-exhaustively indicate additional opportunities of exchanging information through simple direct triggers. For example, an application or routing protocol may receive a trigger from the PHY layer announcing impending connection loss.

Instead of propagating discrete information, other authors co-design several layers at once. In the example of Figure 2.19c, functionalities and information at the MAC and network layers is shared in such way that there is no longer a clear distinction between both layers. Co-designing layers is especially done when information is shared extensively between the layers, such as in [55], where MAC and routing features are integrated to support packet transmission in low duty cycle networks.

Finally, generic cross-layer architectures and frameworks have been presented in literature. These approaches are based on some sort of cross-layer information database spanning all layers of the networking stack. In addition to the default layered behavior of the networking stack, the access to the cross-layer information database is standardized, with the goal of exchanging information across layers. This approach is illustrated in Figure 2.19d. Examples of these frameworks in literature are the following;

In [56], the authors present their cross-layer architecture called ECLAIR. In ECLAIR, a *tuning layer* is provided as an interface to read and update data-structures from the different protocol layers. These tuning layers are then used in their turn by *protocol optimizers* containing the cross-layer algorithms. Changing code at a specific layer or porting the optimizations to a new operating system thus only requires changes to the tuning layers. Their architecture is built on four design goals: rapid prototyping, minimal intrusion, portability and efficiency in terms of minimum execution overhead.

The MobileMan architecture [57] targets a full stack cross-layer design as well. Here, the information database is called the *Network Status* component. In contrast to ECLAIR, the authors believe that the only way to achieve cross-layer optimizations is to redesign the network protocols at every layer. Another cross-layer framework based on the same ideas of modularity and efficiency is presented in [58]. In this work, a *cross-layer coordination server* plays the role of the information database and signaling to this server is done through the sending of event messages.

The authors of [59] add a more fundamental extension to cross-layer architectures in their CrossTalk architecture. Through a data dissemination procedure, a global cross-layer view of the network is built. The idea behind their approach is to act locally based on global knowledge. In practice, two information databases are used: one containing the local view and one with a global view.

2.5.4.2 Cross-layer: a means to an end

In order to let the different subsystems or *components* of our wireless mesh node architecture interact efficiently, a cross-layer strategy is required. The aim is to

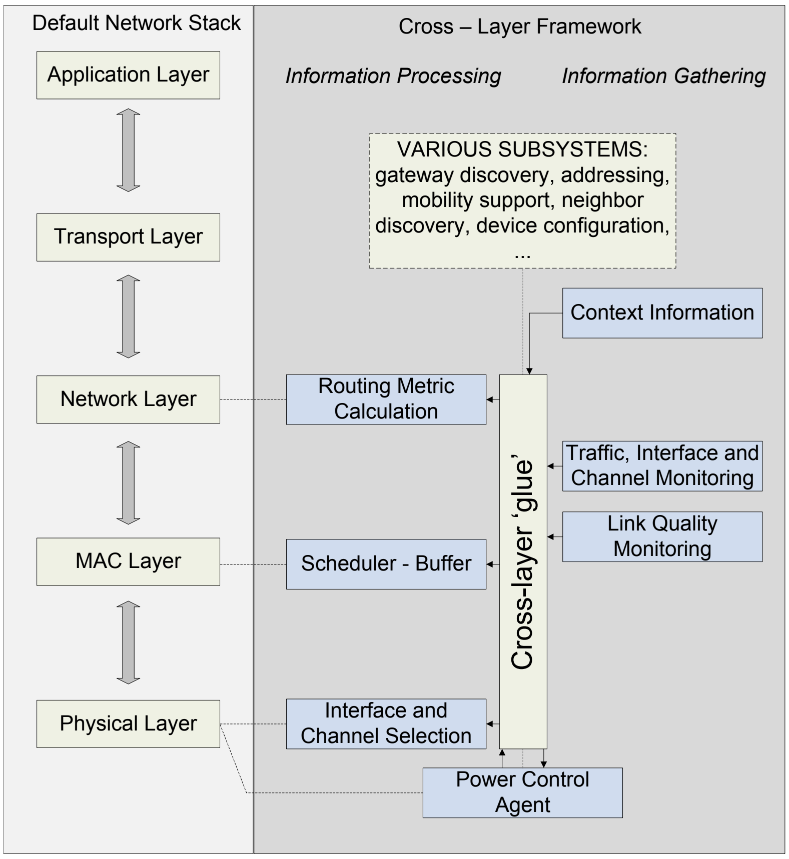


Figure 2.20: Cross-layer optimized wireless mesh node architecture. Applies to mesh routers and mesh clients, although not all architectural components should be supported at all times or by all devices.

avoid a cluttered interconnection of the gathering and processing components from Section 2.5.4 and the original stack, as this would hamper the ability of components to be plugged in on existing algorithms or to be re-used by other protocol designers. Therefore, similar to some of the cross-layer frameworks described above, an approach that keeps cross-layer optimizations out of the original layered stack is needed: the specialized algorithms are developed in a modular way outside of the original stack, as presented in Figure 2.20. The figure shows how the standard OSI protocol stack is left largely unmodified, while the components previously described are grouped inside a cross-layer framework. In this framework, the components that are grouped on the right are used to gather information, while the components on the left are meant to process information and enhance the operation of the default networking stack. The center components indicate additional functionalities that may span the entire networking stack. Not all mesh devices should implement all components or support them at all times.

All implementations are loosely coupled around what is provisionally vaguely described as 'cross-layer glue', describing the need for somehow interconnecting the different components. In comparison with existing frameworks, the presented cross-layer framework included in the node architecture is less generic, and is built with specific objectives for our mesh architecture such as link quality monitoring or channel selection in mind. As such, in this work, building a cross-layer framework is not considered as a goal on itself, but as a means to create stable wireless mesh networking protocols. Instead of a top-down approach where an architecture is theoretically defined and algorithms then have to be fit into the framework, the proposed cross-layer framework is inspired by practical considerations. The advantage of this modus operandi is that the cross-layer framework is able to grow more naturally. Moreover, no limitations are imposed on the way how optimization components should be developed internally.

In order further specify the 'cross-layer glue' block and its requirements, restrictions and assumptions for its use are first determined:

1. The glue block should provide a link between normal protocols used for information gathering, information processing, and the specific actions leading to an adaptation of the networking stack in order to increase the reliability or performance of the wireless system.

2. Any protocol of the default networking stack or cross-layer component may access or modify the contents of the glue block. However, it is assumed that all protocols act in good faith and lead to a global optimization of the wireless system. When there are conflicting interests (e.g. financial cost versus level of quality), the end-user or network administrator should be able to set his or her preference through the context information.

3. There are no limitations on the contents of the glue block: any conceivable way of exchanging information is accepted. Examples of glue entities include a single parameter, a boolean, a vector of parameters, or more complex systems taking into account past, current or future predictions of parameter values.

4. There is no limitation on the number of glue entities that are used. For example, a boolean might be used to indicate whether a node is completely configured or not, and a vector may simultaneously maintain a list of the number of missed ACK messages at every interface. The boolean might be set by the auto-configuration subsystem and trigger an action in the neighbor discovery subsystem, while the vector may be used by the interface selection component and the routing metric calculation.

5. If additional components are connected to the glue block, no changes should be required to any of the existing components, as such assuring modularity of the approach.

6. The protocols and components modifying the contents of the glue block are responsible for assuring the convergence of the glue entities. To avoid instabilities caused by multiple components accessing the same glue component simultaneously, a temporary lock may be set.

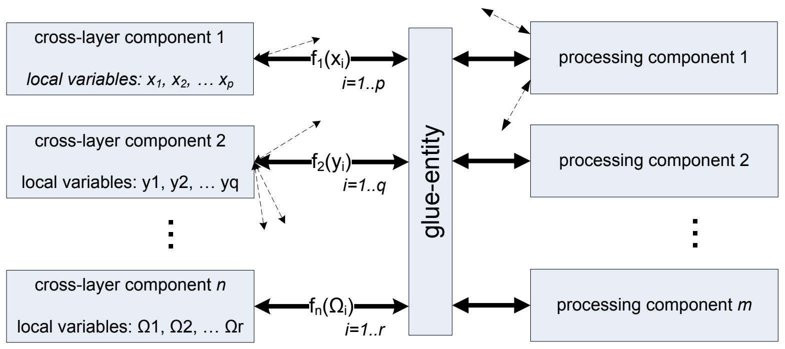


Figure 2.21: Different cross-layer components are loosely coupled using a glue entity. Gathering components measure and process their information in order to adjust the glue entity, which may be used by one or more processing components. Striped arrows indicate optional access to other entities.

Figure 2.21 illustrates how information gathering and information processing components are loosely coupled through a glue entity. In this example, n gathering components collect parameters from some location in the networking stack and store them in their local variables. Next, the local variables are processed and combined, and used to modify a glue entity. This entity may then be used and modified by m processing components in order to carry out their specific task. The striped arrows indicate that one gathering component may adjust several glue entities, and one processing component may use several input entities.

The concept of cross-layer glue is further clarified through the example of a glue entity used for interface and channel selection for IEEE 802.11 compatible networks, that will be used for the development of an interface and channel selection scheme in Chapter 3. Consider a parameter, called *Channel Quality Parameter* (CQP). A CQP is defined as a dimensionless variable attributed to a specific communication channel at a specific node, indicating the abstract notion of channel 'quality'. CQP_α^x denotes the CQP corresponding with channel x at node α. Now, define a CQP_α^x equal to 0, to correspond with node α proclaiming channel x to be a channel of perfect quality, meaning that based on the knowledge of node α, whenever it would initiate a new wireless link, it would prefer the link to be started on channel x. A CQP_α^y higher than CQP_α^x indicates that channel y is less suitable for starting new links than channel x. As such, a CQP value acts as an inverse measure to the channel quality.

In a next step, a glue entity called *Channel Quality Vector* (CQV) is created. The channel quality vector for node α, CQV_α, able to configure its radio to η different channels, is then defined as the following vector of size η:

$$CQV_\alpha = \left[CQP_\alpha^1, CQP_\alpha^2, \ldots, CQP_\alpha^\eta\right]$$

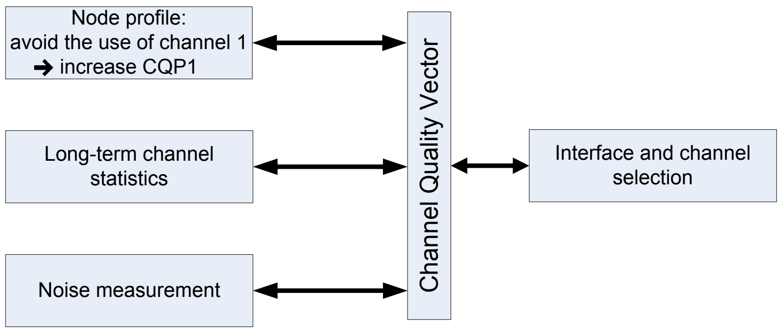

Node profile:
avoid the use of channel 1
➔ increase CQP1
Long-term channel statistics
Noise measurement
Channel Quality Vector
Interface and channel selection

Figure 2.22: Example of the use of Channel Quality Vectors as glue entity between protocols gathering information on the state of the communication channels and an interface and channel selection protocol.

Under the assumption that CQP values are a reliable representation of the channel quality observed on the specific channels, the CQV now represents a full view of the quality attributed to the different channels. As a result, the CQV may be used as an input entity for an interface and channel selection protocol. Figure 2.22 shows how the CQV are determined by protocols and settings at different layers of the application stack. In the example, three modules are influencing the CQV: a setting in the node profile indicates that for some reason, the user wants to avoid the use of channel 1, therefore, the CQP^1 is increased with a large value. At the same time, another subsystem collects long-term statistics of the channel use in the environment of the node, and increases values either strongly or not at all, depending on its interpretation of impact of long term statistics on the channel quality. A last subsystem performs noise measurements and once again, adjusts the CQVs according to its judgments. The CQV is then used by the channel selection protocol to actually configure the wireless interface(s) of the device. Additional information collected by the channel selection protocol might also be translated to additional CQV adjustments.

As such, any developer is free to add additional sources of information to improve the quality of the CQP metrics, and any protocol is able to use the CQV as an input source. The only interface that is needed, is a correct translation of the CQV parameter. As such, a simple yet effective cross-layer framework is created, that ensures full flexibility for developers while guaranteeing modularity of the developed algorithms. The cross-layer framework is not considered to be a rigid box in which algorithms and protocols should be crammed, but rather a flexible membrane that grows naturally as cross-layer optimizations are added. In the long run, a standardization of glue entities might enable globally interchangeable wireless protocols, and create a sound base of powerful metrics which can be used for the development of novel wireless networking algorithms.

As indicated, the cross-layer framework and discussed CQV parameters will directly be used in Chapter 3, where a channel selection mechanism is presented. Not all protocols and subsystems depicted in Figure 2.20 will be treated, however, every realization in this dissertation is built with the discussed framework in mind: the protocols and systems that will be designed are implemented in a modular way, such that they may be added to existing networking stacks with minimal interventions in the stack. In Chapter 3, an interface and channel selection component is presented, and it is discussed how a traffic, interface and channel monitoring component might increase the reliability of the interface selection decisions through interaction with the CQV parameters. Furthermore, a throughput capacity estimation technique that could serve as strategy for link quality monitoring is presented, and in Chapter 4, a modular approach for automatic device and network configuration is developed. While the throughput capacity estimation and auto-configuration strategies are developed as separate subsystems of the node architecture, these different realizations could in theory be combined.

2.6 Conclusion

Despite the fact that wireless ad-hoc networks have been a popular research topic for many years, they are seldom used in our everyday life. In this chapter, an analysis was made of reasons contributing to this discrepancy. Because wireless spectrum is a scarce and shared resource, wireless links are less stable and of lower quality compared to wired links. Furthermore, the scalability of traditional single-interface multi-hop ad-hoc networks is limited as each additional hop in the network results in additional interference.

Research results are often purely theoretical or obtained via simulations based on simplified models. While many suggestions for the improvement of ad-hoc networks are presented in literature, the practical realization of stable ad-hoc network topologies on top of several types of Wi-Fi hardware was found to be more challenging than expected. During various experiments under controlled circumstances, it was found that assumptions that are made when designing algorithms for wireless ad-hoc networks may not be valid when deploying solutions on the selected COTS Wi-Fi hardware; The communication range of ad-hoc nodes cannot be represented through a simple on/off state because the quality of a link is not constant and relies on the physical data rate in use. Measurements were performed to accurately model the relation between received signal strength and maximal achievable data rate.

Although a fitted shadowing model was proven to provide a reasonable estimation of the received signal quality, it was also shown that there is not a strict relation between transmitter distance and received signal strength, as attenuations and reflections in the wireless network environment lead to expected and less expected signal quality variations. Furthermore, in the considered test set-ups, adapting transmission power was found to be a necessity rather than a measure of freedom in order to guarantee network stability, especially when developing multi-channel multi-interface solutions. Moreover, it was argued that errors introduced by low quality hardware impede the development of wireless ad-hoc network protocols that can be used on a wide range of devices.

On one hand, these results and experiments indicate that making absolute performance claims based on simulation results only is dangerous and may lead to faulty conclusions. As a consequence, promising theoretical results do not always lead to corresponding advantages when a solution is deployed in reality. On the other hand, it was observed that a single successful wireless ad-hoc network deployment in a laboratory environment does not guarantee the stability of a solution in another environment.

It was argued that the specific characteristics of hierarchical wireless mesh networks help to relax the discovered ad-hoc issues, and a loosely coupled cross-layer design was proposed as a way of realizing the wireless mesh networking node architecture.

With the above conclusions in mind, protocols for wireless mesh routers will be developed in the next chapters. In contrast with the research approach of many wireless researchers, these protocols will not be developed solely based on theoretical models, but will also follow a bottom-up approach where the observed limitations of the selected ad-hoc enabling Wi-Fi hardware are taken into account. By observing the behavior of the developed wireless solutions in experimental set-ups, practical deployment issues can be identified. As such, these protocols are expected to better cope with the dynamics of wireless network environments and may prove to be the missing link to make ad-hoc networks become part of our lives.

References

[1] Dragor Niculescu, Samrat Ganguly, Kyungtae Kim, and Rauf Izmailov. *Performance of VoIP in a 802.11 Wireless Mesh Network*. In Proceedings of INFOCOM, pages 1–11, April 2006.

[2] Vineet Srivastava and Mehul Motani. *Cross-layer design: a survey and the road ahead*. Communications Magazine, IEEE, 43(12):112–119, Dec. 2005.

[3] Imrich Chlamtac, Marco Conti, and Jennifer J.-N. Liu. *Mobile ad hoc networking: imperatives and challenges*. Ad Hoc Networks, 1(1):13–64, 2003.

[4] Vijay T. Raisinghani and Sridhar Iyer. *Cross-layer design optimizations in wireless protocol stacks*. Computer communications, 27(8):720 – 724, May 2004.

[5] Christos Verikoukis, Luis Alonso, and Giamalis. *Cross-Layer Optimization for Wireless Systems: A European Research Key Challenge*. Global Communications Newsletter, pages 1–3, July 2005.

[6] Harald T. Friis. *A Note on a Simple Transmission Formula*. Proceedings of the IRE, 34(5):254–256, 1946.

[7] Jochen Schiller. *Mobile Communications*. Addison Wesley, 2003.

[8] WiFi Alliance. *Home Page*. http://www.wi-fi.org/.

[9] IEEE Standard. *IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*.

[10] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. *MACAW: A Media Access Protocol for Wireless LAN's*. pages 212–225, 1994.

[11] Joao Luis Sobrinho, Roland de Haan, and José Brázio. *Why RTS-CTS is not your ideal wireless LAN multiple access protocol*. In Proceedings of WCNC'05, March 2005.

[12] Peter Thornycroft. *Designed for speed- Network Infrastructure in an 802.11n World*. Technical report, Aruba Networks, 2007.

[13] Paramvir Bahl, Ranveer Chandra, and John Dunagan. *SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks*. In MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking, pages 216–230, New York, NY, USA, 2004. ACM.

[14] Songwu Lu, Vaduvur Bharghavan, and Rayadurgam Srikant. *Fair scheduling in wireless packet networks*. In SIGCOMM '97: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication, pages 63–74, New York, NY, USA, 1997. ACM.

[15] Raffaele Bruno, Marco Conti, and Enrico Gregori. *Mesh Networks: Commodity Multihop Ad Hoc Networks*. IEEE Communications Magazine, pages 123 – 131, March 2005.

[16] Joshua Robinson, Konstantina Papagiannaki, Christophe Diot, Xingang Guo, and Lakshman Krishnamurthy. *Experimenting with a Multi-Radio Mesh Networking Testbed.* In WiNMee - 1st workshop on Wireless Network Measurements, Riva Del Garda, Italy, January 2005.

[17] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris. *Performance of multihop wireless networks: shortest path is not enough*. SIGCOMM Comput. Commun. Rev., 33(1):83–88, 2003.

[18] Danilo Valerio, Fabio Ricciato, and Paul Fuxjaeger. *On the feasibility of IEEE 802.11 multi-channel multi-hop mesh networks*. Computer Communications, 31(8):1484 – 1496, 2008.

[19] Vivek Shrivastava, Dheeraj Agrawal, Arunesh Mishra, Suman Banerjee, and Tamer Nadeem. *On the (in)feasibility of fine grained power control*. SIGMOBILE Mob. Comput. Commun. Rev., 11(2):65–66, 2007.

[20] David Kotz, Calvin Newport, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliott. *Experimental evaluation of wireless simulation assumptions*. In MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, pages 78–82, New York, NY, USA, 2004. ACM Press.

[21] Jean-Michel Dricot and Philippe De Doncker. *High-accuracy physical layer model for wireless network simulations in NS-2*. In International Workshop on Wireless Ad-Hoc Networks, pages 249–253, 31 May -3 June 2004.

[22] John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun chan Lan, Ya Xu, Wei Ye, Deborah Estrin, and Ramesh Govindan. *Effects of detail in wireless network simulation*, 2001.

[23] Robert Bultitude. *Measurement, characterization and modeling of indoor 800/900 MHz radio channels for digital communications*. Communications Magazine, IEEE, 25(6):5 – 12, jun 1987.

[24] David Plets, Emmeric Tanghe, Kris Vanhecke, Tom Deryckere, Wout Joseph, Luc Martens. *DEUS WP4: Planning tool*. Presented at the fifth IBBT-DEUS plenary, june 2009.

[25] Ram Ramanathan and Regina Rosales-Hain. *Topology control of multihop wireless networks using transmit power adjustment*. In Proceedings of INFOCOM. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, volume 2, pages 404–413 vol.2, August 2000.

[26] Eun-Sun Jung and Nitin H. Vaidya. *Power Aware Routing using Power Control in Ad Hoc Networks*. Technical report, CSL, University of Illinois, Urbana, February 2005.

[27] Qunfeng Dong. *Maximizing system lifetime in wireless sensor networks*. In IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks, page 3, Piscataway, NJ, USA, 2005. IEEE Press.

[28] Vikas Kawadia and Panganamala Ramana Kumar. *Principles and Protocols for Power Control in Wireless Ad Hoc Networks*. IEEE Journal on Selected Areas in Communications, 23(1):76–88, January 2005.

[29] Jian Tang, Guoliang Xue, Christopher Chandler, and Weiyi Zhang. *Link Scheduling with Power Control for Throughput Enhancement in Multihop Wireless Networks*. In Proceedings of the Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, page 1, Washington, DC, USA, 2005. IEEE Computer Society.

[30] Fehmi Ben Abdesslem, Iannone Luigi, Marcelo Dias de Amorim, Konstantin Kabassanov, and Serge Fdida. *On the feasibility of power control in current IEEE 802.11 devices*. In Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on, pages 5 pp.–473, March 2006.

[31] Jian Tang, Guoliang Xue, and Weiyi Zhang. *Interference-Aware Topology Control and QoS Routing in Multi-Channel Wireless Mesh Networks*. In Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, pages 68–77, Urbana-Champaign, IL, USA, May 2005.

[32] Pradeep Kyasanur and Nitin H. Vaidya. *Routing and interface assignment in multi-channel multi-interface wireless networks*. In Wireless Communications and Networking Conference, 2005 IEEE, volume 4, pages 2051–2056 Vol. 4, New Orleans, LA, USA, 2005.

[33] Wim Vandenberghe, Kristof Lamont, Ingrid Moerman, and Piet Demeester. *Connection Management over an Ethernet based Wireless Mesh Network*. In WRECOM, Wireless Rural and Emergency Communications Conference, Rome, Italy, February 2007.

[34] Chen-Mou Cheng, Pai-Hsiang Hsiao, Hsiang-Tsung Kung, and Dario Vlah. *Adjacent Channel Interference in Dual-radio 802.11a Nodes and Its Impact on Multi-hop Networking*. In IEEE Global Telecommunications Conference, 2006., pages 1 –6, Nov 27 -Dec. 1 2006.

[35] Arogyaswami J. Paulraj, Dhananjay A. Gore, Rohit U. Nabar, and Helmut Bölcskei. *An overview of MIMO communications - A key to Gigabit wireless*. Proceedings of the IEEE, 92(2):198–218, February 2004.

[36] Regis J. Bates and Donald W. Gregory. *Voice & data communications handbook*, chapter Modulation and Multiplexing, pages 143–178. McGraw-Hill Companies, Tow Penn Plaza, New York, 2007.

[37] Vivek Shrivastava, Shravan Rayanchu, Jongwoon Yoonj, and Suman Banerjee. *802.11n under the microscope*. In IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, pages 105–110, New York, NY, USA, 2008. ACM.

[38] Motorola. *802.11n Demystified: Key Considerations for n-abling the Wireless Enterprise*. White Paper, 2009.

[39] Allen Miu, Hari Balakrishnan, and Can Emre Koksal. *Improving loss resilience with multi-radio diversity in wireless networks*. In MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking, pages 16–30, New York, NY, USA, 2005. ACM.

[40] PC Engines. *Alix system board*. http://www.pcengines.ch/alix.htm.

[41] Madwifi. *Multiband Atheros Driver for Wifi*. http://madwifi.org/.

[42] 4G Systems. *Home Page*. www.4g-systems.com.

[43] Tadeusz A. Wysocki and Hans-Jürgen Zepernick. *Characterization of the indoor radio propagation channel at 2.4 GHz*. Journal of Telecommunications and Information Technology, 1(3-4):84–90, 2000.

[44] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. *Wireless mesh networks: a survey*. Computer Networks, 47(4):445–487, 2005.

[45] WiMAX Forum. *Home Page*. http://www.wimaxforum.org/.

[46] IEEE Task Group S. *Meetings update*. http://www.ieee802.org/11/Reports/tgs_update.htm.

[47] Joseph D. Camp and Edward W. Knightly. *The IEEE 802.11s Extended Service Set Mesh Networking Standard*. IEEE Communications Magazine, 46(8):120–126, August 2008.

[48] Michael Bahr. *Proposed routing for IEEE 802.11s WLAN mesh networks*. In WICON '06: Proceedings of the 2nd annual international workshop on Wireless internet, page 5, New York, NY, USA, 2006. ACM.

[49] Michael Bahr. *Update on the Hybrid Wireless Mesh Protocol of IEEE 802.11s*. In Proceedings of the IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, MASS, pages 1–6, 2007.

[50] Xudong Wang and Azman O. Lim. *IEEE 802.11s wireless mesh networks: Framework and challenges*. Ad Hoc Networks, 6(6):970 – 984, 2008.

[51] Myung Lee, Rui Zhang, Chunhui Zhu, Tae Rim Park, Chang-Sub Shin, Young-Ae Jeon amd Seong-Hee Lee, Sang-Sung Choi, Yong Liu, and Sung-Woo Park. *Meshing Wireless Personal Area Networks: Introducing IEEE 802.15.5*. IEEE Communications Magazine, 48(1):54–61, January 2010.

[52] Vikas Kawadia and P. R. Kumar. *A cautionary perspective on cross-layer design*. IEEE Wireless Communications, 12:3 – 11, February 2005.

[53] Sanjay Shakkottai, Theodore S. Rappaport, and Peter C. Karlsson. *Cross-layer Design for Wireless Networks*. IEEE Communications Magazine, October 2003.

[54] Sally Floyd. *TCP and Explicit Congestion Notification*. ACM Computer Communication Review, 24(5):10–23, 1994.

[55] Antonio G. Ruzzelli, Gregory M. P. O'Hare, and Raja Jurdak. *MERLIN: Cross-layer integration of MAC and routing for low duty-cycle sensor networks*. Ad Hoc Networks, 6(8):1238–1257, 2008.

[56] Vijay T. Raisinghani and Sridhar Iyer. *Cross-layer Feedback Architecture for Mobile Device Protocol Stacks*. IEEE Communications magazine, January 2006 Volume 44, pages 85–92, 2006.

[57] Marco Conti, Gaia Maselli, Giovanni Turi, and Silvia Giordano. *Cross-Layering in Mobile Ad Hoc Network Design*. IEEE Computer, 37(2):48–51, 2004.

[58] Karim M. El Defrawy, Magda S. El Zarki, and Mohamed M. Khairy. *Proposal for a cross-layer coordination framework for next generation wireless systems*. In IWCMC '06: Proceeding of the 2006 international conference on Communications and mobile computing, pages 141–146, New York, NY, USA, 2006. ACM Press.

[59] Rolf Winter, Jochen H. Schiller, Navid Nikaein, and Christian Bonnet. *CrossTalk: Cross-layer decision support based on global knowledge*. IEEE Communications magazine, 44(1), January 2006.

3 Multi-Channel, Multi-Interface Wireless Mesh Networks

3.1 Introduction

In the previous chapter, it was shown that after years of wireless ad-hoc research, many issues remain which can make the deployment of wireless ad-hoc networking protocols on top of Wi-Fi hardware a challenging task. The observations led to conclusions about the factors contributing to the limited success of ad-hoc networks. Several possible optimizations were identified, including the specification of a cross-layer optimized wireless mesh network architecture.

Among many issues, it was indicated how the limited performance of single-channel multi-hop communication quickly reduces the available end-to-end bandwidth when the number of hops increases. Therefore, this chapter focuses on developing protocols for the support of wireless multi-interface, multi-channel networks that fit in the developed wireless mesh network architecture. As a use case, the set-up of emergency networks is selected.

More specifically, first, a low-overhead, fast interface and channel selection protocol for the wireless backbone routers from the example emergency scenario of Section 2.5.3 is developed. The protocol automatically provides a fast initial channel configuration of mesh backbone routers mounted on emergency vehicles arriving at a disaster scene in a distributed way. This protocol, called FRESME (FREquency Selection based on Message Exchange), collects information on the channel conditions in the node's environment, and optimizes the local spectrum us-

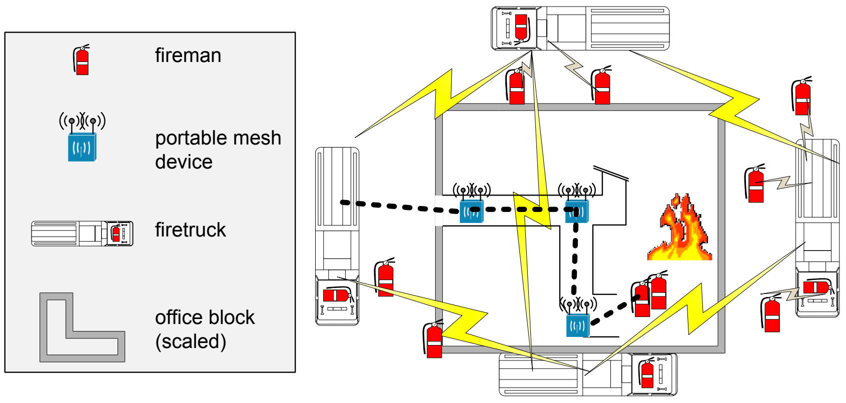


Figure 3.1: Combined use case for the developments considered in this chapter. (Outdoor:) interface and channel selection between mesh routers mounted atop firetrucks arriving at the disaster scene. (Indoor:) End-to-end throughput capacity estimation for the path between the firemen and firetruck.

age at the mesh backbone routers by dynamically distributing wireless data traffic streams across the available interfaces and communication frequencies. FRESME enables the use of multiple wireless network interfaces with minimal adjustments to a single-interface networking stack, and is fully transparent to the routing protocol and upper networking layers.

Second, a monitoring technique providing end-to-end throughput capacity estimation in multi-interfaced wireless networks is developed and evaluated. Multi-hop throughput capacity estimation may also be used during emergency situations; for example, when firemen enter complex environments such as a collapsed building, an underground car parking or a ship, a single hop radio connection may no longer suffice to provide communication with the commanding officer outside the building. In these cases, a multi-hop connection can be used to connect the firemen inside a complex structure with the outside world. An estimation of the end-to-end throughput is then important to judge the quality that can be expected from the wireless link between

The above two scenarios are summarized in Figure 3.1. In this combined scenario, a large office building is on fire. As soon as the fire trucks arrive at the scene, a robust multi-interface, multi-channel mesh network is formed between the fire trucks, in order to provide a high bandwidth network for the rescue workers at the incident scene. In this backbone mesh network, the channel configuration is performed by the FRESME protocol. Furthermore, some employees were not able to evacuate on time. As such, the firemen decide to enter the burning building. Because there are a lot of concrete walls inside, no connection to the outside world

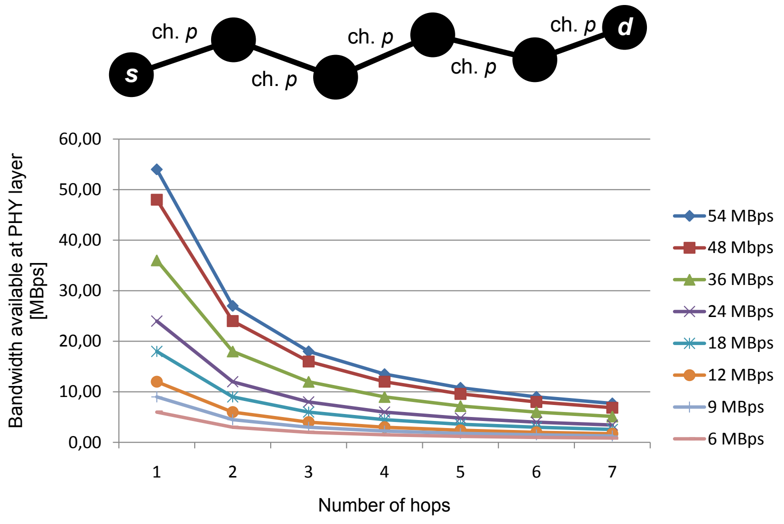


Figure 3.2: IEEE 802.11a/g multi-hop network using a single communication channel p. All nodes are within each other's interference range. Physical layer bandwidth available per wireless link in a multi-hop network for different data rates and increasing number of interfering hops.

is possible using standard communication radios. Therefore, at strategic points within the structure, portable mesh routers are deployed. As new portable devices are deployed in the network, a low-overhead capacity estimation technique, running in the background, determines the quality that can be expected from the network. While not shown on this figure, one or several firetrucks might also provide an uplink to the fire station, or, in case of a bigger disaster, towards a crisis center.

Before the newly developed protocols are treated in detail, the next section provides a more detailed investigation of the performance problems of single-interface multi-hop networks. Furthermore, related work in the field of multi-interface, multi-channel multi-hop networks is discussed.

3.1.1 The capacity of multi-hop wireless networks

As previously stated, single interface wireless networks are fundamentally limited in terms of throughput and delay. This is illustrated with the help of Figure 3.2, showing an example multi-hop network in a simple line topology and a throughput graph for different physical layer settings of an IEEE 802.11a/g network. In this illustrative example, nodes are placed close to each other, as to enable communication at the highest physical layer data rate; as explained in the previous chapter, in order to successfully decode a transmitted signal at a receiving node, a sufficiently high SNR is required. Unfortunately, in this case, the resulting interference range stretches far beyond the intended receiver of the packet. Thus, because of this node density, all nodes are within each other's interference range. Since only one node can transmit at any given time, the available wireless bandwidth is at best divided fairly between the different links. The graph shows that the available bandwidth at physical layer for an increasing number of hops quickly reduces when the number of hops increases. This issue is especially relevant when dimensioning a wireless network to operate at the highest physical layer data rate of $54\,MBps$. When operating links at lower data rate, packets can be received at a larger distance of the transmitters, therefore, in the considered example topology, it would be possible to transmit a packet from source to destination in fewer hops. As such, in this illustrative scenario, the relative throughput reduction is in practice less likely to be as drastic for low data rate multi-hop links as it is for high data rate multi-hop links.

The multi-hop throughput degradation is not only an issue in single-interface networks. In literature, several authors have captured the effects of multi-hop communication on the throughput and/or delay performance of wireless networks [1–5]. To this end, a performance metric called *capacity* has been introduced. As indicated by [6], the 'capacity' term may be defined differently in different research fields and is only meaningful with respect to a specific communication layer. In this chapter, the terms *channel capacity*, *end-to-end throughput capacity* and *network capacity* and are used as follows;

The capacity of a wireless *channel* is measured in bits per second, indicating the actual capabilities of the wireless channel to transport traffic at a given time. Using this terminology, if the physical transmission rate is locked at $54\,Mbps$ in the example of Figure 3.2, the capacity of channel p is $54\,Mbps$. When node s sets up a continuous data stream to node d in a multi-hop way, multi-hop interference causes channel p to be divided in 5 logical subchannels, each with a capacity $\frac{54}{5}\,Mbps = 10.8\,Mbps$. As such, in this example, the end-to-end *throughput capacity* at physical layer from node s to note d is limited to $10.8\,Mbps$.

Similarly, the capacity of a wireless *network* measures the performance of the entire wireless network, and is expressed in bits per second per source-destination pair, or bit-meters per second if the location of the nodes is taken into account. In [1], Gupta and Kumar determine the capacity of a wireless network built out of uniformly distributed single-interfaced static nodes with random source-destination pairs. They show that as the number of nodes per area n increases, the available throughput per source-destination pair scales proportionally to $\frac{1}{\sqrt{n}}$. Consequently, this model indicates that for large numbers of nodes, the throughput per node pair approximates zero.

In [2], Grossglauser and Tse study the same capacity problem, but now for mobile wireless nodes with loose delay constraints, tolerating delays of minutes or

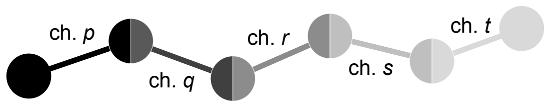


Figure 3.3: IEEE 802.11a/g multi-hop network using multiple interface and multiple communication channels p, q, r, s and t. The wireless interfaces are tuned to non-interfering wireless channels, avoiding interference between the individual links.

even hours. Under these conditions, they show that wireless networks may have a constant throughput per node pair with an increasing number of nodes.

The authors of [3] have further refined the conclusions of the previous authors by better defining delay in the network and capture the Gupta & Kumar and Grossglauser & Tse models in a unified framework.

The above models show that in a single-interface wireless multi-hop network, throughput can be traded for delay and vice versa. However, many situations exist in which both low delay and high bandwidth networks are required. For these situations, multi-interface, multi-channel networks provide a solution. Figure 3.3 depicts a multi-interface, multi-channel version of the simple chain topology from Figure 3.2. By tuning the different links to different, non-interfering wireless channels, multi-hop interference is avoided. Under the same idealized assumptions as before, and additionally assuming 5 fully non-interfering wireless channels p, q, r, s and t to be available, the capacity of each individual channel is $54\,Mbps$. This time, the theoretical end-to-end physical layer throughput from node s to node d remains undiminished at $54\,Mbps$.

3.1.2 Channel selection in wireless networks: related work

In order to design robust broadband wireless mesh networks, multi-interface nodes and channel selection schemes have been explored in the past, both in wireless infrastructure networks as in wireless mesh networks. The key challenge for these different protocols is identical: to configure wireless interfaces and wireless channels in such way that network performance is optimized, either in terms of throughput, delay, power consumption, or other metrics.

In infrastructure wireless networks, the individual single-interface wireless access points are interconnected over a wired network (Figure 3.4). Several channel optimization approaches for wireless infrastructure networks exist. In the simplest versions, all access points are owned by a single administrator, and no other setups are interfering. The channel selection protocol needs to provide a one-time static channel configuration in such way that spatial reuse of the frequencies is optimal, given a fixed location of the nodes. The problem grows more complex as the influence of external interfering networks, long term traffic profiles, or varying

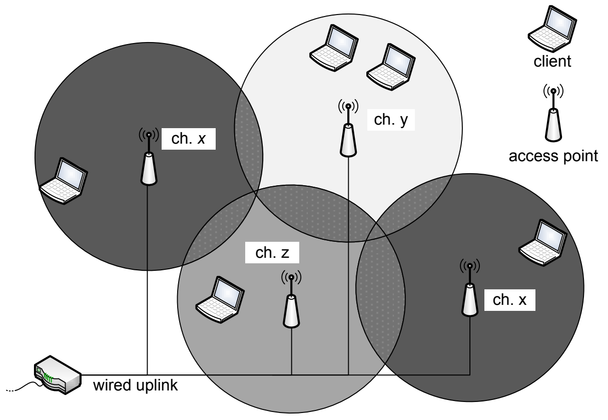


Figure 3.4: Channel selection in wireless infrastructure networks: neighboring access points are configured to non-overlapping channels in order to avoid interference.

load conditions are taken into account [7, 8]. There are as many solution methods for this problem as there are variations on the scenario: channel selection in infrastructure wireless networks is performed through graph coloring [9, 10], game theory [11], by scanning the environment for the least congested channel [12], or based on interference measurements [13]. Since client nodes generally connect to their access point using a network name, there is no risk for prolonged connectivity loss when the channel of an access point is changed: when a client detects that the access point is no longer available, it scans all channels in search for the access point. As soon as it is found, the connection is recovered.

Changing channels in a wireless mesh network is considerably more complex. Since wireless mesh backbone routers may have multiple interfaces and possibly maintain multiple links with several other mesh routers, changing the channel for one wireless link might result in other wireless links getting lost. For example, consider the scenario depicted in Figure 3.5. As the fire truck equipped with a single interface mesh router E approaches its planned location marked with an $\times$-sign, new wireless links need to be configured. It is reasonable to assume that on arrival of the rescue teams, the rescue workers will be uniformly distributed over the disaster area. As such, peer to peer traffic between the different teams is expected, resulting in both short and long paths through the network, dividing the load across the different backbone links. If it were up to node C to decide which channel to use under these assumptions, the dark gray diagonally striped channel z would be chosen, since the black channel x is already used for two wireless links, and a third interface is not available at node C to tune to yet another channel. However, node E observes heavy interference on channel z caused by an interferer

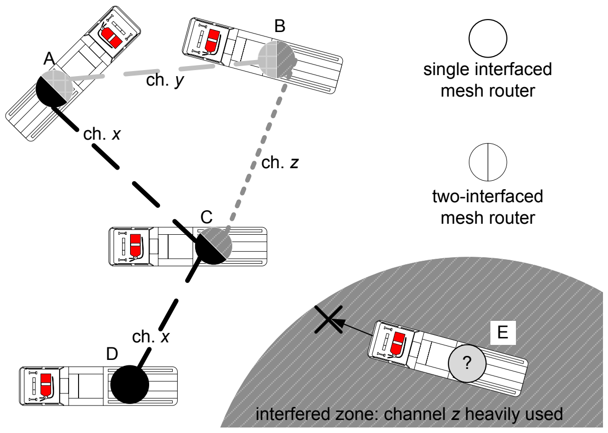

A
B
C
D
E
ch. y
ch. x
ch. z
ch. x
?
single interfaced mesh router
two-interfaced mesh router
interfered zone: channel z heavily used

Figure 3.5: Channel selection in wireless mesh networks. A firetruck, equipped with a single-interfaced mesh router, E approaches its planned location, requiring wireless links to be configured.

unnoticeable by C. As a result, node E might request that the second interface of node C changes its interface from channel z to the light gray channel y. However, this would result in link $A-B$ and link $B-C$ sharing the same channel y; unless link $A-B$ is reconfigured to operate on channel z. As can be seen even in this simple scenario with only a limited number of nodes, a decision at one location in the network might require changes at another location in the network. Since no wired channels are available, the situation is further complicated as the continuity of a network connection might be jeopardized in case of synchronization errors.

Despite the complexities, many authors have engaged in channel selection techniques for wireless multi-interface networks [14]. These approaches are classified in three ways. One option is to classify the mechanisms based on the input data source: while some algorithms make decisions based on measurements such as round-trip latency [15], others make status based decisions, for example by monitoring the exchange of control packets [16].

Second, mechanisms can be classified as distributed or centralized. In distributed schemes such as [13], the decision to configure a wireless link to a particular channel is made locally at the different nodes participating in the network. In centralized channel selection schemes, the channel configuration for the entire network is determined at a single node. For example, in [17], based on the location and traffic profile between each pair of nodes, a channel is assigned to each link and the optimal routes inside the network are determined, with the goal of optimizing the useful bandwidth inside the network. Some of these algorithms, like [18], presume a complete and perfect knowledge about the network state in terms of topology, transmission power, traffic load or other parameters.

Finally, channel selection research can be classified based on feasibility using current generation commodity hardware: as noted above, some works are purely theoretical. Although this type of research is interesting in order to get a better understanding of the impact of certain network parameters, without additional modifications, the resulting solutions may not always be able to –or may not be designed to– dimension dynamic wireless networks under real-life conditions in the short term; The used propagation models may not correspond with reality and therefore may be less feasible to result in real-life deployments, the computation complexity may be too high to support dynamic networking environments, or certain input variables for the models may be unavailable. For example, for the previously mentioned approach in [17] to work, traffic profiles between each pair of nodes should be known centrally. This assumption might not be realistic in every deployment.

Within the context of this dissertation, where real-life implementation of protocols is considered to be important to discover and overcome practical issues, a channel selection scheme is developed that can be implemented on top of COTS Wi-Fi hardware without requiring drastic hardware re-design. As such, the goal of the channel selection protocol to be developed is not to claim theoretical optimality, but to design and evaluate a feasible channel selection protocol. This choice should not be interpreted as a statement to criticize theoretical approaches: both theoretical and practical designs are of importance, and different boundary conditions apply.

3.2 Frequency selection based on message exchange

3.2.1 Design considerations

Keeping in mind the observations from Chapter 2, following design goals are set for the FRESME dynamic channel selection protocol for wireless multi-interface backbone routers. First, implementing the protocol should be possible using the COTS Wi-Fi hardware that was evaluated in the previous chapter.

Second, as tens of routing protocols for wireless ad-hoc and mesh networks are already available and new protocols are continuously being developed [19], the goal is to develop a channel selection protocol which can be used in combination with a broad range of wireless routing protocols. In addition, no support for multi-interface networks is expected from the routing protocol.

Third, the presented emergency scenario is dynamic in nature, and demands a very fast way of providing initial channel configuration on arrival of new mesh routers at the incident scene. Furthermore, since communication in an emergency situation may save lives, an instantly available, highly reliable, high quality wireless network is expected. However, given the dynamic and time-varying nature of the network, in contrast with related work, the FRESME protocol cannot make use of a centralized control entity, long-term traffic profiles, nor assume complete knowledge of all network parameters at all times. Therefore, the protocol should operate in a distributed way.

Fourth, the overhead of the solution is to be kept at a minimum.

3.2.2 Design principles and assumptions

In several of the cited channel selection approaches, the channel configuration of new nodes joining the network is dictated by nodes already available in the network. In case of dynamic channel selection in wireless infrastructure networks, access points often decide for themselves which channel to select after a scanning procedure. Neither of these solutions can avoid the configuration issues that were illustrated in 3.5 when node E was joining the network. When the channel to be used on a new link is dictated by the existing nodes, the medium occupation as observed by the new node is not taken into account. Besides, a new node may not be aware of the network state of the existing network.

However, the operation of the IEEE 802.11 protocol requires favorable channel conditions both at sender and receiver side: when a unidirectional packet is successfully transferred from sender to receiver, an ACK packet is immediately sent on the same channel in the other direction. As such, to enable packet reception, interference on the channel should be avoided both at the sender and the receiver side. When the interference is too high at at least one side, either the data packet itself will get lost, or the ACK packet will not be received. In the latter case, the lost ACK packet is interpreted by the sender as packet loss, resulting in packet retransmission and thus causing additional delays and interference.

While scanning procedures may be used to collect information on the state of neighboring nodes, a short scanning procedure might not produce an accurate view on the environment. Besides, in emergency situations, there is no time for a long scanning procedure. Furthermore, the results from long scanning procedures might not be meaningful because of the dynamic nature of the network. Therefore, the FRESME protocol initially builds a local view on the network by monitoring control packets, and the channel configuration for a new link is decided upon in mutual agreement between the two nodes configuring the wireless link. Rather than estimating the channel state of a corresponding node, the FRESME protocol simply asks for the corresponding node's most recent view on the environment using a three way handshake. A similar approach is found in [16], where channel selection is performed for multi-channel single-interface networks by modifying the IEEE 802.11 RTS/CTS messages (cf. Section 2.2.2). The presented protocol is also partially inspired by the RTS/CTS mechanism. However, the FRESME protocol targets multi-interface nodes, implements its own control messages, and does not require changes to the IEEE 802.11 MAC itself.

In order to limit the overhead of the handshake approach, wireless links are configured on demand, only when data traffic is to be sent. In the next sections, it will be detailed how the handshake packets contain the channel quality vectors (CQVs) previously introduced in Section 2.5.4.2. Based on the CQVs of the two nodes starting a new wireless link, the 'best' common channel is determined. The new channel is then reserved for as long as data traffic is sent in between the nodes.

Although the channel selection protocol can be used in combination with any radio technology, the current implementation of FRESME targets IEEE 802.11b/g devices, given their availability at the time of implementation. Because only three non-overlapping channels are available in IEEE 802.11b/g networks, an identical and fixed channel to interface assignment at every node is assumed. All mesh backbone routers are equipped with three interfaces, tuned to wireless communication channels 1, 6, and 11. In the FRESME protocol, as new wireless links are initiated, they are automatically attributed to one of the available interfaces, and thus to the corresponding frequencies. In [17], Raniwala *et al.* correctly argue that fixed channel to interface assignment schemes are less optimal compared to channel selection schemes which dynamically reconfigure the channel setting at the different interfaces. However, for IEEE 802.11b/g based implementations where every node has three interfaces, the chosen fixed channel configuration of the interfaces has no negative impact on the use of the spectrum, since the entire available spectrum is covered by the three channels. More importantly, the fixed channel assignment of the interfaces is strictly a choice used to simplify implementation and to describe the operation of the protocol more clearly. In Section 3.2.5, the fixed channel assignment scheme is extended in order to transparently support dynamic channel assignment in combination with any number of wireless interfaces and wireless channels.

3.2.3 Node architecture

3.2.3.1 Overview

Figure 3.6 depicts the three-interface mesh node architecture used in the implementation of the FRESME protocol. The architecture fits into the cross-layer framework previously defined in Section 2.5.4.2, and requires only minimal adaptations to the networking stack. In order to support multiple routing protocols, the packet flow that is normally followed in the OSI stack is interrupted at MAC layer. Every node in the network has a single IP address, but has multiple network interfaces which are hidden from the upper part of the networking stack by a *multiplexer/demultiplexer* (MUX/DEM) unit. The MAC addresses PHY1, PHY2 and PHY3 of a single device are interrelated, enabling two-way MAC address translation for outgoing as well as incoming packets. When IP packets are sent in a

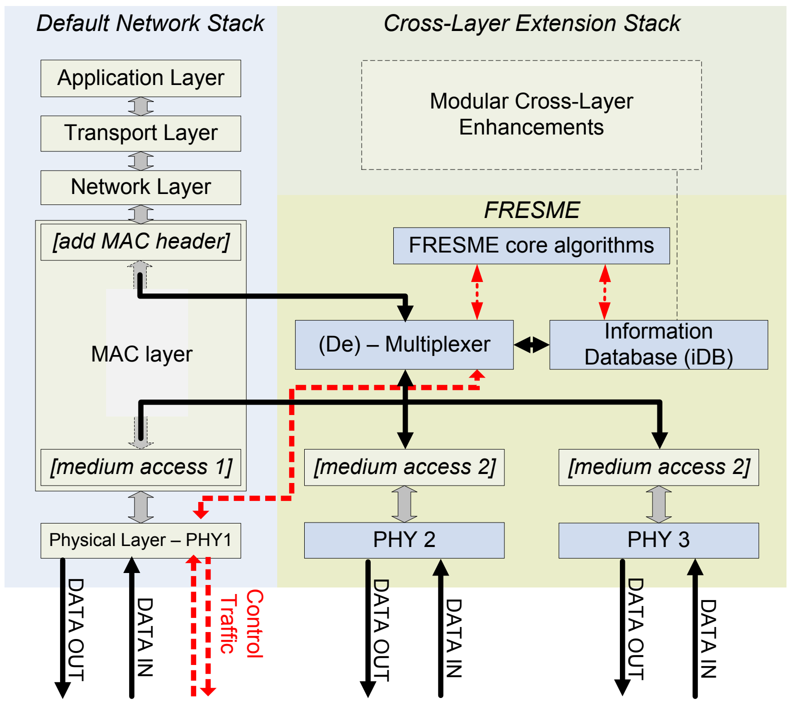


Figure 3.6: FRESME mesh node architecture. The channel selection functionality is implemented in an individual subsystem, requiring minimum intervention in the default networking stack.

default networking chain, data is first encapsulated in an IP header. Then, a MAC header is added before the packet is queued and subsequently transmitted by the physical layer. The MUX/DEM modifies this behavior: after the MAC header is added to an IP data packet, the normal routine of the MAC layer is interrupted, and the packet is sent to the MUX/DEM unit.

The multiplexer unit takes the next hop MAC address from the packet, and verifies whether the address is known by the *information database* (iDB). This iDB holds a table which binds destination MAC addresses to outgoing interfaces, thus performing channel selection on a per-link basis. The iDB is built and maintained by the channel configuration protocol within the *FRESME core algorithms* block. The protocol is detailed in Section 3.2.4, and optimizes the local spectrum usage at a node. When sending or receiving packets, the MUX/DEM translates the MAC addresses in such way that from the point of view of the routing protocol, all packets are sent and received using a single wireless interface. In reality, the packets are transmitted over different interfaces.

The high-level system architecture presented in [15] shows similarities with the

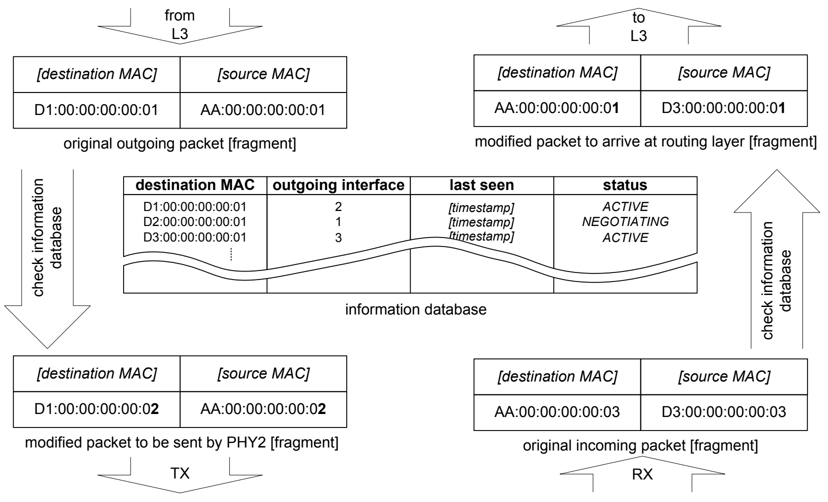

from L3
[destination MAC]
[source MAC]
D1:00:00:00:00:01
AA:00:00:00:00:01
original outgoing packet [fragment]
to L3
AA:00:00:00:00:01
D3:00:00:00:00:01
modified packet to arrive at routing layer [fragment]
check information database
destination MAC
outgoing interface
last seen
status
D1:00:00:00:00:01 2 [timestamp] ACTIVE
D2:00:00:00:00:01 1 [timestamp] NEGOTIATING
D3:00:00:00:00:01 3 [timestamp] ACTIVE
information database
D1:00:00:00:00:02
AA:00:00:00:00:02
modified packet to be sent by PHY2 [fragment]
TX
AA:00:00:00:00:03
D3:00:00:00:00:03
original incoming packet [fragment]
RX

Figure 3.7: MAC header translation by the multiplexer/demultiplexer component at the node with primary MAC address AA:00:00:00:00:01. The network layer is not aware that multiple PHY interfaces are used. MAC addresses are translated based on the contents of the information database.

FRESME architecture: no changes are required to the IEEE 802.11 MAC and the use of multiple network interfaces is hidden from upper network layers by using a demultiplexer software component. The authors perform intelligent channel selection based on a *channel quality metric*, which is determined by sending probes on a periodic basis. The channel decisions are then made at a node locally, without the need for agreement between sender and receiver. This is in contrast with the operation of the FRESME protocol, which does not require periodic probing, but only exchanges information between sending and receiving node prior to initializing a new channel reservation. As such, the presented solution imposes less overhead on the wireless network.

3.2.3.2 MAC header translation

Figure 3.7 illustrates the MAC header translation process performed by the MUX/-DEM at the mesh node with primary MAC address AA:00:00:00:00:01. In this example, the MAC address of the first wireless interface ends with :01, and the derived addresses with :02 and :03 respectively. Due to the fixed channel configuration of the interfaces in the implementation, specifying the interface implicitly specifies the chosen channel. Packets were originally sent to MAC address D1:00:00:00:00:01. This MAC address is found in the information database of the sending node; the iDB shows an active channel reservation for this destination, using outgoing interface 2. As such, the packet will be sent by the second physical interface of the source node to the second wireless interface of the destination node. The MAC address of the outgoing packet is modified accordingly to D1:00:00:00:00:02, and then sent via PHY2.

When receiving packets, similar adjustments are made: packets arriving on any of the physical interfaces are translated as if they were received by a single wireless interface. Furthermore, as packets are received, the existence of multiple remote interfaces is hidden from the routing layer by the demultiplexer.

3.2.3.3 A hybrid control channel approach

In order to have a robust mesh network, it is essential that control traffic is delivered in a reliable way. One way to increase the probability of this control traffic being delivered correctly, is to reserve a channel for control traffic specifically. Under the assumption that no interference is generated by external networks, this control channel may be fixed. When interference from external networks is expected, a scanning procedure may decide which channel to (temporarily) reserve for control traffic. In our basic implementation, the control channel is fixed to channel one. This decision is motivated by the fact that it is reasonable to assume that in the considered disaster scenario, most existing infrastructure is destroyed. Furthermore, this assumption makes the implementation more feasible.

When few control messages are sent, a fixed control channel means a waste of spectrum. Therefore, a hybrid approach is proposed: the control channel is primarily used as a signaling channel for the channel selection protocol and for the exchange of routing protocol messages. However, the control channel may also be used for data traffic under the following two conditions:

(i) A link to a new neighbor needs to be set up and the channel reservation is not yet completed or fails. This includes failed negotiation with nodes which do not run FRESME, thus ensuring backward compatibility with non-FRESME nodes.

(ii) The non-control wireless interfaces are heavily loaded. In this case, the control channel is also used as additional data channel, as it does not make sense to receive all control traffic via an almost empty channel, while application traffic is blocked because data channels are fully occupied.

As such, this hybrid approach allows efficient spectrum occupation with priority for control traffic, while avoiding delays during the channel set-up phase or when channel negotiation fails.

3.2.4 Core algorithms

The core algorithms of FRESME are now explained in detail under the assumptions described above: every mesh router is FRESME enabled and has exactly three wireless interfaces. A first wireless interface is tuned to the control channel 1, and is used to transmit and receive all control and routing traffic, or as a data carrier under the specific conditions listed above. The second and third interface are respectively tuned to channel 6 and 11. Both interfaces will only be used to transmit and receive data traffic and the corresponding ACKs. All interfaces connect to the wireless network named $channel\_x$, x being the channel number of the interface. At this moment, the default routing protocol and addressing mechanisms of the mesh network can start operating using the default interface on channel 1.

As stated before, FRESME is an on demand protocol: as long as no traffic other than broadcast traffic –including traffic generated by the routing protocol– is sent, nothing happens. An example using a simple topology can be seen on Figure 3.8a. Before any useful data packet is sent, the routing protocol is able to build its routing tables. However, since the topology may still change, for example because an emergency vehicle is still moving into place, no reservations are made when no data traffic needs to be transported. This approach limits the channel selection control overhead and ensures that no superfluous reservations are made for wireless links that are not used. When a non-broadcast packet is detected by the multiplexer, the destination MAC address is verified against the information database. If the address is known, the packet is forwarded over the previously configured channel.

If no entry is found, the packet is forwarded over the default interface (PHY 1). Subsequently, the channel negotiation process between sending and receiving node is started in order to determine the most appropriate channel for communication between the nodes. As input for the channel selection algorithm, Channel Quality Parameters (CQPs) are used. Recall from Section 2.5.4.2 that a CQP is a dimensionless variable which acts as an inverse measure to the quality observed by a node at a specific channel, and that every node holds a vector of Channel Quality Parameters called a Channel Quality Vector (CQV). The dimension of the CQV equals the number of orthogonal channels a specific node can tune to. The lowest CQP in the CQV at a specific node indicates which channel is best suited to be used as a (new) communication channel. In our example, $CQV_A = \left[CQP_A^1, CQP_A^6, CQP_A^{11}\right]$. Channels that are reserved for the transport of data packets are initialized with CQV = 0. The default channel gets an initial penalty and is initialized with a non-zero CQV.

When running FRESME without extensions (cf. Section 3.2.5), the CQP_α^x of node α is increased (decreased) in case of following two events:

1. a destination MAC address is added to (removed from) channel x in the iDB:

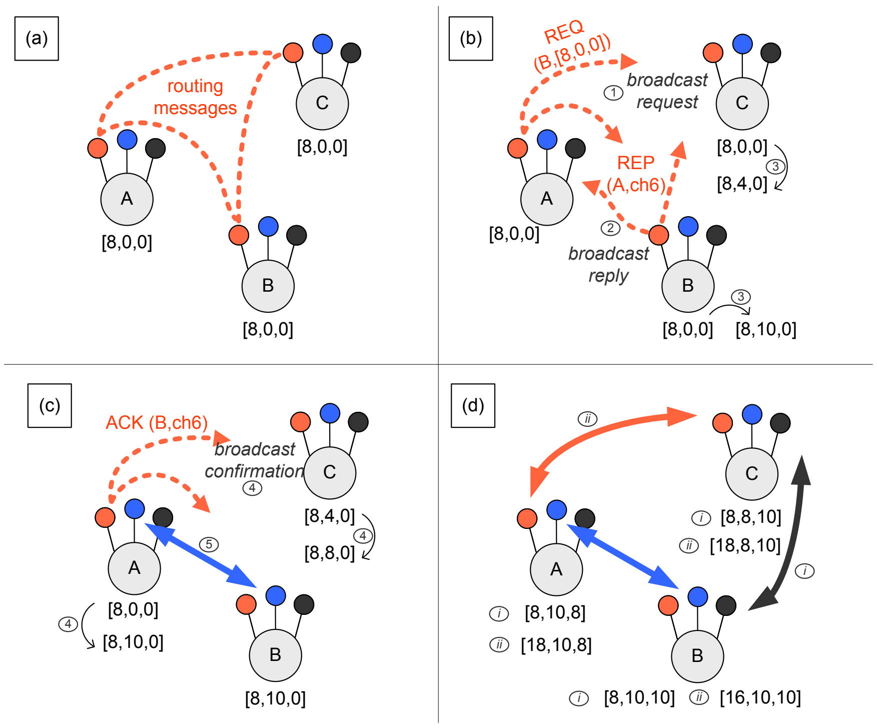


Figure 3.8: Simple topology illustrating the FRESME protocol. All nodes are within communication range. (a) Starting situation. Routing protocol messages are exchanged, CQVs are initialized. (b) Node A requests a communication channel from B. B replies with a message containing the chosen channel. The reply message is overheard by C. (c) After sending an ACK message, all traffic between node A and B will be sent on channel 6. (d) The selected channels and impact in CQVs after setup, first by selecting a channel for use between C and B, and then additionally between A and C.

+(-) *link reservation penalty*.

2. a node interfering on channel x is detected (removed):
   +(-) *interference penalty*.

For our implementation, the link reservation penalty is set to 10, and the interference penalty to 4. The rationale behind increasing (decreasing) by 10 or 4 is the following: suffering interference from one (+4) or two (+4,+4) nearby nodes on a particular channel is considered less harmful than adding an extra flow (+10) to a particular channel on the node itself. In the current configuration, +8 is selected as initialization value for the default channel: if more than 2 wireless connections per node are set up and 3 interfaces are available, the default channel (with CQP +8) is preferred above a channel that is already in use (with CQP +10) on that node.

Note that the values are not rescaled to 2, 4 and 5 in order to allow more granularity when changing the CQV using protocol extensions. Furthermore, CQPs may not be negative.

In the example of Figure 3.8, the initialization is as follows: $CQP_i^1 = 8$ and $CQP_i^{6,11} = 0, i = \{A, B, C\}$. Suppose node A needs to send data packets to B. As no entry of B's MAC address is found in A's iDB, the channel negotiation procedure is started. In Figure 3.8b, node A sends a request message indicating that it wants to negotiate with node B in order to make a channel reservation. This message contains the CQV_A vector and the intended receiver in the payload, and is sent to the broadcast address using the default interface operating at the lowest data rate. The nodes which are in communication range, i.c. node B and C, receive the request. While it is discarded by node C, the intended receiver B immediately calculates the sum of its own and the received CQV vector, $CQV_{sum} = CQV_A + CQV_B$.

Node B determines the smallest value in CQV_{sum} and sends out a broadcast reply message, containing the intended receiver A, announcing that the corresponding channel will be selected (i.c., channel 6). Node B also raises its CQP_B^6 with 10, because traffic will be sent through his interface on channel 6. On a side note: in the current protocol implementation, if a unique smallest variable cannot be found, the first minimum is selected. Large-scale simulations in Section 3.2.6 will show that this creates an unwanted bias favoring channels with a small channel number. This issue is easily resolved by upgrading the ‘first minimum’ policy with a random selection policy.

The reply message, announcing the link on channel 6 with node A, is received by node A. Node C overhears the message, learns that it can expect interference on channel 6 and increases its CQP_C^6 from 0 to 4. Node A in its turn responds to the reply message by sending an extra channel configuration acknowledgment confirming its connection with node B on channel 6, and raising CQP_A^6 to 10, cf. Figure 3.8c. If a node would only be in the communication range of node A and not of node B, that node would still be aware of the fact that traffic was about to start flowing on channel 6. Node C again raises its CQV_C^6 by 4 after receiving the acknowledgment, as it is now very clear that it will suffer interference both from A and B. Figure 3.8d shows the topology and resulting CQVs after adding extra reservations. A single interface can be used to transmit and receive data to and from multiple nodes.

If something goes wrong during the channel agreement procedure, or no traffic is received for a predetermined period of time on a particular channel, for example because of link breaks, a changed topology or termination of an application session, the corresponding reservations are removed from the database and control messages are sent as an announcement. Based on these events and the corresponding messages, the CQPs are decreased accordingly.

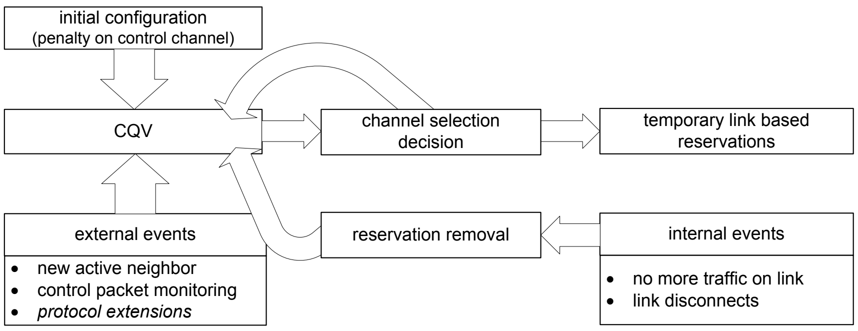

initial configuration
(penalty on control channel)
CQV
channel selection decision
temporary link based reservations
external events
• new active neighbor
• control packet monitoring
• *protocol extensions*
reservation removal
internal events
• no more traffic on link
• link disconnects

Figure 3.9: Schematic overview of the channel selection protocol operation. Additional optimizations are possible by adjusting CQVs through optional protocol extensions.

Additional mechanisms are in place in order to avoid multiple configurations taking place at the same time: in order to ensure that no decisions are made based on outdated information, two negotiation procedures involving a common node cannot take place at the same time. When this happens, one of the procedures is delayed for a configurable period τ. During the configuration delay period τ, the data packets are temporarily forwarded over the control channel. As such, no packets are lost.

The mechanisms described in the last two paragraphs explain the presence of the 'last seen' and 'status' fields in the information database depicted in the center of Figure 3.7: every time a data packet is received or sent through a reserved path, the 'last seen' timestamp is updated with the current system time. A background process checks for outdated links every second. The 'status' field indicates whether a reservation process is ongoing or finished. The reservation status is also used to identify partially failed handshake procedures and trigger reconfiguration attempts when required.

3.2.5 Protocol extensions

3.2.5.1 Additional CQV manipulations

The basic operation of the FRESME protocol is summarized in Figure 3.9: after setting an initial penalty for the control channel, new active neighbors trigger the channel reservation protocol, based on the value of the CQVs, and, leading to modifications of the CQVs. The reservation process leads to new temporary link based reservations, which are maintained for as long as data is sent in between the nodes. In case no more traffic is sent over a reserved link or the link disconnects, reservations are removed after a timeout period.

As indicated in the previous chapter, the channel quality vector is used as a glue entity for developing cross-layer protocol extensions without interfering with the basic operation of the protocol. For example, if the application layer indicates that a link to a certain neighbor requires absolute priority, the CQV of the corresponding channel could be raised considerable, making it very unlikely that additional links will be added to the same channel. As such, a certain degree of quality of service can be integrated in the protocol.

Furthermore, after the initial configuration which happens very fast, slower measurement techniques can alter the CQVs. This leads to a hybrid approach where initial configuration can happen very fast using a status based method, and is refined afterwards using measurement based methods.

Since the protocol relies on the exchange of control messages to judge whether links will be interfering or not, the protocol requires nodes to be within communication range to detect possible interfering nodes. In order to account for the effect of interfering nodes which are out of the communication range, an agent running in the background can perform noise measurements and carrier sensing. If interference is detected on a certain channel, the responding CQV can be raised. Such approach does not necessarily contradict with the message based approach, as the initial configuration is only a way to enable a fast spreading of the channel use.

3.2.5.2 Supporting any number of interfaces and channels

The FRESME scheme can easily be extended to support other physical layers such as 802.11a using more than three non-interfering frequencies. A naive approach would be to increase the number of additional PHY blocks to match the amount of non-interfering channels. However, a more elegant approach is possible, where the number of channels used can be higher than the number of interfaces available. Furthermore, the mesh routers do not need to have the same number of interfaces.

Suppose that a certain mesh node has n interfaces and η non-overlapping channels are available. The concept of channel quality vectors can still be used, now with a vector of size η: one CQV for every channel the node could tune to. When a new traffic link needs to be configured, a request message can be sent in the same way as described above. However, a second vector of size n should be added to the packet, indicating how many interfaces are already in use at the requesting node, and which channel they are configured to. A zero value indicates an unused interface.

If both sender and receiver still have an unused interface left, the channel with the lowest CQV_{sum} index is selected as a data channel for the data packets after the reservation procedure is completed; the unused interfaces are reconfigured to use the particular channel. If either sender or receiver do not have an unused network interface left, the communication side which has a free interface left can tune its last interface to the channel that best matches the other side's needs. The most difficult situation occurs when sending and receiving side both have all data interfaces configured without any matching channels. Two solutions are possible: *(i)* the default channel can be used; *(ii)* similar to the example of Figure 3.5, attempts can be made to reconfigure a wireless interface, e.g. at receiver side, in order to match a channel at the sending side. This can be done by broadcasting a reconfiguration request at the receiver side: the other nodes which already had made channel arrangements with the receiver then can answer this request by telling if still other connections are made using that particular channel. If not, reconfiguration can take place. In the other case, that node can send a reconfiguration message in its turn.

3.2.6 Performance analysis

The FRESME protocol was simulated and implemented using the Click Modular Router [20] software. Click allows programming and modifying networking protocols in a fast and easy way: it supports sending raw packets over wired and wireless interfaces, and allows building advanced router configurations by linking multiple fully customizable *elements*, each responsible for a small task. For example, by linking the 'RandomSource()', 'Queue()', and 'ToDevice()' elements after passing configuration arguments, packets with random content can be created, put into a queue, and sent over a wired or wireless interface.

By using Nsclick [21], Click configurations can be evaluated in the ns-2 [22] network simulator. In order to benefit from the full flexibility offered by the Click platform when simulating our protocol in nsclick, the nsmadwifi [23] extension for nsclick is used. Nsmadwifi enables the use of wireless features of the Click Modular Router platform such as rate setting, RTS/CTS and Wi-Fi packet transmission in the ns-2 environment. As such, the same code can be used both on actual hardware and in the simulator with minute adjustments.

As a routing protocol, the OLSR protocol is selected [24]. As indicated in the introductory chapter, OLSR is one of the experimental routing protocols for wireless ad-hoc networks under evaluation by the MANET working group. As the behavior of OLSR is relatively well-known, choosing this popular routing protocol makes it easier to distinguish problems caused by the routing protocol from problems caused by the channel selection protocol under evaluation.

3.2.6.1 Implementation

The platform of choice for the FRESME implementation is a Linksys WRT54GL router. This common type of COTS Wi-Fi router has only a single wireless interface. In order to overcome the interface limitation of the device, the device interconnection from Figure 3.10 was used. In this construction, one wireless node is configured as a master node, running the FRESME protocol. Two other nodes run

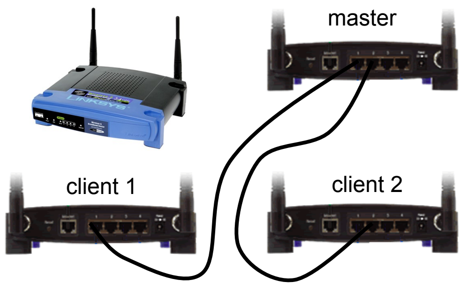


Figure 3.10: Three interconnected Linksys WRT54GL routers represent a single, three-interfaced node used as implementation platform for the FRESME protocol.

a simple configuration. The task of this simple configuration is twofold: first, it allows the node parameters such as the communication channel to be configured by the master device. Second, the configuration dictates the devices to forward any packets not related to device control over the wireless wireless interface, and forward all received wireless packets to the master device.

The master node is configured to channel 1, the client nodes to channel 6 and 11 respectively. Whenever a packet should be forwarded over channel 6 or 11, it is forwarded to a client node in order to be transmitted over its wireless interface. Vice versa, any packet that is received on a wireless interface of a client node is transferred to the master node. There, the packet is processed as if it were received locally. While this approach is more complex than selecting a three-interface node, it enables to conduct experiments with adequate antenna separation.

The implementation on top of the WRT54GL router was made possible thanks to a bilateral cooperation between the IBBT research institute and Cisco Systems within the scope of the IBBT-Eyesense project. With support from Cisco, a development toolkit was created that allows building new firmware images that can be flashed to the router platform. Using this development toolkit, the Click modular router software was adjusted to run on top of the WRT54GL router platform. The routing core was implemented in Click by Nicolas Letor, based on the OLSR implementation in Click made available at [25]. An overview of the interconnection of the OLSR implementation and FRESME multi-channel implementation is found in Figure 3.11. The packets leaving the OLSR core are in Ethernet format and already contain the next hop destination MAC address corresponding to the destination IP address, as the Address Resolution Protocol (ARP [26]) functionality is included in the Click OLSR implementation through the Click ARP-Querier [20] element.

The core of the FRESME protocol is implemented as a single Click element, which keeps track of the CQVs, and generates and processes the FRESME control

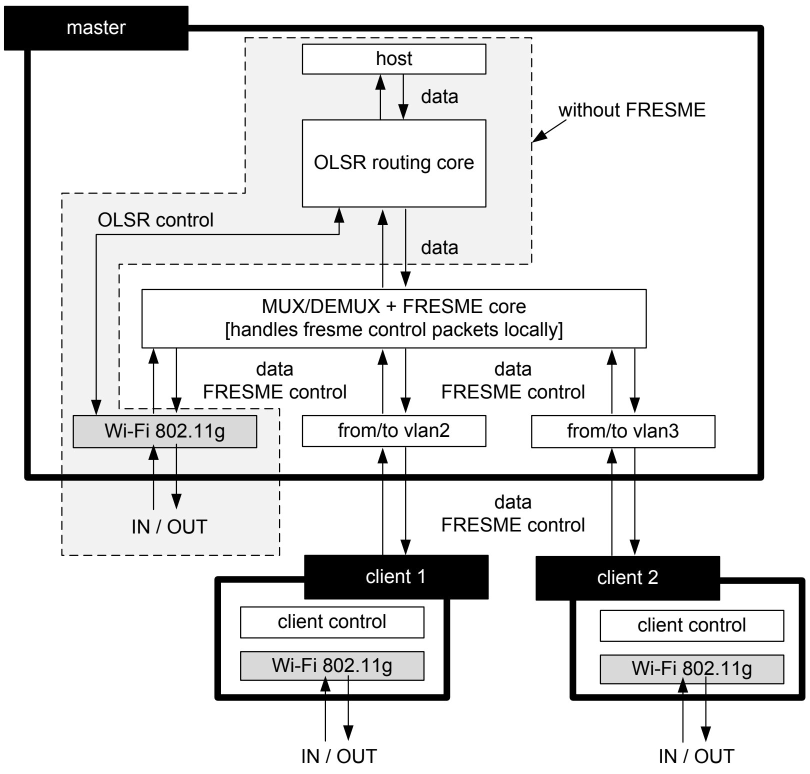


Figure 3.11: Implementation structure of the FRESME protocol on top of an OLSR implementation. The gray zone that is outlined with a striped line indicates the normal flow of the OLSR implementation, without FRESME being added. The master and client labels indicate the different WRT54GL routers that are used for the implementation.

messages (link reservation request, link reservation reply, link reservation ACK, link removal announcements). The FRESME control messages are transmitted over the (unofficial) UDP port 100, to the broadcast IP address. When packets enter the router, they are classified according to their type: packets and control packets of the OLSR routing protocol (official UDP port 698) are sent to the routing control packet processing functions, FRESME control messages are sent to the FRESME control packet processing functions, all other packets pass through the FRESME element for processing (e.g. update lifetime of reservation in case an IP packet from a known sender was received) before being forwarded to OLSR routing core.

Vice versa, IP packets that would have directly been sent to the wireless interface in the absence of the FRESME implementation, are now intercepted by the

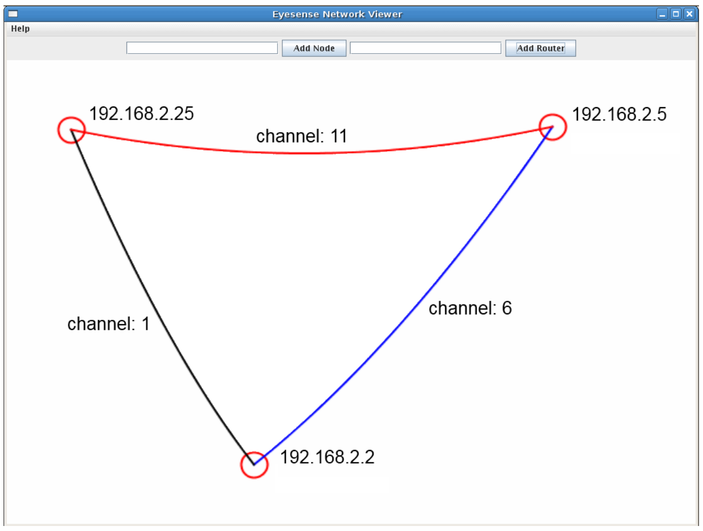


Figure 3.12: GUI presenting a live view on the test topology and channel reservations in place.

FRESME core. On arrival of such IP packet, the transmission of new FRESME control packets might be triggered (in case the IP address of the receiver is not included in the interface database). Based on the MAC address to interface mapping, the FRESME core routes the packet directly to the Wi-Fi interface of the master router, or encapsulates the packet in order to forward it over Ethernet to one of the client devices. On receiving an encapsulated packet, the client devices remove the forwarding header and transmit the packet over their Wi-Fi interface. Broadcast packets such as the control packets of FRESME and OLSR, are always forwarded over the Wi-Fi interface of the master device.

3.2.6.2 Real-life evaluation

Nine routers are available, as such, limiting the test topology to three three-interfaced nodes. However, even from this relatively small test set-up, much can be learned about the protocol. In order to help interpreting the behavior of the protocol, a simple GUI was built to visualize the wireless links in the network, as well as the channel reservations that are in place at a specific time. A screen capture of this GUI is shown in Figure 3.12. The screen was captured after initiating a ping session at all three nodes towards their clockwise neighbor.

A first qualitative observation is that the protocol succeeds in setting up channel reservations in a fast and distributed way, unnoticeable to the user: channel reservations are made near instantly as soon as data is sent over any wireless link. When a certain wireless link is no longer used for the transportation of data traffic,

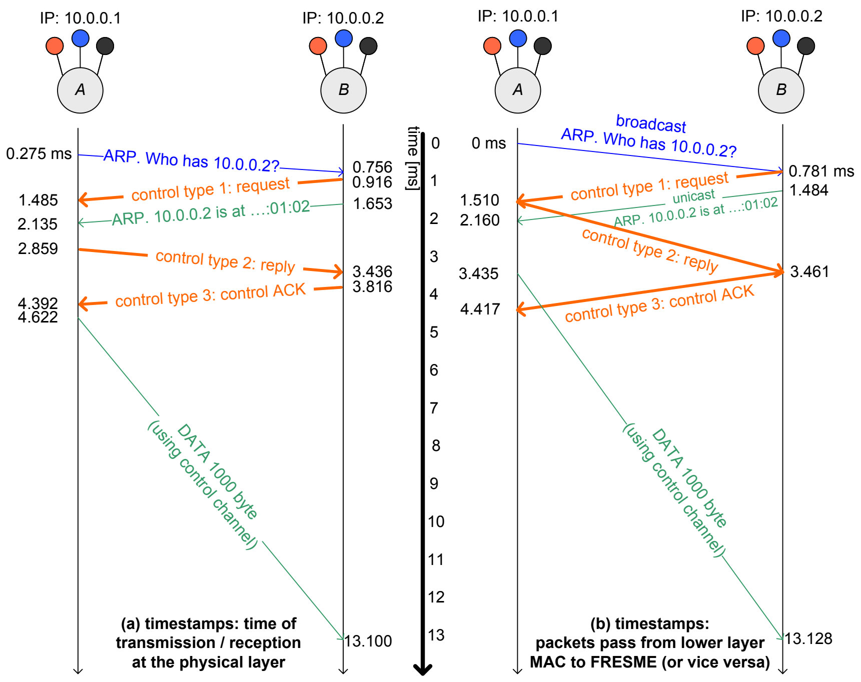


Figure 3.13: (a) Node A sends a 1000 byte IP packet to node B. Flowchart of the exchanged packets, with corresponding timestamps indicating when the packets are transmitted by or received at the physical layer. (b) Identical experiment, with timestamps taken when packets are transferred between FRESME subsystem and lower MAC layer functionality.

the reservation is canceled and all CQVs are adjusted as expected. Several simple experiments show that the FRESME protocol is a feasible and efficient way to optimize single-hop and multi-hop channel use.

Second, through the use of a packet sniffer and by recording timestamps on the devices, the duration of a configuration procedure is measured. To this end, a simple experiment is performed with two nodes. After booting the nodes, the OLSR routing messages are exchanged, and the routing table is constructed within seconds after the node boot. Next, a 1000 $byte$ packet is transmitted from sending node A to receiving node B every second. While the configuration handshake seemingly is completed even before the first data packet is sent, the first data packet is still forwarded over the control channel, instead of through the reserved channel. This behavior is further studied in a simple simulation where exactly the same scenario is repeated, at the same time illustrating the use of the three configuration message types (channel reservation request, channel reservation reply, channel reservation acknowledgment) more clearly.

Figure 3.13a shows a trace log of the first 13 milliseconds, during which the channel configuration takes place and the first packet is sent. IEEE 802.11 ACK packets are not shown in the figure. The timestamps shown indicate the exact time when which a packet is sent or received by the physical layer. The packet generated packet trace closely resembles the packet logs found from the real set-up: when A tries to send a 1000 $byte$ packet, first, an ARP request is broadcast, informing about the location of the wireless node with the IP address of B, 10.0.0.2. When this packet is received, node B wants to transmit the ARP reply using an unicast packet to node A. As this packet passes through the MUX/DEM, the destination MAC address (i.c. of node A) is not found in the iDB, thus, the negotiation process is quickly started by sending a channel configuration request before sending the ARP reply. First, the reservation procedure completes after sending a reservation reply and reservation ACK. Immediately afterwards, the data packet is transferred, although still using the control interface.

The apparent contradiction is solved by observing the same experiment, but now with timestamps obtained on the exact moment that packets are transferred from the FRESME subsystem to the default networking stack when sending packets, or received by the FRESME subsystem from the MAC layer when receiving packets (see Figure 3.13b). This flow graph clearly shows how node A is already queuing the data packet at the physical layer, before the configuration is completed. As soon as the wireless medium becomes available, the queued packet is sent, ignoring the fresh channel reservation.

More importantly, the single-hop experiment indicates that the user observed time between activating the channel reservation and completing channel reservation is limited to $4.622 - 0.916 = 3.706$ milliseconds. This configuration duration value sets the absolute bottom value for the configurable delay period τ previously defined near the end of Section 3.2.4 in case a channel reservation conflict is encountered.

3.2.6.3 Large-scale simulation

In order to verify the behavior of the FRESME channel selection set-up in larger topologies, the protocol is further analyzed based on ns-2/nsclick/nsmadwifi simulations. At the time of implementation, ns-2 version 2.26 was configured with a communication range of 200 meter when using a physical layer data rate of 1 $Mbps$. Again, the OLSR protocol is used. Since the OLSR metric uses a shortest path metric to determine its path, the nodes in our test topologies are separated by 200 meter: a denser topology makes little sense, as only few short paths would be chosen in the network anyway.

First, the protocol is tested on the 10-node line topology, where a path is set up from one side of the line topology to the other side. Figure 3.14 shows the resulting topology after the configuration protocol completes: data is alternately transmitted using channel 6 and channel 11: the load is ideally spread across the two interfaces reserved for data traffic.

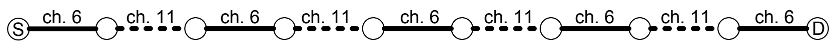


Figure 3.14: Resulting data channel configuration after the completion of the distributed channel selection procedure, for 10 mesh nodes in line topology with inter-node distance set to 200m.

In order to measure the efficiency of the FRESME protocol to optimize the channel use at mesh nodes in larger set-ups, simulations were done using the 10x10 raster topology from Figure 3.15. In this experiment, the routing protocol is activated at the beginning of the experiment, and 10 continuous UDP streams with packet size of 500 bytes are started sequentially between two random source-destination pairs. The first stream is activated after 10 seconds, and every 2 seconds, an additional stream is added. The experiment is repeated 10 times, each time with different random source destination pairs. The resulting random streams, routing paths and channel reservations at the end of one of these experiments is shown on Figure 3.15, generated through a custom built trace processing GUI, allowing easy interpretation of ns-2-generated topologies and the FRESME packet log files. The output of the GUI is based on preprocessed ns-2 traces (containing the node locations) and additional information added to the trace logs via output of the FRESME Click element, indicating the channel reservations. The information that is collected in this automated way is then also used to calculate the performance metrics detailed below.

As the routing protocol choice affects various performance metrics such as throughput or delay, new metrics are introduced focusing only on the channel distribution:

- ***Global relative channel use:*** the global relative channel use determines the global use of the different communication channels throughout the entire network as seen from the individual nodes. It is determined by counting the number of unidirectional reserved links configured to each individual channel as observed from a node, in relation with the total number of unidirectional links that are observed by the node. In our example raster configuration, it is calculated as follows: at each node, the number of reserved data links that either start or arrive at that node within an interference area of 250m around the node are counted. This includes links which are used by the node itself, as well as links between two other nodes which cause interference at the node. Both starting and arriving links count as 1. The sum of this link count is then calculated for all nodes, finally dividing it between the

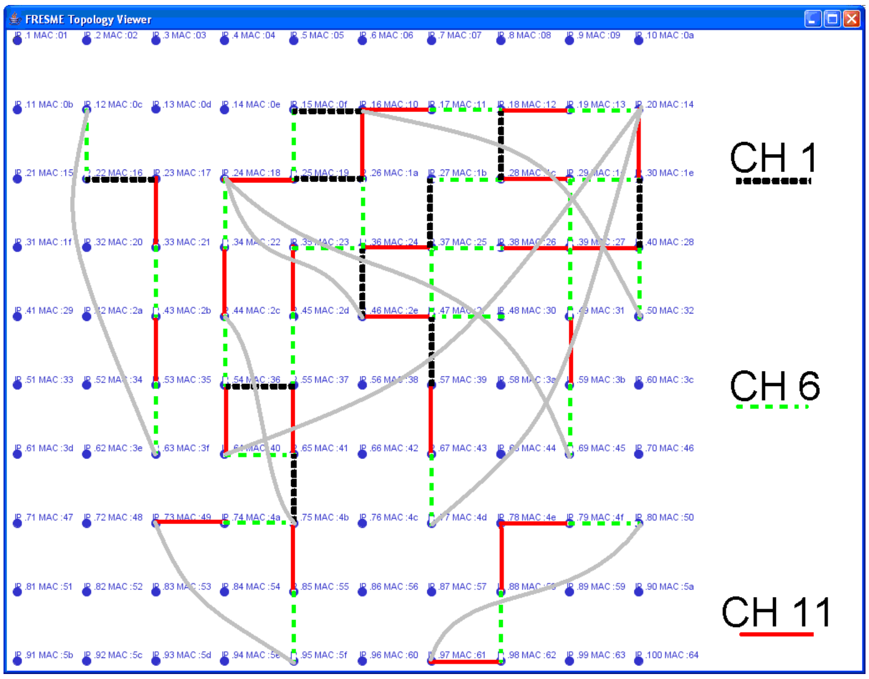


Figure 3.15: Custom built log file visualization tool, showing 10 x 10 raster topology with 10 random source-destination pairs. The horizontal and vertical distance between the nodes is 200m. The data configuration is displayed on the figure.

total number of observed unidirectional links. This global relative channel use of the different data channels is a first indication of the network-wide channel distribution.

- ***Local unbalance:*** the local unbalance of a node is the biggest difference in number of links configured to any one channel reserved exclusively for data communication at the node, with the number of links configured to any other channel used exclusively for data communication at the node. This is mathematically expressed as follows: with ${N_i}^{\alpha}$ the number of links at node α reserved to channel i, with $i = 1, ..., \eta$ and $i \neq controlchannel$, the local unbalance U^{α} of a node α equals:

$$U^{\alpha} = max(|{N_p}^{\alpha} - {N_q}^{\alpha}|) \text{ with } p, q = 1..\eta \text{ and } p, q \neq controlchannel$$

In the example implementation, the local unbalance of a node is determined by $|N_6 - N_{11}|$. For each node, the metric indicates whether the channel reservation is locally balanced or not. The local unbalance of a perfectly balanced node equals 0 in case as many data link reservations were made on the first data channel as on the second. A node is called locally unbalanced as soon as the unbalance is larger than 1. Nodes that do not make any reservations are excluded from the statistics.

- ***Neighborhood unbalance:*** The neighborhood unbalance is determined in similar way as the local unbalance. However, this time, the number of links is not only counted at the node itself, but also includes the interfering links. With $IF_j(\alpha)$ the j-th node of M nodes interfering with node α, the neighborhood unbalance at node α is expressed as:

$$U_{NB}{}^{\alpha} = max(\mid N_p{}^{\alpha} + \sum_{j=1}^{M} N_p{}^{IF_j(\alpha)} - N_q{}^{\alpha} - \sum_{j=1}^{M} N_q{}^{IF_j(\alpha)} \mid)$$
$$\text{with } p, q = 1..\eta \text{ and } p, q \neq controlchannel$$

Nodes that do not make any reservations themselves and do not observe link reservations in their interference range are excluded from the statistics.

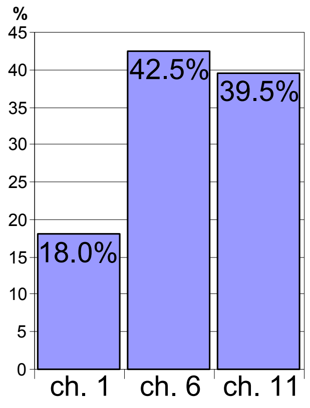


Figure 3.16: Global relative channel use of the raster topology, averaged over 10 test runs.

The global relative channel use of the raster topology is depicted in Figure 3.16. The bar chart shows that on average, a comparable amount of reservations on data channel 6 as on data channel 11 can be observed at a node. Furthermore, only 18% of the data channel reservations are made using the control channel, which compares to half the number of reservations as on the data channels. As such, in this topology, on average one fifth of the data channel reservations are made on the control channel, while the data channels each carry two fifths of the data. This result corresponds with the expectations.

Averaging results over several nodes might lead to misleading conclusions. Therefore, the local and neighborhood balance is determined for the raster set-up.

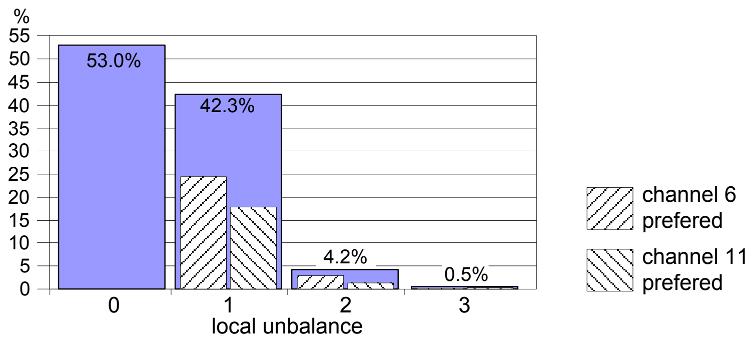


Figure 3.17: Local unbalance statistics of the 10x10 raster topology.

Over the 10 test runs, (interfering) link reservations are observed at on average 87.4% of the nodes. From these nodes, 68.0% participate in the channel reservation procedure. For the nodes participating in channel reservation procedures, Figure 3.17 shows the distribution of local unbalance values. In case of unbalance, the specific preference for channel 6 or 11 is detailed. The chart shows that 53% of the nodes actively participating in link reservations do not show any unbalance on their data channels: both channel 6 and channel 11 are used an equal amount of times. An additional 42.3 % of the nodes shows an unbalance equal to one. In this case, the uneven balance is caused by the need to reserve an additional link on channel 6 or 11 when they are already used an equal number of times, or because the node lies at the end of a network path and only has a single reservation. As such, an unbalance of 1 is simply unavoidable at times and is no real error of the protocol. Less than 5% of the nodes observe an unbalance equal or greater than 2, meaning that in over 95% of the cases the local channel distribution is optimal.

The chart also shows that in case of unbalance, there is a tendency to prefer channel 6 over channel 11. This behavior is caused by the previously explained 'first minimum' policy which was in place during the experiments.

Finally, the neighborhood unbalance of the 874 nodes observing channel reservations is shown in Figure 3.18. Of those nodes observing a channel reservation within their environment or participating in the channel reservation process themselves, $(36.8 + 49.3)\% = 86.1\%$ observes a neighborhood unbalance of zero or one, indicating that the channel selection technique works well even when considering interfering links that are not used by the nodes themselves. As such, for the considered raster topology the distributed message based channel approach is suboptimal in less than 14% of the cases.

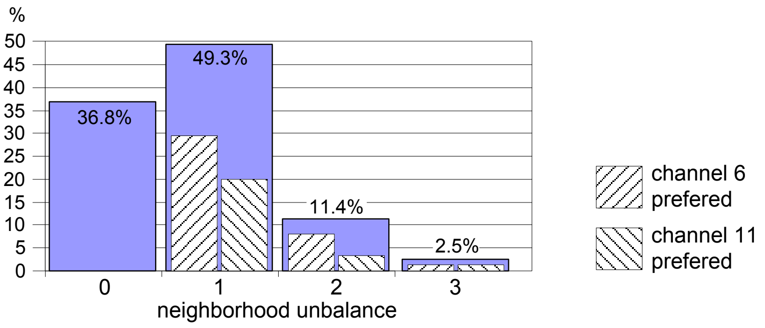


Figure 3.18: Neighborhood unbalance statistics of the 10x10 raster topology.

3.2.6.4 Discussion and possible optimizations

While the FRESME simulation results are promising, there are also limitations to the current FRESME approach. First, since the channel selection protocol is transparent to the routing protocol, it has the disadvantage of not being able to influence the macroscopic routes in the network. As such, even if the CQV show that a node is heavily loaded, the decisions of an off-the-shelf routing protocol cannot be influenced, making it impossible to pursue advanced routing strategies such as re-routing traffic over a less loaded traffic path. However, a cross-layer routing protocol could be developed which takes into account the CQV glue entity as a way of determining routes in the network; Within the presented cross-layer node architecture, this could be realized by replacing a shortest path routing metric which makes its decisions entirely based on the topology of the network, by a routing metric which also includes the reservation state at the different nodes, derived from the CQV entities.

Second, in the considered scenario and above analysis, it was assumed that the traffic load imposed by each additional reservation is equal, and that that all wireless links in the network are characterized by the same link capacity. For most scenarios, this assumption will prove to be invalid. However, within the cross-layer node architecture of Figure 2.20, this limitation could be overcome by reflecting the actual state of the wireless links and (average) traffic load in the CQPs of the corresponding channels.

Third, while the hybrid control channel approach results in a control channel that is less interfered than the data channels, the protocol in its current implementation is not able to automatically deal with control channels that are interfered by external networks. A scanning procedure and control channel selection scheme is needed to overcome this issue.

Fourth, the protocol was evaluated in a basic yet realistic version on top of IEEE 802.11b/g wireless mesh interfaces, which have only three non-interfering channels available. The protocol may be re-evaluated for wireless technologies capable of using more channels simultaneously.

Finally, small-scale testbed experiments and large-scale simulation results have indicated that the protocol successfully deals with discrete topology changes, removing outdated links after a predefined timeout period. However, it would be interesting to verify the performance of the scheme under highly mobile conditions and more complex topologies. As wireless nodes essentially gather information by listening to control packets, nodes arriving into a new environment need some time to build up their view on the channel quality. As such, when topology changes continuously occur, it might be necessary to introduce a new packet type to be able to actively query for existing channel reservations in the node's neighborhood and instantly receive a more accurate view on the channel reservations.

Nevertheless, even without the above optimizations, the current FRESME protocol can immediately be used on operational networks. At the cost of adding additional interfaces to an existing solution, and provided the interference between the different interfaces of the node is kept under control, the FRESME protocol increases the reliability of control traffic and thus the stability of the network, while spreading the wireless links over the available data interfaces. As such, the performance gap with wired connection alternatives is reduced.

3.3 Capacity estimation of multi-channel paths

In the previous section, the first of two developments of this chapter, namely the distributed channel approach for wireless mesh networks was developed and analyzed. While the FRESME protocol is a feasible and efficient approach to provide a quick link-to-channel reservation, no information is provided on the actual link quality that may be expected. Therefore, in this section, a secondary aspect of channel configuration in multi-interface, multi-channel wireless mesh networks is treated: a protocol is developed and evaluated that estimates end-to-end application layer throughput capacity of a multi-channel wireless network path. The protocol is based on distributed per-link capacity measurements using the packet pair probing technique described in [27]. In the cited work, the packet pair probing technique is used to determine the maximum IP-layer throughput of wired networks by transmitting packet pairs across an end-to-end path. In this chapter, the concept of packet pair probing is re-used to evaluate the end-to-end throughput capacity in wireless multi-interface, multi-channel networks, based on per-link capacity measurements.

As a use case, the wireless multi-hop chain topology for emergency services scenario previously depicted in the center of Figure 3.1 is used. Although the developed technique is applicable to an end-to-end path in any type of topology, the relative simplicity of a multi-hop line topology allows the development and

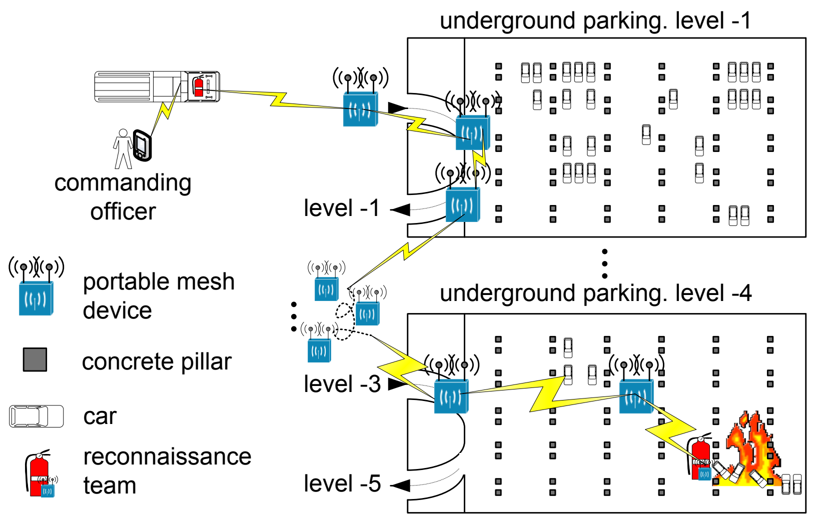


Figure 3.19: The reconnaissance team of the fire brigade enters an underground parking where a fire is located at parking level -4. Since traditional communication systems cannot penetrate the thick layers of concrete, portable mesh devices are positioned at strategic locations to maintain connectivity.

performance analysis of the capacity estimation technique in absence of the unpredictable effects caused by routing protocols in more complex test set-ups. The scenario is further detailed in Figure 3.19. Here, the challenge is to maintain a high-performance wireless path from the reconnaissance team of the fire brigade inside a structure to the commanding officer on scene outside the building. A car fire needs to be extinguished in an underground parking, four floors beneath the ground.

Today, reliable voice communication in similar complex building structures is not a guaranteed success: although existing TETRA (Terrestrial Trunked Radio) [28] communication systems for professional safety services and traditional walkie talkies are widely used by public services, their wireless signals may not be able to propagate far enough inside a building.

Moreover, while these traditional techniques are suited to transport voice signals, new applications require high-capacity networks. As an example application, consider the transmission of video feeds captured by helmet cameras of firemen in the reconnaissance team (RT). Such video feed may help the commanding officer to better judge the situation inside a building, or may help paramedics outside the building to prepare the right equipment for victims that are to be brought outside: a person with severe burns requires a completely different treatment as a victim with an open fracture. Making the right judgments at the right time and having the appropriate medical equipment at the ready may save valuable seconds. Under these extreme circumstances, saving seconds means saving lives.

Streaming video images from inside a burning structure is no scenario from the future: for example, the Fire and Incident Camera Observation Team FICOT [29] has designed camera equipment with infrared and thermal view, for use during fire-fighting operations. However, due to the limited performance of traditional wireless networks, a special group of people is needed in order to drag the cable.

As such, there is a need for a high capacity wireless multi-hop network. Therefore, similar to the previous section, a multi-hop, multi-interface wireless network is needed, which avoids the use of identical communication channels within a single wireless collision domain. Furthermore, in order to know which bandwidth can be expected from a specific wireless path, a capacity estimation strategy is required. Such capacity estimation strategy is useful for two reasons: *(i)* When deploying nodes in line topologies, one may want to drop a new mesh device whenever the capacity of the path is almost falling below a certain desired threshold. In the described scenario of the underground parking, the capacity of the path between the reconnaissance team and the firetruck could constantly be monitored. *(ii)* Knowing the actual state of a wireless network is a first step in being able to judge whether additional data may still be sent over the network without affecting the reliability of the network. In our example, a commanding officer may want to know whether an additional video stream may be activated without affecting the voice communication sent over the same communication infrastructure, or, without losing another vital already enabled video stream.

This capacity information may also be used by other algorithms within the wireless network stack of a node, such as the routing protocol. As such, the presented capacity estimation technique represents a possible implementation of the link quality monitoring block from the cross-layer optimized mesh node architecture.

3.3.1 Determining the capacity of wireless links

3.3.1.1 Overview

Determining the capacity of wireless network links without interfering with the operation and performance of the network is difficult [30–33]; while it is relatively simple to get an accurate view on the capacity of a link by flooding the wireless link with as much traffic as possible, using TCP and UDP bandwidth measurement tools such as iperf [34], these methods are intrusive by design. As the packet streams generated by similar tools consume the entire available bandwidth, these tools are mainly suited for use in static networks such as wired networks, where the capacity does not vary over time.

It is a challenging task to provide capacity estimation of a path in an opera-

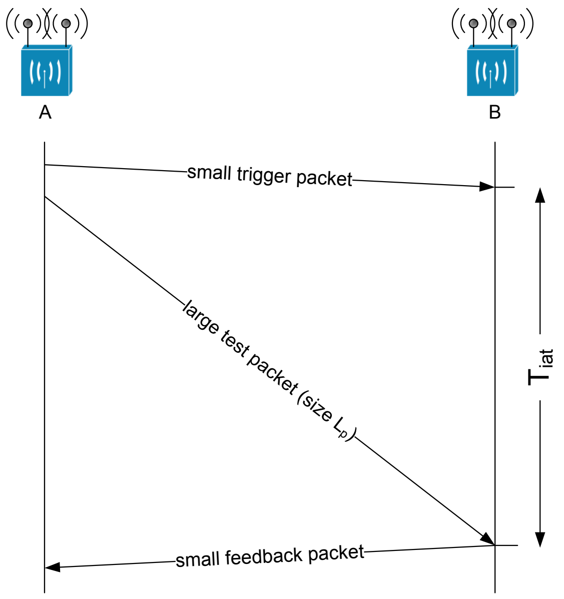


Figure 3.20: Principle of the packet pair probing technique.

tional dynamic wireless network, without imposing a large packet overhead on the network. One technique that has been applied in literature is the packet pair probing technique [27]. The principle of this technique is explained in its simplest form in Figure 3.20, omitting the IEEE 802.11 MAC layer acknowledgment packets for clarity reasons. In this single-hop network, the wireless path corresponds with a single wireless link.

In order to estimate the throughput capacity of the source-destination path $A - B$, two packets destined to node B are queued at node A and transmitted immediately after each other. The first packet is small while a second packet is of a larger size L_p. The small packet is used as trigger to notify node B that a larger test packet is coming. At receiver side, the inter arrival time $(T_{iat})^{AB}$ between the trigger packet and test packet is determined. The measured $(T_{iat})^{AB}$ value is then communicated back to node A. Since the relative time difference between the trigger and test packet is determined, there is no need for synchronization between the nodes.

The throughput capacity of the path $A - B$, in this case corresponding with the capacity of the link $A - B$, is then determined as:

$$C_{AB} = \frac{L_p}{T_{iat}{}^{AB}} \tag{3.1}$$

Most of the cited works using packet pair probing as a technique use packet pair probing over end-to-end multi-hop paths. However, these techniques are burdened by slow convergence [35]: the packet pair probing technique should be applied to each individual end-to-end path in the network. Thus, in large networks with varying topologies, stable measurements related to all possible paths in the network may not always be available. Unfortunately, per-hop capacity estimation techniques do not provide the end-to-end estimation which is essential in the emergency mesh network. Therefore, in this work, a hybrid approach is presented that combines the best of both extremes: first, the packet pair probing technique is applied on a per-hop basis. Then, the capacity estimations are disseminated through the network by piggy-backing information on the OLSR-based routing protocol messages to avoid additional packet overhead as much as possible. Because OLSR is a proactive link state protocol, every node is aware of the entire topology of the network by default. By adding capacity information to the control packets of the OLSR routing protocol, every node is also aware of the estimated throughput capacity of each individual link. By combining the available information, each node is capable of judging the throughput capacity of any path in the network at any time.

3.3.1.2 Implementation

In order to calculate the end-to-end path capacity of any path, the approach illustrated in Figure 3.21 is used. The numbers in the paragraph below relate to the numbers indicated in this figure.

First, (1) each node checks the contents of its routing table. Then, each node performs the packet pair probing technique with all of its individual neighbors. In the example of Figure 3.21, the middle node is the first to start the process (2), with its neighbor on the right side of the figure: both a small $100\,Byte$ trigger packet and a larger $1000\,Byte$ test packet are queued at the wireless driver, which transmits these packets as soon as possible; on reception of these packets, the receiving node on the right side of the figure registers the timestamps at which the trigger packet (3) and test packet (4) are received. The receiver is capable of detecting which test packet corresponds with which trigger packet, because a unique sequence number is included in each of the packets, and by identifying the sender of the packets through its IP address. When two matching packets are identified, the receiver calculates the difference between the respective timestamps, and includes this value in a feedback packet that is sent towards the originator of the test packets. The middle node stores the n consecutive time differences locally (6), while repeating the process with its other neighbor(s). Steps (3)–(6) and (3')–(6') are continuously repeated for a configurable number of times every second on every link. More cycles per second increase the reliability of the measurement, at the cost of adding overhead.

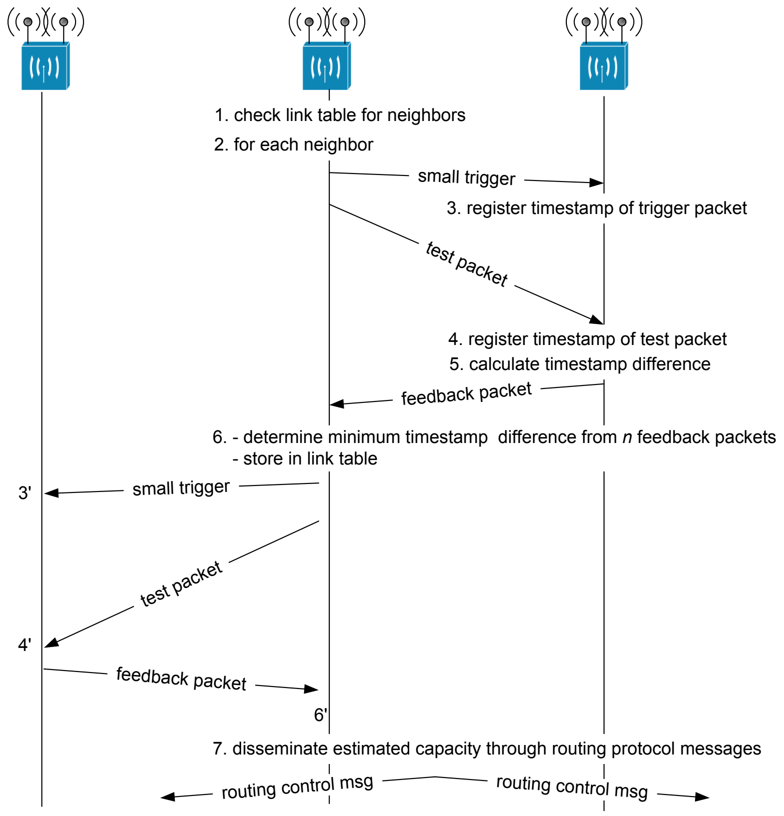

1. check link table for neighbors
2. for each neighbor
small trigger
3. register timestamp of trigger packet
test packet
4. register timestamp of test packet
5. calculate timestamp difference
feedback packet
6. - determine minimum timestamp difference from *n* feedback packets
- store in link table
3'
small trigger
test packet
4'
feedback packet
6'
7. disseminate estimated capacity through routing protocol messages
routing control msg
routing control msg

Figure 3.21: Packet exchanges during in the implemented distributed packet pair probing approach. The packet pair probing technique is performed with each individual neighbor of the node. The estimated capacity values are then distributed through the routing control messages.

The minimum T_{iat} registered after n samples is used to estimate the link capacity, as it is assumed that at least one packet pair within this window of time was successfully transmitted back-to-back without being delayed because of external interference, and the technique is targeted towards identifying the maximum throughput capacity.

The estimated capacity values are distributed across the network by means of the existing OLSR routing messages (7) that are already present in the network in order to disseminate the topology. The channel configuration at the nodes is distributed in the same way. As a result, the topology, channel configuration and link capacity estimations are available at every individual node in the network

The end-to-end application layer throughput capacity of a path may then be determined as follows. Assume that m different non-interfering channels are used along the end-to-end path. First, for each of these m channels, the throughput capacity of each 'virtual' sub-path that is formed by all interfering links configured to a particular channel is determined. This sub-path is labeled 'virtual', since is not necessarily connected. To reduce the complexity of the formulas, assume in a first approximation that all links configured to the same channel are causing interference to each other.

Denote C_x the end-to-end throughput capacity of the sub-path configured to channel x, with x one of the m channels. The end-to-end throughput capacity of the sub-path is defined as:

$$C_x = \frac{L_p}{T_{iat,x}} \tag{3.2}$$

In this equation, L_p is the length of the probe packet, and $T_{iat,x}$ the total time to transmit the test packet over the subset of s_x links on the considered sub-path set to channel x.

Since the total inter arrival time for the links on channel x is the combined sum of the individual inter arrival times ${T_{iat}}^j$ for each link j, $T_{iat,x} = \sum_{j=1}^{s_x} {T_{iat}}^j$.

This allows C_x from equation 3.2 to be rewritten as follows:

$$C_x = \frac{L_p}{\sum_{j=1}^{s_x} {T_{iat}}^j} \tag{3.3}$$

Now, let $C_{(j,x)}$ be the throughput capacity of link j, with $j = 1..s_x$ set to channel x. From equation 3.1, it is known that:

$$C_{(j,x)} = \frac{L_p}{{T_{iat}}^j} \tag{3.4}$$

Finally, combining 3.3 and 3.4, shows that the end-to-end throughput capacity of the sub-path configured to channel x, C_x, may be determined as:

$$C_x = \frac{1}{\sum_{j=0}^{s_x} \frac{1}{C_{(j,x)}}}$$

This equation proves that the throughput capacity of a sub-path may be calculated based on the throughput capacity estimations of the different links using the specified channel.

This calculation is then repeated for every other of the m channels that are used along the original end-to-end path. Finally, the end-to-end throughput capacity of the wireless path is determined by the bottleneck throughput capacity of all sub-paths along the wireless path.

The above approach is further refined by adding the knowledge of a k-hop interference neighborhood. Wireless links that are configured to the same communication channel, but are separated by more than k-hops along the end-to-end path are considered not to cause interference to each other. In the considered scenario that operates in a high path loss environment and thus is sparsely connected, a k value of 3 is sufficiently conservative. This refinement is implemented by applying the above method recursively as follows: for each individual wireless link in the end-to-end path, the set of interfering links is determined, thus forming a sub-sub-path that may not include all links that were previously included in the sub-path linked to the particular channel. For this sub-sub-path, the throughput capacity is determined in an identical way as above. This process is then repeated for all other sub-sub-paths configured to the same channel. Finally, the throughput capacity estimation of the sub-path is determined as the minimum throughput capacity of the individual sub-sub-paths.

In some topologies with some channel configurations, the end-to-end throughput capacity estimation calculated using the refined method may underestimate the actual capacity, as within each sub-sub-path, each node is assumed to contribute equally to the load of the wireless medium; In reality, a wireless link at the border of a sub-sub-path might share the medium with yet another interfering link, and may thus not always be active, leaving a larger part of time to the other nodes in the sub-sub-path available for packet transmissions. The detailed analysis of this phenomenon is left as future work.

3.3.2 Performance analysis

The throughput capacity estimation technique was implemented on customized 4G Mesh Cubes, as previously used to obtain the spectral measurements of Section 2.4.4. In order to be able to experiment with the platform in a real-life context, several modifications were made to the design. Since firemen cannot rely on power sockets in a burning building, three light-weight batteries (Li-ion, 3.7V, 2200mAh) replaced the standard power adapter, allowing the device to run on batteries for three to four hours. Figure 3.22 shows the compactness of the design.

In order to evaluate the performance of the implemented packet pair probing technique, it is first tested on a single-hop link in the shielded environment previously described on page 31. A one-hop network is set up, and the packet pair probing starts. The maximum physical layer data rate of the nodes is manually limited to the different physical layer data rates available in IEEE 802.11a/b/g, and the capacity of the link is estimated every five seconds. The results of the experiment using IEEE 802.11a data rates is plotted in Figure 3.23. From this graph, it can be seen that the packet pair probing technique produces accurate results for lower

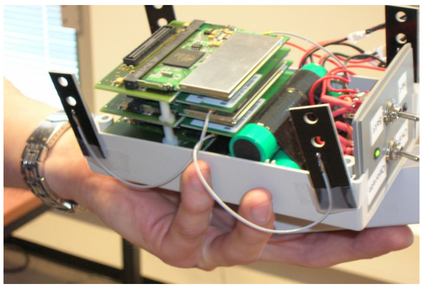


Figure 3.22: Implementation platform for the capacity estimation protocol.

technology	IEEE 802.11a		IEEE 802.11bg	
rate $[Mbps]$	iperf $[Mbps]$	pp-probing $[Mbps]$	iperf $[Mbps]$	pp-probing $[Mbps]$
auto	31.30	27.26	31.10	29.78
54	30.20	26.60	29.50	28.72
48	31.00	23.95	31.80	26.35
36	26.50	20.21	25.20	22.75
24	19.00	16.50	18.20	16.76
18	14.40	12.38	13.70	13.28
12	9.89	9.41	9.70	9.66
9	7.97	7.52	7.30	7.49
6	5.25	5.25	4.99	5.33
11	n.a	n.a.	7.91	8.04
5.5	n.a	n.a.	4.2	4.85
2	n.a	n.a.	1.78	1.78
1	n.a	n.a.	0.90	0.90

Table 3.1: Comparison of single-hop capacity estimations with iperf measurement for IEEE 802.11a and IEEE 802.11b/g technologies (averaged).

physical layer data rates. However, at higher rates, the capacity estimation is less stable and at times lower than expected. Two factors are contributing to this difference. First, the T_{iat} is measured by the packet pair probing algorithms, which are implemented above the physical layer. As such, it would be more correct to state that the measured time is $T_{iat} + \delta$, where δ is a factor accounting for delays caused by processing in the wireless driver and at system level. While δ is always small, it has a relatively larger influence at higher physical layer data rates, since T_{iat} is considerably lower at higher data rates. Second, the reduced performance to what is expected can reflect actual instabilities and performance drops of the wireless links.

Therefore, the capacity estimations are compared with capacity measurements

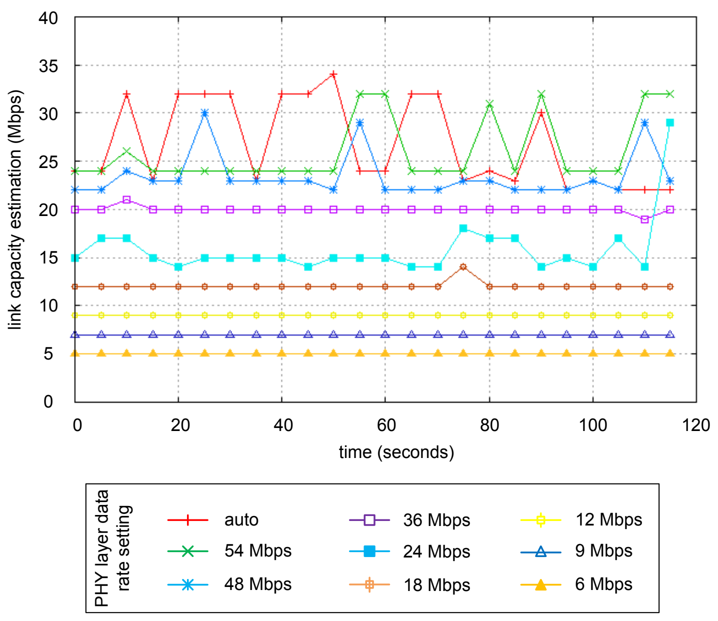


Figure 3.23: Link capacity estimation values reported by the capacity estimation protocol in time, for varying physical layer data rate settings using IEEE 802.11a.

obtained by flooding the single-hop wireless link using the iperf capacity estimation tool. Table 3.1 compares the iperf measurement with the average estimations of the packet pair probing implementation. The comparison of these values shows that packet pair probing tends to underestimate the capacity at higher data rates, although not problematically.

Finally, the multi-hop, multi-channel throughput capacity estimation protocol is tested outside of the shielded environment in a two-hop testbed depicted at the top of Figure 3.24. The two wireless links are configured at theoretically non-interfering frequencies. However, in practice, there is no perfect separation between the different interfaces at the middle node. This causes instabilities, mainly while operating the network at higher data rates. In order to avoid these side effects, one link is configured to a fixed physical layer data rate of 18 $Mbps$, while the other link is varied between 6, 9, 12 and 18 $Mbps$. Figure 3.24 shows bar charts comparing the throughput capacity measurement with the throughput capacity estimation. The figure shows that the multi-channel multi-hop capacity estimation approaches the measured capacity, as such proving the feasibility of the distributed measurement technique.

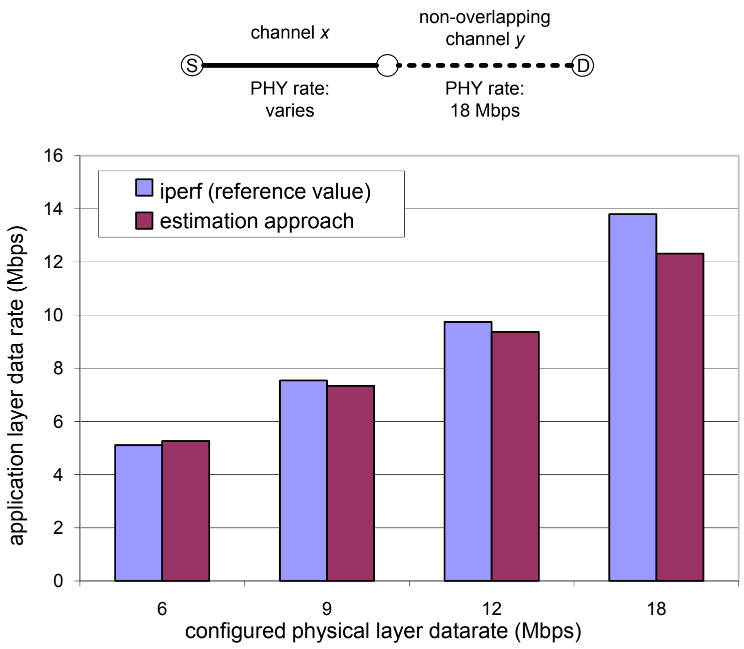


Figure 3.24: (above:) Two-hop topology. One of the two wireless links is configured to a fixed physical layer data rate of 18 MBps. The data rate of the other link is varied. Links are not interfering. (below:) Comparison of end-to-end application layer data rate estimated in this topology using the packet pair probing based technique, with the application layer data rate measured with iperf, for different physical layer data rates of the first link.

3.4 Conclusion

In a first part of this chapter, the FRESME protocol was developed, implemented and analyzed. It was shown that the protocol is able to configure channels on demand on a per link basis in a fully distributed way. In the absence of communication errors, a channel reservation is completed in less than 4 milliseconds using only 3 control packets (and 3 802.11 ACK packets). The protocol does not rely on long-term measurements, does not require periodic broadcast of control messages, and can cooperate with a broad range of routing protocols. The efficiency and feasibility of the protocol was demonstrated through small-scale implementation and large-scale simulation. It was shown that in a raster topology, globally, the channels reserved for data communication are equally loaded. Locally, the link to data channel mapping is optimal at 86.1% of the nodes.

In the second part of the chapter, a throughput capacity estimation technique was developed for use in multi-channel, multi-interface networks. Based on the packet pair probing technique, an estimation of the capacity of each individual link is determined. After dissemination of the per-link estimation and the channel configuration parameters, each node in the network is able to determine the end-to-end throughput capacity of any possible path in the network. Small-scale tests indicate the feasibility of the technique, and show the estimated capacity of the test links to be similar to the capacity measurements performed by using a capacity measurement tool that floods the network. While the latter technique consumes all bandwidth in the network, the overhead of the developed technique is minimal.

Although these results are promising, additional measurements are necessary to evaluate the protocol in larger test set-ups while varying topologies, background noise levels and the amount of background traffic generated in the network. These measurements are left for future research.

References

[1] Piyush Gupta and Panganamala Ramana Kumar. *The Capacity of Wireless Networks*. IEEE Transactions on Information Theory, 46(2):388–404, March 2000.

[2] Matthias Grossglauser and David N.C. Tse. *Mobility increases the capacity of ad hoc wireless networks*. IEEE/ACM Transactions on Networking, 10(4):477 – 486, aug 2002.

[3] Abbas El Gamal, James P. Mammen, Balaji Prabhakar, and Devavrat Shah. *Throughput-Delay Trade-off in Wireless Networks*. In INFOCOM, 2004.

[4] Ahmed Bader and Eylem Ekici. *Throughput and delay optimization in interference-limited multihop networks*. In MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing, pages 274–285, New York, NY, USA, 2006. ACM.

[5] Jinyang Li, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris. *Capacity of Ad Hoc wireless networks*. In MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking, pages 61–69, New York, NY, USA, 2001. ACM.

[6] Phil Chimento and Joseph Ishac. *Defining Network Capacity*. RFC informational 5136, Internet Engineering Task Force, February 2008.

[7] Lei Yang, Lili Cao, Heather Zheng, and Elizabeth Belding. *Traffic-aware dynamic spectrum access*. In WICON '08: Proceedings of the 4th Annual International Conference on Wireless Internet, pages 1–9, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[8] Koushik Kar, Xiang Luo, and Saswati Sarkar. *Throughput-Optimal Scheduling in Multichannel Access Point Networks Under Infrequent Channel Measurements*. In INFOCOM. 26th IEEE International Conference on Computer Communications, pages 1640 –1648, may 2007.

[9] Arunesh Mishra, Suman Banerjee, and William Arbaugh. *Weighted coloring based channel assignment for WLANs*. SIGMOBILE Mob. Comput. Commun. Rev., 9(3):19–31, 2005.

[10] Janne Riihijarvi, Marina Petrova, and Petri Mahonen. *Frequency Allocation for WLANs Using Graph Colouring Techniques*. In WONS '05: Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services, pages 216–222, Washington, DC, USA, 2005. IEEE Computer Society.

[11] Magnús M. Halldórsson, Joseph Y. Halpern, Li (Erran) Li, and Vahab S. Mirrokni. *On spectrum sharing games*. In PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing, pages 107–114, New York, NY, USA, 2004. ACM.

[12] Cisco Systems. *Cisco Aironet access points online help*. http://www.cisco.com/web/techdoc/wireless/access_points/online_help/eag/122-15.JA/1400br/h_ap_network-if_802-11_c.html.

[13] Douglas J. Leith and Peter Clifford. *A Self-Managed Distributed Channel Selection Algorithm for WLANs*. In 4th Intl. Symp. on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pages 1–9, Boston, MA, April 2006.

[14] Priyank Porwal and Maria Papadopouli. *On-demand channel switching for multi-channel wireless MAC protocols*. In 12th European Wireless Conference, Athens, Greece, 2006.

[15] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou. *A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks*. In Broadnets '04, pages 344–354, Los Alamitos, CA, USA, 2004.

[16] Jiandong Li, Zygmunt J. Haas, Min Sheng, and Yanhui Chen. *Performance evaluation of modified IEEE 802.11 MAC for multi-channel multi-hop ad hoc network*. In 17th International Conference on Advanced Information Networking and Applications, pages 312–317, Xi'an, China, March 2003.

[17] Ashish Raniwala, Kartik Gopalan, and Tzi cker Chiueh. *Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks*. volume 8, pages 50–65, New York, NY, USA, 2004. ACM Press.

[18] Jian Tang, Guoliang Xue, and Weiyi Zhang. *End-to-end rate allocation in multi-radio wireless mesh networks: cross-layer schemes*. In QShine '06: Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks, Waterloo, ON, Canada, 2006.

[19] Daniel Lang. *A comprehensive overview about selected ad hoc networking routing protocols*. Master's thesis, Technische Universität München, München, Germany, 2003.

[20] Click. *The Click Modular Router Project*. http://pdos.csail.mit.edu/click/.

[21] nsclick. *The nsclick Simulation Environment*. http://systems.cs.colorado.edu/Networking/nsclick/.

[22] NS2. *The Network Simulator*. http://www.isi.edu/nsnam/ns.

[23] Nicolas Letor, Peter De Cleyn, and Chris Blondia. *Enabling cross layer design: adding the MadWifi extensions to Nsclick*. In Proc. First International Workshop on Network Simulation Tools 2007 in conjunction with Valuetools 2007, October 2007.

[24] Thomas Clausen, Christopher Dearlove, Philippe Jacquet, The OLSRv2 Design Team, and The MANET Working Group. *Optimized Link State Routing Protocol (OLSR) version 2*. Ietf draft, Internet Engineering Task Force, April 2010.

[25] University of Antwerp - PATS. *An implementation of OLSR in Click*. http://www.pats.ua.ac.be/software/olsr.

[26] David C. Plummer. *An Ethernet Address Resolution Protocol*. Ietf rfc 826, Internet Engineering Task Force, November 1982.

[27] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. *What Do Packet Dispersion Techniques Measure?* In INFOCOM, pages 905–914, 2001.

[28] TETRA association. *Home Page*. http://www.tetramou.com/.

[29] Danny Bontinck. *Wij zijn de verkenners van de brandweer*. Newspaper article - available online in dutch at http://www.nieuwsblad.be/article/detail.aspx?articleid=G78GOGND, July 2005.

[30] Richard Draves, Jitendra Padhye, and Brian Zill. *Routing in multi-radio, multi-hop wireless mesh networks*. In MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking, pages 114–128, New York, NY, USA, 2004. ACM Press.

[31] Zhenyu Yang, Chandrakanth Chereddi, and Haiyun Luo. *Bandwidth Measurement in Wireless Mesh Networks*. Project Report, 2004.

[32] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. *Link-level measurements from an 802.11b mesh network*. SIGCOMM Comput. Commun. Rev., 34(4):121–132, 2004.

[33] Kyu-Han Kim and Kang G. Shin. *On accurate measurement of link quality in multi-hop wireless mesh networks*. In MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking, pages 38–49, New York, NY, USA, 2006. ACM Press.

[34] University of Central Florida. *iperf- The TCP/UDP Bandwidth Measurement Tool*. http://sourceforge.net/projects/iperf/.

[35] Tony Sun, Guang Yang, Ling-Jyh Chen, Medy Yahya Sanadidi, and Mario Gerla. *A measurement study of path capacity in 802.11b based wireless networks*. In WiTMeMo '05: Papers presented at the 2005 workshop on Wireless traffic measurements and modeling, pages 31–37, Berkeley, CA, USA, 2005. USENIX Association.

4 Auto-Configuration and Easy Management

4.1 Introduction

In the previous chapter, protocols were designed to enable the efficient use of the wireless spectrum in dynamic multi-interface wireless mesh set-ups: both a distributed channel selection approach as well as a way to monitor the quality of an operational network were presented. As such, these protocols implement two important building blocks of the presented mesh architecture.

Among various other subsystems that could be investigated in order to further increase the stability and performance of the defined mesh architecture, the focus of this chapter is on finding a way to automate the deployment and expansion of wireless mesh and ad-hoc networks. While the algorithms from Chapter 3 enhance traditional network stacks with multi-interface and monitoring capabilities, an initial manual configuration from the mesh nodes is still needed. For example, a routing protocol needs to be selected and configured, an IP address is needed, and certain parameters of the protocols (e.g. number of network interfaces) need to be adapted according to the characteristics of each individual node.

Performing these initial configurations is a tedious and error prone task, and requires a thorough knowledge of (wireless) communication networks. If wireless ad-hoc and wireless mesh networking technology is to be used by the large public any time soon, reducing the complexity of installing, expanding and monitoring wireless mesh networks is of vital importance.

Therefore, in this chapter, an auto-configuration mechanism for the deployment of wireless mesh networks is introduced. These mechanisms allow wireless mesh networks to be set up a in seconds, requiring minimal user interaction. The resulting mesh network is automatically secured through the use of existing encryption techniques. Although the presented technique is generic in nature, as a use case, the deployment of IEEE 802.11a/b/g based wireless mesh networks is considered. Once deployed, the mesh network can be easily managed through a graphical user interface.

The auto-configuration, expansion and monitoring approach is then developed as part of a large integrated implementation, providing all functionalities needed to efficiently deploy, operate and expand wireless mesh networks. This ranges from device discovery over a routing protocol to a monitoring and configuration application which allows enabling access point interfaces installed on the mesh devices. However, the focus of this chapter especially lies with the development of the bootstrapping protocols providing initial configuration of mesh backbone nodes.

4.1.1 Applications

Auto-configurable wireless mesh networks may be used in many application domains. A first example is the use in home and office environments. An increasing number of devices around the home and office such as computers, smart phones, or Internet radios rely on a permanent connection to the Internet. Even in home environments, a single access point might no longer be able to provide the required network coverage. Whenever a wired backbone is too expensive or not practical to install, an auto-configuring wireless mesh network could be used as an alternative. However, mesh networks are seldom installed in homes or offices, as they are unknown by the general public and installation is too much of a challenge.

Secondly, a fast deployment solution enables to set up temporary networks in a cost-effective way at fairs or public events. The presented solution allows a large wireless mesh network to be deployed within minutes, without requiring any pre-configuration to be performed by the person(s) installing the network. New nodes can be used directly after unpacking, allowing fast and easy set-up.

Thirdly, a secure mesh network that is easily installed may be used while deploying an emergency communication infrastructure as previously depicted in Figure 2.18: quick deployment assures that no valuable time is lost, and network security protects sensitive information such as medical files or confidential voice communication that may be transmitted during the operation.

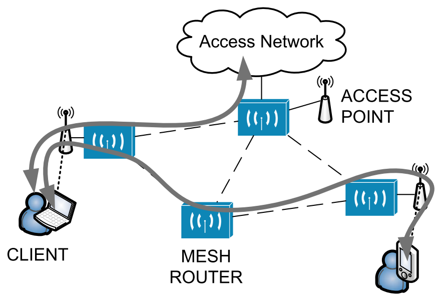


Figure 4.1: Backbone mesh architecture. Clients connect to the backbone via access points.

4.1.2 Features and design considerations

To achieve the goal of a secure auto-configuring mesh architecture, the protocols were designed with following key aspects in mind:

- **Auto-deployment, without sacrificing security.** Since ease-of-use is essential to encourage the use of wireless mesh networks by the general public, the focus of this chapter is on the design of a mesh network that can be installed and secured, right from the box. No knowledge about security or networking should be required, and setting up a network should be possible in an intuitive way. Furthermore, wireless networks may grow larger over time. The method must support the addition of network nodes at any time. Anyone authorized to access the administration interface is able to securely install new nodes at any location within the communication range of the already available network through the use of an intuitive GUI.

- **Prototype implementation.** Full auto-configuration of the system, device and protocol settings of a wireless mesh network is only possible through a complex interaction of protocols operating at different layers of the OSI stack. Some auto-configuration aspects have been extensively studied in the past and many implementations are available. For example, the Dynamic Host Configuration Protocol (DHCP, [1]) is a suitable solution for auto-configuring the IP parameters of the end-user devices in a mesh network. Other functionalities such as scanning for wireless networks are already included in wireless drivers. There is no need to re-invent existing and well-proven solutions; as will be detailed in this chapter, the presented auto-configuration solution makes use of several existing implementations. The interaction of protocols with existing implementations is very hard to study using simulations only.

- **Compatibility.** In the proposed mesh architecture from Section 2.5, *end-*

user device compatibility is supported by default since end-user clients connect through legacy access points (see Figure 4.1).

In order to support the *compatibility* of the auto-configuration protocols with *other mesh and ad-hoc architectures*, the auto-configuration subsystem is implemented in a modular way, enabling it to be plugged into existing network solutions with minor adjustments. The solution does not rely on existing, physical layer technology dependent beaconing mechanisms to perform neighbor detection. This enables the designed algorithms to run on top of multiple physical layer technologies, thus mixing several technologies in a single network overlay. While the routing protocol in the current implementation is based upon the OLSR routing protocol, it could be replaced with any proactive or reactive routing protocol.

Since the focus of this chapter is on the auto-deployment mechanisms, the algorithms will be developed and studied based on single-interface wireless mesh routers. Although this results in all backbone nodes being part of a single wireless collision domain thus negatively impacting the network capacity, potential issues that may be caused by channel selection protocols are avoided.

The chapter is organized as follows; first, Section 4.2 gives an overview of related work, and indicates how our approach contributes to the state of the art. Then, in Section 4.3, the auto-configuration solution is presented. A high-level description of a practical application scenario demonstrates the ease of use and advanced possibilities of the solution from a technical point of view, after which Section 4.4 details the techniques that are used behind each phase of the configuration mechanism and analyzes the resilience of the solution against different security attacks. Next, a prototype implementation of the solution is presented in Section 4.5, demonstrating automatic deployment of a secure wireless network, this time from a user point of view. Additionally, the feasibility of the solution is illustrated with measurements obtained from experiments on a wireless testbed. In Section 4.6, management extensions and future developments to the auto-configuration approach are treated. Finally, the chapter is concluded in Section 4.7.

4.2 Related work

As the primary contribution of this chapter lies in the development of an integrated secure auto-configuring wireless mesh network, the overview of related work focuses on related secure auto-configuration and bootstrapping techniques.

In order to build a wireless mesh network with as less effort as possible, auto-configuration and management strategies should be implemented at various layers of the OSI stack: mesh nodes need a network interface configuration, a valid security configuration and unique IP parameters. In addition, no reliable IP network can be built without a robust, well configured routing protocol that is able to recover from network and node failures. Likewise, parameters can be configured automatically to assist the application layer, such as providing a name list of service locations or configuring application layer protocol settings depending on the location or expected service quality of the network.

Each of these topics is a research area on its own, resulting in a wide range of literature related to auto-configuration topics at all layers of the OSI stack; for the automatic configuration of IP addresses, the authors of [2] and [3] give an overview of stateful and stateless address auto-configuration schemes that are found in literature. Within the IPv6 Internet standard specification [4], the IP address length is changed from 32 bits to 128 bits, which should not only lead to a greater number of addressable nodes, but also allow for a simpler auto-configuration of IP addresses. This IP address auto-configuration for IPv6 nodes is described in separate documents [5, 6]. Once an IP address is available, self-organizing routing protocols exist that are able to cope with the dynamics of wireless ad-hoc and mesh networks [7, 8], taking care of automatic recovery after link breaks. In order to provide network security, networks often rely on a shared secret key, for example, a WPA (Wi-Fi Protected Access, described in IEEE 802.11i, included in [9]) key might be required to gain access to a Wi-Fi network, or, when pairing bluetooth devices a common PIN (Personal Identification Number) code might be used. In order to avoid manual device configuration, proximity based binding mechanisms exist for wireless devices, such as used when binding a wireless keyboard to a USB receiver by pressing a button on each device. To assist the configuration of network devices, generic networking protocols such as proposed by the UPnP (Universal Plug and Play) Forum [10] industrial initiative have been developed, providing a base to provide network device auto-configuration ranging from addressing to device discovery and configuration.

The above list of example auto-configuration techniques for network nodes is by far exhaustive. However, most auto-configuration techniques in use today or described in literature expect a direct (wired or wireless) interface to be available on new devices for administrator configuration. As a result, end-users typically have to configure every device manually using an unsecure connection or default login settings. This makes the configuration of large networks tedious, time-consuming and error prone. Other auto-configuration techniques such as DHCP only provide solutions for incremental (and not initial) (wireless) device configuration, assume a certain trust relationship to pre-exist between the different devices (e.g. UPnP), or do not consider security at all. While wireless networking devices may work directly from the box, security settings are often disabled, and the lack of information on security settings in the user manual may cause the end-user or less experienced network administrator to experience configuration difficulties, thus causing usability issues [11]. Without modifications, these solutions are thus unsuitable for use in a potentially hostile wireless mesh environment where a direct interface with the devices is unavailable or unwanted.

In [12], the authors *do* consider the security issue and provide a certificate based solution mainly focusing on the address configuration in a dynamic ad-hoc network. As the authors consider an administrator-free network, nodes are accepted into the network after verifying the certificate of the new node with a common certificate authority, requiring nodes to be configured for a specific network owner at manufacturing time.

In [13], Balfanz et al. present a solution to authenticate devices in ad-hoc networks, based on exchanging authentication keys over a wireless link. However, their method requires physical contact to the devices which are to be connected (e.g. pressing a button on a printer at a public location to connect it to a personal mobile device), and relies on so called *location-limited channels* for initial information exchange. The idea of this channel is that the user can exactly determine which devices can communicate with each other.

The Cisco *Zero Touch Configuration* solution [14] lets network administrators add nodes to a network by adding MAC addresses to a controller's whitelist. Security during set-up phase is guaranteed by installing a default shared key on all mesh devices at the manufacturing stage. The use of the default shared key can be overruled by disabling the Zero Touch Configuration option and setting a different key manually.

In contrast with the above approaches, the targeted installation, expansion and management solution should not only configure network addresses of one single type of device from which all characteristics are known in advance: all relevant parameters to integrate any node into the mesh backbone should be configured, as long as the node supports the solutions that are developed in this chapter. Although this requires some pre-configuration to take place at manufacturing time, this pre-configuration should be completely independent of the network that the mesh nodes will eventually end up in. For example, a profile which indicates the capabilities of the device in terms of number of interfaces or technologies is an acceptable and required preconfiguration, however, a single default shared key which should be reconfigured afterwards is not. As such, the auto-configuration and management solution that is developed in this chapter should support the deployment of secure mesh networks using network nodes manufactured by different providers or having different characteristics.

Furthermore, while an initial information exchange over location-limited side channels may be user-friendly for interconnecting a limited number of devices, the physical installation of mesh nodes in large scale wireless mesh networks should be possible without requiring network nodes to be physically co-located at the time of first introduction. This is also a major advantage when large networks need to be expanded, as the physical node installation can be performed by handymen without worrying about any configuration details of the nodes. Once physically deployed, no physical access to the new node(s) should be required to complete the node configuration.

Finally, in contrast with a "configure MAC and wait"-procedure as followed by the Cisco solution, the presented protocols use a "verify and accept"-approach. When adding a node to a network using this approach, the person installing the network is informed about the presence of a new auto-configurable node through an administrative GUI which may be installed at any (authorized) device that is connected to the already existing backbone. As will be clarified in Section 4.3.2, the secure integration and configuration of a new node is then performed fully automatically by the auto-configuration protocols after verifying the identity of the new device with the help of the GUI. This approach is believed to be more intuitive to users without technical knowledge on wireless networks, since the users may act on an event rather than having to enter configuration details before deployment.

For a more specific overview of security issues and authentication protocols in ad-hoc networks, the reader is referred to [15] and [16].

As will be detailed in the remainder of this chapter, the presented and implemented wireless mesh auto-configuration method contributes to the state of the art by *(i)* providing an integrated and fully implemented solution based on the integration of novel auto-configuration concepts with existing security techniques, *(ii)* enabling secure device configuration anywhere inside the operational network without requiring a direct connection to the device, *(iii)* enabling auto-configuration across different platforms and transmission technologies, *(iv)* presenting a simple way to authenticate new nodes in the network, that does not require a deployment site specific factory pre-configuration.

4.3 Automatic configuration procedure

4.3.1 Functional requirements

In order to identify the different configuration steps needed when developing an auto-deployment mechanism for wireless mesh networks, it is verified how such networks are set up in the absence of auto-configuration techniques. Assume that wireless nodes are available, which already have an operating system installed. Typically, following steps are required *for every node* that is added to a Wi-Fi based mesh or ad-hoc IP network.

1. Gather essential connection details such as network name, security settings and IP ranges.

2. Configure the wireless network interface(s) of the device. At a minimum: set the network name. Optionally: set the communication channel, transmission power levels or potential other custom settings.

3. Configure the security parameters of the device. Typically, a WEP or WPA key is used.

4. Configure the IP settings of the device.

After completing these steps, provided a multi-hop routing protocol is installed and properly configured, a new node is integrated in an existing network, and the wireless mesh or ad-hoc network is expanded.

The above steps might not pose a challenge to a network administrator with experience in wireless networks, but is quite complex for the average computer user. Nevertheless, even for an experienced administrator, the above approach is tedious and has several drawbacks: first, in most cases, an administrator will need physical (wired) access to each device to be configured. Alternatively, a device might be pre-configured with e.g. a WPA passphrase that is labeled on the node. In the latter case, the physical access requirement is somewhat relaxed, but an administrator still needs to be in the communication range of the new device while configuring. Second, suppose a wireless node can no longer be part of the network, for example because an employee is leaving a company (in an ad-hoc network case), or a wireless backbone node is stolen (in a wireless mesh network case). As all devices in the network share the same passphrase, security is compromised and every device should be updated with new security settings using a failover secure channel. If no such failover mechanisms are available, wired access to the nodes is the only secure alternative to update the key on all devices.

An automatic configuration procedure should cover the same steps as listed above for manual configuration and thus needs to provide following functionalities when nodes are added to an existing network:

1. Find a way to detect available networks supporting the auto-configuration procedure, and configure the interface(s) accordingly.

2. Get the security parameters needed to integrate the node in the existing secure environment.

3. Get and set an IP configuration for the device.

As an additional requirement, an administrator should be able to accept new nodes which are added anywhere into the network, without the need for wired access or proximity to the new node, through a single interface. For those cases where security requirements are less stringent, the system should be able to fully integrate new nodes in the network, without requiring any form of user intervention at all.

4.3.2 Deployment Scenario

As stated in the introductory chapter, the design of a wireless ad-hoc network or mesh architecture is ideally based on distributed algorithms whenever possible, avoiding a single point of failure. However, there needs to be some form of centralized control for one-time registration and general coordination purposes: unless factory pre-configuration is taken to a level where every node is built on demand for a specific network of a specific customer, full auto-configuration and full security can never be combined without sacrificing either one or the other; first, there needs to be a way to identify new nodes as trusted in order to avoid malicious nodes from joining the network using the auto-configuration mechanism without the administrator's knowledge. Second, it must be guaranteed that when a new node is booted in the presence of multiple networks supporting the same auto-configuration mechanism, the new node will join the network of the user's choice. If not, there is a risk of unintentionally "losing" a newly acquired node to an other administrator's network.

The deployment protocols require all network nodes to have the same auto-configuration algorithms installed. However, the nodes react differently to the custom auto-deployment control frames, based on a *node profile*: one of the parameters in this profile indicates whether the node should act as a configuration *coordinator*, or as a generic wireless mesh node. This central coordinator node *does* need a onetime upfront configuration: an administrator needs to set a network profile (defining IP ranges and other options) and needs to generate a *Certificate Authority Certificate*, which will be used for securing the links during a later phase. If this one-time upfront configuration is unwanted, it is perfectly possible to pre-configure nodes as "coordinator node" at manufacturing time, as long as the user is satisfied with default network configuration and default IP ranges. Any new node which is added to the network should of course be compatible with the configuration mechanism as well. This compatibility is assured by providing the new nodes with *(i)* the algorithms needed for requesting, receiving and forwarding auto-configuration data, *(ii)* the algorithms for auto-configuration, *(iii)* a factory installed temporary certificate and corresponding private key, and a node profile containing at least a nodeID and a list of available network interfaces, *(iv)* an authorization code which is installed on the node, and made available to the buyer of the node (e.g. on a (removable) label on the device).

Figure 4.2 clarifies the configuration procedure. The principle is explained, omitting network security implementation details for clarity reasons. These details will be discussed in Section 4.4. After the coordinator node C boots, it automatically creates a new ad-hoc network. The idea is to add nodes gradually to the network, starting with a coordinator node, and adding as much nodes as required, simply by placing them *anywhere* in the coverage area of the network that has been created so far. In Figure 4.2a, the coordinator is up and running, thus is part of a

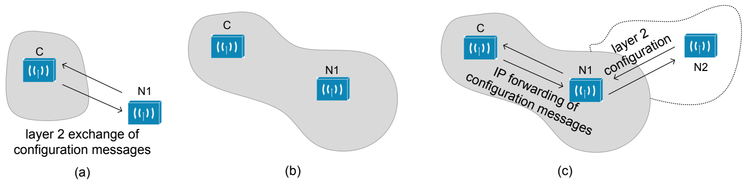

C
N1
layer 2 exchange of configuration messages
(a)
C
N1
(b)
C
N1
N2
layer 2 configuration
IP forwarding of configuration messages
(c)

Figure 4.2: Expanding the network through the auto-configuration mechanism. The gray areas indicate that a node is part of the secure mesh network. Node $N1$ is in the communication range of C and $N2$. $N2$ is out of the communication range of node C.

trusted service area indicated in gray. It is shown how a node $N1$, which is booted within the coverage area of the coordinator node, scans for beacons of available auto configurable networks, and subsequently sends a layer 2 configuration challenge to the coordinator. At this event, an administrator is notified of the fact that a new node wants to join the network. This notification can arrive at any device somewhere in the network: either a fixed administration terminal, or a mobile device carried by the admin, connected to the pre-existing secure network. Upon this notification, the administrator can either reject the request, or accept it by entering an authorization code. When a request is accepted, the coordinator automatically creates the necessary security configuration data and securely delivers it in one or more layer 2 packets to the new node. This is called the *security configuration phase*. The new node then verifies the security configuration data and data source, and if it is found valid, stores this new security information. In a second phase called the *profile configuration phase*, the new node asks for a network and node profile. Upon this request, the coordinator node generates profile data, and sends it to the new node in one or several layer 2 configuration packets. The new node installs the profile information and is now able to fully and securely integrate in the mesh network. This is when the routing protocol starts receiving and sending messages, expanding the mesh network to cover a larger area (Figure 4.2b). Finally, Figure 4.2c shows that, when new nodes such as $N2$ are deployed, there is no need for them to be in the proximity of the coordinator. The same layer 2 configuration procedure can be repeated in the coverage area of any available and configured backbone node, which will forward the layer 2 request over the previously established secure layer 3 backbone, through a multi-hop path, towards the node that is configured as a coordinator.

Every node needs to execute the configuration procedure only once, as the node stores the received configuration details locally. If a node fails and needs to be rebooted, it uses this stored data. The coordinator thus needs to be present only when new nodes are added to the network. If it fails or becomes unreachable, the

The device manufacturer is responsible for: pre-installed temporary certificate and corresponding private key; authorisation code (AC); node Profile		
administrator	**autoconf @ coordinator**	**autoconf @ new node**
power up node		
		scan and find existing auto configurable network
		send security configuration challenge
accept new node in network		
enter AC		
	create and send security configuration	
		receive response and verify AC
		install received security config.
		send profile challenge
	create and send profile	
		install received profile config.
		IP configuration and routing
New node is added to the network and completely configured. Additional configuration and / or monitoring can now be performed using the established and secure IP link		

Table 4.1: Overview of the actions during the configuration procedure.

mesh backbone remains operational. Note that storing the configuration details in case of power loss or node failure is optional: mesh nodes might be configured not to store any local configuration details.

4.4 Security and Auto-Configuration Details

Table 4.1 gives an overview of the auto-configuration steps described in previous paragraphs. The auto-configuration information is exchanged through a series of layer 2 and layer 3 configuration packets shown in Figure 4.3. In order to not overload the figure, every information exchange is shown as a single packet, while in reality, some steps of the procedure require multiple configuration packets because of link layer MTU (Maximum Transmission Unit) limitations. On the right side of the figure, four operational phases are indicated: the normal operation phase (n), the security configuration phase (sc), the profile configuration phase (pc), and the secure link setup phase (sl). In what follows, Sections 4.4.1 to 4.4.3 detail the security mechanisms of the initial set-up phases, after which Section 4.4.4 analyzes the security threats.

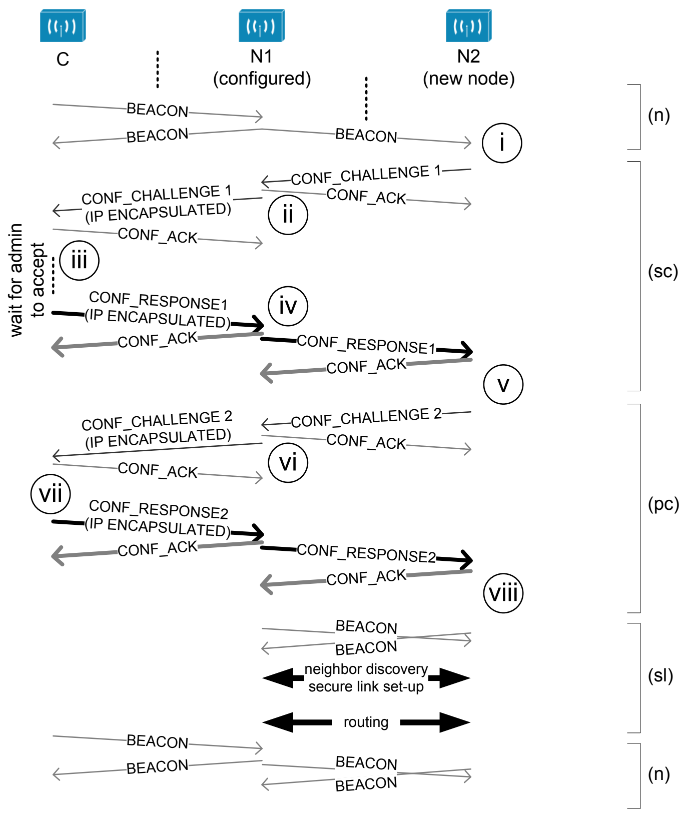


Figure 4.3: Schematic overview of the packet flow needed for configuring node $N2$ from Figure 4.2c. (n) normal operation; (sc) security configuration phase; (pc) profile configuration phase; (sl) secure link setup phase.

4.4.1 Security Configuration Phase

A node that needs to be configured scans for beacons that already configured nodes are broadcasting periodically to discover neighboring nodes. Detection of such *neighbor discovery beacons* indicates the presence of an auto-configurable network. When such a beacon is received, a unicast layer 2 *CONF_CHALLENGE1* packet is sent to the MAC address of the beacon sender, as shown in Figure 4.3. The *CONF_CHALLENGE1* configuration packet is depicted in Figure 4.4, and contains the unencrypted certificate that was pre-installed by the manufacturer, and an RSA signature –named after its developers Rivest, Shamir and Adleman– guaranteeing the message integrity [17]. The certificate includes the ID of the new node and the public key that goes with the pre-installed private key of the new

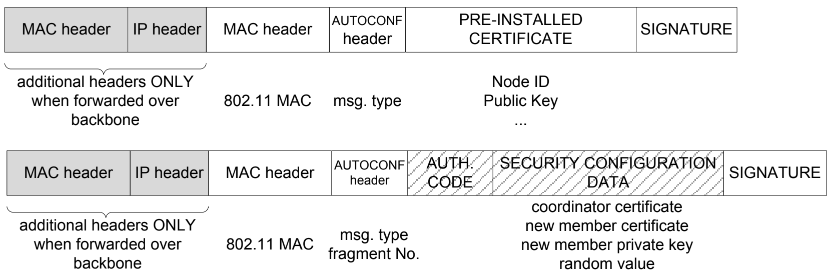


Figure 4.4: CONF_CHALLENGE1 (top) and CONF_RESPONSE1 packet format. The cross hatched data is encrypted. The gray headers are only present when packets are forwarded over the available secured links. L3 encryption details are not shown.

node.

When the challenge arrives at the coordinator node, either directly or through an already available secured path, the network administrator is informed of this event through a software administration interface. The administrator then has to inspect the validity of the certificate. Therefore, the administrator is presented with a public key fingerprint which can be compared to a fingerprint made available through a secure channel, such as an extra sheet of paper in the manual, a label on the device, a smartcard or any other data carrier included in the device box. The described way of supplying additional configuration data to the administrator is a feasible alternative to proximity-based solutions to carry out device pre-authentication. Alternatively, the administrator could authorize the new device by relying on an external certificate authority. After entering the authorization code, supplied in a similar way to the administrator, the system creates a sequence of *CONF_RESPONSE1* packets. The cross hatched parts of the message, depicted in Figure 4.4, are encrypted with the public key that was pre-installed on the node, so only the new node is able to decrypt the received authorization code and security configuration data. The new node can ascertain that the configuration data is produced and sent by the genuine administrator after verifying the authorization code, and installs the received certificates that were generated by the coordinator node. From this point on, the new node is authorized by the administrator, and the administrator is authorized by the new node. The new node now has a certificate which is signed by the coordinator node. As all nodes in the network have the coordinator certificate available, all nodes are able to verify the validity of this new, automatically installed member certificate. Note that while the proposed method thus still requires an action to be taken for every node that is added to the network, all actions are taken through a single administration interface: no individual connections to the new nodes are required.

4.4.2 Profile Configuration Phase

During the profile configuration phase (cf. Figure 4.3), the new node is provided with configuration data, enabling full integration in the wireless mesh network. This phase is explicitly separated from the security configuration phase, as this enables the new node to broadcast the minimum of unsecured information: only the temporary certificate is sent unencrypted. Sensitive data such as the device capabilities or device profile are always sent encrypted with the coordinator public key which is retrieved from the coordinator's certificate.

The configuration packets used during the profile configuration phase are similar to the security configuration packets, but now the newly installed keys from the security configuration phase are used. The *CONF_CHALLENGE2* configuration information is encrypted with the public key of the coordinator. The *CONF_CHALLENGE2* message also contains a random value that was earlier included in the (encrypted) security configuration data, confirming the origin of the *CONF_CHALLENGE2* message. Consequently, as the challenge packet arrives at the coordinator, no further administrator intervention is required. The coordinator node creates a node and network configuration profile based on the received pre-installed profile information, and encrypts it with the (new) public key of the new member. The network profile informs the new node about network wide parameters such as IP ranges and routing protocols used in the network. The node profile holds additional configuration options that are specific for the node, such as its name.

4.4.3 Normal Network Operation

After the security configuration and profile configuration phases are completed, all configured nodes obtain a common trust relationship through the use of certificates. Every configured node has a private key, a signed certificate (a public key signed with the private key of the coordinator) and the coordinator CA certificate. Together with the received profile information, everything is present to start establishing or join the secure mesh network.

The network mechanisms during normal network operation are based on the Virtual Private Ad Hoc (VPAN) framework designed and developed by Jeroen Hoebeke et al. [18, 19]. In the cited work, the design of the VPAN platform is detailed. The VPAN system allows the creation of virtual overlay networks consisting of distributed groups of devices (i.e. clusters) at different geographical locations. One of the VPAN functionalities is the organization of (previously configured) local devices in secure clusters. The local clustering mechanism and security provisions of the VPAN platform are used to organize and secure the operational mesh network. As such, the security and profile configuration phases add a modular node initialization approach to the existing VPAN framework, either through a direct connection with a coordinator node, or through a multi-hop connection with an already configured node.

The configured mesh nodes are broadcasting beacons periodically over all network interfaces that are used for the mesh connectivity. Upon reception of a beacon of another trusted mesh node, a three-way challenge-response session will take place. Using the installed certificates and 1024-bit public key cryptography, the nodes perform mutual authentication, store all link information and exchange a 128-bit short-term pairwise unicast key, and node specific broadcast session keys. These short-term keys are then used to encrypt all further unicast and broadcast communication using symmetric AES [20]. This encryption is further complemented with a 160-bit SHA1 digest [21] and replay counter to offer data integrity and to protect against replay attacks, resulting in a fully secured communication link. The exchanged neighbor information additionally includes the MAC addresses of the interfaces as to avoid the use of ARP messages, protecting against ARP spoofing [22].

In order to enable layer 3 IP connectivity between the neighboring mesh nodes, address assignment and routing capabilities are needed. Every mesh node will therefore automatically generate an IP address within the addressing range specified in the received profile information and the VPAN duplicate address detection mechanism will guarantee the uniqueness of the generated addresses. Alternatively, a fixed address could be assigned and distributed during the profile configuration phase. Every node is only assigned a single IP address, independent of the number of underlying interfaces used. Similar to the multiplexing system that was described in Section 3.2.3.1, a convergence layer hides the interface details from the networking layer and transparently takes care of the management of the interfaces, selection of the best link in case multiple links are possible and the encryption of the forwarded traffic using the corresponding unicast and broadcast session keys.

Using the assigned address and the neighbor information for the detection of new links and link breaks, an OLSR based proactive ad-hoc routing protocol exchanges encrypted routing information, enabling fully self-organizing and secure end-to-end connectivity between all mesh nodes.

Finally, note that the VPAN technology used during the normal network operation empowers the mesh network with additional advanced mesh networking functionality, such as running different secure mesh networks on top of the same hardware, securing remote access to the mesh network, or interconnecting mesh networks at different locations over the Internet.

4.4.4 Security Threat Analysis

In this section, potential attacks to the auto-configuration system are listed and it is indicated how the system is able to detect and resist attempted security breaches during the initialization phase of a new node. Section 4.4.3 described how during the normal network operation phase, all traffic between the mesh nodes is secured. Provided the certificates that are used to exchange a symmetric key were received in a secure way, and the unicast and broadcast keys are periodically refreshed, it can be assumed that this encryption guarantees full security. As public key cryptography is well described in literature [23], the analysis in this section thus is limited to the initial key installation phases shown in Figure 4.3, which are sent at layer 2. This layer 2 communication can either happen between a new node and the coordinator node directly, or between a new node and an intermediate, already configured mesh backbone node. However, as from the viewpoint of an external (malicious) or new node, configuration through an already configured node is essentially indistinguishable from direct configuration, both cases can be treated identically. As a final restriction, nodes that are already part of the trusted backbone are considered fully trustworthy. Should any node become physically compromised, its certificates should be revoked.

Fake beacon. (Figure 4.3.*i*). Neighbor discovery beacons are not encrypted. A malicious node might generate a fake beacon. This will result in a genuine *CONF_CHALLENGE1*, revealing the pre-installed public key of the new node. Although the malicious node might generate a fake *CONF_RESPONSE1* in response, the attacker does not know the authorization code. This will be detected by the new node, and the security configuration data will be rejected. While not implemented in the current version of the code, repeated guessing of the authorization code could be detected by the new node, which could then refuse any further configuration attempts. However, even without such detection mechanism, it is highly unlikely that an attacker would be able to guess the code before the genuine node configuration is completed. If such detection mechanism would be available, the described attack could be used as a denial of service (DoS) attack. However, as there are far more easier ways to trigger a DoS attack in a wireless environment, such as the continuous transmission of interfering RF signals, presenting a solution to this type of DoS attack is not considered a priority.

Fake *CONF_CHALLENGE1*. (Figure 4.3.*ii*). Fake *CONF_CHALLENGE* messages cannot be detected automatically by the system. They will be forwarded to the coordinator node and show up in the administration GUI. It is up to the person installing the network to identify these requests as fake requests based on one of the suggested verification methods presented in Section 4.4.1. In order to avoid overloading the network administrator with configuration challenges, the administrator might choose only to allow nodes to join during a limited period of time. Alternatively, if, for example, a smart card based authorization system is chosen, the administrator application could filter out false requests automatically.

Fake *CONF_ACK*. (Figure 4.3.*iii*). The payload of a *CONF_ACK* message contains the type and sequence number of the message that is being acknowledged. When a new node receives a *CONF_ACK* on its request, it is programmed to stop sending requests to the originator of the *CONF_ACK* message in order to decrease network load while waiting for a *CONF_RESPONSE*. This will only influence the auto-configuration operation in case the intended receiver of the *CONF_CHALLENGE* did not receive the genuine message, which would result in the new node not receiving its configuration data, but never in compromised information. Fake *CONF_ACKS* are however unlikely to cause any trouble for two reasons. Firstly, in case the node is able to reach several configured nodes, a single successfully received *CONF_CHALLENGE* message is all it takes to get the node configured. Secondly, if after a certain amount of time the reception of the challenge message is acked but no challenges are subsequently received, the node transmits a new request.

Confidentiality and integrity of *CONF_RESPONSE* messages. (Figure 4.3.*iv*). An attacker, eavesdropping on configuration response messages, cannot decode the authorization code or security configuration data, as they are encrypted with the public key of the requesting node. Data integrity is guaranteed by the signature. On the other hand, (Figure 4.3.*v*) the new node is able to decode the message using its private key, and can then also verify the validity of the message thanks to the authorization code. An attacker might try to replay certain *CONF_RESPONSE* messages. However, the new node is able to detect duplicates because the fragment number is included in the packet.

The security analysis of the configuration profile steps *(vi)* to *(viii)* from Figure 4.3 can be treated identical as steps *(ii)* to *(v)* and is therefore omitted.

4.5 Prototype Implementation and User Experience

4.5.1 Implementation Overview

The auto-configuration mechanism was implemented using the Click Modular Router platform [24], running on top of Alix system boards [25], using Madwifi as wireless network driver. Figure 4.5 shows an Alix system board. Its form factor is close to what may be expected from a commercial platform. A high-level overview of the implementation is displayed in Figure 4.6, and shows how the

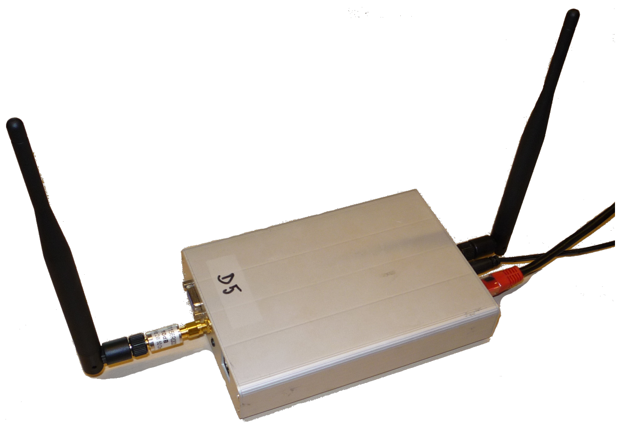


Figure 4.5: The mesh routers are implemented on Alix system boards.

auto-configuration algorithms are implemented in a subsystem, which runs separately from the *core networking functionalities*, thus allowing the implementation to be easily plugged into existing architectures with minimal adaptations. The latter functional block holds, amongst others, the routing and forwarding algorithms. In this example, it can be seen how a single Ethernet and a single IEEE 802.11g interface are hidden from the core and auto-configuration algorithms by an *abstraction layer*, enabling the system to work with any interface configuration. In order to support scanning and network cycling (see further, Section 4.5.2) for wireless interfaces, a *Wireless Interface Controller* is added.

After passing through the abstraction layer, all packets are sent to a *classifier* which determines whether the received packet is routed to the auto-configuration subsystem or the core networking functionalities. On initialization of a new node, the classifier is decoupled from the core networking functionalities and all traffic is sent to the auto-configuration subsystem. When the auto-configuration procedure completes, it configures parameters such as the IP address in the network core. From this point on, all packets, except those related to the auto-configuration procedure, are routed to the core networking functionalities. If at this point, the auto-configuration algorithms need to forward a packet over IP, e.g. relay a configuration challenge to the coordinator, the control messages are forwarded through the core networking functionalities. Layer 2 packets are sent directly to the interface abstraction layer.

4.5.2 Network Cycling Subsystem

As stated before, the auto-configuration system is able to configure networks using any type of wired or wireless interface. When a new auto-configurable node boots

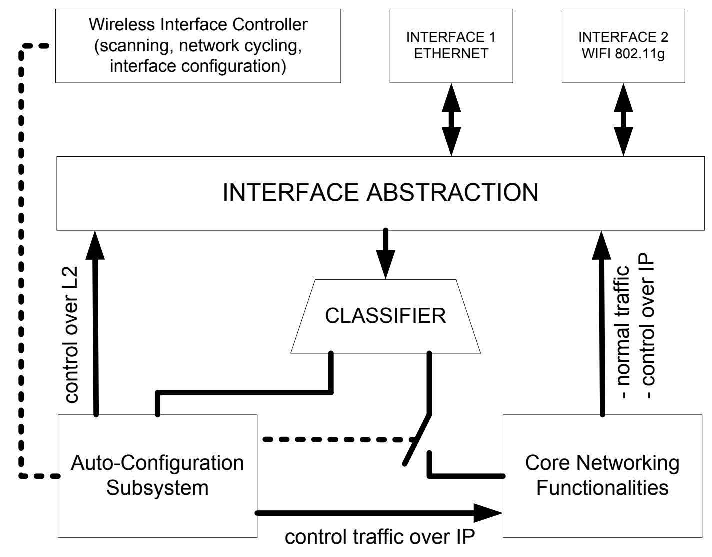


Figure 4.6: High-level overview of the auto-configuration scheme.

in the network, it scans for neighbor discovery beacons on all network interfaces. For wired systems or systems where only a single wireless channel is used, the configuration procedure is trivial: after sending the *CONF_CHALLENGE1* request, a new node remains in the same state, waiting for an administrator to acknowledge the request and the *CONF_RESPONSE1* message to arrive. However, when multiple wireless channels need to be scanned because multiple auto-configurable networks were detected on different wireless channels, the configuration procedure is more complex. First, a list of available networks is determined by letting the scanning procedure of the wireless driver scan for ad-hoc networks. Since it is impossible to know which of the available networks will accept the configuration request, the new node has to keep cycling the different channels on which a *CONF_RESPONSE1* message might arrive. Say the time that the node stays on every candidate channel is denoted as T_c. Successful configuration can then be guaranteed by retransmitting the first packet of the *CONF_RESPONSE1* sequence every $T_r < T_c$ seconds from the last hop after the administrator's approval, until the new node acknowledges reception. Optimized values for the retransmission time T_r and channel scanning time T_c are determined in Section 4.5.6.

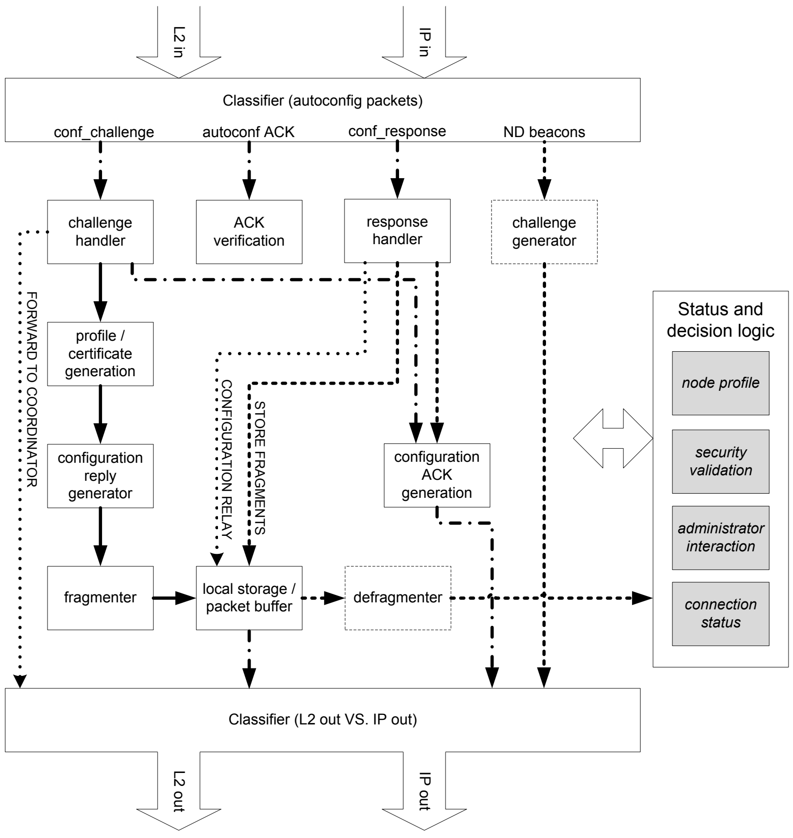


Figure 4.7: Schematic overview of the Auto-Configuration Subsystem.

4.5.3 Auto Configuration Subsystem and Reliability

Figure 4.7 shows a schematic overview of the internal construction of the auto-configuration subsystem, implementing the functionality from Section 4.4. Both L2 and L3 configuration packets are sent to an initial classifier, which sorts the auto-configuration packets according to their type. The coordinator nodes and normal nodes run identical code, but behave differently according to their node profile. When first booting a node, only the dashed flows from Figure 4.7 are active: a new node can interpret neighbor discovery beacons (NDB), generate configuration challenge messages, and store partial configuration response messages which are sent to one of the MAC addresses of its interfaces. When all parts are received, they are reassembled, and processed.

The dash-dotted lines indicate flows which are in use during several node roles;

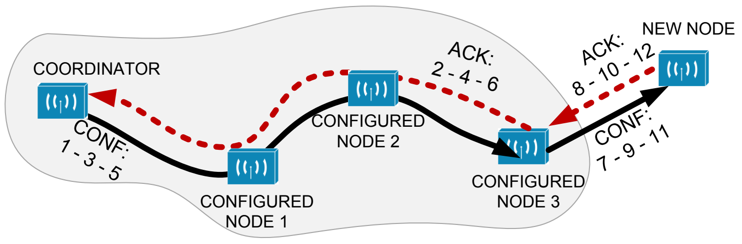


Figure 4.8: Illustration of reliable delivery of configuration messages when sending a three-fragment configuration message from the coordinator node to a new node in the network. The numbers indicate the packet sending order.

for example: auto-configuration ACKs are generated by new nodes, by coordinator nodes, and by configured nodes. The full lines indicate flows that are only followed by the coordinator node, and the dotted lines by configured nodes, when forwarding challenges to the coordinator, or responses to new nodes that are about to be configured.

Depending on the node role, the *packet buffer* contains packets holding fragments for different purposes. In cooperation with the ACK functional blocks, full transmission reliability is guaranteed. After creating and fragmenting profile and certification data, the buffer of the coordinator holds packets to be sent to the new node. If the coordinator cannot contact a new node over a direct L2 path, the packets will be forwarded to an already configured node that does have a link to the new node. This configured node uses the packet buffer to store all configuration packets originating from the coordinator, before sending it to the new node. Buffering the configuration messages at the last configured node before transmission to the new node is done because of the previously described network cycling problems: configuration packets which are sent in vain, because the new node is awaiting the possible arrival of configuration messages on a different channel, are now only sent in the environment of the node to be configured, while they otherwise would trigger a large amount of useless multi-hop transmissions. Especially in networks where new nodes are separated by many hops from the coordinator, this would result in a large amount of interference.

Finally, the new node stores incoming configuration packets until all are received, after which the entire configuration message is defragmented and, if the security parameters are found to be valid, processed and stored.

Packets are removed from packet buffers only after receiving a *configuration ACK (CONF_ACK)* message. The *CONF_ACKs* should not be confused with the ACK packets as used in IEEE 802.11: the former are required to guarantee full reliability, even in case multiple MAC retransmissions fail. Furthermore, they guar-

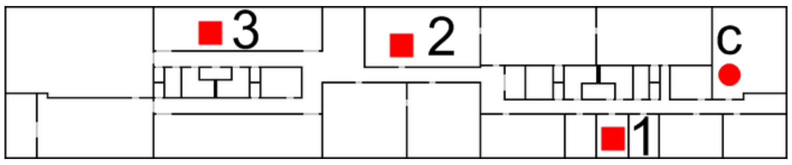


Figure 4.9: Floor plan of test area. Area of approximately 90m x 18m.

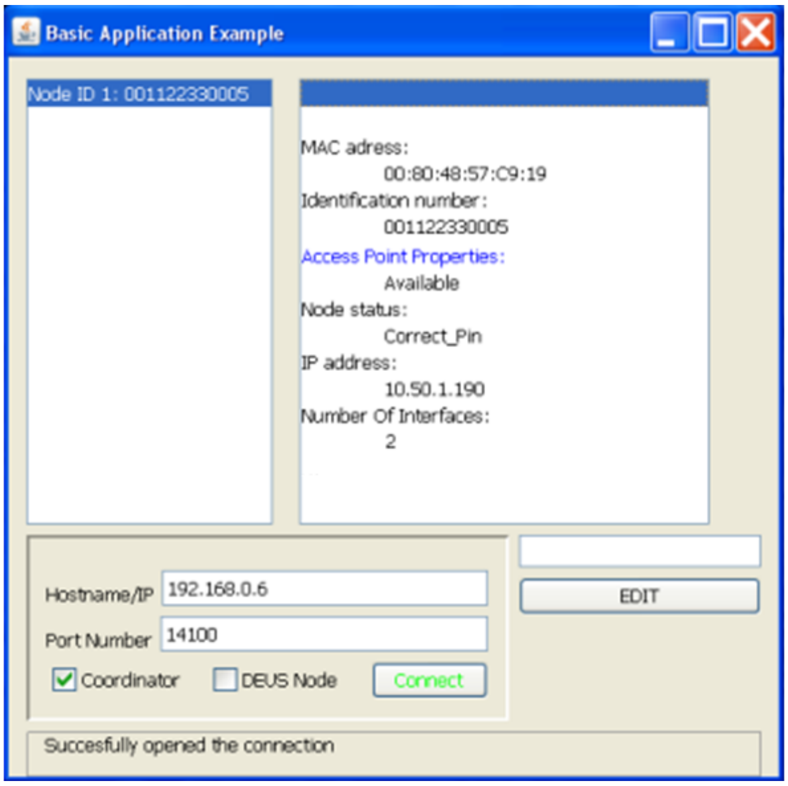
antee the operation of the protocol over any transmission medium and technology. The sender waits for a configuration ACK message on every packet, before sending the next packet from the buffer. If a *CONF_ACK* does not arrive within a predetermined period T_{ACK}, the packet is retransmitted. When packets are forwarded over the IP link, *CONF_ACK* messages are sent from the last configured node able to receive the secured IP messages in order to indicate the correct reception to the coordinator node. The acking mechanism over the IP link is not implemented on a link-by-link basis, as the pre-existing mesh backbone is generally considered to be stable.

Figure 4.8 shows an example in which a three-fragment configuration message is sent (without any retransmissions) to a new node. Each number represents a packet, and indicates the order of packet transmissions. Note that, if the third configured node would fail during the configuration procedure before the new node was completely configured, this is detected by the new node. If, despite the failure of node 3, the new node is still within the coverage area of the operational mesh, it will transmit a new request for configuration through an alternative path.

4.5.4 Qualitative test results

Qualitative tests indicate the simplicity of the proposed scheme and were performed as follows. Figure 4.9 shows the floorplan of the third floor of the IBCN research group building. For this test, a first node is configured as a coordinator node and is put on the location of the dot labeled C at the right side of the figure. Three other mesh nodes labeled 1 to 3, each with their own "factory" pre-installed certificate and authorization code are made available to the person installing the network. Although this person does not need to have any knowledge about communication networks, this person is called the administrator for simplicity reasons. While in a real-life situation, the administrator would find the authorization code and a hash of the pre-installed certificate, or any of the other previously discussed alternatives (cf. Section 4.4.1), included in the (sealed) box of the newly acquired device, this information is now presented on an information sheet.

After the coordinator node boots, a computer is connected to its wired interface. On this computer, our administrator GUI is started. The JAVA GUI is shown in Figure 4.10 and makes a TCP connection to the coordinator device. Through



MAC adress:
00:80:48:57:C9:19
Identification number:
001122330005
Access Point Properties:
Available
Node status:
Correct_Pin
IP address:
10.50.1.190
Number Of Interfaces:
2

Hostname/IP: 192.168.0.6
Port Number: 14100
[✓ Coordinator] DEUS Node

Succesfully opened the connection

Figure 4.10: Administrator GUI.

this connection, the administrator can monitor and control the deployment of additional nodes. After booting node 1 (cf. Figure 4.9), its node ID appears on the administration interface. The information sheet enables the administrator to verify whether the node on the GUI is allowed to join the network. To complete the action, the corresponding authorization code is entered through the GUI. The GUI also shows an up to date view of the connection status of the new node, and informs the administrator of any problems, such as a wrong authorization code. The system now automatically creates the necessary security and network profile information for the new node, performs the necessary packet exchanges, and the procedure is completed. Should it be desired, the user can now adjust properties of the device through the same user interface. In case the auto-configuration procedure is used to add a wireless mesh node with two wireless interfaces, one of the interfaces can be configured as an access point in order to support client traffic over the mesh.

The new node is now integrated in the network, and both nodes 2 and 3 are booted and show up on the administrator GUI after about 50 seconds. From the

node boot time	45 s
security configuration phase (coordinator side)	1.68 s
security configuration phase (new node side)	0.13 s
network delay	$O(ms)$
profile configuration phase	0.18 s
secure link setup and routing	$< 1\ s$

Table 4.2: Typical durations of configuration steps.

point of view of the administrator, both nodes are equivalent, although, behind the scenes, the configuration challenge of node 2 arrives through a direct layer 2 path at the coordinator node, while the challenge of node 3, being out of direct range of the coordinator node, is forwarded by node 1. After entering the corresponding authorization codes, both nodes are integrated in the secure network environment.

4.5.5 Quantitative test results

In order to further quantify the solution, additional experiments were performed, in which the coordinator node was configured in an "accept by default" mode, such that every *CONF_CHALLENGE1* was automatically followed by the generation of security and profile information, sending a *CONF_RESPONSE1* message as soon as the security information was available. Without the administrator acceptance delay, node boot time (about 45 seconds) and scanning delay, measurements show that the time between reception of a neighbor discovery beacon and node profile installation is on average 1.99 seconds if a direct connection exists between the new node and the coordinator. About 1.68 seconds of this time is spent generating and processing the security certificates at the coordinators side, and an additional $T_{completion}$ = 0.31 seconds are needed to complete packet exchanges, and the security and profile configuration phase.

The test results show that the transmission delay is in the order of milliseconds. Consequently, multi-hop configuration increases the total configuration duration only marginally. Since the control packets are prioritized at the output queue, the influence of background traffic on the configuration duration is minimized. It can be concluded that the total time for full secure and automatic integration of a new node after completing the boot sequence in the absence of networking errors, and with only a single auto configurable network present is typically less than 3 seconds, administrator approval time and scanning overhead excluded. This means that a secure multi-node mesh backbone or ad-hoc network covering a large area can literally be configured in minutes with a few simple steps: unpack the new nodes, distribute them across the area to be serviced, boot the nodes, and approve the new nodes' configuration requests by entering the authorization codes.

In total, only 2 (i.e. 1 $\times$ security configuration phase, 1 $\times$ profile configuration phase) configuration challenge packets, 3+2 configuration reply messages and the 7 corresponding *(CONF_ACK)* packets are needed. The typical durations of the configuration steps when using Alix3c3 devices are summarized in Table 4.2. Obviously, when the protocols are installed on different devices, the duration of configuration steps such as the node boot time and time needed for the generation and installation of certificates may be different. The parameter optimizations in the next sections are based on the implementation on top of the Alix3c3 devices; however, the formulas are generic and may be used to evaluate protocol settings when other hardware is used.

4.5.6 Optimized parameters and overhead

During the scanning procedure, a mesh node scans every network for T_c seconds. As a configuration challenge is triggered after receiving a neighbor discovery beacon, and these beacons are sent every T_{NDB} seconds from all interfaces of every configured mesh node in the network, T_c should be configured to at least the neighbor discovery beacon interval T_{NDB} in order to ensure a beacon is received.

In addition to the trigger functionality, the beacons are also used to enable discovery of new links, initiate the secure link set-up, and to detect link breaks. In order to guarantee fast detection of new and broken links, T_{NDB} should be set to a small value. On the other hand, increasing T_{NDB} reduces the overhead. With the method described in [26], it is calculated that in an IEEE 802.11g network, in a lossless situation with our beacon payload size of $24bytes$ and a broadcast transmission rate of $1Mbps$, each beacon requires a maximum share of $T_{beacon} = T_{DIFS}+T_{BACKOFF}+T_{DATA} = (28+139.5+548)\,\mu s \approx 716\,\mu s$ of the wireless medium. Thus, when in a certain area B nodes are sending beacons, following percentage of time is minimally available for sending other traffic:

$$P_{available} = 100 \cdot \left[1 - \left(\frac{1}{T_{NDB}} \cdot T_{beacon} \cdot B\right)\right] \% \qquad (4.1)$$

This equation is plotted in Figure 4.11. While $100\,ms$ is the default beacon time on most Wi-Fi devices, the figure clearly shows that choosing a similar value for our own beaconing system is unacceptable in dense network deployments. The wireless medium busy time is significantly reduced by configuring T_{NDB} to 2 seconds, leaving over 99% of the wireless medium available when up to 27 backbone mesh nodes are used in a node's local interference range.

The network cycling method described in Section 4.5.2 introduces a scanning delay to the total configuration time; recall that during a scan cycle, the new node first gets a list of available wireless networks, after which it configures itself to each of the possible candidate networks for a predefined scanning time T_c. In the wireless driver, the actual scanning for networks is performed as a background process, making the time needed to get the list of available networks negligible

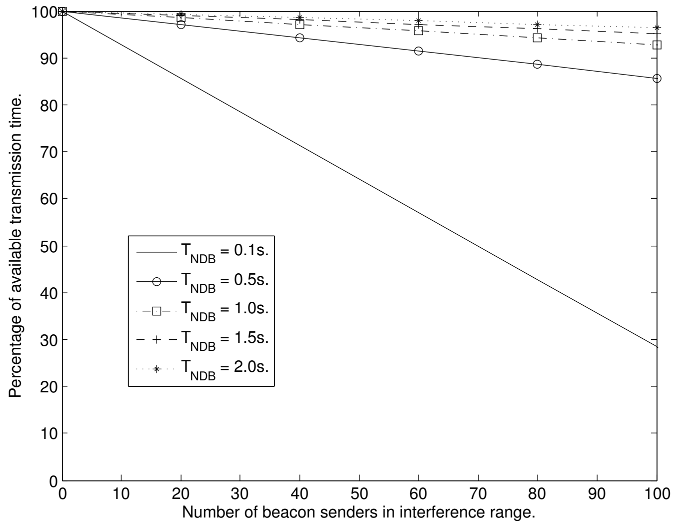


Figure 4.11: Percentage of remaining bandwidth after beacon overhead, for networks of varying density and for different neighbor discovery beacon intervals.

compared to T_c. A network scanning delay is suffered each time the new node is configured to a different channel than the target network by which it will eventually be configured. This may occur when new nodes are booted in an environment where several auto-configurable networks are available. Furthermore, when the "accept by default" mode is disabled and manual authorization is required, the exchange of auto-configuration packets is interrupted during the security configuration phase; while waiting for administrator approval, the new node keeps cycling the different candidate networks, thus causing an additional delay. Even under an "accept by default" policy, scanning delay may be suffered when a large number of nodes is booted at the same time, flooding the coordinator with configuration requests such that the procedure is not completed in a single cycle.

More specifically, a first additional scanning delay is suffered after booting the node: the statistical chance of the new node immediately contacting the intended network when selecting a network from its scanned network list equals $1/N$, where N is the number of networks found. Every failed attempt results in a pre-authorization delay equal to the scanning time T_c, leading to a *maximum* pre-authorization scanning delay equal to:

$$(N-1) \cdot T_c + T_{NDB} \tag{4.2}$$

T_{NDB} is added to account for the maximum time a node can wait on the correct channel for a beacon to arrive. Furthermore, in order to assure that a beacon is

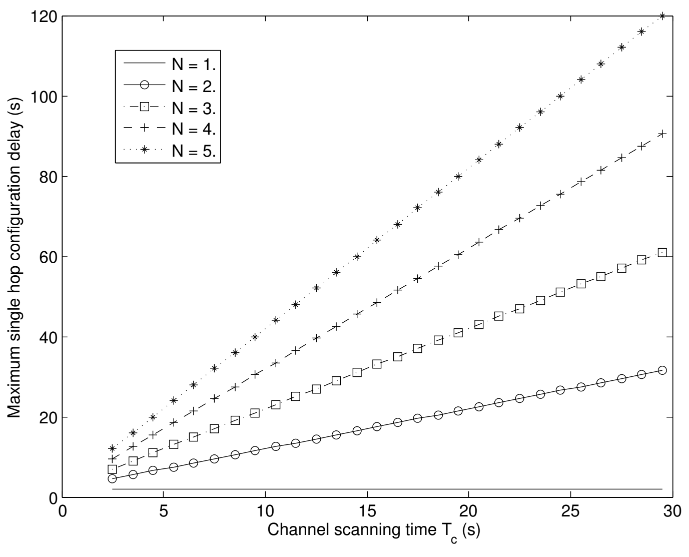


Figure 4.12: Maximum pre-authorization delay for a varying number of discovered networks N and T_{NDB}=2 seconds.

received during T_c, following relation is required:

$$T_c = T_{NDB} + \delta_1 \tag{4.3}$$

In this equation, δ_1 is a small positive time margin to maximize the chances that sufficient time is left for generation of the configuration request and receiving the configuration acknowledgment after receiving the beacon, say e.g. $\delta_1 = 0.5\,s$. Figure 4.12 visualizes equation 4.2 with T_{NDB} set to 2 seconds. T_{NDB} determines the vertical offset in the graph. The figure shows how the maximum single-hop configuration delay increases linearly with the channel scanning time T_c and how the maximum configuration delay rises faster with T_c as the number of networks found increases.

A second, post-authorization delay can be caused by the new node not necessarily being configured to the correct channel as the configuration reply messages arrive. In addition to a delay, identical to equation 4.2 but replacing T_{NDB} with the retransmission interval T_r, a configuration packet overhead is suffered, *maximally* equal to:

$$\lceil (N-1) \cdot T_c / T_r \rceil \tag{4.4}$$

This formula can be understood as follows: under the assumptions that one of the N networks in the candidate list holds the actual target network, the maximum number of timeslots with duration T_c within which configuration is not possible

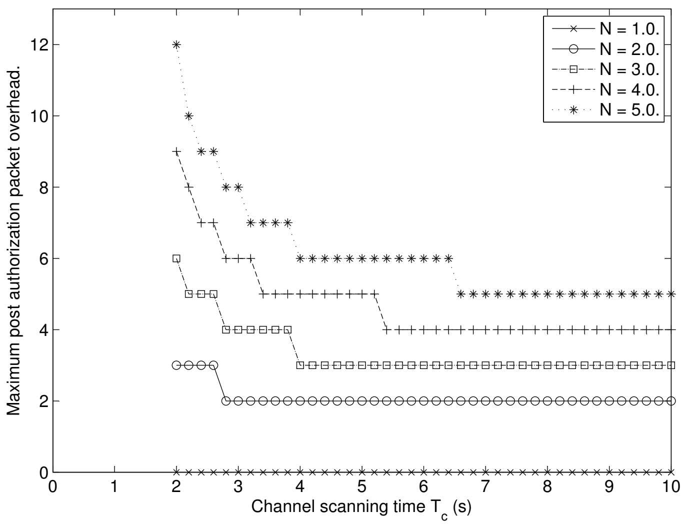


Figure 4.13: Maximum packet overhead in the post-authorization phase, for a varying number of discovered networks N.

equals $(N-1)$, accounting for a period of delay of $(N-1) \cdot T_c$. During any given time period t before successful configuration, the number of configuration packets that are sent maximally equals $\left\lceil \frac{t}{T_r} \right\rceil$. In this case, t equals $(N-1) \cdot T_c$.

In order to allow successful configuration, the T_c should at least be equal to:

$$T_c = T_r + T_{completion} + \delta_2 \tag{4.5}$$

In the right-hand side of the equation, T_r is the longest time before a configuration reply message is received during the scanning cycle (i.e. if the reply was just missed before being configured to the correct network), $T_{completion}$ is the average time needed by the system to complete the profile configuration phase after receiving the first $CONF\_RESPONSE1$ message, previously determined in Section 4.5.5, and δ_2 a positive time margin, needed as a safety margin to account for minor variations of the configuration duration, set to 1 second. Using the values of Table 4.2, equation 4.5 is concretized to $T_c = T_r + 0.31 + 1$. Given this relation between T_r and T_c, the maximum packet overhead (expr. 4.4) can be rewritten as a function of T_c and N:

$$\lceil (N-1) \cdot T_c/(T_c - 1.31) \rceil \tag{4.6}$$

This expression is plotted in Figure 4.13 for $T_c > T_{NDB}$ (cf. eq. 4.3), and shows that for a certain number of networks found N, with $N > 1$, the retransmission overhead measured in number of packets is smaller for larger values of T_c.

T_{NDB}	2000 ms
T_c	2620 ms
T_r	1310 ms

Table 4.3: Optimized settings for neighbor discovery interval, channel scanning time and retransmission time, minimizing configuration delay and packet loss.

When $T_c/(T_c - 1.31) \leq N/(N-1)$, the maximum packet overhead converges to N.

While both the pre and post-authorization delays are smaller for smaller values of T_c, the (maximum) packet overhead of the configuration procedure increases, albeit not dramatically. While the packet overhead increases when several networks are co-located, the overhead measured in packets per second is independent of N. In most wireless mesh use cases, a fast configuration will be preferred to saving a few packet transmissions. However, because of equation 4.3, T_c cannot be chosen arbitrarily low. Since in most cases, not too many networks will be co-located, T_c is configured to an aggressive $2620ms$, satisfying (4.3) while minimizing the packet loss for N up to 2. Table 4.3 summarizes the selected (optimal) configuration parameters for the considered implementation on top of alix3c3 boards, given the chosen safety margins for δ_1 and δ_2. Note that even more aggressive settings are possible by reducing these safety margins. However, reducing the safety margins decreases the chances on full configuration during a single scanning cycle, which could eventually result in a more slow configuration and an increased packet overhead.

4.5.7 Testbed results

In order to quantify the stability and scalability of the solution under real-life test conditions, experiments were performed on a subset of the alix3c3 based Wi-Fi testbed described in Appendix A, with an increasing number of nodes. Table 4.4 summarizes the total time needed to complete configuration of a network of different node sizes, in "accept by default" mode, using the configuration parameters from Table 4.3. The time is listed with the reception of the first *CONF_CHALLENGE1* message at the coordinator as reference starting point, and the last *CONF_RESPONSE2* to reach a new node as end point. Figure 4.14 shows the cumulative number of configured nodes as the experiment advances, for tests with a varying number of nodes. The figure shows that the solution always results in a fully configured network. The measured configuration time is relatively close to the theoretical maximum configuration rate (leftmost graph), which is calculated to be 1.81 nodes/second using the values of Table 4.2. Note that this maximum rate is mainly limited by the processing capabilities of the coordinator node, which is responsible for the certificate generation. While the initial configuration rate is rel-

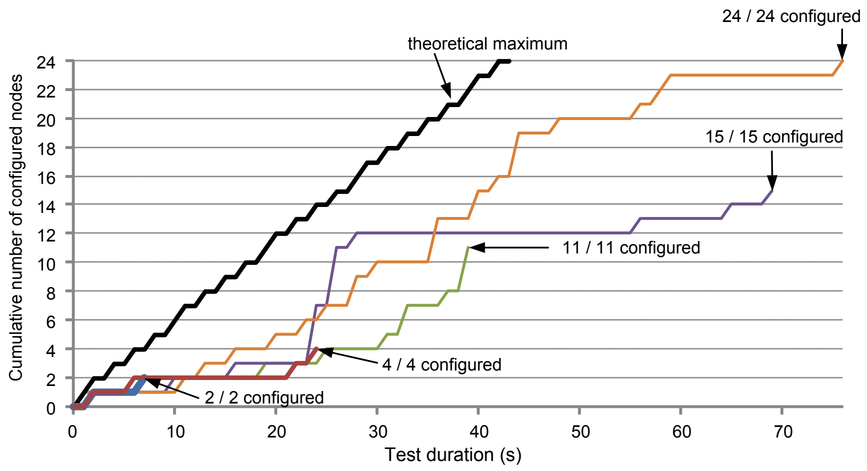


Figure 4.14: Cumulative number of configured nodes for varying number of nodes in the network.

atively low due to multiple simultaneous configuration requests triggering parallel certificate generations at the coordinator, the configuration rate rises towards the end of the experiment: if multiple certificates become available at the same time, several nodes might be configured nearly simultaneously.

The tests were performed in the same office environment, with the coordinator node at the same location, and all new nodes booting simultaneously. Note that, even at the lowest possible transmission power, the largest number of nodes were able to contact the coordinator directly. These large-scale tests prove that the algorithms can be used in larger and dense networks, and provide the necessary robustness to cope with packet loss that inevitably goes with dense network deployments. The average configuration rate is only listed in the table in order to give an idea on the configuration speed. As the value is totally determined by the last node to be configured (which, in the case of the tests of 24 new nodes, was well behind the configuration completion of the penultimate node due to accidental repeated packet loss), the value should not necessarily be interpreted as the average configuration duration per node. However, it shows that in these real-life tests, the measured configuration rate exceeds the theoretical minimum configuration rate by a factor 1.75 to 3.35.

This deviation is explained by individual links to new nodes suffering packet loss, thereby failing to complete the configuration during a single scan cycle. Moreover, the measurements were performed in an environment where –depending on the specific node– multiple networks were picked up during the scanning phase, leading to pre-authorization delays.

number of nodes in network	configuration completion time (seconds)	configuration rate (seconds per node)
coordinator + 2	7.35	3.67
coordinator + 4	24.27	6.06
coordinator + 11	39.24	3.56
coordinator + 15	69.96	4.66
coordinator + 24	76.15	3.17

Table 4.4: Time needed to complete configuration of a network of various sizes.

4.6 Management and deployment extensions

4.6.1 Advance planning

While the developed deployment mechanism provides an answer to easily and securely configure new mesh nodes, no answer was currently given to the question *where* to deploy new mesh nodes. While an RSSI based deployment mechanism such as the one described in Chapter 3 can be used, alternative solutions exist when there is sufficient time to plan a mesh roll-out. Planned roll-outs are more likely to immediately provide the required connectivity with the minimum of nodes, and will avoid spots without network coverage. This might be the preferred solution when, for example, an office building is to be covered with mesh nodes: if 'good' locations for deploying backbone nodes are known in advance, roll out is as simple as walking from office to office, each time unpacking and placing a new node on the planned location.

While planning of the roll-out may involve an intensive site survey or complex 3D modeling, other solutions are available that allow a quick sketch of the deployment environment to be made within minutes, and device locations to be generated shortly afterwards using simplified propagation models. As such, basic planning does not necessarily conflict with a fast network roll-out.

4.6.2 Interfacing with administration tools

In the above sections, it was demonstrated how a secure IP mesh backbone network can be rolled out in seconds after unpacking newly bought devices. While the administrative GUI was deliberately designed to be as simple as possible for non-technically skilled persons, it is easy to extend the GUI in order to provide more details and configuration options to advanced users.

During the configuration procedure, the coordinator node maintains a list of the mesh backbone nodes in the network and their capabilities, which were exchanged during the profile configuration phase. Provided a secondary interface is available on a specific mesh node, a user may enable and configure an access point from

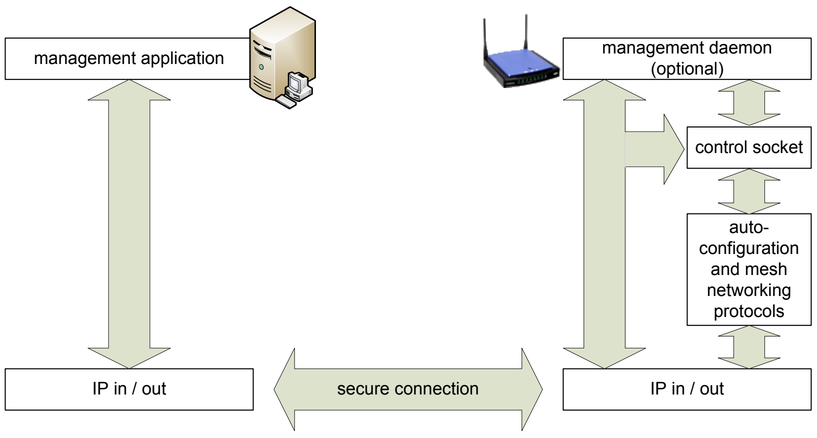
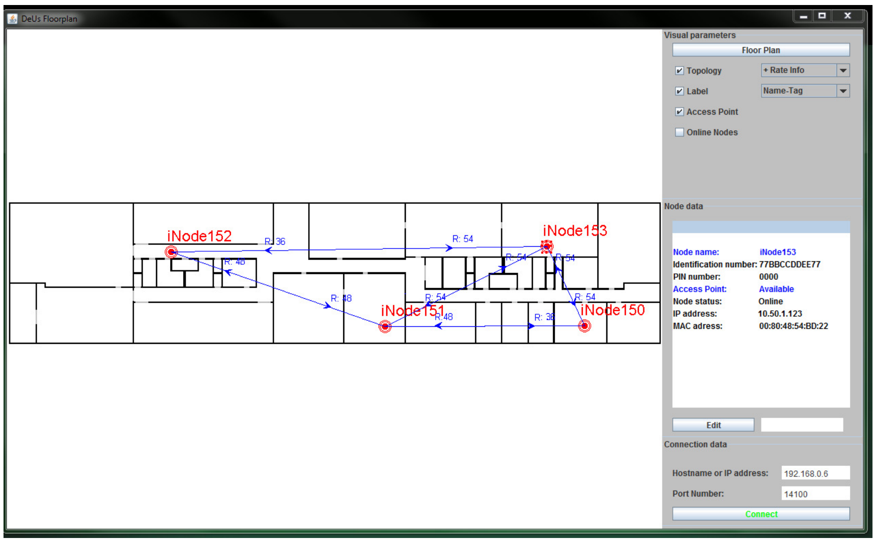

management application
management daemon (optional)
control socket
auto-configuration and mesh networking protocols
IP in / out
secure connection
IP in / out

Figure 4.15: Remote or local administration applications may access management information through a TCP socket.

the administrative GUI. Management of the configured network is easy: since IP connections are available between the nodes, any management application that is able to run on top of IP may be used to manage the network. Furthermore, as the underlying links are secured, there is no strict need for implementing security into the management application.

The mesh core and auto-configuration protocols have built-in functions that can be called through a TCP socket, simplifying interaction with the wireless mesh device. As such, management applications can be easily developed without the need for any knowledge on the internal workings of the mesh algorithms. Figure 4.15 shows a schematic overview of how a management server can contact the mesh nodes in the network. The socket allows both access to various types of information such as various packet counters and administrative information from new nodes in the network, as well as manipulation of configuration settings such as enabling access points, changing network names, or communication channels. While the socket can be contacted directly, an optional management daemon might be installed on the mesh devices. The advantage of the latter is that the management daemon is also able to collect information from other sources than the auto-configuration and mesh networking protocols, for example through interfacing with the operating system.

Figure 4.16 shows an example of an extended administration GUI which works by directly contacting the control socket of the coordinator node. When new nodes appear, the user can drag them to a specific location on an imported map of the area to be serviced. The GUI then visualizes the available links in the network.



Visual parameters

- [✓] Topology: + Rate Info
- [✓] Label: Name-Tag
- [✓] Access Point
- [ ] Online Nodes

Node data

Node name:	iNode153
Identification number:	77BBCCDDEE77
PIN number:	0000
Access Point:	Available
Node status:	Online
IP address:	10.50.1.123
MAC adress:	00:80:48:54:BD:22

Connection data

Hostname or IP address:	192.168.0.6
Port Number:	14100

Figure 4.16: Extended deployment GUI with graphical representation of node locations and additional information on the links in the network.

4.6.3 Future extensions

Extensions to the described planning and deployment mechanisms are easily imaginable. Given the fact that an IP backbone is available, that interaction with the developed auto-deployment algorithms is already supported, and a quick planning tool is available, it would be relatively easy to develop an advanced, integrated planning and deployment application as to be able to fully support following wireless mesh network deployment scenario, illustrated in Figure 4.17.

1. The integrated tool is installed on a portable device, such as a laptop or tablet computer.
2. A map of the environment to be serviced is imported or sketched through the planning part of the tool (cf. Fig. 4.17(1)).
3. The required connectivity at different zones of the map is configured, allowing the planning tool to estimate the optimal locations of the access points and/or mesh backbone node locations.
4. The coordinator node is unpacked and powered, and put on one of the predetermined locations. An access point interface for administration purposes is automatically enabled.
5. The administrator connects his or her tablet to the access point; the integrated tool detects the availability of the coordinator.

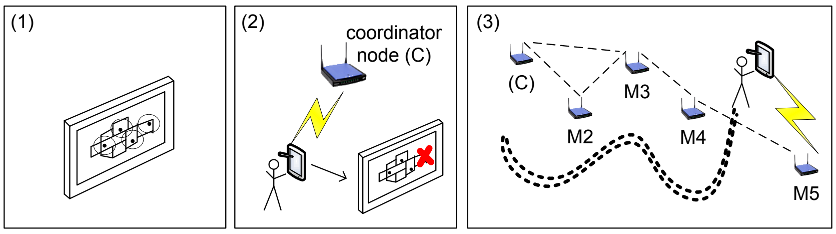


Figure 4.17: Network installation using integrated planning/deployment tool. (1) Handheld device with a map of the environment showing estimated optimal deployment locations. (2) After the coordinator node is placed, the administrator indicates the exact position of the deployed nodes on the map. (3) The process is repeated for the other nodes, wile roaming through the freshly established mesh network. Simultaneously, location estimations for the remaining nodes to be installed are potentially updated by the planning algorithm.

6. Using the tablet GUI, the administrator inputs the correct location of the coordinator node on the map (cf. Fig. 4.17(2)). This location could differ from the estimated optimal location, since e.g. no power socket is available at the previously estimated spot or because of aesthetic reasons.

7. A new node is unpacked and powered, is detected by the tool and configured using the protocols from this chapter.

8. Using the tablet GUI, the administrator inputs the new node's installation location.

9. Immediately afterwards, the nodes start to gather performance data on the connection between the new node and coordinator node. This information is then processed and fed back to the planning module of the integrated tool, which uses the exact location information and measurement data to re-evaluate the remaining node locations. These locations might be different if, for example, the collected measurement data shows that a wall in a building does not attenuate the wireless signal as much as was initially estimated.

10. As the administrator continues the deployment of the nodes, he or she can roam with the tablet PC from the coordinator access point to any other access point that can be enabled on any of the newly deployed nodes (cf. Fig. 4.17(3)).

11. The process is repeated for as many nodes that need to be installed.

12. A few minutes later, a new mesh network is installed on the location. Furthermore, a map is available which shows the exact locations of all nodes.

> From then on, the GUI could show status information on the nodes and network, and by clicking on a specific node, more details about the device could be shown.

The presented solution and implementation can easily be modified in several ways, in order to suit the varying needs of an auto-configuration mechanism for different use cases. Depending on the case, a user or group of users might want to trade-off security for additional installation comfort, or vice versa. As an example, it was previously mentioned that the current implementation saves the configuration parameters locally after a node is fully configured, enabling to recover from e.g. a power failure without re-requiring an authorization code to be entered. If the storage of information is considered to be insecure, because there is a risk of nodes being physically hacked, local storage can be disabled.

4.7 Conclusion

In this chapter, an integrated solution enabling automatic deployment, expansion and management of secure wireless mesh networks was presented. No strict security relationship with the pre-existing network is assumed prior to node deployment. Nodes can be added *anywhere* in the network, with minimal administrator intervention.

The presented mechanism is not limited to a theoretical study: all subsystems such as scanning, neighbor discovery, node initialization, link security and routing are implemented and integrated, allowing an operational multi-hop mesh network providing secure data transport to be set up in minutes. Additionally, the solution allows a network to be built out of heterogeneous commodity hardware provided by different vendors. Even though the mechanism was primarily developed for simplifying wireless network set-ups, the mechanisms works equally well on top of wired interfaces, or even on devices having both wired and wireless interfaces, seamlessly and securely interconnecting networks. For the experienced network administrator, using the presented solution means avoiding tedious installation procedures. At the same time, the procedure opens a new world to the occasional end-user, enabling installation and expansion of a secure multi-hop mesh network through a single uniform interface, requiring a minimum of user interaction.

References

[1] Ralph Droms. *Dynamic Host Configuration Protocol*. RFC 2131, Internet Engineering Task Force, March 1997.

[2] Carlos J. Bernardos, Mara Caldern, and Hassnaa Moustafa. *Survey of IP address autoconfiguration mechanisms for MANETs*. draft-bernardos-manet-autoconf-survey-04 (work-in-progress), November 2008.

[3] Novi I. Cempaka Wangi, Rangarao Venkatesha Prasad, Martin Jacobsson, and Ignas Niemegeers. *Address autoconfiguration in wireless ad hoc networks: protocols and techniques*. Wireless Communications, IEEE, 15(1):70–80, February 2008.

[4] Stephen E. Deering and Robert M. Hinden. *Internet Protocol, Version 6 (IPv6)*. RFC 2460, Internet Engineering Task Force, December 1998.

[5] Ralph Droms, Jim Bound, Bernie Volz, Ted Lemon, Charles Perkins, and M Carney. *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. RFC 3315, Internet Engineering Task Force, July 2003.

[6] Susan Thomson and Thomas Narten. *IPv6 Stateless Address Autoconfiguration*. RFC 2462, Internet Engineering Task Force, December 2008.

[7] Ian Chakeres and Charles Perkins. *Dynamic MANET On-demand (DYMO) Routing*. Ietf draft, Internet Engineering Task Force, March 8 2009.

[8] Thomas Clausen, Christopher Dearlove, Philippe Jacquet, The OLSRv2 Design Team, and The MANET Working Group. *Optimized Link State Routing Protocol (OLSR) version 2*. Ietf draft, Internet Engineering Task Force, April 2010.

[9] IEEE Standard. *IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*.

[10] UpnP Forum. *Home Page*. http://www.upnp.org/.

[11] Steven Furnell and Bogdan Ghita. *Usability pitfalls in Wireless LAN security*. Network Security, 2006(3):4 – 8, 2006.

[12] Ana Cavalli and Jean-Marie Orset. *Secure hosts auto-configuration in mobile ad hoc networks*. Ad Hoc Networks, 3(5):656 – 667, 2005.

[13] Dirk Balfanz, Diana K. Smetters, Paul Stewart, and Hao Chi Wong. *Talking To Strangers: Authentication in Ad-Hoc Wireless Networks*. In Proceedings of Network and Distributed System Security Symposium 2002 (NDSS'02), San Diego, CA, February 2002.

[14] Cisco Systems. *Cisco Mesh Networking Solution Deployment Guide*. http://www.cisco.com/en/US/docs/wireless/access_point/mesh/4.0/deployment/guide/config.html.

[15] Yih-Chun Hu and Adrian Perrig. *A Survey of Secure Wireless Ad Hoc Routing*. IEEE Security and Privacy, 2(3):28–39, 2004.

[16] Nidal Aboudagga, Mohamed Tamer Refaei, Mohamed Eltoweissy, Luiz A. DaSilva, and Jean-Jacques Quisquater. *Authentication protocols for ad hoc networks: taxonomy and research issues*. In Q2SWinet '05: Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks, pages 96–104, New York, NY, USA, 2005. ACM.

[17] Fred Cohen. *A cryptographic checksum for integrity protection*. Computers & Security, 6(6):505 – 510, 1987.

[18] Jeroen Hoebeke, Ingrid Moerman, and Piet Demeester. *Virtual private ad hoc networks*. In Terena Networking Conference, Bruges, Belgium, May 2008.

[19] Jeroen Hoebeke. *Adaptive Ad Hoc Routing and Its Application to Virtual Private Ad Hoc Networks*. PhD in Computer Engineering Science, Ghent University, IBCN - INTEC, B-9050 Ghent, Belgium, 2007.

[20] FIPS. *Advanced Encryption Standard (AES)*. Technical Report 197, pub-NIST, November 2001.

[21] Donald Eastlake and Paul Jones. *US Secure Hash Algorithm 1 (SHA1)*. RFC Informational 3174, IETF, September 2001.

[22] Donald Welch and Scott Lathrop. *A survey of 802.11a wireless security threats and security mechanisms*. Technical report, Department of Electrical Engineering and Computer Science United States Military Academy, 2003.

[23] Arto Solomaa. *Public-Key Cryptography*. Springer-Verlag, Berlin Heidelberg New York, 1996.

[24] *The Click Modular Router Project*.

[25] *PC Engines - Alix system board*.

[26] Jangeun Jun, Pushkin Peddabachagari, and Mihail Sichitiu. *Theoretical maximum throughput of IEEE 802.11 and its applications*. Second IEEE International Symposium on Network Computing and Applications, pages 249–256, 2003.

5 Wireless Network Experimentation Methodology

5.1 Introduction

Irrespective of the specific wireless ad-hoc subtopic that is studied, researchers are presented with a large choice of performance evaluation methods while analyzing newly developed hardware components, wireless architectures, algorithms or protocols at different layers of the OSI stack. These methods range from mathematical evaluation of a single aspect of a protocol, over various types of simulations and emulations, to a full implementation with field trials performed by the target audience.

Choosing the appropriate evaluation method for a specific research goal is a non-trivial task. Traditionally, in situations where analytical models are inapplicable or unavailable due to the complexity of the wireless problem under investigation, most wireless ad-hoc paper results are based on simulations. This is not surprising, as it is often difficult or too costly to conduct real-life wireless networking tests with a large number of (mobile) devices under varying topologies or varying traffic conditions. Furthermore, researchers often lack the tools, hardware or manpower to implement their solutions on a real platform and perform measurements. However, the use of simulators for evaluating wireless ad-hoc protocols is not undisputed. In Chapter 2 of this work, it was argued that the origin of many of the current wireless ad-hoc issues lies in the misinterpretation of performance results of networking protocols obtained through theoretical research based on wrong assumptions, or simulation results based on oversimplified models. In literature, the authors of [1] indicated significant differences while simulating a single protocol using three different popular network simulators: Opnet, ns-2 and GloMoSim. Through an extensive literature survey, Andel and Yasinac question the credibility of MANET simulations and show how misleading results are easily produced if research is exclusively based on simulations [2].

Therefore, during the realization of this work, simulation results and theoretical expectations were complemented with implementations and real-life experiments whenever possible. Moreover, within the IBCN research group, being frequently involved in wireless projects that lead to proof-of-concept set-ups [3, 4], it was learned to appreciate the added value of demonstrator implementations on many occasions.

As will be detailed in this chapter, through failures and successes, it was found that there are many advantages to implementing wireless solutions, but also many pitfalls; while results obtained through theory or simulation might not always be fully representative for the actual behavior of a solution once it is deployed in a real-life environment [2], experimentally-driven research is not always the best performance evaluation option either. As the international wireless research community gets increasingly interested in performance evaluation through measurements on real-life testbeds, and experimentation is becoming an important way of analyzing the performance of wireless networks, it gets increasingly important to avoid misinterpreted experimental test results. Unfortunately, while there are publications related to potential issues with wireless network simulations [1, 5] and high-level methodologies have been proposed to design wireless network protocol architectures [6, 7], no generic *experimentation* methodologies for wireless network protocol developers exist. Consequently, mistakes are easily made while setting up wireless networking experiments, and results obtained from wireless network experiments are easily misinterpreted. In case of unexpected behavior of the test set-up, locating the origin of a specific problem may be a complex task, causing researchers to resort to 'best guesses' or vague error descriptions such as 'the behavior is caused by driver errors'.

Therefore, the goal of this chapter is twofold: first, to assist researchers in choosing a performance evaluation when developing wireless networking protocols, a detailed classification and overview of performance analysis techniques for wireless protocols, their fields of application, and common mistakes is provided. To my knowledge, no detailed overview is currently available in literature. Second, a methodology for building wireless test set-ups and conducting field trials is developed. The methodology aims at providing those who are new to wireless experimentation with a guideline that will help them to obtain more reliable results with less effort, by avoiding errors that were made in the past or observed in literature. Furthermore, to those who are experienced in performing wireless experiments, the presented methodology helps to optimize future experimental work and to gain additional insight in using implementation as an evaluation method for designing wireless networking protocols.

5.2 Performance evaluation methods for wireless ad-hoc protocols

5.2.1 Overview

As stated before, several research papers in literature indicate that performance analysis of wireless ad-hoc protocols based exclusively on simulation might lead to misleading or faulty conclusions. In [8], the authors propose to combine simulation, emulation and real world experiments in order to analyze the performance of the AODV, DSR and OLSR protocols. By comparing the results from the three evaluation methods, implementation problems are more easy to detect. The authors compare the characteristics of the real-world, emulation and simulation methods and determine whether the routing logic, hardware, stack, mobility and radio of a particular method is real or not.

In Table 5.1, an analogous yet highly extended overview of performance analysis methods is introduced. Seven methods are identified:

(i) mathematical and statistical models

(ii) full simulation

(iii) real device but with emulated wireless radio interface

(iv) real device, real wireless network interface card (NIC), but an emulated wireless medium

(v) real-world experiment, but in an emulated environment

(vi) real-world experiment by technically skilled persons, in a realistic environment

(vii) real-world experiment, performed by the target audience

For each method, the table indicates whether certain aspects related to the hardware or wireless environment that could impact the performance of the wireless ad-hoc protocols are mathematically modeled ('m'), simulated ('s') or real ('r'). An influence that is not relevant or not possible to verify is labeled 'n'; a method that can be used to analyze a certain external influence is labeled 'y', meaning yes, the behavior can be verified. Whenever a minus suffix ('$-$') is added to any of the above characters, it should be interpreted as 'not entirely'. For example:

	device influences									external influences							other			
	RF propagation	wireless NIC	wireless driver	protocols under test	host operating system	interface with other programs	self-interference	form factor	available memory and CPU	external interference	enviroment	topology	traffic pattern	usability (from a user POV)	quality of experience	interoperability	reproducability	financial cost	required effort (time)	time to market
mathematical and statistical model	m	n	n	m	n	n	m	n	n	m	m	m	m	n	n	n	H	L	L-H	H
simulation	s	s	n	s	n	n	s	n	n	s	s	s	s	n	n	n	H	L	L	M-H
real device, emulated radio interface and medium	s	s	n	r	r	r	s	r-	r	s	s	s	r-	y-	y-	y-	M	L-M	M-H	M-L
real device, real wireless NIC, emulated medium	s	r	r	r	r	r	s	r-	r	s	s	s	r-	y-	y-	y-	M	M-H	M-H	M-L
all real, except emulated environment (testbed)	r-	r	r	r	r	r	r-	r-	r	r-	r-	s	r-	y-	y-	y	M	M-H	H	L
all real, testbed in realistic environment	r	r	r	r	r	r	r	r	r	r	r	r	r-	y-	y-	y	M	M-H	H	L
all real/realistic, used by intended audience	r	r	r	r	r	r	r	r	r	r	r	r	r	y	y	y	L	M-H	H	L

Table 5.1: Performance analysis methods and their characteristics. m: mathematically modeled. s: simulated. r: real. n: not relevant/cannot be verified. y: yes, verifiable behavior.

the '$y-$' indicator in the quality of experience (QoE) column of the third performance analysis method (real device, emulated radio interface, emulated medium) indicates that while measuring the quality of experience (QoE) of end-user applications is possible to some extent, the results will be biased since the radio interface of the devices is not real but emulated, and the user cannot experience the application in the environment where it would normally be used.

In a third subdivision, the table provides a relative ranking of the aspects of reproducibility, financial cost, required effort and time to market, each aspect categorized as *relatively* low ('L'), medium ('M') or *relatively* high ('H').

5.2.2 Performance analysis without hardware devices

Mathematical and statistical models are one of the pillars of engineering. When used in wireless ad-hoc research, these models usually try to capture the behavior of the wireless medium [9], explore the limits of a certain metric such as throughput given a set of assumptions [10] or model a specific aspect such as the overhead introduced by a protocol. Both model types are closely related: where mathematical models describe phenomena in a closed form expression, statistical models take into account the probabilistic behavior introduced by varying factors such as user mobility, traffic patterns and/or variations in the wireless channel.

These models are typically the cheapest way of studying problems, and, once models are available, model parameters are easily varied, quickly leading to research results. On one hand, the validity of generic models with respect to the assumptions that are made can easily be verified or further refined by other researchers. On the other hand, in order to reduce the number of variables in an expression, a model may be designed in such way that is only valid for a specific topology or specific traffic pattern. The lack of generalization reduces the applicability of these simple models and may make side by side comparison difficult. As such, the main difficulty with mathematical and statistical models is the conflict between model simplicity and correctness [5].

In addition to modeling a system, mathematical formulas are also used in the core of wireless networking algorithms. For example, imagine an algorithm determining the optimal output power settings for a set of wireless ad-hoc nodes at a specific time, given the detailed traffic patterns, location and the number of available interfaces. While such algorithm might be able to produce a theoretical optimal configuration given a specific target function such as maximum average throughput, the assumptions regarding the availability of input parameters may be hard to achieve in a real deployment. Even if these models are valuable to discover general trends, they should not be used to make claims about the general performance of a specific system or algorithm in the real-world. Mathematical and statistical models are therefore especially useful to study trends in wireless subsystems and to determine upper and lower bounds of specific quality metrics in best case and worst case scenarios.

The complexity of capturing the performance of an entire system in a mathematical model has motivated researchers to build network **simulators**. These simulators essentially use mathematical and statistical models to simulate the behavior of a specific protocol or algorithm under varying conditions. As a consequence, the correctness of the simulator output depends on the correctness of the models in use. Because the user of a simulator is often not the programmer of the simulator, researchers should not blindly trust any result obtained through simulation. As an example, consider a wireless ad-hoc network with multi-interface IEEE 802.11g compatible nodes. Suppose a channel selection protocol is simulated and the sum of the throughput of several traffic flows is used as a benchmark. The throughput in this situation depends on the interference model embedded in the simulator. If a simulator does not take into account any interference between neighboring channels of the radio spectrum, the simulator results will obviously give an overestimation of achievable throughput. However, as indicated by [2], many authors fail to use the wireless ad-hoc simulator in the intended way or make other mistakes such as basing results on unrealistic traffic patterns, unrealistic topologies, use the wrong radio model for their specific situation or do not specify the simulation package or version number, making independent verification of simulation claims impossible.

Simulation is a valuable tool during a development process of wireless networks, allowing evaluation and debugging of wireless ad-hoc protocols, as long as the user of a simulation tool is aware of the simulator's capabilities and limitations, and any results that are discovered through the use of the model are not used to make hard statements about the performance of the protocols under real-life conditions. Most simulators require code to be written specifically for the simulator, and make abstraction of any host platform or specific driver. Some simulator environments such as nsclick [11], previously used for the evaluation of the FRESME channel selection protocol in Chapter 3 allow to run simulations using program code that can also be used on a real device. The absence of perfect driver simulation, host platform or interference simulation is not necessarily a bad thing as it allows isolating coding errors in the protocol under test from errors introduced by the wireless driver or interactions with the host platform.

Interesting additional thoughts on simulation and an overview of (wireless) network simulation tools are found in [12].

5.2.3 Performance analysis using hardware devices

One of the major challenges of performing wireless experiments with real hardware without any emulation is to control the topology of a test set-up. In contrast with mathematical models or simulators that use simplified propagation models with fixed wireless transmission and interference ranges, real wireless network interface cards are subject to *all* laws of physics, are influenced by the specific time varying conditions of their environment, and, as shown in Chapter 2, may suffer from flaws in the hardware or driver design. Moving objects, external sources of RF signals and small variations of antenna or node positions are just a few of the elements that may have a profound impact on the packet delivery ratio in the network.

Methods three to seven from Table 5.1 require the use of a hardware device. The wireless node behavior is no longer fully modeled or simulated, but (partially) replaced by a hardware device. This device can be the actual target device or a prototype thereof, or a generic device such as a desktop or laptop computer used as a replacement platform during development. In what follows, performance analysis alternatives that do use a real hardware device, but emulate part of its functionality or environment are presented and discussed.

Figure 5.1 shows a logical chain of the principal components of a single wireless communication link between a sender and receiver. The chain should not be confused with the OSI-layer model. The layered model is applicable to most wireless communication systems. While source and destination are represented by a person in the figure, person-to-machine and machine-to-machine communication are possible as well. For example, in wireless mesh networks, a user might access a server through a multi-hop wireless connection. In this case, the intermediate mesh routers forwarding the request do not use their end-user application. It is assumed that the MAC and PHY layer of the OSI stack are located in the *wireless NIC driver* and *wireless radio chip* component. All upper layers, except for the application are abstracted in *upper layer processing* functional block.

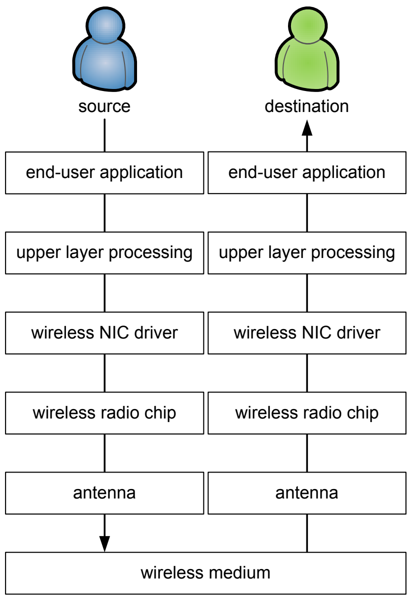


Figure 5.1: Layered view of logical chain of components between wireless source and wireless destination.

5.2.3.1 Real device, wireless NIC emulation

Figure 5.2 represents a hybrid evaluation method, in which the upper layer protocols of the wireless network are fully implemented on a hardware device, while the **wireless NIC and wireless medium** are **fully emulated**. The shaded blocks represent functional blocks that are replaced in order to support partial emulation. Every node in the network is connected over an Ethernet interface to a central emulation server. After configuring a virtual topology and mobility pattern, the emulation server starts receiving packets which, according to the implementation, may or may not be encapsulated in the wireless frame format, and processes them using an RF propagation model such that packets are discarded or delayed before being forwarded to the nodes within the virtual transmission range. The WINES (Wireless Network Emulation System) set-up, built several years ago at our research group, works according to this principle. The WINES system uses GloMoSim [13] as the underlying RF propagation emulation engine, and was built to support the emulation of IEEE 802.11a/b/g cross-layer protocols, by providing an interface to the upper layer protocols to access all network parameters down to the physical layer. Commercial alternatives such as Opnet's System-in-the-loop [14] exist, having similar functionality.

The method can thus be used as a way to functionally test cross-layer proto-

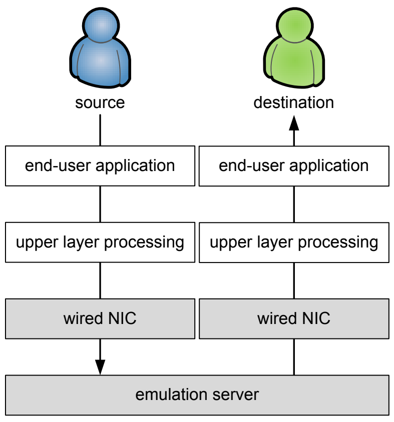


Figure 5.2: Layered view on the wireless communication chain when using wireless NIC emulation.

cols such as fast roaming protocols under controllable and reproducible physical propagation characteristics, while still allowing realistic –user mobility excepted– real time interaction by end-users using the target end-user device. In contrast with the previous evaluation methods, interaction with any end-user application is supported natively, allowing more realistic traffic patterns to be evaluated. Furthermore, while it is not always possible to fully control the radio chip of a hardware platform e.g. up to packet level, an emulated radio interface and medium are not affected by this limitation. Because the device under test is already the actual target platform, the developer is confronted with limitations of the device such as limited memory or limited processing power, and most programmed code can be re-used should the designed solution be further developed towards a final product.

5.2.3.2 Real device, emulated medium

The previous method enables development using an end-user device while still allowing the programmer to focus on the functionality of a wireless solution without having to worry about unreliable and varying wireless devices and medium. The downside of the abstraction is that the robustness of developed algorithms cannot easily be verified against typical wireless problems generated by unstable links, unexpected interference or issues introduced by (misconfiguration of) the wireless driver such as scanning delays or (temporary) node malfunction, which may lead to performance issues when deploying the solution in a real-life test environment. The reason is that the RF models on the emulation server are the same as those used in simulators. While it can be argued that it is not the responsibility of a researcher

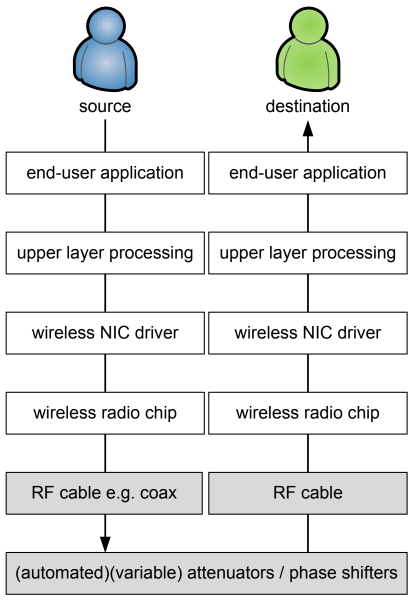


Figure 5.3: Layered view on the wireless communication chain when using wireless NIC emulation.

developing algorithms at a higher layer to compensate errors generated by wireless network drivers, the reality is that if they are not taken into account, several algorithms will fail once fully deployed on a wireless testbed or in an operational environment. As long as no perfect simulation/emulation model is available, the only way to predict deployment issues –and detect differences between the emulation model and the actual implementation platform– is by increasing the level of detail of the implementation.

In order to take into account the influence of the wireless driver and in order to develop an interface with a real wireless NIC, while still being able to control the topology and external interference sources, the performance evaluation method of Figure 5.3 may be chosen. In this method, a real device and real wireless NIC are used, but instead of connecting an antenna to the radio, a cable or set of (coaxial) cables, attenuators and optionally phase shifters and noise injectors are installed between the NICs, hereby substituting the wireless medium with wired transmission lines.

This technique was previously used in Section 2.4.1 and is, for example, also used by Kaba and Rachle in [15], where they present strategies to support multi-hop wireless network development. Their goal is to create a ‘testbed on a desktop’, allowing to test a wireless solution under network topologies that are otherwise hard to reproduce. Topologies are recreated using attenuators and splitter/combiners, while mobility is simulated through the use of variable attenuators. One of the main advantages of this technique is that no modifications are needed to the hardware or software under test. Several topologies are discussed. A drawback of this solution reported by the authors is the inability to create arbitrary topologies in which the attenuation between any two wireless ad-hoc nodes can be configured.

Since fixed attenuators and splitter/combiners are relatively cheap, this method is an affordable solution to emulate wireless multi-hop networks without the need for a large test location or a lot of equipment. In addition to this low cost solution, commercial alternatives exist offering fully flexible and programmable interconnections through a variable attenuator unit [16]. To eliminate all external interference, RF shielded cases exist. Such solution was used in previous chapters, whenever signals from external nodes were expected to interfere with our measurements. However, when using shielded cases, each node requires its own case, rapidly increasing the cost of commercial solutions with the scale of the experiment. Still other solutions exist which translate RF signals to the digital domain, combining signals from different sources via digital signal processing, this way also supporting MIMO experiments.

Whether a commercial product is chosen or not, the scalability of analog solutions is limited: at every splitter/combiner, the RF signals suffer insertion loss, making it hard to recreate large topologies. Therefore, this method is especially suited for testing wireless ad-hoc protocols with a limited number of nodes in an emulated multi-hop environment. The result is guaranteed not to be affected by external interference or antenna positions, and allows easy emulation of mobility patterns using only a limited testing space.

5.2.3.3 Real device, real medium, emulated environment: wireless testbed in a lab

The logical next step is to run performance tests on a testbed in a laboratory or office environment accessible to the network developers. The output power of the radio chips is no longer emulated or guided across a wired medium, but broadcast through an antenna. While small-scale testbeds might be portable, most ad-hoc nodes in large scale testbed are typically installed at fixed locations for practical reasons [17]. These fixed locations result in a ‘natural’ default topology of the testbed. If wireless network researchers do not want to be stuck with this default topology, techniques are needed to modify the (perceived) topology in the testbed. Especially for wireless technologies operating at a relatively high transmission power such as IEEE 802.11a/b/g, the default topology is often too dense.

Recently, our w-iLab.t testbed was designed and installed at the buildings of the IBCN research group and IBBT research institute in Ghent, Belgium. The w-iLab.t testbed consists of nearly 200 sensor nodes and an equal amount of Wi-Fi

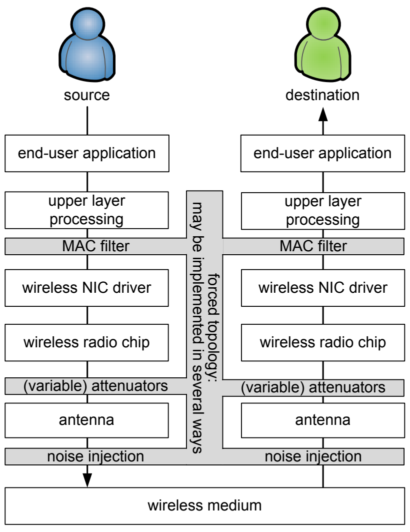


Figure 5.4: Layered view on the wireless communication chain when testing solutions on a wireless testbed.

nodes with two interfaces, which are mounted to the ceilings in the offices and hallways, and is further detailed in Appendix A. Even with the nodes mounted across three $18\,m$ by $90\,m$ floors subdivided in tens of separate offices, the Wi-Fi topology at each individual floor is close to a full mesh network because the transmission power of the nodes cannot be set to a lower value than $0\,dBm$. For most experiments, this is an undesired level of connectivity.

Several solutions to this problem exist. Figure 5.4 indicates the different points of stack modification to implement a multi-hop topology in a testbed. If the 'natural topology' is not a full mesh and many nodes are available, a first solution is to limit the experiment to run only on a subset of all available nodes, in such way that the expected topology is created. However, one should be aware that the topology of a specific subset might change over time – which is not necessarily an unwanted side effect.

A second and equally simple solution, implemented in different ways by many researchers but most likely first reported on by Maltz et al. in [18] is to filter out packets based on the MAC addresses of the sending nodes, either by discarding all packets except from those nodes listed on a whitelist, or by allowing all packets except from those nodes listed on a blacklist. However, unless a good management system is available for 'enabling' or 'disabling' links in the network, configuring new topologies is a tiresome task. Furthermore, while packets might not arrive at the upper layers of the wireless network stack, the packets are still detected and received by the radio chip, thus interfering with performance measurements.

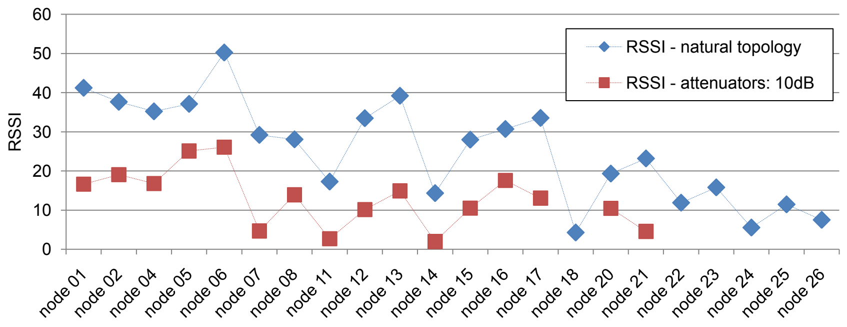


Figure 5.5: RSSI values measured at different nodes at the third floor of the w-iLab.t testbed, in the natural topology, and after installing 10 dB attenuators at every interface.

A third solution is to configure the nodes to fixed and high transmission rates. Since data cannot be sent at high data rates over low quality links, the number of links in the testbed is effectively reduced. However, this solution does not lower global interference levels and cannot be applied when certain protocols such as rate selection schemes are developed.

The fourth solution is similar: for those networks that are not fully connected when the lowest transmission power is selected, the topology may be changed by varying the transmission power. Lowering the transmission power does reduce the general interference level, but cannot be used to develop and test performance of e.g. a dynamic power adaptation protocol. When multi-interface nodes are studied, lowering the transmission power may have a positive effect on the ability to operate several simultaneous radios. However, because of the interference between different wireless interfaces of a single wireless node, previously described in Section 2.4.3, results obtained from a multi-interface testbed operated at low power are not necessarily (fully) representative when the power and the distances between testbed nodes are scaled.

Fifth, in case the transmission power cannot be set arbitrarily low, as was the case with the Wi-Fi side of our testbed, the topology density can be reduced by installing RF attenuators between the radio interface and antennas of the nodes. This approach does not require any modifications to the software running on the nodes, and reduces the general interference level in the testbed. This technique is currently used in the w-iLab.t testbed. Figure 5.5 illustrates the effect of adding attenuators of $10\,dB$ to every node of the testbed. The graph shows RSSI measurements of test packets transmitted at $0\,dBm$ transmission output power by a single sending node. The measurements are performed at several nodes positioned around the transmitter, once in the natural topology, and once after installing the attenuators. It can be seen that for most nodes, the expected RSSI reduction of $20\,dB$ is achieved. Furthermore, some distant nodes do no longer receive packets, effectively reducing the density of the testbed. Whenever tests in a dense network are required, the effect of the attenuators can be reduced or canceled by increasing the output transmission power. Note that since there is no correlation between the distance from the sending node and minor variations to the $20\,dB$ RSSI reduction, these variations are believed to be caused by the manual intervention needed to install the attenuators, after which slight variations of the antenna positions and orientation are an unavoidable side-effect, and antenna connectors which might have gotten slightly decoupled in time are now firmly reattached.

Still other techniques are found in literature. In [19], the authors demonstrate the use of additive white Gaussian noise to raise the noise level in their testbed. Doing so, the signal to noise level is lowered, thus reducing the transmission range. With their technique, they are able to create a four hop string topology on a testbed that covers an area of $8\,m$ by $8\,m$.

Whichever topology control technique is used inside the testbed, in order to get complex wireless ad-hoc protocols run successfully and stable on a wireless testbed, in our experience, the developer or developing team will have to spend several additional days, weeks or even months compared to any of the previous performance analysis techniques. This is due to the fact that the algorithms are now fully exposed to external interference and inherent variations of the wireless radio environment, causing intermittent link connectivity and unexpected link failures generally leading to effects that are hard to detect through simulation or emulation. Furthermore, scanning procedures and other algorithms based on information dissemination and monitoring of the network environment (e.g. in order to select 'the optimal path' through the network) are now also influenced by interfering packets sent by third party networks.

While wireless development on a testbed generally requires more effort, successful testbed deployments assure more stable and reliable wireless ad-hoc protocols. Whenever a testbed produces unexpected or unstable results, it is important to try and find the source of a problem. The methodology described in Section 5.3 helps to avoid basic errors that may appear when a solution is deployed on a testbed. Finally, as previously indicated in Chapter 2, it is important to realize that any result obtained from a testbed is strictly only valid for that particular testbed environment. Especially if the solution is only tested in a single topology, the danger exists that a solution is tuned to work perfectly in the specific testbed situation, but fails to work when deployed in another environment or topology.

5.2.3.4 Fully realistic field tests

For the final two development and performance analysis methods of Table 5.1, the wireless nodes are taken out of a laboratory environment and put under test in the target end-user environment. The distinction between these two last methods is based on the identity of the users testing the solution: a solution may be tested by the developers of the solution or by technically skilled persons, or by the target end-users. Especially when it comes to testing the usability of a solution, tests performed by real end-users are required. Furthermore, there is always an uncertainty factor related to how a solution is used by the target audience: while system designers might develop their system to be used in one way, the final product may be used differently. A popular example is the unexpected success of the Short Messaging Service (SMS) in GSM.

It is not unusual for different wireless drivers to implement certain algorithms slightly different. For example, in case of Wi-Fi compatible nodes, variations on the access point registration procedure were observed: probe request and responses may or may not be used, or the timing in between request and response messages might vary. In a closed laboratory environment, all devices and their different ways of reacting to certain packet types are known. However, when the developed wireless system is used in cooperation with a broad range of end-user devices, unexpected errors may surface, triggered by the aforementioned differences in implementation or by 'illegal' actions of the end-user.

Field tests by non technically skilled end-users provide valuable additional technical and non technical feedback to wireless protocol developers. For example, additional software bugs may surface when using the prototype in different environments or with different types of end-user hardware, users might find a solution either simple or complex to work with and suggest optimizations or additional features, or the test public may indicate that the developed wireless solution to be useful for their everyday activities [3]. Unfortunately, such tests are expensive, and the circumstances (location, mobility pattern, interference sources at the time, specific user behavior and history of the user's actions) in which a certain issue was observed are difficult to log and reproduce.

5.3 A wireless protocol implementation methodology

In the previous paragraphs, an extensive overview of different performance analysis methods for wireless ad-hoc protocol development was given. Recently, there is a slowly growing awareness within the wireless research community that it is dangerous to claim wireless ad-hoc issues to be solved based on simulation results only. This observation is reflected in the topics of international conferences that increasingly welcome experimentally-driven research and the recent interest of the European Commission in experimental facilities [20].

Although, given the context of this dissertation, this evolution is applauded, it is also believed that by analogy with the cautionary warnings concerning wireless simulations, it is necessary to adopt a critical attitude towards the use of experiments as a tool for wireless ad-hoc protocol development and evaluation. Therefore, in this section, a wireless ad-hoc protocol implementation methodology is introduced to help developers obtaining reliable results from experimental setups, either from a testbed or through a field test.

Wireless ad-hoc implementation methodologies are seldom found in literature. One of the few examples is the work performed by the authors of [21]. In this work, a test methodology is presented for increasing the repeatability and reproducibility of ad-hoc protocol experiments through the use of the publicly available APE (Ad-hoc Protocol Evaluation) testbed [22]. The APE testbed is built out of laptop computers and allows creating mobility scenarios that both trigger devices to perform specific events (e.g. start a traffic stream) at a certain time, and instruct volunteers carrying laptops through messages on screen to follow a predetermined path. As such, experiments can be repeated several times under similar conditions, after which log files are compared in order to draw conclusions on the performance of a certain solution.

In [6], the authors present a high level overview of an iterative methodology for the design of wireless protocols based on formal models. The goal of their methodology is not to be a guideline for experimental evaluation, but to help the design of the protocol itself by defining architectural functional blocks which are then mapped to a specific architectural resource which can either be a software component in a network layer, or a hardware component. By following their methodology, a complete description of the system in terms of hardware and software components, as well as a first approximation of the performance and cost of the system is hoped to be achieved. From the example in the paper, it is understood that the methodology particularly aims at designing hardware components.

The authors of [7] discuss their service-oriented design methodology for wireless sensor networks based on agile development principles [23, 24]. Their design and implementation strategy consists of three phases: the overall solution and architecture design, the protocol and application design, and the actual implementation. These three cycles are then iterated throughout the entire project. To motivate the iterative approach, the authors present a list of wireless sensor network design characteristics, including the desire for rapid prototyping, the fact that requirements of the project may change during development and the fact that wireless sensor systems are often developed by a small group of highly qualified developers working in close collaboration. After describing the required functionality in terms of main parameters characterizing the system performance, network parameters, service parameters and hardware and software parameters, the design process is started. Their design process loosely follows a sequential software development process known as the waterfall model [25], and allows a wireless sensor network solution to be built from the initial specifications by making choices such as using a proactive routing approach versus a reactive routing approach, caching messages to increase network lifetime versus using piggybacking, or using a client-server versus a cluster-based architecture. Little information is included on how to ac-

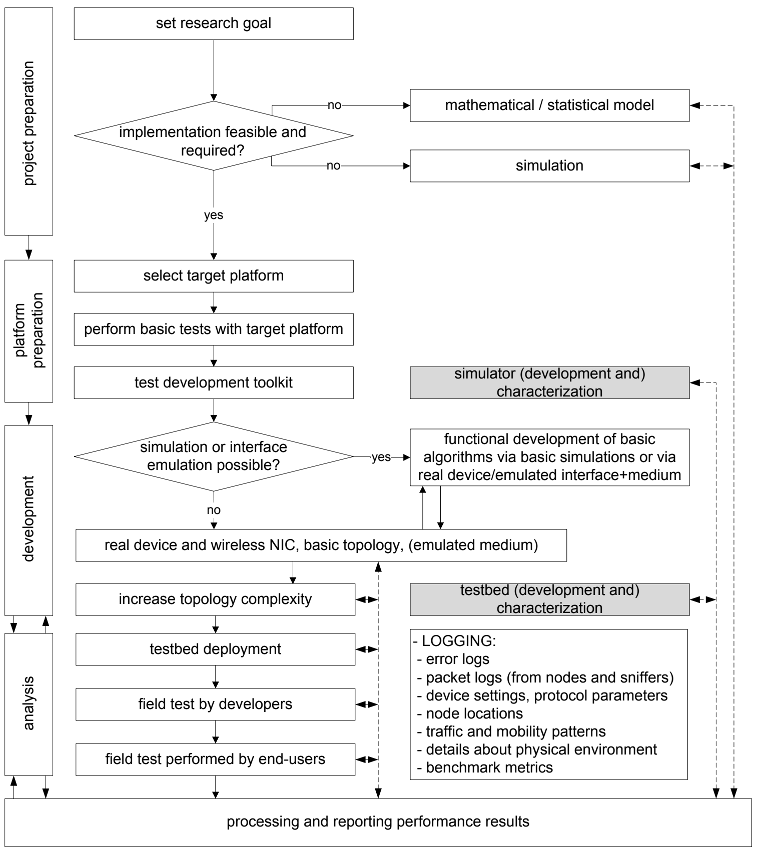

project preparation
set research goal
implementation feasible and required?
no
mathematical / statistical model
no
simulation
yes
platform preparation
select target platform
perform basic tests with target platform
test development toolkit
simulator (development and) characterization
development
simulation or interface emulation possible?
yes
functional development of basic algorithms via basic simulations or via real device/emulated interface+medium
no
real device and wireless NIC, basic topology, (emulated medium)
increase topology complexity
testbed (development and) characterization
analysis
testbed deployment
- LOGGING:
- error logs
- packet logs (from nodes and sniffers)
- device settings, protocol parameters
- node locations
- traffic and mobility patterns
- details about physical environment
- benchmark metrics
field test by developers
field test performed by end-users
processing and reporting performance results

Figure 5.6: Overview of the wireless ad-hoc performance analysis methodology phases.

tually implement the solution once it is fully specified or how to benchmark the solution: the authors suggest to find a compromise between complexity, comprehensiveness and efficiency of the code, for example by using suitable programming abstractions and clear naming.

In contrast with the APE methodology, the methodology presented below has a broader scope, as it is not tied to a specific testbed. Moreover, and in contrast with [6, 7], the methodology in this chapter is not targeted at optimizing the internal construction or programming structures of wireless networking protocols. It complements the related work by providing a methodology to help wireless protocol designers in determining which implementation method is most suited while working towards a specific goal. Furthermore, if the answer to this question is a full implementation and real-life evaluation, the methodology guides wireless protocol developers from concept to field test and to reliable evaluation, combining

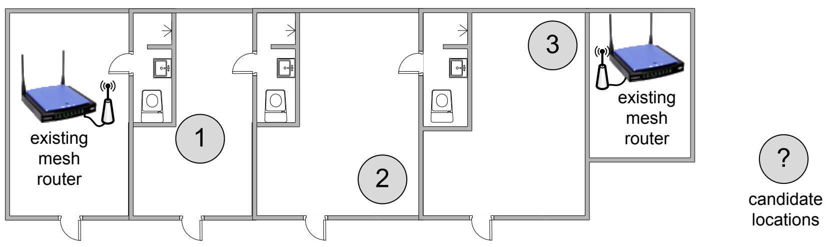


Figure 5.7: Hotel use case: design of an easy deployment extension for wireless mesh networks, determining which location(s) are suited for the deployment of an additional mesh backbone router.

the methods described in Section 5.2. A general overview of the methodology is depicted in Figure 5.6. It consists of following five phases: project preparation, platform preparation, development, analysis and reporting. The methodology focuses on the technical aspect of wireless ad-hoc research and implementation: non-technical aspects such as the search for funding or techno-economic feasibility are not considered.

Example case. The methodology is illustrated through the example implementation of a simple deployment extension for wireless mesh network routers. When deploying or expanding wireless mesh networks, mesh routers are spread over the area where coverage is required. Due to the unpredictable nature of wireless signal propagation [26], positioning the devices is a non trivial task. While a measurement campaign may be used to determine the ideal locations for mesh backbone routers, there are many cases in which it is impractical or too expensive for such campaign to be performed. In order to overcome this issue, one can imagine a small screen or status LEDs on the mesh backbone routers indicating which 'connection quality' can be expected if a node is dropped on its current spot. If the service quality is shown to be acceptable, the node can be dropped. If not, the person installing the network is instructed to move to a different location. An example scenario is depicted in Figure 5.7: a hotel already has installed two Wi-Fi compatible wireless mesh routers with attached wireless access points configured to non-interfering frequencies at a particular floor, but guests still observe a weak access point signal. Therefore, the hotel management decides to install an additional mesh backbone router with attached access point. Three candidate locations have been selected, and a member of the staff without any knowledge on RF signal propagation needs to find out on which location(s) the backbone router can be placed in order for it to have a good connection quality to the existing backbone. For simplicity reasons, in what follows it is assumed that if the backbone router observes a good connection quality to the existing backbone, the attached access point will be able to provide the necessary connectivity to the hotel guests. As such, the problem is reduced to finding a good deployment spot for the backbone router.

5.3.1 Project preparation

While obvious and applicable to any project, sufficient time should be spent preparing a deployment prior to starting implementation. A first step is to determine the research goal. Basic questions to be answered in this phase are: *(i)* what is the problem to be solved, *(ii)* what could be a solution to this problem, *(iii)* which technique(s) can be used to analyze or implement the solution, and *(iv)* how can the effectiveness of the presumed solution be measured. The former two questions assume that the researcher or research team is sufficiently familiar with the research field and effort was put in the search for related work. These questions apply to any type of research project and are therefore not treated in detail. In order to answer to the third and fourth question, there are certain aspects specific to wireless ad-hoc implementation that require particular attention. These aspects are detailed in the next sections.

Example case. In the example use case of the deployment tool, the problem is that it is not always easy to find a suitable location for the deployment of an additional mesh backbone router. A possible solution to this problem is to let the mesh device indicate what service quality can be expected at the node's location.

5.3.1.1 Determining the evaluation technique

The evaluation technique overview of Section 5.2.1 is a starting point to answer the third preparation question. At this point one should verify whether implementation on real hardware is feasible, and if so, if it is required. Many reasons exist why an implementation is not feasible. For example, it might be technically unfeasible for a research group or individual to modify certain device parameters or protocols. Imagine a conceptual modification to an IEEE 802.11g compatible device allowing to adjusts power settings and channel bandwidth for every single packet sent using a Wi-Fi based device. While this might lead to theoretical performance gains, and in the long run to newly developed devices able to support such demand, the average academic researcher proposing the modification probably will not have such device at design time of the algorithm. Moreover, even if devices exist that could theoretically support these technical demands from a hardware point of view, there are still no guarantees that the developer of the conceptual modification himself is able to implement the solution, as he might not have access to the source of the device drivers, or, because a very large number of devices is needed for the considered scenario, making it too expensive or unfeasible for him to acquire the devices for academic research purposes. Obviously, the feasibility of an implementation is not necessary (only) a matter of theoretical possibilities, but also one of time constraints and financial budget.

Among the reasons to invest in a real-life implementation are: the conviction that only a fully implemented solution running on the target devices provides reliable performance results, the development of a complex solution that interacts with other already implemented programs, the desire to collect usability information from a field test, the wish to increase visibility of a developed solution through a demonstrator, or the drive or need to design a solution which may grow sufficiently mature to evolve towards a commercial product.

Which analysis method fits best is obviously project specific. For those researchers choosing mathematical/statistical models or simulation as the only evaluation technique, the papers on the risks of modeling and simulating wireless networks previously cited are a must-read to guarantee the reliability and reproducibility of the achieved results. However, whenever any form of experimental implementation is set as a goal, this does not mean that mathematics and simulations are to be neglected: although negative and positive effects of specific implementation choices may be discovered by trial and error, experimentally driven research should eventually lead to scientific output, under the form of a detailed analysis of the observed effects and contributing causes, or by creating an empirical model. While a working solution is preferred to no solution at all, it is important to know why or when a certain solution works or fails, whether optimizations to the solution are possible, and what additional steps or modifications could lead to this improvement. To answer the latter questions, mathematical models or carefully planned simulations might be required.

Experience has taught that when experimentation is the preferred method for wireless ad-hoc protocol analysis, one should take into account a considerable longer research process compared to producing research results using simulations only. As such, sufficient time should be allocated in the research schedule to solve issues that are not necessarily directly related to the algorithms under investigation. Due to the complexity of a full implementation, experimental research is often based on code that is already available, possibly created by people external to the organization. One should be particularly careful not to blindly trust off-the-shelf code: existing implementations will probably have been developed on different nodes using different wireless drivers and may require considerable efforts to modify. Furthermore, the programmer of the original code might have chosen not to implement specific parts of e.g. some protocol, since they were not interesting for him or her at the time. Unimplemented functions or errors in these implementations may be difficult to detect.

While a similar remark could be made for other than wireless research projects, this issue is especially relevant for wireless network experiments due to the fact that it is hard to measure and reproduce conditions leading to errors. For example, when debugging wired Ethernet based protocols, any network protocol analyzer can produce a fully accurate view of all packets that are sent in the network. In contrast, while it is relatively easy to sniff wireless packets, a (set of) wireless sniffer(s) based on commodity hardware will never be able to produce an accurate view of all packets that are sent on all frequencies of a wireless network. In operational environments, interference caused by devices external to the test set-up may temporary disable random links. Even if a sniffer node receives a certain packet, the same packet is not necessarily received by a nearby node participating in the test due to minor variations in signal strength and the potential presence of additional noise sources. As a result, there are plenty of reasons why a wireless packet is not received by a specific node.

If closed source software is used and support from the authors is not available or difficult to get, implementation issues may surface that cannot be resolved at all. For complex implementations, whether or not (partially) based on off-the-shelf code, one must typically be prepared to spend at least as much time to issues that are not directly related to the specific research goal under investigation, as to the real problem itself. As such, it is a specious idea that experimentation is a quick and easy alternative to performance analysis through mathematical models or simulations.

Example case. In our example case, the reason to develop the mesh router deployment tool is the knowledge that node positioning is not straightforward due to the unpredictable RF propagation and interference in complex environments. As such, studying this solution by simulations only makes little sense. Moreover, in order to find out if such tool can be used by non-technically skilled persons, a field test is required. From the previous chapter, the implementation of a self-configuring wireless mesh network is available. All code is developed in-house, and the only additional implementation needed is a subsystem somehow measuring the availability and connection quality to mesh routers that were previously deployed. Recall that every mesh backbone router periodically sends a proprietary neighbor discovery beacon. The idea for the example experimental implementation is to find out whether the RSSI values of these beacons can be used as a way to predict the quality of the mesh link(s) during deployment. Given the familiarity with the existing code, an implementation seems a realistic target.

5.3.1.2 Performance metrics

The selection for a specific technique, whether involving experimentation or not, has an impact on how the effectiveness of the solution can be measured. Results are usually more easily obtained from theoretical models and simulation results than collected from integrated implementations. When using models and simulations, all parameters that influence the measurement are inherently known and available. Statistics and measurements from different nodes can unambiguously be obtained and correlated, and parameter variations are easily performed. As a consequence, once models are available, they quickly lead to reproducible –but not necessarily correct or scientifically sound– results. For example, in a simulation, the exact position of nodes is either known in advance or can be recorded in case statistical models are used. Every packet that is sent by the simulated network can be logged with an exact time stamp, and it can be verified if and when these packets arrive at their virtual destination.

Moreover, performance results based on theoretical models or simulations do not necessarily require an accurate simulation of the entire system. For example, consider an algorithm determining an optimal fixed channel assignment in a multi-interface ad-hoc network through integer linear programming (ILP) models [27]. The typical approach used in an ILP model is to propose a model and assumptions, formulate a problem and objective function, and use an ILP solver to get the results. In this example, the model for the optimal channel assignment is focused around the distances between the nodes, an RF propagation model, and an assumption of a set of mutually non-overlapping channels. This ILP model directly provides tangible results, without needing to worry about, for example, how node positions should be determined, whether channels are fully non-overlapping or not, or which routing protocol is needed. As such, in this case, the ILP model is a perfect choice when trying to determine the theoretically best channel configuration and may be used as a guide to evaluate possible performance gains of using multi-channel techniques.

In contrast, in order to analyze the practical feasibility of such approach through an experimental set-up, all subsystems to organize and control an ad-hoc network such as scanning, neighbor discovery, address configuration or hooks for the modification of physical layer parameters are needed before a judgment can be made on the performance of the protocols. Moreover, gathering the information to produce an accurate view on the history of events leading to the decision of the channel selection protocol might generate several sub-challenges. Studying an integrated system is more complex than studying a single model, since the influence of subsystems may heavily impact the performance measurement of the protocol under test. Worst case, the experimental system may suffer from stability issues in other subsystems which may render evaluation of the target algorithm impossible.

Qualitative measurements In order to avoid a long development process that cannot deliver performance data in the end, it is good practice to think about how performance will be measured prior to development. Typically, real-life exper-

Qualitative results	Examples (non-exhaustive)
feasibility	*positive results*: the routing protocol/channel selection scheme/roaming protocol works in a real-life environment
	negative results: using this or that sensor network, the fire could not be detected due to failing links/delay issues /wrong assumptions regarding X or Y
QoE	*positive results*:the users did not notice their wired connection being replaced by a wireless connection
	negative results:connections are frequently dropped when roaming through the wireless network
compatibility	*with devices*: people were (not) able to use their own laptops (due to a difference in implementation of protocol X) in cooperation with our system Y
	with other protocols: our protocol is able to cooperate with several routing protocols
popularity	people (do not) regularly use our application because of X, people prefer tracking sensor nodes sewed in their clothes to carrying an extra device in their pockets
effort	the implementation was (not) easy to realize (since we ran into problem X or Y)
environment	the solution works good in an indoor environment but often fails outdoor
stability	since we started using this solution, we did not experience any downtime
topology	the solution works when a large number of people are within short distance of each other
QoS	traffic streams with high priority receive more access to the wireless medium

Table 5.2: Qualitative result categories and examples obtainable from implementation.

iments are a good source of qualitative performance data and allow to answer questions that cannot be answered through modeling or simulations only.

Table 5.2 shows examples of qualitative results that can be obtained from experimental test set-ups. One of the principal qualitative observations of wireless experimental set-ups is the feasibility or non-feasibility of a specific approach. From a research point of view, a lot can be learned even if a solution fails to work as expected. Furthermore, experiments allow to evaluate the quality of experience or to find out whether or not the solution is compatible with existing devices or protocols. Most qualitative observations are characterized by the need for many experiments, multiple testing locations, many prototype or end-user devices, and sometimes an external testing audience. By studying qualitative measurement opportunities during the preparation phase, one can verify if the logistics and time needed to perform measurements are realistic within the considered research project.

Quantitative results	Examples (non-exhaustive)
timing	how long until the system boots or network is stable; what is the end-to-end delay; how long does a handover procedure take to complete
throughput	maximum/average/total system throughput
ratio of success	how many times did the protocol or system behave as intended, how many times did it fail?
scalability	how many simultaneous users does the system support, how many wireless VoIP calls are supported?
link parameters	packet loss, link breaks, link lifetime, PHY layer data rate, received signal strength
system parameters	system power use, memory use, processor use
variance	what is measured, what was expected, what is the difference?
protocol specific	*positioning*: what is average/maximum error on the predicted location.
	network architecture: how many servers/access points are needed to provide full coverage
	routing metric X succeeds in finding the 'optimal route' in terms of metric Y or Z
	aggregation: the average number of packets sent is reduced by factor X

Table 5.3: Quantitative measurement categories and examples obtainable from implementation.

Quantitative measurements The drawback of qualitative experiments is that they generally do not lead to a lot of graphs and numbers, thereby sometimes making it harder for peer groups to appreciate the value of a certain implementation and its underlying realizations, and compare the achieved results with the state of the art. Fortunately, many quantitative measurements are possible as well. The main categories of quantitative measurements obtainable from implementation are listed in Table 5.3. Acquiring quantitative data may require additional non-trivial invasive implementations. For example, suppose an aggregation protocol for large scale wireless sensor networks is implemented, and a researcher plans to perform delay measurements through experimentation. First, a lot of devices should be acquired, and should be configured with the latest firmware. Due to the limited storage and processing capabilities of the sensor devices, it may be far from trivial to implement time synchronization or to store a large number of test packets with a time stamp on the devices. As such, delay measurements may not be possible without designing and implementing time synchronization on resource limited devices, or researching a low-overhead (multi-hop) data collection scheme. Moreover, these additional developments can introduce new errors.

Even for devices that are less constrained in terms of processing and memory power, some measurements are hard to acquire from real-life set-ups. For example, while a wired interface may be added to a powerful wireless mesh device for the collection of test data, or packets may be stored temporary on the device during the test, a physical link data rate or packet loss rate can still not be monitored without packets being sent. As such, continuous output of test parameters is not always possible without interfering with the actual test. By anticipating possible measurement difficulties during the planning phase, the necessary hooks to get the required statistics can immediately be included in the code, thereby assuring meaningful and accurate measurements after completing the protocol or system implementation.

Example case. In the wireless mesh deployment tool implementation example, once the tool is available, the feasibility of using RSSI values of received beacon frames as a way to determine backbone router locations can be verified. By interviewing people testing the solution, the end-user quality of experience may be evaluated. A quantitative analysis is also possible: for example, if the tool predicts a good connection quality, what is the actual average data rate that can be achieved to its neighbors? How stable is the quality prediction? If it is unstable, what is causing the instability, and how can this be resolved? How often does the approach fail? No insuperable problems are expected for these performance measurements, as such, the implementation work is likely to be rewarded with interesting performance measurements, justifying the implementation effort.

5.3.2 Platform preparation

5.3.2.1 Considerations for choosing a wireless platform

If any form of implementation was chosen during preparation, the search for a suitable hardware and software platform can start. While the design of fully customized hardware is an option, many researchers resort to combining commercial off-the-shelf components.

Depending on the specific topic under investigation, the hardware configurability may be rather limited. For example, if sensor networks are studied, most commercially available sensor nodes do not have any easily interchangeable components. In contrast, if the topic concerns IEEE 802.11a/b/g based wireless networks, a large number of COTS integrated products such as laptops or Wi-Fi routers are

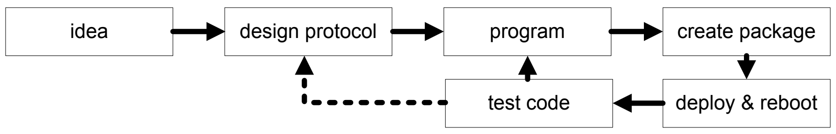


Figure 5.8: Development cycle using a node with limited or end-user inaccessible operation system.

available, and countless combinations of motherboards and processors, wireless interfaces, antennas and cases are possible.

In some cases, multiple platforms will be used during development. For example, this is the case when a solution is initially developed on testbed with desktop computers, and is then migrated to an integrated portable device in order to perform a field test. On many occasions, it was learned that migrating a wireless solution from one platform to another, even if they are similar, may require considerable additional efforts. This is mainly due to the fact that other hardware/driver combinations can behave differently, because of slightly different interpretation of communication standards. Moreover, as previously discussed in Chapter 2, when designing multi-interface solutions, embedded platforms generally suffer more from self-interference than desktop computers. Therefore, if a field-test is planned, it is recommended to immediately use the target hardware components, or approximate them as close as possible.

The initial choice for a specific wireless node is often based on a comparison of technical parameters (type and number of interfaces, power consumption, memory capacity, storage capacity, processing power, physical form factor), device cost, and support for development tool chains. The decision may further be influenced by previous experiences with or availability of other types of hardware, or project partners providing a specific type of hardware. Obviously, the technical properties should at least meet the technical requirements. However, in most cases, it is wise to choose nodes that have more capacities than are actually needed to increase the development comfort and prepare for future projects. In the example of the data aggregation protocol for sensor networks, the protocol may be expected to require only a very limited amount of memory and battery power. However, if the goal of the implementation is to functionally develop aggregation approaches and test the overall performance in terms of delays and success ratios, it makes little sense to choose devices with extremely limited capabilities, as this will only complicate the development process and limit future expansion possibilities of the protocols.

Development cycle The device choice has a large impact on how algorithms can be developed and deployed. For wireless devices, two main techniques exist. A

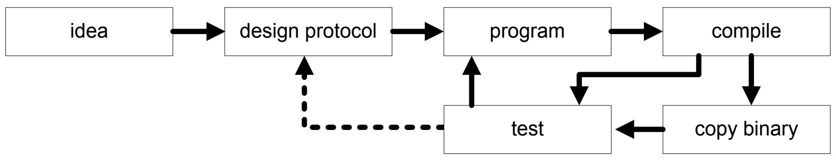


Figure 5.9: Development cycle using a node with mature operation system.

first group of devices is not able to run a complex (end-user accessible) operation system, and expects one or more packages containing drivers and protocols to be uploaded to the device; the device then runs the newly designed code after a reboot cycle. This technique is often used in integrated nodes such as sensor nodes or COTS Wi-Fi routers. Typically, the programming, (cross)compiling and packaging is done on a separate computer device that has little in common with the platform for which the code is developed. Once the code is deployed and running, limited or no control over the device is possible, unless specifically implemented through monitoring and control functions. The resulting development cycle is depicted in Figure 5.8 and shows how any modification to the code under test requires a complete re-build, deploy and reboot cycle. During past tests with such devices, for example during the development of the FRESME channel selection approach in Chapter 3, this approach was found to be tiresome. Furthermore, apart from the memory chip that stores the packages and is only accessible during the deployment of the code, there is often no access to non-volatile memory, complicating data logging during tests.

A second group of devices is able to run complex operating systems. Due to their configurability and open source nature, Linux based operating systems are a popular choice of wireless protocol designers. Many specialized Linux distributions have been designed for x86 embedded systems, some of them such as Voyage Linux [28] fully supporting packet managers while requiring less than 64 megabytes to install. By combining these distributions with a commercially available x86 router board with sufficient storage and memory capacities and one or multiple wireless interfaces, cheap and small wireless computing devices are created that are an excellent alternative to the use of laptop computers for wireless ad-hoc tests. The typical development cycle for these devices is displayed in Figure 5.9.

In this case, programming and compiling may be done directly on the target device. However, since the performance of these devices is still limited compared to a laptop or desktop computer, it is also possible to install the same distribution on a more powerful (virtual) computer, program and compile on this machine and simply copy the resulting binaries to the device(s) under test. In many cases, there is no need to reboot the devices, considerably reducing the deployment overhead.

Furthermore, a lot of mature (open source) network analysis tools and wireless network card drivers are available for Linux systems, shortening the development cycle. The availability of writable device storage simplifies data logging and allows the use of external configuration files, as such allowing configuration of network parameters more easily. Note that some tiny Linux distributions may be compiled into packages that can be installed on integrated devices with very limited memory capacities. An example is the OpenWrt [29] distribution for embedded devices, which can be run on top of dozens of popular Wi-Fi routers such as the previously used Linksys WRT54G family. However, due to the limited storage capacity of these devices, the functionality of tiny distributions is limited, and each modification still requires a full packaging, deployment and reboot cycle.

An example development platform of the second group of more complex devices was used for the implementation of the auto configuration and easy management protocols in the previous chapter. More precisely, an Alix3C3 [30] router board running the Voyage Linux distribution was used, in combination with the Madwifi [31] driver and the Click Modular Router [32].

Example case. The mesh routers in the example implementation are based on the mesh routers as developed in the previous chapter. As such, they will be implemented on the platform described above. More specifically, Alix 3c3 system boards are used with Voyage Linux 0.6 distribution on top of Linux kernel 2.6.24.7. The embedded system boards are equipped with an Ethernet NIC, a serial port, VGA output, compact flash storage, onboard audio, two mini-PCI slots and two USB ports. Two IEEE 802.11a/b/g compatible Compex wlm54sag23 mini-PCI wireless interfaces are installed, each connected to one dual band omnidirectional antenna. A one gigabyte flash card assures sufficient non-volatile storage space such that there is no need to worry about space constraints and full attention may go to the implementation.

5.3.2.2 Basic performance test

From past experiments, we have learned that it is a good idea to run a selection of performance tests on the chosen platform, prior to implementation of the protocols and before ordering large quantities of nodes. For example, the choice for the specific wireless interfaces used in the described test platform are based on performance measurements (throughput, stability, wireless spectrum measurements) of multiple mini-PCI cards in combination with the Alix router board.

Testing the basic performance of the chosen development platform allows to familiarize oneself with the default behavior of test hardware and development cycle, and helps to identify missing development tools. When (partially) relying on third party implementations, this is also the time to test their quality and stability. For example, when the research goal is to compare the performance of different

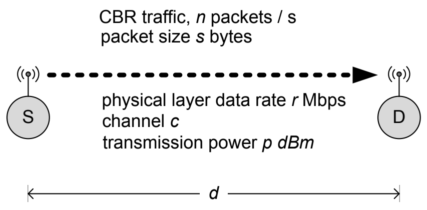

CBR traffic, *n* packets / s
packet size *s* bytes
S
D
physical layer data rate *r* Mbps
channel *c*
transmission power *p dBm*
d

Figure 5.10: Test topology for basic performance evaluation and parameters to be varied.

routing protocols in a large scale wireless test set-up using publicly available code, one must first verify whether the code is actually sufficiently stable to be used for measurement purposes: a bad implementation of a routing protocol cannot be used to make statements about the performance of a routing protocol – only about the specific implementation.

Hidden implementation issues. It is important to gain knowledge on 'hidden' mechanisms that are often included in drivers. For example, a wireless driver automatically performing power adaptation in the background is obviously not a good choice when measuring a relation between power setting and another parameter. Unknown or misinterpreted driver functionality may cause difficulties when trying to explain phenomena that are observed on a testbed implementation and may require a lot of effort to understand. As a first example, in [33], the authors study the effect of background scan procedures of the Madwifi driver on the throughput and delay of UDP and TCP traffic. The authors indicate a considerable impact on the performance, especially when aggressive settings are used. A second example is found in [34], where the side effects of proprietary solutions for fading and interference mitigation implemented in Atheros chips are described. The authors rightly indicate that many researchers are unaware of certain mechanisms implemented in their test hardware that may cause significant performance degradation. An actual case that was witnessed on many occasions is using an IEEE 802.11a/b/g interface with a single antenna, while leaving the Madwifi driver configuration at its default settings to use 'antenna diversity'. The latter setting assumes the availability of two antennas connected to a single interface, and, among other things, will transmit broadcast packets alternately via the first and second, non-existent antenna. Obviously, this leads to a high loss of broadcast packets. While some mistakes may seem very basic, 'hidden' mechanisms may be very hard to discover in wireless networks, due to a combination of a large parameter space and difficulties in monitoring wireless experiments.

Performance test: initial set-up. A basic performance test can be done by putting two test devices in the basic single-hop topology of Figure 5.10. In this test set-up, a sending node and destination node are separated by distance d. Next, a constant bit rate (CBR) packet stream is configured, where n packets per second are transmitted with a size of s bytes. If the physical layer data rate r can be adjusted, it is initially set to the lowest possible value, and both devices are further configured to operate on channel c using a transmission power of p dBm. If any of the values cannot be adjusted, they are necessarily left at the default value.

In an environment that is not too heavily interfered, put the devices in line of sight at a relatively small distance d such that they are easily accessible during the experiments. However, unless the devices are intended to be used at very short distances (e.g. in body area networks [35]), it is safer to separate the devices by at least 2 meters, avoiding the near field of the sender, and to place them away from any electronic devices such as CRT monitors or server racks that might cause a high amount of interference. Set the transmission power p to the lowest possible value and first channel c of the channel range. Next, configure a broadcast CBR packet stream. Which packet size s and packet rate n to choose depends on the chosen technology: configure these values such that the chosen technology should theoretically be able to easily cope with the requested CBR stream at the lowest physical layer data rate. For example, when IEEE 802.11g interfaces are used, set s to 100 bytes and n to 10 packets per second.

Following measurements can now be performed:

- ***Packet loss under low load conditions.*** To account for the effects of temporal fading, measure the packet loss at the receiver side during a sufficiently long period of time of for example 15 minutes. With these settings, the packet loss should be limited or zero. If packet loss did occur, verify whether the loss was uniform over the test duration or not. If packet loss did occur in bursts, the packet loss may for example have been caused by someone downloading a file over a nearby interfering access point. If the loss is relatively high but uniform over the test duration, re-run the test while increasing the transmission power p until the packet loss is minimal. Take note of all test settings and according results. If after several attempts, performance is still not satisfactory, a wireless packet sniffer may be used to find out whether the test packets are sent by the sending node and which other wireless traffic is observed on the communication frequency and neighboring frequencies. Keep in mind that packet sniffers are often limited to analyzing a single radio technology: obviously, a Bluetooth sniffer will not detect WLAN traffic. Other technology independent hardware components exist allowing a technology independent spectrum scan of the used RF range. While they are helpful to identify unexpected interfering sources such as microwave ovens or cordless phone signals, in our experience, these tools often do not provide much additional information.

  If after several tries, a stable wireless unidirectional link is still not achieved, it is a good idea to migrate the full wireless set-up to a set-up with emulated wireless medium as described in Section 5.2.3.2, as to exclude any sources of external interference. If at this time stable results are still not observed, there might be an issue with the wireless interface card, the driver, a setting, or a combination thereof. By replacing any of these components, the source of the issue is likely be found.

- ***Variation of physical layer parameters.*** As previously discussed in Chapter 2, some wireless nodes were observed to consistently fail to operate on certain channels. For many implementations, a single 'bad channel' is not necessarily a major issue. However, when designing e.g. a dynamic channel selection protocol, important performance degradations are to be expected during experimental validation. Similar problems may be observed when modifying transmission power, especially at high transmission power settings. Therefore, the low load packet loss experiment is best repeated while varying the communication channel c and power setting p. Measurements of the output spectrum of the wireless interface similar to Figure 2.16, might help to analyze the origin of the observed phenomena.

- ***Throughput evaluation.*** When comparing different hardware/driver combinations, it may be worthwhile to check the maximal performance of the devices by increasing the packet size b to the maximum packet size before fragmentation and increasing the packet rate r. In the absence of interference, the measured value should be close or equal to the calculated theoretical maximal value. If not, an unknown potentially wrongly configured background mechanism might be interfering with the measurement, or the hardware may not be as good as hoped for. For example, the processing power of the device might be too small to support high data rate applications.

- ***Long term stability.*** If the performance of the development platform is satisfactory, the long term stability of the development platform may be tested by running a high throughput test for several hours or days. During development phase, most tests will probably only last seconds or minutes, but if the solution is eventually going to be deployed in an external testing environment where researchers cannot always easily reach the devices, the long term stability should be guaranteed. The described test is ideal to discover memory leaks in an early stage, or find out unexpected effects such as a temperature sensor which does not report an accurate temperature due to self-heating caused by its own control unit.

- ***Additional tests.*** Depending on the specific situation, additional tests of the development platform may be performed. For example, if protocols for a

parameter	unit
packet sending rate	packets per second
PHY transmission rate	Mbps
number of test packets	
packet size	bytes
transmission power	dBm

Table 5.4: Adjustable parameters at the sending side of our benchmark tool.

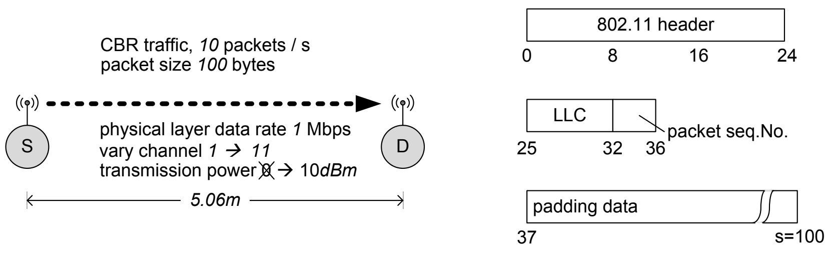


Figure 5.11: Basic performance evaluation of our example platform and packet format used for custom test tool.

multi-interface solution are developed, a simple two-hop test where the two links are configured to theoretically non-interfering frequencies, similar to the experiments performed in Chapter 2, may indicate to which extent the devices will suffer from self-generated interference. Additional tests may also be performed if invasive modifications to the hardware or drivers of the development platform are needed. For example, it could be verified if the platform allows changing the transmission channel of every individual packet, should this feature be required.

In the next section, several of the above tests are carried out for the example implementation, illustrating the importance of a basic performance test.

5.3.2.3 Basic performance test for the example case

Two nodes are placed at a distance of 5.06 meter. The example implementation requires a measurement of the RSSI values reported by the wireless drivers. To access these values, a tool called ‘phyclick’ is developed through the programming of custom Click elements. A sender script is installed at the sending node that allows creating and broadcasting custom built test packets, which are displayed alongside the test set-up shown in Figure 5.11. Through the sender script, the parameters from Table 5.4 can be varied. A receiver script is installed at the receiver

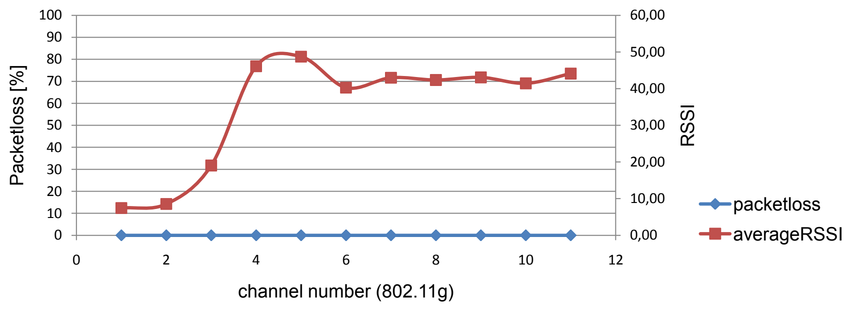


Figure 5.12: Packet loss and average RSSI recorded during the basic performance test of the demonstration implementation, for varying communication channels. Set-up and parameters as in Figure 5.11

node. The receiver script filters out the test packets and creates a log of all received packets, containing a timestamp, the sender MAC address, packet sequence numbers, and physical layer parameters such as the physical layer data rate that was used, the observed antenna noise, and measured RSSI value. During the basic tests, a predefined number of packets is sent, and packet loss is recorded at receiving side. In the event of packet loss, the packet sequence numbers help to determine whether the loss was uniform over the test period or not.

During a first packet loss test with an output power of $0\,dBm$ using channel 1 and IEEE 802.11b/g technology at a physical layer rate of $1\,Mbps$, in contrast with expectations, not a single packet is received. Therefore, the transmission power is increased to $10\,dBm$. This time, the packet loss is zero. The basic packet loss test is now repeated for every other channel with only 3 packets lost during the entire range of tests. This positive result is plotted on Figure 5.12, together with the naive arithmetic mean of the received RSSI values. The absence of packet loss is a first indication that our development platform is likely to be sufficiently reliable for the planned tests. However, the shape of the RSSI graph is a surprise: instead of being relatively flat, the average RSSI is considerably lower for the lowest channel numbers. This graph provides a clue to finding out why the tests on channel 1 at the lowest transmission output power of $0\,dBm$ failed: further analysis of the log files shows that the antenna noise observed at the first three channels ($-53\,dBm$, $-55\,dBm$, $-63\,dBm$ respectively) is considerably higher than the antenna noise at the other channels, which is, as expected [36], almost constant around $-95\,dBm$. This might indicate that an external test set-up is interfering with the test devices. However, a Wi-Fi scan of the environment does not show any nearby active access points.

Additional measurements using a Fluke Etherscope network assistant suggest that a non Wi-Fi device in the testing environment is emitting a RF signal of

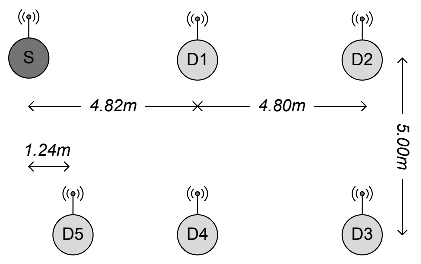


Figure 5.13: Test topology used for the additional tests regarding the example case. The sender S broadcasts test packets to the five destination nodes $D1 - D5$.

$-55\,dBm$ at the frequency corresponding with channel two, thus confirming the hypothesis of external interference. As such, this first performance test already provides valuable information:

1. The chosen development platform is operating as expected.
2. At least one device in the testing environment is causing an important level of background noise. Possible further actions are to find and shut down the interference source –which might not always be possible in a testing environment shared with other researchers –, move to another testing location, or simply to accept the interference and consider it as an additional testing opportunity to verify the behavior of the algorithms in the presence of noise. In any event, the simple knowledge that strange phenomena that might be observed during future experiments may be caused by an external interference source is valuable information.

Next, a two hour long low load stability test with same test parameters at channel 1 shows that 71174 out of 72000 packets are received (98,85%), and that the lost packets are uniformly distributed throughout the test. Moreover, throughput tests using the development platform at higher transmission rate show that the maximum application layer throughput is approximatelly 34Mbps, which is close to the theoretical maximum. From these generic tests, the development platform proofs to be sufficiently reliable. However, as RSSI values will be used as the main metric, a few additional tests are carried out in order to verify if the reported RSSI values are stable for static nodes. If these measurements do not show stable results at all, then there is little sense in continuing the development according to the original plan.

For the additional tests, consider the test topology and test parameters of Figure 5.13. During this test, the same 'phyclick' test tool is used, with five nodes configured to receive the test packets. The interfaces are configured to the IEEE

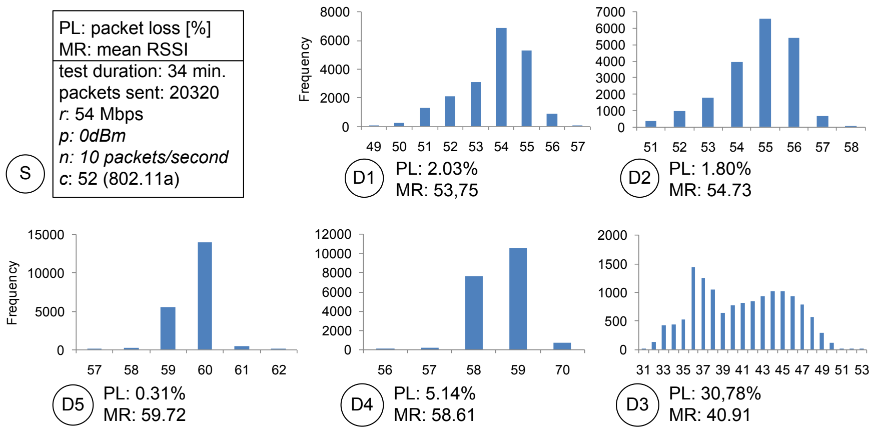


Figure 5.14: Histograms of results obtained from the test set-up of Figure 5.13. For each node, the y-axis shows the number of packets received for the corresponding measured RSSI values labeled on the x-axis. Different axis scales are used.

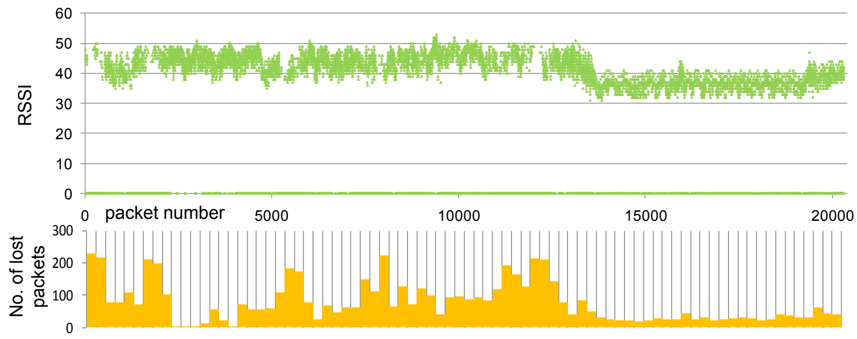


Figure 5.15: RSSI measurements at node $D3$ in function of the packet sequence number. The bar graph shows the total loss for a corresponding packet window with a size of 250 packets.

802.11a channel 52 as to avoid as much interference from other test set-ups as possible, and the physical data rate is set to $54\,Mbps$ in order to induce packet loss in the size limited test set-up. At each receiving node, the packet loss and mean RSSI are determined. The results are shown in Figure 5.14, this time also presenting the measured RSSI values in a histogram. The figure shows that the packet loss observed at the receiving nodes is limited, except for node $D3$. Since less packets were received by $D3$, less RSSI samples are available. Nevertheless, the RSSI measurements performed at this node are most dispersed. In order to get a view of how the measurements at node $D3$ vary in time, and in order to know if lost packets are especially observed under low-RSSI conditions, Figure 5.15 plots the RSSI measurements in function of the packet sequence number, and compares the measured RSSI values with the total loss during a corresponding packet window with a size of 250 packets. A first observation from this figure is that the RSSI trend drops from a mean value of 45 to a mean value of 35 at about 13000 packets or about 21 minutes into the test. A second observation is that, surprisingly, the observed packet loss is generally lower when the lowest RSSI values are reported. This large variation of reported RSSI values and contra-intuitive packet loss behavior may cause the example approach to become unfeasible. Since the example application is used illustratively, and all other nodes except $D3$ report relatively stable RSSI values, no deeper analysis of the observed issue is performed. If this application was the goal of this research, the next steps would be to analyze the hardware (e.g. wireless interface or antenna connector) and settings of the device, and re-run the test, potentially in a shielded environment. In any case, this simple test is a reminder of the fact that wireless set-ups do not always behave as expected. Furthermore, the test results may inspire new metrics such as the standard deviation of a set of RSSI measurements, rather than only considering the mean value.

To summarize, a basic performance test of the target development hardware is invaluable to discover the behavior of the development platform, to learn how to work with development tools, and/or create new tools needed for the implementation. Development platform issues discovered during an early phase are far more easy to solve or avoid, since at this point it is still possible to switch to a different platform or modify the original measurement approach. If the development platform is highly unstable even during simple tests, or unable to perform the actions that will be required by the protocol or system to be developed, continuing the implementation makes little sense. Unfortunately, all too often, these practical issues with the development platform are only discovered after months of research.

5.3.2.4 Simulator characterization

Even if real-life implementation is the method of choice for the eventual performance analysis, simulations may be helpful to develop and test the functionality of a developed algorithm. A few basic simulated topologies are often sufficient to verify the operation of an algorithm in an idealized environment. However, in order not to have to develop and maintain two separate implementations, it is best to resort to techniques allowing to use (nearly) identical code both in the simulator and on the real device. An example of such tool used in this work is nsclick. Other similar tools exist, such as the TCP/IP network simulator proposed by Wang and Kung [37].

Simulation may also help to evaluate the performance of a developed solution in a larger topology than can be realized in reality. If comparison of test results achieved from a real-life test set-up with results obtained from an identical simulation shows that simulations are representative in smaller scale test set-ups, additional simulations in a large scale topology may be used to extrapolate results more reliably than would be achieved by simulation only. Traces and measurements achieved from the test set-ups may be used to create or enhance the reliability of the simulator environment. For example, the previously determined values of Table 2.2 can easily be used as input in a packet simulator to replace an on/off link model with a simple yet far more realistic connectivity model.

By analogy with the characterization of the implementation platform, the basic behavior of the simulator and potential third-party protocols is best studied through some simple experiments. Even if the performance results collected with the simulator are not entirely realistic, this does not mean the simulator cannot be used during the implementation process. For example, this approach was followed while designing the handshake approach used in the FRESME channel selection protocol in Chapter 3. In an initial stage of development, it often does not matter whether the link rate, packet loss, or interference is modeled accurately: in this phase, basic functional analysis is more important. Example information that may be gathered from simulations are whether the packet structure of the newly created packet types is as designed, or whether the correct response function is triggered on receiving a specific type of challenge. At this time, idealized behavior is even preferred to unexpected behavior such as packet loss, as this considerably simplifies functional debugging.

5.3.3 Development

5.3.3.1 Development Methodology

Now that the device and simulator characteristics are known, the development phase can start. In Figure 5.16a, a traditional top-down development approach is depicted, where code is fully written and tested in a simulator environment, then ported in case the code for the simulator and experimentation platform are not compatible, and finally used for experimental analysis. Although this is a natural approach to follow when first using real-life implementation for performance evaluation, the main drawback of this approach is that basic flaws of the code may not be revealed until far into the development process. For example, an implementation may have worked in simulation when no packets from external networks were interfering, but fail to work on the experimental hardware when unknown packet types transmitted by devices external to the test set-up are received. Even if the source of the experimental problems can be determined, adding a solution to the implementation may require a redesign of the entire algorithm or system, leading to considerable time loss.

During the implementations in this work, the approach illustrated in Figure 5.16b

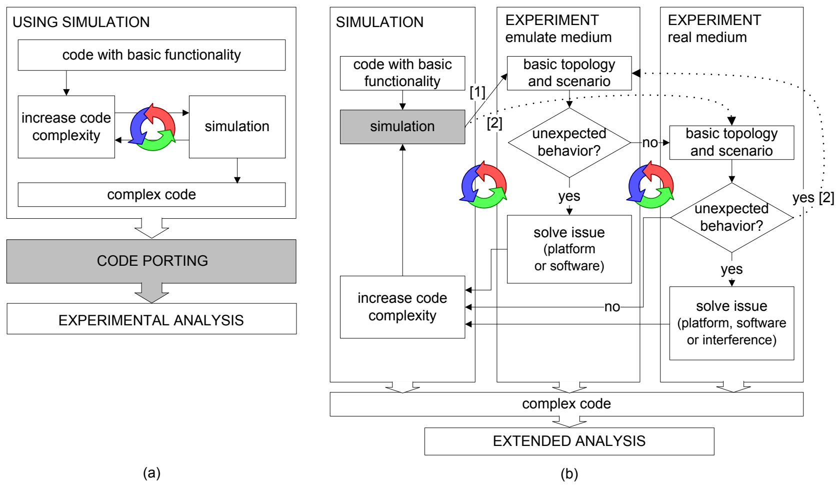


Figure 5.16: (a) Traditional development approach: simulated code is ported and used for experimental analysis. (b) Alternative development approach: simultaneous use of simulation and experimental environments. Gray blocks are optional.

was found to be far more efficient. In this approach, experimental analysis is no longer postponed until after completing the code, but simulation and basic experiments are carried out simultaneously.

As already stated, in case the protocol under development can be simulated, simulations are an excellent way to verify the functionality of a solution under idealized conditions. Therefore, they are still used as a first step during development. Since simulation is not always possible, the simulation block in the figure can be left out, or be replaced by experiments with emulated wireless interfaces and emulated wireless medium.

As soon as code with basic functionality is available, it is tested in parallel using simulations and a basic experimental set-up. Much can be learned from comparing the same experiment over an emulated medium as described in Section 5.2.3.2 with the results from a fully wireless testbed (see Section 5.2.3.3). For such a comparison, two approaches are possible. In a first approach, the level of reality is gradually increased. This is depicted in Figure 5.16b with the line marked with [1]. If an experiment with emulated medium shows no unexpected behavior, the experiment can be repeated over the wireless medium.

The other approach, marked with [2] and presented using dotted lines on the figure, is to immediately test the developed code in a fully wireless experiment, and only perform a test on the emulated medium when unexplainable errors are encountered. This approach leads to faster results when only few implementation issues are encountered.

As long as no errors are found, the protocol or system complexity is gradually increased. In the event of unexpected behavior of the test set-up, the combined analysis of simulation, emulation and testbed makes it possible to localize the failing subsystem more easily using the following approach, which exploits the differences that were presented in Table 5.1. It is assumed that the code under test used in simulation and during experiments is identical, and that the basic performance test was successfully passed and the basic configuration of the development platform (e.g. configuration of interfaces, deployment and activation of the developed code, firewall settings) is correct.

1. If the **problems occur when using an emulated wireless medium, but not in simulation**, the problem may be caused by the development platform or by the results of packet loss, causing the developed algorithm or system to fail because a specific situation popped up that cannot be observed in simulations.

   Reassess the basic performance of the development platform by repeating some of the tests performed during the platform preparation phase, especially when performing experiments in a new test set-up or after enabling additional nodes. The connectivity of the test set-up and quality of the connections may be different from what is expected, for example due to a node or interface malfunction, or a loose cable. Verify if any updates to any subsystem such as driver or package updates were performed during the last couple of experiments that might cause the observed problem.

   Even when connecting wireless devices with a real wireless interface over a cable, packet loss may be suffered, although the quality of each individual link should be reasonably stable. As such, verify the protocols under test and potential third party code to assure that the protocols are able to handle any type of packet loss or packet delay; this will be a requirement for the final implementation anyway.

2. If a **problem occurs in a wireless testbed, but not when the medium is emulated**, the issue is not likely to be caused by the development platform, but by the consequences of interference, changes in the experimentation environment like moving persons or objects, or other effects such as shadowing and multipath fading [38] could be at the root of the observed problems. As a result, in comparison with the emulation strategy, the developed system now has to account for frequently changing link qualities, unidirectional wireless links, and packets sent by other wireless equipment. As in the previous case, it is often worthwhile to run a quick connectivity test on a specific testbed topology before running an experiment.

5.3.3.2 Development hints

Several lessons were learned during past development cycles. First, coping with the unreliable wireless medium often requires additional subsystems to be implemented, or certain protocol parameters to be tuned. While these additions and modifications might seem rather trivial during the implementation, some of these additional subsystems are worth being documented on their own, as they may help other researchers to produce stable experimental results. Therefore, it is a good idea to take note of any additional implementation or modification during development.

Second, when performing frequent experiments during development, it may be inviting to rely on intuition in order to make important decisions. However, it is important not to forget about the mathematics behind the implementation: an implementation does not justify using ‘out of the blue‘ values for protocol settings, timing settings, or other parameters. For example, it may be hard to judge the overhead caused by a specific implementation; a quick calculation may show that subsystems, which were believed to cause significant overhead, are actually not causing that much overhead as expected after all. When testbed experiments using tens of nodes fail to operate as expected, it is easy to blame ‘interference’ and collisions, for example caused by e.g. signaling beacons, while in reality the routing protocol could be failing due to scaling issues.

Third, when several subsystems are developed in parallel by several people which may work at different location, it is a good idea not to postpone the integration until the end of the research project. If unexpected behavior is observed after code integration, comparing results using the different performance evaluation mechanisms may show the source of the problem. In our experience, many integration issues are not caused by the specific code under development itself, but by a mismatch of development platform versions or parameters. This fact again shows the importance of logging even minor adjustments to the development environment.

Finally, a graphical representation of the topology and collected test parameters are often very helpful to better understand the dynamics of the wireless network and the developed protocols. The initial cost of GUI development is quickly paid back when a large amount of log files can be visualized.

5.3.3.3 Testbed development and characterization

During development, the topology of the test set-up may gradually increase in complexity. While initial experiments may start on a desktop, it is hard to maintain this approach when more nodes are added to the set-up, both because of space limitations and side effects of dense networks. Therefore, a larger scale testbed may be developed.

For static wireless set-ups, a first aspect of installing a testbed is finding appropriate locations for the nodes, developing the tools needed for easy deployment and manipulation of the deployed algorithms, and finding a way to easily collect log files from the test set-up. Second, additional devices such as wireless packet sniffers may be added to monitor the testbed externally. For details on the creation of our w-iLab.t testbed, the reader is referred to Appendix A.

When node mobility is required, testbed development is even more challenging. In order to guarantee reproducibility of tests using mobility, random movements are best avoided during development. Emulating the wireless network card or wireless medium are best used in an initial phase. If fully wireless mobility tests are carried out, mobile nodes should always be carried along a planned path. In outdoor situations, GPS coordinates can be used to log and control the exact movements of mobile nodes [39]. In indoor situations, one should strive to follow one or multiple predefined benchmark paths as good as possible, or resort to GPS-less location determination methods [40]. If moving people are the source of mobility in the network, techniques such as used in the previously mentioned APE testbed [21] may help to guide people along the planned path. In case faster mobility is required, it may be necessary to resort to network card or wireless medium emulation. As with other test set-ups, whenever modifications are made to a testbed, it is a good idea to verify the basic performance before running any tests.

During the development phase, both the developed algorithms and test topology grows more complex step by step. Once the developed protocols are sufficiently mature, a deeper analysis of the developed system may start. The analysis may reveal additional issues or inspire new functionalities. As a result, the border between the development and analysis phases narrows in many cases.

5.3.3.4 Development of the example application

A first step in realizing the example easy deployment tool that will help to determine suitable locations to add new mesh nodes to a mesh backbone is to perform a breakdown of the required functionality into several subsystems. Recall that this example implementation is seen as an extension to the auto-configuration approach developed in the previous chapter, and that the idea of the approach is to determine the suitability of a specific candidate location by performing RSSI measurements on the custom neighbor discovery beacons used in the previous paragraph.

Following implementation steps are identified:

1. Capture all packets sent on the wireless medium, and filter out the custom neighbor discovery beacons.

2. Retrieve the RSSI measurements corresponding with these packets from the wireless driver.

3. Keep a list of every configured mesh node from which beacons are received. With W a configurable parameter, continuously keep track of the latest W RSSI values per beacon sending node. W is the windows size used to calculate a naive moving average of RSSI values, for each beacon sender.

4. Interpret the RSSI mean value and give feedback to the user.

To realize the above steps in individual subsystems, the second approach of Figure 5.16 marked with [2] is followed; the code is developed in a virtual machine, compiled, and then copied to a single testing node to be tested in the presence of a single mesh node transmitting beacons. No simulator is used, as few implementation difficulties are expected and implementing the beaconing mechanisms in the simulator would cause a relatively large overhead with respect to the target implementation.

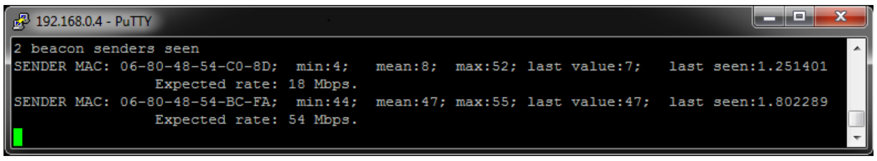
The first subsystem is realized by configuring the node under test to promiscuous mode. For the example implementation, the node running the deployment tool is configured manually to match the channel on which the beacons are sent. In a final version, a scanning procedure should precede the above steps. Neighbor discovery beacons are easily identified since their structure is fixed. A first test on the node shows that messages are filtered out effectively.

For the second subsystem, the solutions that were previously designed for the 'phyclick' benchmarking tool from Section 5.3.2.3 can largely be re-used. After some minor adjustments, an initial version of the deployment tool is successfully completed. The program continuously displays the RSSI values that are retrieved from the beacon messages.

The third subsystem is a pure programming matter: organizing data which was already available after the second step. After implementation, new test shows that RSSI values are organized and stored per sending node. If a new beacon transmitter comes within range, it is added to the overview dynamically.

To provide the logic behind the fourth subsystem, the results of Table 2.2, previously determined in Chapter 2, are used. Using this table, a mapping between observed RSSI and estimated maximal physical layer data rate is made. In order not to base the estimation on a single value, the naive mean RSSI value, averaged over a window of size W is used. When used in an actual deployment in a user-friendly way, the deployment tool should provide an indication of the connectivity to be expected through a small screen on the device or through signaling LEDs. However, for the proof-of-concept implementation, the connectivity information is retrieved by making a wired connection from a laptop to the mesh device that is to be deployed. The statistics are updated on the laptop screen every second.

The finalized implementation is now tested on a desktop testbed in a simple scenario: two mesh nodes are transmitting beacons. The antenna of one mesh node is removed as to emulate a distant node. Figure 5.17 shows the resulting output of



```
2 beacon senders seen
SENDER MAC: 06-80-48-54-C0-8D;  min:4;   mean:8;  max:52; last value:7;   last seen:1.251401
                Expected rate: 18 Mbps.
SENDER MAC: 06-80-48-54-BC-FA;  min:44;  mean:47; max:55; last value:47;  last seen:1.802289
                Expected rate: 54 Mbps.
```

Figure 5.17: Screenshot presenting the output of the deployment tool. Two mesh backbone nodes are observed. Minimum, mean and maximum observed RSSI values are presented, as well as a counter indicating how long ago a mesh node was last seen. Based on the mean RSSI value, an estimation of the expected physical layer data rate after deployment is displayed.

the prototype example deployment application. In this case, the deployment tool indicates that two nodes sending beacons were observed. For each sending node, the minimum, maximum and average RSSI measurement are displayed. Furthermore, the last RSSI value that was retrieved is shown, as well as an indication of the time since the last beacon was received (in seconds) from this node. The *last value* indicator is used to quickly get an idea of the RSSI value at a new location, without the need to wait for W RSSI samples to be retrieved before a new, mean value can be read which reflects the actual situation at the new location. The *last seen* value enables to detect whether the measurement data is still fresh. Based on the mean RSSI value, an estimation of the physical layer data rate to be expected is displayed to the user.

With the proof-of-concept implementation finished, and after completing the desktop test successfully, the deployment tool can now be analyzed in the next phase.

5.3.4 Analysis

While several experimental tests were already conducted during the development, a deeper analysis of the developed wireless protocols or system cannot be performed after every minor upgrade to the code. Therefore, once the design goals are (almost) achieved, sufficient time is needed to perform an in-depth analysis and gather the qualitative and quantitative results that were aimed for during the project preparation. The results may be measured through extensive testbed measurements or through a field test, either performed by the developers or by an external test audience representing the target end-users.

Before performing field tests, it is recommended to perform a thorough analysis of the behavior of the solution in the testbed environment that was also used during the development, as the collection of test data and manipulation of the devices is normally a lot more comfortable in the testbed environment than after deploying the same solution in the field. Furthermore, the development tools can still easily be accessed in case minor adjustments are needed to the solution. Once in the field, the options to control the nodes, collect logging data or modify the algorithms under test are more limited, complicating the analysis.

The analysis phase typically requires a large amount of tests, resulting in a high volume of log files. The quality of the quantitative results that are derived from this test data is strongly affected by the availability of as much details about the experiment and environment as possible. The importance of logging the relevant parameters used during a test cannot be stressed enough: packet logs are useless in case the specific configuration used during the test are not available. Important parameters include settings of the developed protocols, test topology, mobility patterns, details about the set-up and environment such as geographical location of the nodes, type of building, vegetation, external sources of interference, as well as details about the device configuration, such as version of the drivers and protocol(s) under test, output transmission power, or communication channel.

If the wireless device has sufficient processing power and non-volatile memory, packet logs may be stored on each device. Certain wireless systems will only allow to retrieve a list of packets at layer 3 of the OSI stack. These packet logs may hide a lot of information available at lower layers, such as MAC layer retransmissions or ACK frames. When packet logs on the device are not an option or more detail about the packets on the wireless medium is wanted, strategically placed sniffer nodes configured to the relevant channels may lead to a better view on the packets that are sent by the test devices and by nodes interfering with the test set-up. Recall how during the analysis of the configuration handshake used in the FRESME protocol (see Section 3.2.6.2), studying packets at different layers of the stack may explain the specific behavior of a certain solution.

In order to guarantee the relevance of measurements, a specific test should be repeated several times before drawing conclusions about a solution. Furthermore, these results are strictly spoken only valid for the test set-up under consideration. If time allows, testing a single solution under different topologies adds credibility to the general performance of the implementation.

Despite all development efforts, the wireless system may not behave as expected when deployed in a field test. In our view, this is no reason to not publish any test results, on condition that a thorough analysis of events leading to the failure of the solution is made. This may be done by varying between different performance analysis methods in search for the root cause of a specific problem.

Ideally, a final set of tests can be performed by the target end-users. This can only be done under the assumption that all subsystems are implemented, sufficiently stable, and if no technical knowledge is required from the end-users. Especially when tests are carried out in the absence of the developers, it may be very difficult to produce reliable logging information.

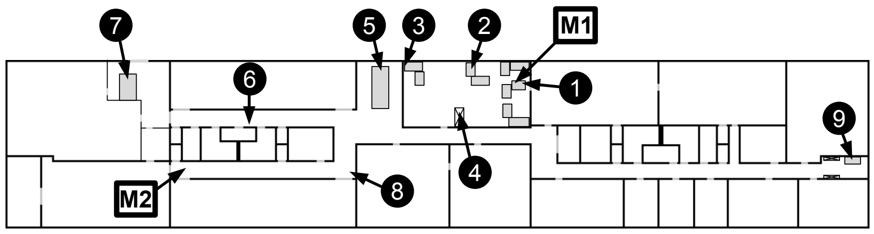


Figure 5.18: Floor plan of the testing area, sized approximately 90m x 18m. The existing mesh routers $M1$ and $M2$ are indicated with squares, the candidate locations for placement of an additional node are depicted as circles.

		to *M1*		to *M2*	
location	description	mean RSSI	PHY rate estimation	mean RSSI	PHY rate estimation
1	on desk	55	54	12	24
2	on desk	38	54	16	36
3	on desk	34	54	29	36
4	on closet	44	54	21	54
5	on desk	34	54	29	36
6	on ground	25	54	18	36
7	on desk	18	36	19	48
8	on ground	24	54	41	54
9	on desk	9	18	9	18

Table 5.5: Measurements and estimations collected using the example deployment tool at the different placement locations. Transmission power during the test was set to 1dBm.

Example case. In order to analyze the behavior of the example deployment tool, the tests that were previously planned in Section 5.3.1.2 are taken under consideration. To keep the analysis manageable within the scope of this chapter, a real-life deployment with two existing wireless mesh nodes is targeted, in which a third mesh backbone router needs to be introduced. Consider the testing location displayed in Figure 5.18. Two mesh nodes, $M1$ and $M2$ are installed at an office building, on the locations indicated by squares on the map. In accordance with the example scenario from Figure 5.7, a third node is now to be added at one of several candidate locations, indicated with circles on the map. Without a deployment tool, there is no easy way to determine the 'best' location with respect to optimizing the connectivity in the mesh backbone. The developed deployment tool could prove to be a feasible way of determining node locations if: (i) the mean RSSI value is sufficiently stable in time at a given spot, and if (ii) the RSSI-to-data rate mapping proofs to be correct.

To this end, during the analysis phase, the node that is to be deployed is

node location	PHY rate estimation $[Mbps]$	calculated maximal appl. layer bandwidth $[Mbps]$	measured application layer bandwidth $[Mbps]$
1	54	30.5	30.1
2	54	30.5	30
3	54	30.5	29.3
4	54	30.5	30.7
5	54	30.5	28.9
6	54	30.5	27.0
7	36	23.7	20.6
8	54	30.5	28.6
9	18	14.1	7.69

Table 5.6: Comparing the physical layer rate estimations for the link between the deployed node and $M1$, with the maximum application layer bandwidth measurements collected from the test set-up with the iperf tool, for the different candidate node locations.

equipped with a battery, enabling it to be quickly moved in between the different candidate locations. The measurements that are collected through the use of the tool are presented in Table 5.5. The test values are obtained with the window size for moving average calculation W equal to 20, and tests were performed during the evening, with no people present in the building. A neighbor discovery beacon is sent every 2 seconds from every node on channel 7 (auto-selected by the auto-deployment algorithms from the previous chapter), using a transmission power of $1\ dBm$. This means that an accurate mean value is obtained the soonest after 40 seconds.

During the experiment the reported ‘last RSSI value’ is observed to be stable. As a result, the mean value quickly converges to match the ‘last value’. After the 40 seconds convergence time passed, the ‘last RSSI value’ never deviated from the mean RSSI value by more than 2 dB, meaning that the measurements obtained through this technique are relatively stable. The measured RSSI values indicate that at most locations, a relatively good connection to both the mesh routers is observed. The results suggest that location 4 (where the new node is put on top of a closet) and location 8 (with the new node on the floor in the hallway leading to existing mesh node $M2$) are the most suitable locations, with an estimated physical layer link rate of $54\ Mbps$ to each of the two mesh nodes.

Finally, the estimated physical layer connectivity at each location is compared with an application layer bandwidth measurement obtained at the same location using the iperf capacity measurement tool, previously used in Section 3.3.2. For the wireless link between the newly deployed node and the existing mesh router $M1$, Table 5.6 compares the estimated physical layer bandwidth with the corresponding calculated maximum application layer bandwidth, and with the average of three one-minute iperf UDP measurements. The calculated application layer throughput is determined through the method presented in [41], using a packet size of 1500 bytes corresponding with the UDP test packet size, and an average-sized contention window of 7.5 IEEE 802.11 time slots [42], reasonably assuming few other nodes contending for the wireless medium at the time of the experiment. Except for location 9, the calculated and measured bandwidths largely correspond, suggesting the feasibility of the followed deployment approach, especially when a good connection is indicated.

The fact that the estimations for the highest rate can be explained as follows: Table 2.2 shows how the minimal required RSSI values for the lower data rates are all within a relatively small range, while all RSSI values higher than 20 result in the maximal physical layer rates. As such, mean RSSI values that are considerably higher than 20, leave a lot of margin to compensate for variations in the wireless medium. In contrast, a minor variation on the mean RSSI in the range between 2 and 8 may quickly lead to a relatively large degradation in maximal obtainable physical layer data rate.

More measurements could be performed to further analyze the behavior of the example implementation; measurements in other environments could either confirm or refute the promising first analysis; one could verify the influence of the moving average window size W; additional data sources could be included in the link quality analysis (e.g. estimate the neighbor discovery beacon time, then keep track of the number of beacons that are lost, or, take into account the influence of new routing possibilities in the network); or, one may further develop the solution to a more final prototype which can easily be used by non-technically skilled persons to enable usability testing. These interesting aspects are left for future research.

5.3.5 Reporting

In the final phase, the test results that were gathered during the analysis phase are put together and a report is made. The most important aspect of reporting the results is to ensure that all relevant aspects that might have influenced the test results are listed. Test results become irrelevant in case no details are known concerning the used technology, topology, protocol parameters, measurement strategy, measurement tools or environment in which results were obtained. In order to enable independent verification of results, details on the development platform such as type of device, device settings, drivers, driver settings, and version numbers of used software are essential.

While sometimes omitted in research papers, it may be very interesting to report on problems that were encountered during the implementation. These practical experiences are helpful in two ways: first, they may help other researchers to avoid losing time on similar issues, and thus help to advance the state of the art more quickly. Second, in case fundamental issues are discovered during the implementation or analysis, new research topics might be identified, which eventually may lead to a new generation of wireless networks.

Development tools, platforms or architectures that were developed as a by-product during the research process may lead to additional reports by being described as topics on their own. Furthermore, measurement results can and should be used as input to increase the reliability of simulation models. Finally, sharing the source code of a developed solution is worth considering, as it *(i)* enables independent verification of a solution more easily, *(ii)* might lead to valuable feedback from peer groups, and *(iii)* increases the chances of a designed solution being used as a reference case during future research alternatives. Eventually, a successful experiment might lead to end-users benefiting from the designed solution.

For the example case, all relevant details were already discussed in the previous paragraphs.

5.4 Conclusion

Many research approaches for the design and evaluation of wireless networking protocols exist. In this chapter, an in-depth overview was given of different performance analysis techniques that may be used by wireless networking researchers today, each with their own advantages and disadvantages. There is no single answer to which development technique is best; when investigating the theoretical gains of a certain concept under idealized assumptions, mathematical models or well-considered simulations are likely to be the best choice, as controlled parameters can easily be varied and measured, and the influence of unpredictable behavior of RF signal propagation, wireless hardware and wireless drivers may be eliminated.

However, the fact that even after years of research, the simplest wireless ad-hoc scenarios are hard to realize in real-life, proves that the actual state of the art may not be moving forward as fast as it sometimes appears. Within the international wireless research community, there is an increasing number of cautionary perspectives on studying wireless ad-hoc networks purely by theory or simulations, and the use of experimental methods is gaining attention. However, over-enthusiastic use of experimental methods may be causing as much confusion and wrongly interpreted results as thoughtless simulations. As experimental evaluation is gaining momentum, now is the time to launch a similar warning on ill-considered experimental evaluation techniques.

In contrast to what may be believed, using experiments as an evaluation technique is not an easy alternative to extensive simulation studies. In fact, theory, simulations and experiments are complimentary in achieving a research goal. When using experimentally driven research as a way to determine feasibility and performance of a wireless protocol or system, there is typically more effort required than when only considering theory or simulations. While a lot can be learned from implementations, sufficient time and resources are needed to produce meaningful results. More specifically, during planning phase, one should reserve sufficient time to tackle issues that are not directly related to the implementation of the protocol or system under tests itself, but to other subsystems of the implementation and to developing data gathering techniques once the implementation is finished. Furthermore, it is important to know in advance how the performance of the solution is going to be measured and whether sufficient time and resources are available to perform large scale tests, in order not to lose precious development time leading to no other results than 'the system does (not) work'.

The implementation platform determines the basic stability of any protocol that is implemented on top of the device, decides which development cycles are possible, and impacts the way that results can be logged during test runs. Therefore, the choice for a specific development platform should not be taken lightly. Moreover, testing the basic performance of any test set-up, small or large, provides invaluable performance data and is a must-do before carrying out any experiment. Failure to grasp the basic performance may lead to continuously failing tests or misinterpretation of test results. Finding the development platform to be causing basic issues that make performance measurements hard or impossible after months of research, is a frustrating experience.

Ideally, development is done in small incremental steps through simulations and real-life experiments in parallel, using the same code base and the target end-user device. This approach assures that unexpected issues are discovered promptly. By comparing the results from simulations and different experimental performance analysis methods, the subsystem causing the specific problem is more easily identified. The ability to visualize operational networks or packet logs through a GUI is a great way to process large amounts of test data more quickly.

Keeping track of as much relevant details and packet logs as possible of the experiments that were carried out assures that during analysis and reporting of the results, meaningful conclusions can be drawn. Furthermore, the collected data may afterwards be used to enhance simulator models for use during future research. Even if the end results are not as positive as expected, a detailed analysis of the events leading to an issue which could not be foreseen through the use of simulations only may help other researchers to avoid making the same mistakes or inspire new research topics.

In order to illustrate the research methodology, an example use case was tackled, in which a deployment extension for the auto-configuration approach from the previous chapter was developed. Based on the previously determined relation between RSSI and maximum physical layer data rate for the chosen development platform, an estimation of the physical layer data rate that can be expected when deploying new mesh nodes at a specific location is presented. During this example implementation, realistic targets were set, and a well-known development platform and well-known development tools were used. Through stepwise implementation and testing, the deployment tool was realized in an efficient way. Moreover, the deployment approach was tested in an office environment and showed to be a feasible way of generating reliable physical data rate predictions for high data rate links. As such, the example implementation has not only an illustrative value but is also a valuable contribution on its own, further simplifying the deployment of reliable wireless mesh networks.

By following the hints and research methodology introduced in this chapter, more reliable research results may be collected from test set-ups with less effort. Solutions that are developed using the methodology are guaranteed to better cope with real wireless environments than solutions which were only designed through simulations. By tackling both theoretical difficulties and practical issues, we believe that the state of the art of wireless ad-hoc networks may truly advance, eventually resulting in wireless ad-hoc networks to be used to their fullest potential.

References

[1] David Cavin, Yoav Sasson, and André Schiper. *On the accuracy of MANET simulators*. In POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing, pages 38–43, New York, NY, USA, 2002. ACM Press.

[2] Todd R. Andel and Alec Yasinsac. *On the credibility of manet simulations*. Computer, 39(7):48–54, 2006.

[3] Stefan Bouckaert, Johan Bergs, Dries Naudts, Erik De Kegel, John Baekelmans, Nik Van den Wijngaert, Chris Blondia, Ingrid Moerman, and Piet Demeester. *A Mobile Crisis Management System for Emergency Services: from Concept to Field Test*. In Proceedings of the WiMeshNets06 workshop, part of the third Intl. Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, Waterloo, Canada, august 2006.

[4] Wim Vandenberghe, Kristof Lamont, Ingrid Moerman, and Piet Demeester. *Connection Management over an Ethernet based Wireless Mesh Network*. In WRECOM, Wireless Rural and Emergency Communications Conference, Rome, Italy, February 2007.

[5] John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun chan Lan, Ya Xu, Wei Ye, Deborah Estrin, and Ramesh Govindan. *Effects of detail in wireless network simulation*, 2001.

[6] Marco Sgroi, Julio Leao da Silva, Fernando De Bernardinis, Fred Burghardt, Alberto Sangiovanni-Vincentelli, and Jan Rabaey. *Designing Wireless Protocols: Methodology and Applications*. In Proceedings of ICASSP'00, June 2000.

[7] Elena Meshkova, Janne Riihijärvi, Frank Oldewurtel, Christine Jardak, and Petri Mähönen. *Service-Oriented Design Methodology for Wireless Sensor Networks: A View through Case Studies*. In SUTC '08: Proceedings of the 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008), pages 146–153, Washington, DC, USA, 2008. IEEE Computer Society.

[8] Erik Nordström, Per Gunningberg, Christian Rohner, and Oskar Wibling. *Evaluating wireless multi-hop networks using a combination of simulation, emulation, and real world experiments*. In MobiEval '07: Proceedings of the 1st international workshop on System evaluation for mobile platforms, pages 29–34, New York, NY, USA, 2007. ACM.

[9] Dai Lu and David Rutledge. *Investigation of indoor radio channels from 2.4 GHz to 24 GHz*. In IEEE Antennas and Propagation Society International Symposium, pages 134–137, Columbus, Ohio, June 2003.

[10] Piyush Gupta and Panganamala Ramana Kumar. *The Capacity of Wireless Networks*. IEEE Transactions on Information Theory, 46(2):388–404, March 2000.

[11] Michael Neufeld, Ashish Jain, and Dirk Grunwald. *Nsclick:: bridging network simulation and deployment*. In MSWiM '02: Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems, pages 74–81, New York, NY, USA, 2002. ACM.

[12] William Kasch, Jon Ward, and Julia Andrusenko. *Wireless network modeling and simulation tools for designers and developers*. Communications Magazine, IEEE, 47(3):120–127, March 2009.

[13] Glomosim. *Global Mobile Information Systems Simulation Library*. http://pcl.cs.ucla.edu/projects/glomosim/.

[14] OPNET Technologies Inc. *System-in-the-Loop module*. http://www.opnet.com/solutions/network_rd/system_in_the_loop.html.

[15] James T. Kaba and Douglas R. Raichle. *Testbed on a desktop: strategies and techniques to support multi-hop MANET routing protocol development*. In MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pages 164–172, New York, NY, USA, 2001. ACM.

[16] Qosmotec. *Air Interface Simulator*. http://www.qosmotec.com/.

[17] Ruben Merz, Harald Schiöberg, and Cigdem Sengul. *Design of a Configurable Wireless Network Testbed with Live Traffic*. In Proceedings of the 6th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, May 2010.

[18] David A. Maltz, Josh Broch, and David B. Johnson. *Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed*, 1999.

[19] Sanjit Krishnan Kaul, Marco Gruteser, and Ivan Seskar. *Creating wireless multi-hop topologies on space-constrained indoor testbeds through noise injection*. In TRIDENTCOM 2006. 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006.

[20] European Commission. *ICT Work Programme 2009-10 for ICT research in FP7*. http://cordis.europa.eu/fp7/ict/, July 2009.

[21] Erik Nordström, Per Gunningberg, and Henrik Lundgren. *A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks*. In Tridentcom 2005. First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities., pages 100–109, Feb. 2005.

[22] Erik Nordstrm, Henrik Lundgren, David Lundberg, Christian Tschudin, and Per Gunningberg. *Ad hoc Protocol Evaluation testbed*. http://apetestbed.sourceforge.net/.

[23] Agile Alliance. *Home Page*. http://www.agilealliance.org/.

[24] Craig Larman and Victor R. Basili. *Iterative and Incremental Development: A Brief History*. Computer, 36(6):47–56, 2003.

[25] Ian Sommerville. *Software process models*. ACM Computing Surveys, 28(1):269–271, 1996.

[26] Joshua Robinson, Konstantina Papagiannaki, Christophe Diot, Xingang Guo, and Lakshman Krishnamurthy. *Experimenting with a Multi-Radio Mesh Networking Testbed.* In WiNMee - 1st workshop on Wireless Network Measurements, Riva Del Garda, Italy, January 2005.

[27] Arindam K. Das, Hamed M. K. Alazemi, Rajiv Vijayakumar, and Sumit Roy. *Optimization models for fixed channel assignment in wireless mesh networks with multiple radios*. In In SECON, pages 463–474, 2005.

[28] Punky Tse *et al. Voyage Linux home page*. http://linux.voyage.hk/.

[29] Andy Boyett, Nicolas Hill *et al. OpenWrt home page*. http://openwrt.org/.

[30] PC Engines. *Alix system board*. http://www.pcengines.ch/alix.htm.

[31] Madwifi. *Multiband Atheros Driver for Wifi*. http://madwifi.org/.

[32] Click. *The Click Modular Router Project*. http://pdos.csail.mit.edu/click/.

[33] Gurpal Singh, Ajay Pal Singh Atwal, and B. S. Sohi. *Effect of background scan on performance of neighbouring channels in 802.11 networks*. Int. J. Commun. Netw. Distrib. Syst., 1(1):19–32, 2008.

[34] Ilenia Tinnirello, Domenico Giustiniano, Luca Scalia, and Giuseppe Bianchi. *On the side-effects of proprietary solutions for fading and interference mitigation in IEEE 802.11b/g outdoor links*. Computer Networks, 53(2):141–152, 2009.

[35] Benoît Latré, Bart Braem, Chris Blondia, Ingrid Moerman, and Piet Demeester. *A Survey on Wireless Body Area Networks*. Accepted for publication in Wireless Networks, 2010.

[36] Ramakrishna Gummadi, David Wetherall, Ben Greenstein, and Srinivasan Seshan. *Understanding and mitigating the impact of RF interference on 802.11 networks*. In SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, pages 385–396, New York, NY, USA, 2007. ACM.

[37] Shie Yuan Wang and Hsiang-tsung Kung. *A new methodology for easily constructing extensible and high-fidelity TCP/IP network simulators*. Comput. Netw., 40(2):257–278, 2002.

[38] Daniele Puccinelli and Martin Haenggi. *Multipath fading in wireless sensor networks: measurements and interpretation*. In IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing, pages 1039–1044, New York, NY, USA, 2006. ACM.

[39] Young-Bae Ko and Nitin H. Vaidya. *Location-aided routing (LAR) in mobile ad hoc networks*. Wireless Networks, 6(4):307–321, 2000.

[40] Srdan Capkun, Maher Hamdi, and Jean-Pierre Hubaux. *GPS-free positioning in mobile Ad-Hoc networks*. In Cluster Computing, pages 3481–3490, 2001.

[41] Jangeun Jun, Pushkin Peddabachagari, and Mihail Sichitiu. *Theoretical maximum throughput of IEEE 802.11 and its applications*. Second IEEE International Symposium on Network Computing and Applications, pages 249–256, 2003.

[42] IEEE Standard. *IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*.

6 Overall Conclusion

Wireless communication networks play an important role in our current technological society. Mobile handheld devices and laptops are getting more powerful by the day and are able to operate on batteries for an ever lengthening period of time. Today, wireless cellular based technologies are used everywhere in the world, and wireless Wi-Fi access points are omnipresent. These technologies have revolutionized the way in which communication is experienced, by extending the traditional wired infrastructure with a single-hop wireless link.

As a fully wireless alternative to the above technologies, wireless ad-hoc networks have been studied for years. Wireless ad-hoc networks are defined as self-organizing autonomous networks between wireless nodes that may act both as clients and routers. In wireless multi-hop ad-hoc networks, information is sent from source node to destination node either directly, or via a wireless multi-hop path. Due to their unique characteristics, wireless ad-hoc networks have been called the ideal solution to provide communication in environments without any existing, accessible or reliable communication infrastructure.

Plenty of potential use cases exist: ad-hoc networks may be used in homes and the offices when the installation of cables is unwanted or too expensive, may quickly replace a devastated communication network infrastructure after nature disasters such as earthquakes strike, may be used as a temporary network to distribute documents during meetings or in classrooms, may answer to the connectivity needs in rural areas and third world countries, or may be used as a way to extend the coverage of existing network infrastructure.

Triggered by these interesting applications and the widespread availability of devices and wireless technologies that are theoretically able to support wireless ad-hoc networks, the international research community has been very active in proposing new and updated protocols and solutions at all layers of the OSI network stack. With clock-like regularity, new protocols and algorithms are reported upon.

Despite these research efforts, the promising applications and the claimed benefits, it was observed that wireless multi-hop ad-hoc networks are hardly used in our everyday environment. The question surfaced as to what factors are causing this remarkable discrepancy, and what could be done to move wireless ad-hoc networks forward from a research phase to being actually used in an operational environment.

The analysis in this dissertation indicated that both technical and non-technical aspects are at the root of the limited popularity of wireless ad-hoc networks: many ad-hoc protocols have been designed with the traditional OSI layer approach in mind, which was shown to be a suboptimal approach. Moreover, the physical layer of wireless networks is less reliable and less powerful in terms of throughput and delay characteristics than the physical layer of its wired counterparts, and the scalability of single-interface multi-hop wireless ad-hoc networks is fundamentally limited due to interference between different wireless links sharing the wireless medium. More importantly, many researchers approach wireless ad-hoc networks from a purely theoretical angle, or analyze their solutions based on simulations only. Theory and simulations are valuable tools while designing and analyzing wireless ad-hoc protocols. However, when protocol analysis is done based on oversimplified wireless networking models only, results are easily misinterpreted and may lead to challenges that are deemed solved because they were tackled theoretically.

Through experiments in small-scale test setups based on a selection of commercial off-the-shelf Wi-Fi hardware, popular basic generalized theoretical assumptions were challenged. The results of these experiments indicated that these assumptions do not necessarily hold true when deploying Wi-Fi based ad-hoc networks; wireless links cannot be modeled by a simple on/off state based on distance between sender and receiver, as there is a complex relation between received signal strength, packet loss, and physical layer data rates. While simple on/off models might be justified for evaluating some aspects of a protocol, they should not be used to make general statements on the performance of a protocol in an arbitrary environment. A fitted shadowing model showed to provide a reasonable estimation of the packet loss, although it cannot always explain variations in signal strength existing in a realistic environment.

Furthermore, for these test set-ups, it was shown that while output power adaption of wireless interfaces is often considered as a measure of freedom, in reality it is a simple necessity in order to guarantee stable wireless links. More specifically, the theoretical benefit of using small-scale multi-interface, multi-hop ad-hoc nodes as a solution to increase the capacity in a wireless network was shown to be practically unfeasible, unless the wireless interfaces are set to use a low output transmission power. Experiments demonstrated the positive effect of increasing the antenna separation between different interfaces of a multi-interface node, but at the same time suggest that the use of small-scale multi-interface wireless ad-hoc nodes based on current generation off-the-shelf Wi-Fi hardware may currently be unfeasible. A final set of measurements revealed that there are important differences in quality and stability between hardware of different vendors within the same price range. Even if an individual end-user tries to avoid some of the hardware related issues by buying hardware of better quality, he or she is still not necessarily guaranteed of a high quality wireless link if low-quality nodes participate in the network.

These experimental observations led to a somewhat pessimistic view on the ability of traditional wireless ad-hoc networks and many of the developed protocols to be used on top of Wi-Fi hardware in large scale scenarios. Therefore, the need for a wireless ad-hoc architecture founded on realistic assumptions and expectations was expressed. This resulted in a definition of a hierarchical dynamic wireless mesh network architecture which was shown to relax several requirements of traditional ad-hoc networks. In wireless mesh networks, powerful specialized nodes automatically form a wireless backbone for wireless end-user clients. The clients can connect to the backbone through wireless access points which are connected to the mesh nodes. This way, the quality of the wireless network as observed by the client devices depends largely on the quality of the wireless mesh network. Moreover, in the short term and without the power to automatically force updates at a large number of client devices, it is more realistic to develop algorithms to be deployed on a limited number of high quality nodes and offer backward compatibility with existing standardized end-user technologies, than to expect every end-user to install newly developed non-standardized protocols.

A loosely coupled cross-layer design based on so-called glue parameters was proposed as a way of realizing the interconnection between the different building blocks of the wireless mesh node architecture. The flexible architecture allows the different building blocks to work together in an independent and flexible way, without imposing strong limitations on the designed algorithms: cross-layer interactions are considered as a means, not as a goal. Both the building blocks and the cross-layer architecture were inspired by a bottom-up design: the practical issues that were observed during the experiments were taken into account from the start. This way, the risk of spending a large amount of time in designing algorithms that are doomed to fail when deployed in reality was avoided.

In a second part of this book, multiple building blocks of the mesh architecture were developed. In order to guarantee the practical feasibility of these protocols, theoretical considerations and simulations were each time complemented with a real-life implementation. Because IEEE 802.11a/b/g compatible devices are popular and cheap, and operate in unlicensed wireless spectrum bands, they were chosen as demonstration technology for the implementation. However, with minor modifications, the designed algorithms may be used on top of other wireless technologies.

First, in order to overcome the fundamental limits of single interfaced wireless nodes, a fast channel selection protocol for multi-interface, multi-channel mesh nodes based on message exchange called FRESME was designed, implemented, and analyzed. For the specific case of IEEE 802.11g based Wi-Fi networks with three interfaces, the analysis showed that the protocol is able to configure channels on a per link, on demand basis in a fully distributed way using a hybrid control channel approach. A new channel reservation is completed in less than 4 milliseconds using only 3 control packets. The control packets contain Channel Quality Parameters (CQPs). These dimensionless metrics were created as an abstract notion of channel quality and are used to get a view on the observed channel quality at both sides of a wireless link, hereby avoiding decisions that are imposed by any of the two communicating nodes. It was indicated how different background processes may each modify the CQPs sequentially, in order to enhance the channel selection protocol without requiring changes to the core algorithms.

Furthermore, the FRESME protocol does not rely on long-term measurements, does not require periodic broadcast of control messages, and can cooperate with a broad range of routing protocols as it is hidden from the routing layer by a multiplexer/demultiplexer component. The efficiency and feasibility of the protocol was demonstrated through an implementation which was analyzed in a small-scale wireless testbed and through a large scale simulation. In order to judge the quality of the channel distribution throughout the network, new so-called unbalance metrics were defined. It was shown that in a raster topology, globally, the channels reserved for data communication are assigned an equal number of active wireless links. Locally, the link to data channel mapping is optimal at 86.1% of the nodes. While small-scale experiments in an IEEE 802.11g testbed proved the value of the protocol, several suggestions were made to could provide further refinements. They were left as future work.

Second, an end-to-end throughput capacity estimation protocol for use in multi-channel, multi-interface networks was developed based on the packet pair probing technique. In contrast with related work, packet pair probing is not used on an end-to-end basis nor does it only provide details on the capacity of an individual link. Instead, the capacity of each wireless link is determined individually; then, the link capacity estimations and the channel settings of the network are disseminated by piggybacking information onto the routing protocol messages. It was proved that each node is able to estimate the throughput capacity of any end-to-end path in the network based on the estimations collected from each individual link. Small-scale tests indicated the feasibility of the technique, and showed that the estimated throughput capacity of the test links was similar to the capacity measurements performed using the iperf bandwidth measurement tool. While the measurement tool consumed all bandwidth in the network while measuring the capacity, the overhead of the developed estimation technique was minimal. Although these initial results were promising, additional measurements are necessary to evaluate the protocol in larger test set-ups while varying topologies, background noise levels and the amount of background traffic generated in the network. These measurements are left for future research.

While these first two protocols operate automatically after a one-time manual configuration, manual configuration of each device was experienced to be tedious and error-prone, and cannot be done without having a good knowledge of wireless networks. Furthermore, if an error occurs in the network, for example, a wireless mesh backbone node dies, finding the faulty node without proper monitoring tools proved to be difficult. Performing a manual configuration might not be an insurmountable problem for companies or public services which have system administrators to perform this task. However, tedious installation procedures may keep the general public from using wireless mesh products.

Therefore, a third development concerned an integrated auto-configuration solution enabling automatic deployment, expansion and management of wireless mesh networks. More precisely, an auto-configuration method was developed that enables out of the box configuration of new wireless devices, which are then fully automatically integrated into a wireless mesh network. By re-using certificate based encryption techniques available from literature, the network is fully secured. No strict security relationship with the pre-existing network is assumed prior to node deployment, and new nodes can be added anywhere within the coverage of an existing network without requiring physical access to the device and with minimal administrator intervention.

Again, the presented mechanism was not limited to a theoretical study: all subsystems such as scanning, neighbor discovery, node initialization, link security, routing and reliability mechanisms are implemented and integrated, allowing a real-life large-scale operational multi-hop mesh network providing secure data transport to be set up in minutes. The solution allows a network to be built out of heterogeneous commodity hardware provided by different vendors. Even though the mechanism was primarily developed for simplifying wireless network set-ups, the mechanisms works equally well on top of wired interfaces, or even on devices having both wired and wireless interfaces, seamlessly and securely interconnecting networks.

Wireless testbed experiments showed that a wireless mesh network with over twenty nodes was fully configured from scratch in less than two minutes, administrator approval time excluded. In order to enable the administrator to verify the identity of nodes that are added to the network, a basic GUI was developed which allows the addition of new nodes through a unified interface. For network administrators that want to get more details about the network, a more advanced version of the GUI was developed which allows monitoring and managing settings in the network. Finally, it was indicated how external management tools can easily interface with the mesh network, and how future extensions to the auto-configuration approach could combine planning, deployment and detailed management, with automatic wireless network optimizations.

The performance analysis of the above protocols and the ability to deploy the developed solutions on current generation off-the-shelf hardware have proved the successfulness of the followed research approach, which combines theoretical considerations with well-chosen simulations, practical implementations and observations. However, during the realization of this work, it was found that there are also many pitfalls to using implementations and real-life experiments as a performance evaluation method. Therefore, in the last part of this work, a methodology for wireless network research using real-life implementation was proposed, which aims to allow researchers to generate more reliable protocols and performance analysis results with less effort. Such methodology will become increasingly important in the future, as the interest of the wireless research community in experimental evaluation methods is steadily growing. If mistakes that have been made while simulating wireless networking protocols in the past are to be avoided when adopting an experimentally driven approach in the near future, now is the time to define guidelines that will help researchers to produce high quality wireless networking protocols and performance results.

To this end, first, an in-depth overview of seven wireless performance analysis techniques was given. Even though mathematical and statistical models may fail to capture the full complexity of the wireless networking environment, they are extremely useful to study trends in subsystems and to determine upper and lower borders of performance metrics. Simulations are often the only option to evaluate wireless solutions in large topologies and allow easy variation of configuration parameters. Furthermore, performance metrics and debugging data are easily and unambiguously retrieved. However, simulators should not be used unless one is fully aware of which aspects are modeled in the simulator, and which are not. The fact that simulations often do not account for unpredictable events caused by the unstable wireless medium and the specific behavior of wireless hardware and wireless drivers makes that they cannot be used to make absolute claims about the general performance of a wireless protocol. Nevertheless, imperfect simulation models may be used to an advantage in order to test the basic functionality of wireless ad-hoc algorithms.

In order to test the feasibility of an approach under less idealized conditions, the need for performance analysis using real hardware devices was expressed. In its simplest form, a real wireless device may be used while fully emulating the wireless network card and wireless medium. This way, the application and routing layer of the wireless device are real and semi-realistic interaction between test persons and the device is possible. Moreover, as a real device is used, the protocol developer is confronted with the limitations of the device such as limited memory or limited processing power; however, extreme limitations are better avoided during developments that are not especially targeted at low-end devices. In order to take the influence of the wireless driver and wireless MAC into account, while still retaining full control over topology and interference caused by third-party RF sources, a fully realistic device and wireless network card can be used while guiding the wireless signal over a coaxial cable. Within limits, different topologies and signal degradations can be reconstructed by combining (automated) RF splitters, RF combiners and phase shifters. The method was found to be especially suited for recreating basic multi-hop topologies and reproducible signal level variations.

Testbed deployments allow to verify the robustness of a wireless protocol against a broad range of events that are hard to model such as link quality variations, intermittent connectivity, bad antenna connections, unidirectional links, interference from third party wireless networks, or a varying environment caused by people moving or furniture being reorganized. A wireless testbed is more easily controlled and operated if the testbed node locations are fixed. It was indicated how, even with fixed node locations, topology modifications in the testbed are possible. Among several example topology control strategies such as MAC address filtering or power adaptation, the installation of RF signal attenuators between the wireless interface and wireless antenna was argued to be a solution causing few side effects. This solution was also implemented on the w-iLab.t testbed and showed to reduce the perceived node density in the network. Finally, a solution may be tested outside a laboratory environment, either by the researchers or by the intended end-users of the solution. While the last analysis technique is burdened by practical issues caused by the lack of control on the environment, making it harder to log and reproduce issues that were encountered.

The methodology itself was organized in five steps: project preparation, platform preparation, development, analysis and reporting. It was argued that an experimentally driven approach is in no way an easy alternative to extensive simulations or theoretical studies: it are complimentary tools which can be combined to reveal the source of hidden problems. If implementation is considered as a way to determine the feasibility and performance of developed solutions, more effort is typically required than when only focusing on theory or simulations. It was stated that a lot of time is inevitably spent to issues not directly related to the implementation or system under test itself.

Thanks to a good advance planning, which takes possible performance measurements into consideration from the start, many implementation difficulties are anticipated and avoided. As the choice for a specific implementation *(i)* determines the basic stability that can be expected from any protocol implemented on top of the device, *(ii)* decides which development cycle should be followed and *(iii)* has an impact on how results can be logged during experiments, it is a good idea to thoroughly study different device options before starting implementation. Moreover, testing the basic performance of any test set-up is a must before performing any actual experiment. Failure to understand the basic behavior and performance of a test set-up may lead to continuously failing tests or serious misinterpretation of test results.

By organizing development in incremental steps using parallel simulations and real-life experiments, unexpected issues are discovered promptly. A graphical representation generally allows faster processing and interpretation of test results. In order to ensure that meaningful conclusions can be drawn during analysis and reporting of the test results, logging *all* relevant test parameters is essential. If, in spite of all development efforts, the test results are not as good as hoped for, this should be no reason not to publish results and share experiences: other researchers might be able to avoid the same mistakes, or –provided a detailed analysis of the effects leading to the observed problems is available– might find a solution or workaround. Third party analysis and verification is greatly simplified by making the source code of developed solutions publicly available.

Finally, the methodology was illustrated through the implementation and evaluation of a deployment tool assisting network administrators during the physical installation of mesh networking nodes. The solution was developed as an extension to the previously developed auto-configuration and deployment mechanism. Apart from its illustrative value, the implementation showed the feasibility of using RSSI measurements retrieved from the wireless driver to estimate the physical layer connectivity that may be expected after deploying a node on a given location. With the tool, the quality of the placement location for wireless mesh nodes can easily be assessed even before taking part in the network. As such, the tool was found to simplify the deployment of wireless mesh networks.

With the above conclusions and realizations in mind, should one expect wireless mesh networks and wireless ad-hoc networks to be frequently used in the near future? Obviously, there is no simple yes or no answer to this question, as the future of wireless ad-hoc networks depends on several factors.

A first factor is the practical feasibility of current generation wireless mesh and wireless ad-hoc networks. Using the developments from this dissertation, deploying and maintaining a Wi-Fi based wireless mesh network to provide connectivity to equipment around the house and office or during temporary events, is believed to be a feasible option today. Within minutes, network coverage may be installed throughout any house, office building or event hall, providing sufficient communication bandwidth for many applications. For life-critical communication, operating wireless mesh networks in the unlicensed spectrum is currently less feasible, because the wireless medium cannot be controlled and no hard guarantees to the service quality can be given. However, there is no reason why the developed techniques could not be re-used in a spectrum band especially reserved for public services.

Second, from a technological point of view, WLAN technologies are continuously evolving. Thanks to MIMO technologies, data rates and communication ranges are increasing, and the physical layer is expected to better cope with varying link qualities. As such, more stable yet more complex physical layer technologies are expected. However, successful ad-hoc networks will only exist if the physical layer and upper networking layers work together as efficiently as possible. This will only be possible if researchers cooperate and share experiences across research domains covering the entire OSI network stack.

Third, the future of wireless ad-hoc and mesh networks is influenced by future connectivity needs. The rise of wireless networks is likely to continue. In the near future, an increasing number of devices is expected to depend on a connection to the Internet. For example, printed newspapers might gradually disappear as e-paper readers gain popularity. The refrigerator may communicate with a chip attached to a milk bottle to detect if the milk is still fresh or if the bottle is almost empty. This information may be used to update an on-line shopping list or directly place an order with the supermarket. Elderly people may use wearable sensor devices to ensure a continuous follow-up of vital parameters, and automatically call for help when needed. Portable handheld devices may contact a wide range of powerful services which are located ‘in the cloud’, using computing resources potentially located at the other side of the world. In all of the above scenarios, a wireless connection is the preferred way to provide connectivity to the Internet. Wireless ad-hoc networks might be an ideal candidate to support the connectivity needs, although, especially for handheld devices, strong competition is expected from advanced cellular technologies rolled out by network operators. To some operators, wireless mesh networks operating in unlicensed spectrum are considered as a threat, since there is a chance of losing market share to end-user driven communities providing connectivity services at reduced or no cost. Although the latest generation of wireless cellular technologies provide a relatively high amount of communication bandwidth, in the future, this speed will still have to increase. At the same time, a lot more users are expected to use mobile broadband connections. In order to let the cellular technologies scale with the increasing number of users and increasing data requirements, cellular communications will eventually be driven to higher communication frequencies, requiring a lot more antennas and creating smaller cells. In these cases, multi-hop wireless mesh network technology might prove to be an economically more feasible alternative to installing cables between a dense set-up of antennas. As such, from the application perspective, wireless ad-hoc networks have a bright future ahead.

Fourth, the increasing number of devices and users, and the need for faster connections will undoubtedly lead to an even more crowded wireless spectrum than today. Spectrum limitations may be reduced to some extent by international decisions of the policy makers. However, the part of wireless spectrum usable for RF communication will always be limited. As such, one of the main challenges in the future will be to organize the available wireless spectrum as efficiently as possible through the use of cognitive radio strategies. The resulting networks will be large and will be formed out of devices with heterogeneous capabilities. Both the efficient use of cognitive strategies as the integration of low-end devices provide challenges for the future.

By designing a practically feasible, auto-configuring wireless mesh network, this dissertation contributed to enabling the use of wireless ad-hoc network technology today. In order to tackle future challenges in an ever more complex wireless network environment, not only in theory but also in practice, it will become increasingly important to complement theoretical research with a well-planned experimentally driven approach. To enable large scale experiments and independent verification and comparison of international research results, large scale open testbed infrastructures are needed. The design of flexible, reliable and user-friendly wireless testbeds is an interesting topic for future research.

Although the developed protocols and research methodology provide no definitive answers to all ad-hoc networking issues, it is hoped that they serve as an inspiration to other researchers. The initial wireless mesh deployments may directly be used to develop and test a new generation of applications, further emphasizing the usefulness of wireless ad-hoc technology. In their turn, new applications may increase the interest in the design and realization of practically feasible wireless protocols and systems. The interaction between these developments may eventually result in wireless ad-hoc networks being used anytime and anyplace in order to support our daily activities.

A
The w-iLab.t testbed

Stefan Bouckaert, Wim Vandenberghe, Bart Jooris, Ingrid Moerman, Piet Demeester

To be published in the Lecture Notes of ICST, ICST-Springer, May 2010.

Abstract *In this paper, the W-iLab.t wireless testbed is presented. The testbed consists of nearly 200 sensor nodes and an equal amount of Wi-Fi nodes, which are installed across three floors of an office building. The testbed supports wireless sensor experiments, Wi-Fi based mesh and ad-hoc experiments, and mixed sensor/Wi-Fi experiments. It is explained how changes in the environment of the sensor nodes can be emulated and how experiments with heterogeneous wireless nodes are enabled. Additional features of the testbed are listed and lessons learned are presented that will help researchers to construct their own testbed infrastructure or add functionality to an existing testbed. Finally, it is argued that deep analysis of unexpected testbed behavior is key to understanding the dynamics of wireless network deployments.*

A.1 Introduction

As a research group, frequently involved in interdisciplinary projects with industrial partners, validation of developed algorithms and protocols for wireless ad-hoc, sensor and mesh networks on actual (prototype) hardware has been an important way of proving validity of theoretical and simulated novel concepts, and demonstrating the feasibility of network architectures [1, 2]. Very often, our wireless experiments revealed minor or major flaws in theoretical assumptions [3], requiring time intensive debugging sessions and algorithm modifications that would not have been required if simulation results were the final product of our research.

Over the years, multiple different small-scale wireless sensor and wireless mesh testbeds were set up and torn down in the scope of various projects, master theses and doctoral theses. While a lot of lessons were learned from these experiments on diverse types of hardware, there are also several drawbacks associated with the deployment of multiple individual testbeds. *(i)* Buying new hardware set-ups for every project is costly, and therefore limits the deployment scale. *(ii)* Different hardware architectures require different development approaches. As an example, in the case of IEEE 802.11 based mesh and ad-hoc research, experiments have been performed using off-the-shelf Wi-Fi routers with custom built firmware, custom built multi-interface mesh nodes, PDAs, tablets, laptops and desktop computers with various wireless NICs, and integrated system boards. While experience with diverse network platforms is gained, there is a substantial overhead associated with creating new development environments. *(iii)* Results obtained from different test set-ups cannot easily be compared. *(iv)* Rebuilding old test set-ups is time-consuming and has a negative impact on the reproducibility of test results.

In order to overcome the drawbacks of these individual test set-ups and to enable wireless tests on a larger scale, the w-iLab.t testbed was designed and installed at the buildings of the IBCN research group and IBBT research institute in Ghent, Belgium. The w-iLab.t inherited its name from the larger IBBT iLab.t [4] test infrastructure, where the testbed is a part from. The w-iLab.t testbed consists of nearly 200 sensor nodes and an equal amount of Wi-Fi nodes, which are mounted to the ceilings in the offices and hallways. Although the primary focus of the testbed is to support large scale wireless sensor and actuator network deployments, the testbed architecture supports Wi-Fi mesh and ad-hoc test, and mixed sensor/ad-hoc experiments as well. In the remainder of this paper, the w-iLab.t testbed is presented. The design choices are motivated and the possibilities are demonstrated. Furthermore, we present lessons learned which can help testbed designers to analyze behavior of their own testbed set-up, inspire testbed administrators to add time saving functional blocks to their set-up, or act as a guideline during the initial design phase of a new testbed.

A.2 Goals and Requirements

One of the major drivers to perform real-life experiments, is the fact that a purely mathematical or simulation based approach for designing wireless network solutions is not entirely representative for the real-life performance of the same solutions when deployed in realistic environments. The reason for this discrepancy is a result of simplified traffic pattern and end-user models, wrong assumptions about signal propagation and interference, interactions with other (wired or wireless) network devices and errors introduced by hardware and wireless drivers. While the latter errors should not be solved by upper layer network designers in theory, the success and applicability of a developed algorithm depends on the algorithm's ability to cope with unpredictable behavior introduced by any of the above elements.

Through careful simulations and well designed small-scale testbeds, networking algorithms and protocols can efficiently be debugged to a certain extent. Multihop environments can be emulated on a desktop by interconnecting a small number of wireless nodes through coaxial connections, RF splitters and RF attenuators [5], without the need for a large test infrastructure. However, even with the most advanced simulation models or desktop testbeds it is hard to represent a real networking environment, especially when it comes to simulating interaction with user-level programs and operating systems, evaluating network scanning techniques and channel selection mechanisms, or when modeling dynamic network environments with moving users and external interference. Additionally, measuring user satisfaction and quality of experience is only possible with large-scale testbeds deployed in a realistic environment. Therefore, similar to [6] and [7], it was chosen to install the testbed in an office environment across three $18m$ by $90m$ office floors and thus create a natural network topology. On top of this default topology, additional topology control measures (cf. Section A.4.2) can be taken to vary the perceived node density in the testbed.

In addition to allowing experiments in a realistic office setting, multiple technical and practical requirements were set before designing the testbed:

- Future network environments are expected to be increasingly heterogeneous. Therefore, the testbed should support tests with wireless sensor and actuator nodes, Wi-Fi based mesh and ad-hoc nodes, and mixed scenarios. Since sensor nodes are continuously evolving, it should be possible to easily replace or install additional sensor nodes at the test locations.

- It should be possible to install new software to any sensor or mesh/ad-hoc node from a remote location, and to reboot the nodes remotely in case of node failure. The nodes are preferably powered by external power sources, as to avoid the frequent replacement of batteries.

- Sensor nodes react to environmental changes. Testing protocols that depend on environmental changes is not easily done with current testbeds, as, for example, it is not very convenient to test the reaction of a protocol detecting fire through a fast rise in temperature by holding a flame close to a temperature sensor. Therefore, the testbed infrastructure should be able to emulate environmental changes instead of relying on manual interventions, without

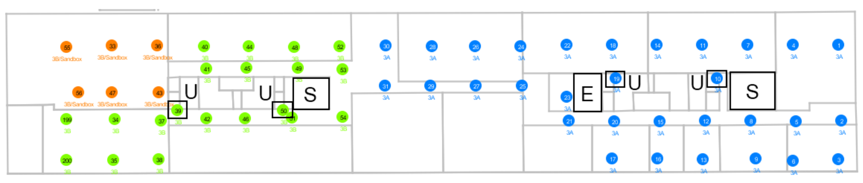


Figure A.1: Third floor of the testbed location. Office area is approximately 90m x 18m. S: staircase. E: elevator. U: utility shaft.

necessarily requiring specific simulation protocols to be compiled with the software under test.

- Researchers should be able to use the testbed from any location. Personalized log in is needed to provide access control and to guarantee a fair share of access to the testbed for each user.

- Advanced logging functionalities are needed, both for wireless sensor network experiments and wireless mesh and ad-hoc experiments.

- Deploying the network devices at the test locations must be as fast and simple as possible, requiring the least possible number of cables to be installed in the offices, reducing the installation cost and minimizing damage to the building.

In the next section, it is explained how the w-iLab.t architecture is able to fulfill all of the above requirements.

A.3 Testbed architecture

A.3.1 Node location

The testbed node locations are distributed across thee similar floors of office space. Figure A.1 shows the location of the nodes on the third floor. Nodes are mounted near the ceiling of both hallways and individual offices which are separated by thermal insulated wooden walls causing little RF attenuation. Several other interesting construction elements are indicated on the floor plan: the elevator and elevator shaft are indicated by E and cause severe RF attenuation. Staircases enclosed in concrete walls (S) and concrete utility shafts (U) which run across the different floors cause an increased RF loss as well. Since the office ceiling is made of metal rasters and the floors of aluminum tiles, there is a large inter-floor signal attenuation inside the building. Therefore, it was chosen to deploy nodes in the utility shafts at every floor, thus constructing inter-floor paths with low signal attenuation.

A.3.2 Hardware components and initial testbed installation

The TMote Sky sensor mote, which is used as the primary type of sensor device in our testbed, is programmed through a USB interface. Since USB technology is not designed to support cable lengths longer than 3 to 5 meters without intermediate USB hubs, a large sensor network cannot be deployed in an office environment using USB cables only. In contrast, Ethernet technology allows longer cable lengths, but is not commonly supported on sensor nodes.

The chosen solution for our testbed was to deploy cheap embedded Voyage Linux operated Alix 3c3 system boards [8] at all node locations. These embedded system boards, which we call *iNodes*, are equipped with an Ethernet NIC, a serial port, VGA output, compact flash storage, onboard audio, two mini-PCI slots and two USB ports. Using the iNodes as relay devices allows the sensor nodes to be programmed remotely.

Two additional advantages are associated with the use of these iNodes: *(i)* by installing two mini-PCI 802.11a/b/g compatible atheros based wireless NICs and adding dual band antennas, the control hardware for the sensor tests can be used as test equipment for Wi-Fi based mesh and ad-hoc tests. *(ii)* The power consumption of the iNodes is low: each iNode consumes only 6.5W in idle state, rising to 7.8W if both Wi-Fi interfaces are enabled and continuously transmitting with a processor load of 100%. The low power consumption allows the iNodes to be powered using only power over Ethernet (PoE). As such, only a single ethernet cable and a PoE converter per node are needed to power the iNodes and connect them to a central administration server. This reduces installation complexity and cost, and allows for remote power switching of the iNodes, and by extension, sensor nodes.

In order to emulate changes to the physical environment of the node, an in-house designed circuit board called *environment emulator* (EE) is added between the USB port of the sensor device under test (DUT) and the iNode. The EE is built around a micro controller, a three port USB hub, and a voltage regulator/measurement chip. It is plugged into a USB port of the iNode, and is equipped with two additional USB ports. The most important goals of the EE are the following: first, one port is used to connect the DUT, the other port allows to additional EEs to be connected in cascade, thus allowing multiple (heterogeneous) sensor nodes to be tested using the same back-end testbed infrastructure. Second, the EE can replace the USB power from the DUT with its own internal power source. Thus, the EE is able to emulate depleting batteries, energy harvesting power sources and node failures. Third, the power that is consumed by the DUT is measured with a sample frequency of 4kHz, allowing to measure the exact power consumption of any sensor node while executing a certain protocol. Fourth, general purpose digital and analog I/O pins are connected to the DUT, allowing to emulate real time digital and/or analog sensor input via programmable events. Fifth, a seven segment LED display and status LEDs provide additional feedback when e.g. flashing the sensor

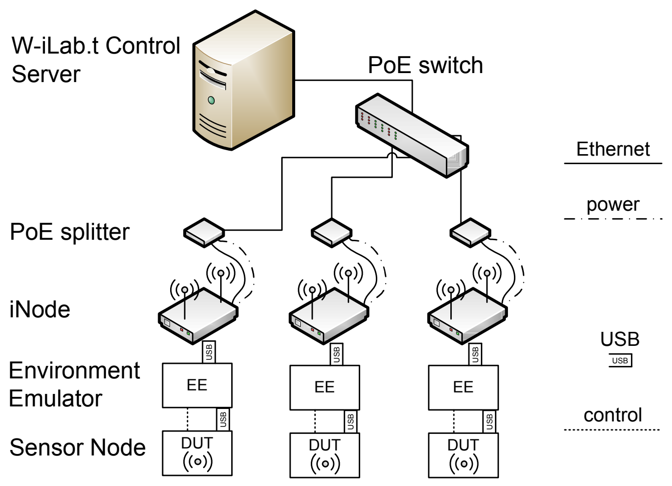


Figure A.2: Logic overview of the w-iLab.t architecture

nodes, writing information to the logs, or during events occurring during normal node operation.

The hardware components of the w-iLab.t testbed architecture are summarized in Figure A.2: the iNodes are powered and connected to a control server through a gigabit PoE switch. Two Wi-Fi cards are installed at the iNodes, allowing to perform Wi-Fi mesh and ad-hoc experiments, and the USB ports of the iNodes are used to connect the sensor node via an environment emulator, which allows advanced testbed manipulation and logging.

A.3.3 Using the testbed

A.3.3.1 Wireless sensor and actuator experiments

The w-iLab.t testbed is accessible by authorized users via a web based interface, which allows users to monitor the testbed status, to upload sensor firmware, to select which nodes will be running what type of firmware during a specific experiment, to schedule an experiment at a specific time for a specific duration, to get an overview of past, current and future tests, and to retrieve results and additional information on completed tests.

The testbed is organized in several geographical zones and sub-zones such as *'third floor'*, *'first half of the third floor'* or *'entire testbed'*. The user can schedule tests in one or multiple zones, or may deploy different code on each individual node. Zone reservations are non blocking, meaning that if one user is running a test on one zone, another user might run a simultaneous test in another, non-overlapping zone. To avoid interference from other experiments, a single user can reserve the whole testbed but only use part of it.

The W-iLab.t control server software is based on the MoteLab [6] software. The software was modified and expanded to support the use of the EE and to allow a more advanced collection and easy representation of test results. Modifications include *(i)* added support for EE scenarios. The user is able to configure events to be triggered at (a selection of) EEs at a user specified time. For example, the user might specify a scenario in which several buttons are pressed at some sensor nodes, while other sensor nodes observe an emulated rise in temperature or fail because of (emulated) battery depletion. The EEs are synchronized and execute the scenario with a maximum error of $100\mu s$. *(ii)* A result processing toolbox, comprising a *sniffer*, *visualizer* and *analyzer* module. Events and sensor node logging information are stored in an SQL database together with the precise timestamps and other test data such as the individual power consumption of the sensor nodes. If the sniffer is enabled, certain sensor nodes are configured as promiscuous nodes and keep a log of all captured frames on a user defined channel. The visualizer and analyzer are universal GUIs allowing both real-time and post-experiment visualization of e.g. packet flows, sensor values or other user measured data, either on a map of the sensor testbed, or by producing a scatter diagram of measured values.

As such, a user is able to easily define tests and emulated scenarios, schedule sensor experiments, and analyze and visualize test results in real-time or after the experiment.

A.3.3.2 Wireless ad-hoc and wireless mesh experiments

As previously stated in Section A.3.2, two Wi-Fi NICs are installed at every iNode. In order to enable mixed Wi-Fi node / sensor node experiments and to keep a uniform interface, it was decided to integrate the support for the Wi-Fi nodes into the same web interface as used for the sensor nodes. Moreover, this fully integrated approach assures that no scheduling conflicts can occur between wireless sensor and wireless mesh experiments. Additionally, when running Wi-Fi experiments, the user should be allowed to operate the devices using a custom Linux kernel, custom drivers and custom application software.

Implementing the above flexibility for Wi-Fi tests might endanger the operation of the sensor testbed: in the default testbed set-up, the iNodes execute a daemon which interprets management information from the central control server, controls the EE and installs the firmware to the DUT. Hence, there could be a certain risk involved in allowing the iNodes to be used for experiments: if a Wi-Fi experiment goes wrong or a user deliberately or unwillingly removes or corrupts crucial files needed for booting the iNode or controlling the sensor nodes, the sensor testbed might become unstable or stop functioning.

These potential issues were avoided as follows. Three subcomponents are required to operate the Wi-Fi testbed: the w.iLab-t central control server acting as a *Preboot Execution Environment* (PXE) server, a user defined *Network File System* (NFS) share, and the iNodes themselves. Two partitions are installed on the iNodes: a first partition holding the original iNode software for controlling sensor experiments, and a second partition used for Wi-Fi experiments only, possibly in combination with a user specified kernel. Whenever an experiment is scheduled, the iNodes reboot using the management functionalities of the PoE switch and contact the PXE server to determine which partition to boot. In case of a Wi-Fi or mixed experiment, the iNode is instructed to load the second partition. The user might specify the location of a custom image using specific kernel located on the NFS share, and also specifies the location of the libraries, binaries and other files or scripts needed to perform the experiment. As a new experiment starts, a user defined start script is executed that e.g. might copy the required files from the share to the iNodes, and/or execute a specific program. Not all nodes need to run the same code, allowing experiments with different node roles.

After the Wi-Fi experiment completes, the iNodes automatically reboot and are instructed to load the first partition. As the first partition is booted again, the second partition is restored to its default state, providing clean iNodes for the next test using Wi-Fi nodes.

Each time a scheduled experiment runs, a logging directory is created on the user defined NFS share. For each iNode in the test, a subfolder is automatically created that uniquely identifies the iNode by its hostname. The respective directories are then mounted to a logging directory on the iNodes. All output that is redirected to this directory on the iNodes is stored on the NFS share. This results in a flexible, fully user specified logging system. Furthermore, as the clocks on iNodes are synchronized through the Precision Time Protocol, logging output can be correlated by adding timestamps to the log messages.

A.4 Additional features and lessons learned

A.4.1 Defining new experiments

The W-iLab.t infrastructure allows fast and easy deployment of newly developed code on a large number of devices. Therefore, it is tempting to not only use the testbed for large-scale deployment of stable algorithms, but also during the development phase for testing incremental adjustments. This results in the testbed not being available for the tests for which it is actually meant, and causes the sensor nodes and/or flash cards of the Wi-Fi node to undergo a large amount of program/erase cycles during a single day, shortening the lifetime of the flash chips in the testbed. Therefore, early development is still performed on isolated small-scale set-ups. Additionally, one zone in the testbed is reserved as a *sandbox* area which is meant for functional testing of new code before switching to another testbed zone. While the use of the ‘normal’ testbed zones is limited by a user-based quota, the sandbox area is not, thus promoting its use.

As for Wi-Fi experiments, it was learned that when no user specified kernel is used, particular care should be taken in keeping the software on the personal testbeds and large-scale testbed synchronized. More specifically, different versions of wireless drivers have shown to cause significant changes to stability and throughput, and result in syntax changes, leading to unexpected results. While obvious, simple driver settings such as disabling antenna diversity when only a single antenna is connected to the wireless NIC are often forgotten but result in considerable stability increases.

Furthermore, it was found that when analyzing a protocol, a researcher often has to create a lot of similar tests, where only a few parameters are changed. For example, in a sensor experiment, one might want to re-run a test on a different transmission power, or change the transmission interval of a certain protocol. Therefore, the option to use parameters in test definitions was added to the testbed: a user might schedule a single test, but with different parameters which are determined at scheduling phase. The system will translate these parameters to individual tests and schedule all of them. This way, a very large amount of test data is collected through a single scheduling action.

A.4.2 Topology control

As previously stated, the w-iLab.t testbed is not located in a separate room but deployed in an office environment. This way, the use of noise injection [9] topology control techniques or attenuators was hoped to be avoided. While this assumption proved to be correct for the sensor network experiments, it was found that it is hard to create topologies with a large number of hops using the Wi-Fi nodes, as their transmission power cannot be set to a value below $0dBm$ due to driver restrictions. After determining the receive sensitivity of the Wi-Fi cards through a measurement campaign using a variable attenuator and modeling the RF propagation characteristics of the office environment, it was decided to add fixed attenuators to all Wi-Fi interfaces of the testbed on the second and third floor of the testbed, with attenuation values of $10dB$ and $20dB$ respectively. The result of this attenuation is a variation of perceived node density at the different floors. Note that the effect of the $10dB$ attenuators on sending and receiving interfaces may be canceled by changing the output transmission power from $0dBm$ to $20dBm$, and that variation of the transmission power of the attenuated nodes allows to emulate environments ranging from sparsely connected (only the direct neighbors are within transmission range) to very densely connected (over 60 nodes in transmission range).

A.4.3 Cautionary perspective on testbed experiments

While new testbed experiments are often characterized by unexpected issues such as protocol failures, node failures or driver errors, it is important to realize that every error happens for a reason. Although this is an obvious observation, authors discussing testbed experiments all too often resort to educated guesses on why a certain error was observed, such as "*we believe that the errors are introduced by the wireless driver*". There are two reasons for these often vague descriptions: first, it takes a huge amount of time to debug all aspects of a testbed deployment, while theoretic calculations and simulations might already be available and are considered to provide adequate proof of an algorithm's or protocol's functionality. Second, the tools to analyze the complex behavior of the testbed might lack.

With respect to the above, some recommendations are the following. *(i)* Test should preferably be run with some nodes acting as a sniffer, since the actual transmitted data is often key to solving problems and better understand the actions (not) taken by the protocol under test. *(ii)* Additionally, even when analyzing upper layer protocols, (basic) knowledge of RF propagation and interference is recommended. *(iii)* Finally, using open source software allows deep analysis of observed behavior.

It should never be forgotten that one of the reasons of using testbeds is to be able to study the behavior of a protocol in a realistic environment. If discovered issues are put aside because the are "*probably* due to X or Y", then the effort of implementing a fully working solution should probably not have been made to begin with.

A.5 Conclusion

The w-iLab.t testbed supports large-scale sensor deployments, Wi-Fi based mesh and ad-hoc tests, and mixed sensor/Wi-Fi experiments, and is therefore able to analyze the behavior of future heterogeneous network deployments. Nearly 200 node locations are available, situated across three floors of an office building. Through an easy-to-use web-based interface, researchers are able to control the deployment of the software to be tested based on network zones or may address individual nodes. Moreover, the environment emulator allows the emulation of sensor network scenarios, provides advanced logging and control, and allows the modular addition of other type of sensor nodes. Test results can be visualized on a map or in graphs in real-time or after the test. The possibility to generate multiple tests based on the same code has proved to be a time-saving functionality, and attenuating Wi-Fi signals is a feasible technique to create a sparser topology in the testbed.

The listed testbed experiences may inspire researchers to design a brand new testbed, or modify or expand their existing testbeds. In order to get a better understanding of the dynamics involved in a real-life deployment, it is necessary to try and explain all erratic behavior observed while conducting testbed experiments. This will eventually lead to the development of robust wireless deployments that are expected to be part of our lives tomorrow.

References

[1] S. Bouckaert, J. Bergs, D. Naudts, E. De Kegel, J. Baekelmans, N. Van den Wijngaert, C. Blondia, I. Moerman, and P. Demeester. *A Mobile Crisis Management System for Emergency Services: from Concept to Field Test*. In Proceedings of the WiMeshNets06 workshop, part of the third Intl. Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, Waterloo, Canada, august 2006.

[2] W. Vandenberghe, K. Lamont, I. Moerman, and P. Demeester. *Connection Management over an Ethernet based Wireless Mesh Network*. In WRECOM, Wireless Rural and Emergency Communications Conference, Rome, Italy, February 2007.

[3] S. Bouckaert, D. Naudts, I. Moerman, and P. Demeester. *Making Ad Hoc Networking a Reality: Problems and Solutions*. Journal of Telecommunications and Information Technology, 1(1):3–11, 2008.

[4] *iLab.t Technology Center*. http://ilabt.ibbt.be/.

[5] James T. Kaba and Douglas R. Raichle. *Testbed on a desktop: strategies and techniques to support multi-hop MANET routing protocol development*. In MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pages 164–172, New York, NY, USA, 2001. ACM.

[6] G. Werner-Allen, P. Swieskowski, and M. Welsh. *MoteLab: a wireless sensor network testbed*. In Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on, pages 483–488, April 2005.

[7] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. *TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks*. In REALMAN '06: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality, pages 63–70, New York, NY, USA, 2006. ACM.

[8] PC Engines. *Alix system board*. http://www.pcengines.ch/alix.htm.

[9] S.K. Kaul, M. Gruteser, and I. Seskar. *Creating wireless multi-hop topologies on space-constrained indoor testbeds through noise injection*. In TRIDENT-COM 2006. 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities., 2006.