Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Informatietechnologie

Protocolarchitecturen van de nieuwe generatie voor heterogene draadloze sensornetwerken

Next-Generation Protocol Architectures for Heterogeneous Wireless Sensor Networks

Eli De Poorter

Proefschrift tot het bekomen van de graad van
Doctor in de Ingenieurswetenschappen:
Computerwetenschappen
Academiejaar 2010-2011

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Informatietechnologie

Promotor: Prof. Dr. Ir. Ingrid Moerman

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur

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:
Computerwetenschappen
Academiejaar 2010-2011

Dankwoord

Dit boek is een eindresultaat van vier jaar doctoraatsonderzoek. Een doctoraat omvat echter meer dan vier jaar gespecialiseerd en geïsoleerd onderzoek. Uit naam van de Universiteit Gent was ik betrokken bij activiteiten zoals het uitvoeren van wetenschappelijk onderzoek, het begeleiden van practica en master thesissen, het opvolgen van wetenschappelijke projecten en het uitstippelen van onderzoeksroadmaps. Met andere woorden: interessant, gevariëerd en veelzijdig werk! Bij deze activiteiten is het opbouwen van sociale contacten cruciaal. Dit is dus het uitgelezen moment om alle personen te bedanken die me in staat stelden om deze activiteiten tot een goed einde te brengen.

Allereerst wil ik vakgroepvoorzitter prof. Daniël De Zutter bedanken voor het ter beschikking stellen van de uitgebreide onderzoeksfaciliteiten. Verder ben ik prof. Piet Demeester, de drijvende kracht achter de IBCN onderzoeksgroep, erg dankbaar voor de vriendelijke en persoonlijke manier van omgaan met de leden van zijn onderzoeksgroep. Hierbij ook mijn respect, gezien de IBCN onderzoeksgroep onder zijn leiding ieder jaar meer naambekendheid en succes kent. Natuurlijk kan mijn promoter, prof. Ingrid Moerman niet ontbreken in dit dankwoord. Zij gaf me de kans te werken omtrent het onderzoeksonderwerp 'sensornetwerken'. Bedankt voor de interessante discussies, voor alle nuttige feedback, voor het vertrouwen en vooral bedankt om te helpen een eigen kritische onderzoeksvisie te ontwikkelen.

Vervolgens wil ik alle instanties en personen bedanken die het mogelijk maakten mijn onderzoek te realizeren. Daarbij bedank ik vooral het IWT en het IBBT voor het financieren van zowel mijn onderzoek als gerelateerde onderzoeksprojecten. De discussies en demonstraties die voortvloeiden uit verschillende projecten waren uiterst waardevol om mijn eigen onderzoek bij te sturen, alsook om mijn onderzoek te evalueren in praktische use-cases. Bedankt dus aan alle partners en onderzoekers van o.a. de volgende projecten: het IBBT WBA project, het IBBT DEUS project, het NGWiNeTs project, het IWT SymbioNets project en het FP7 SPITFIRE project. Ook wil ik o.a. prof. Frank Gielen en Wouter Haerick bedanken voor hun advies omtrent commercializatiemogelijkheden van mijn onderzoek.

Wat hulp betreft verdienen mijn collega's van 'de sensorgroep' zeker een speciale vermelding. Bedankt voor de interessante discussies, voor het vertrouwen in IDRA, voor de gemeenschappelijke papers en projecten, voor het testlab, voor de onderlinge steun en de gezellige babbels. Velen van jullie beschouw ik ondertussen meer als vriend dan als collega.

Ook dank aan de leden van het administratief team voor de hulp om alle ad-

ministratieve rompslomp in goede banen te leiden. Ook de technische support van ons admin-team was onmisbaar: zij zorgden ervoor dat de netwerken, de computers en de technische infrastructuur steeds vlot bleven draaien.

Gelukkig zorgden ook op persoonlijk vlak een hele hoop vrienden de afgelopen vier jaar voor de nodige steun en ontspanning na het werk. Bedankt allemaal!

En tot slot bedank ik natuurlijk mijn ouders en zus. Bedankt voor de warme thuishaven, jullie raad en steun, en bovenal bedankt om mij te laten uitgroeien tot een zelfstandig individu met een eigen persoonlijkheid. En als allerlaatste, bedankt An-Sofie, mijn levensgezellin en beste vriendin.

Dit dankwoord is te kort om iedereen die belangrijk is in mijn leven voldoende te bedanken. Daarom: bedankt aan alle anderen die er voor mij waren, op het werk of in mijn privé leven.

Naar de toekomst toe kan ik enkel hopen dat mijn werk even uitdagend en boeiend blijft en dat ik zelf op mijn beurt mensen uit mijn omgeving kan enthousiast maken, stimuleren en vooruit helpen.

Gent, Februari 2011
Eli De Poorter

Nederlandstalige samenvatting -Summary in Dutch-

Het tijdig verkrijgen van correcte informatie wordt steeds belangrijker in de hedendaagse maatschappij. Informatie die online beschikbaar is, zoals persoonlijke interesses of gebruikersreviews, oefent een grote invloed uit op de werking van bedrijven en kan zelfs worden gebruikt om nieuwe commerciële markten te creëren. Het is dan ook niet verwonderlijk dat informatie steeds toegankelijker wordt. Draadloze communicatiesystemen garanderen dat een veelvoud van informatie steeds beschikbaar en opvraagbaar is, onafhankelijk van de lokatie van de gebruiker. Hoewel er natuurlijk risico's zijn verbonden aan het ongelimiteerd openstellen van (persoonlijke) informatie, heeft dergelijke informatievrijheid vooral vele voordelen. Denk maar aan het tijdsverlies dat wordt vermeden door het gebruik van een GPS-toestel met up-to-date verkeersinformatie.

Een andere opvallende wijziging in onze maatschappij is de aanwezigheid van steeds meer elektronische toestellen. Een steeds verder gaande miniaturisering resulteerde in dalende prijzen voor elektronica. Als gevolg daarvan worden goedkope elektronische microchips tegenwoordig ingebouwd in een groot aantal dagdagelijkse toestellen: van radiowekkers tot high-tech autobesturing. Dankzij deze prijsdalingen is het commercieel interessant om netwerken op te bouwen bestaande uit een groot aantal goedkope identieke sensornodes met beperkte rekencapaciteit. Deze sensornodes meten omgevingsinformatie op, zoals de temperatuur of de vochtigheid, en sturen de opgemeten informatie onderling door naar elkaar. De opgemeten waarden worden tenslotte verzameld in een centrale server, die de gegevens beschikbaar maakt voor de buitenwereld.

Het aantal toepassingmogelijkheden voor sensornetwerken is heel uitgebreid. Enkele voorbeeldtoepassingen zijn de volgende:

Omgevingsmonitoring Door goedkope microprocessoren te gebruiken kan een groot aantal sensornodes worden geïnstalleerd in een (natuur)gebied om zo omgevingsvariabelen zoals temperatuur of vochtigheid op erg accurate wijze real-time op te volgen. Indien bijvoorbeeld abnormale droogte of een bosbrand wordt gedetecteerd, worden de watersluizen automatisch geopend en wordt de brandweer verwittigd.

Domotica Door sensornodes te installeren in huishoudelijke toestellen kunnen deze toestellen draadloos worden geconfigureerd en gecontroleerd. Op deze

wijze worden toepassingen zoals automatische klimaatregeling, inbraakdetectie en automatisatie van het huis mogelijk.

E-health Tot slot kunnen sensoren worden aangebracht op het lichaam voor het draadloos opvolgen van de conditie van topsporters of voor het tijdig detecteren van ziekteverschijnselen zoals epileptische aanvallen.

Om onderling te kunnen communiceren hebben sensornodes een communicatiestack nodig. Deze communicatiestack bevat meerdere netwerkprotocollen die verantwoordelijk zijn voor het verzenden en routeren van de opgemeten informatie naar de correcte bestemming. Om de prijs van sensornodes laag te houden bevatten sensornodes slechts een beperkte hoeveelheid ROM en RAM geheugen (grootteorde: enkele kilobytes). Door deze beperking is het ontwikkelen van zelforganizerende netwerkprotocollen voor sensornetwerken meestal erg complex en tijdrovend. Om dit euvel te verhelpen wordt in dit proefschrift een alternatieve 'informatie-gebaseerde communicatiearchitectuur' ('IDRA') voorgesteld. Door optimalizaties te ondersteunen op architecturaal niveau wordt het ontwikkelen van netwerkprotocollen in IDRA vereenvoudigd. Bijvoorbeeld, wanneer een IDRA-netwerkprotocol informatie wenst te versturen naar een andere node, hoeft dit protocol deze informatie enkel te overhandigen aan de IDRA-architectuur. Het is de verantwoordelijkheid van IDRA om deze informatie in een pakket te verpakken, het pakket op te slaan in buffers en de juiste netwerkprotocollen te activeren die het pakket kunnen verwerken.

Bovendien resulteert hergebruik van gemeenschappelijke bibliotheken in een efficiënter gebruik van het beschikbare geheugen. In gelaagde communicatiestacks worden sommige netwerkfuncties meerdere malen geïmplementeerd in verschillende netwerklagen. Daartegenover is IDRA verantwoordelijk voor alle gemeenschappelijke functionaliteit gerelateerd aan pakketbeheer, bufferbeheer en protocolselectie. Daardoor verbruiken IDRA-netwerkprotocollen tot een factor 10 minder geheugen.

Om eenvoudig nieuwe sensornetwerken te kunnen installeren in afgelegen gebieden worden sensornodes dikwijls uitgerust met batterijen. Om de batterijen van de sensornodes niet regelmatig te moeten vervangen is het erg belangrijk om de radio zoveel mogelijk uit te schakelen. Dit kan bijvoorbeeld door het aantal uitgewisselde pakketten te beperken. Om de levensduur van het netwerk te verlengen is de IDRA-architectuur in staat om informatieuitwisselingen van verschillende protocollagen te combineren in één enkel pakket. Het gebruik van de IDRA-pakketaggregatie verhoogt de levensduur van een sensornetwerk met 30-50%. Omdat deze architecturale oplossing inwerkt op meerdere protocollen is dit resultaat beduidend beter dan de levensduur die kan worden bereikt door het optimalizeren van slechts één enkel netwerkprotocol.

Geavanceerde toepassingen voor sensornetwerken vereisen bovendien ondersteuning voor meer complexe communicatiemogelijkheden. Zo moeten e-health-toepassingen bijvoorbeeld de garantie krijgen dat de informatieuitwisselingen op een betrouwbare wijze hun bestemming bereiken. Om deze nieuwe generatie toepassingen mogelijk te maken bevat IDRA bibliotheken die Quality of Service

(QoS), cross-layer optimalizaties en heterogene toestellen ondersteunen. Door deze geavanceerde functies op architecturaal niveau te ondersteunen worden de protocolontwerpers ontlast van de taak om zelf deze complexe netwerkfunctionaliteiten te voorzien. Bovendien heeft deze aanpak het voordeel dat de ontwikkelde heterogeneiteit en QoS-oplossingen op universele wijze kunnen worden gecombineerd met alle bestaande en/of nieuwe IDRA-netwerkprotocollen.

Hedendaagse succesvolle producten kunnen steeds meer interageren met hun omgeving. Denk maar aan smartphones, die met het internet kunnen connecteren gebruik makende van een veelvoud aan communicatieinterfaces zoals USB, UMTS, bluetooth of Wi-Fi. Op termijn zal ook opgemeten sensorinformatie op elk moment opvraagbaar zijn door middel van een groot aantal technologieën. Om deze evolutie te ondersteunen kan IDRA nu reeds op efficiënte en transparante wijze sensornetwerken integreren met bestaande technologieën. Meer bepaald is IDRA in staat om (i) verschillende soorten ontvangen pakkettypes correct te interpreteren, (ii) uitgaande pakkettypes te converteren naar nieuwe pakketformaten, (iii) automatisch optimale netwerkprotocollen te selecteren en (iv) meerdere communicatieinterfaces te beheren.

Tot slot beschrijft dit proefschrift een toekomstvisie waarin netwerken niet langer manueel moeten worden geconfigureerd. Om deze visie mogelijk te maken wordt een methodologie beschreven waarbij toestellen automatisch op zoek gaan naar naburige (draadloze) toestellen. Op basis van hun netwerkverwachtingen onderhandelen toestellen automatisch met naburige (draadloze) toestellen omtrent de optimale netwerksettings en de optimale netwerkconfiguratie. Door op deze wijze netwerkoverschrijdend samen te werken wordt de efficiëntie van alle betrokken netwerken globaal geöptimaliseerd.

Alle concepten van dit proefschrift werden geïmplementeerd en geëvalueerd in één IDRA-architectuur. Dit bewijst dat nieuwe protocolarchitecturen zowel efficiënt als veelzijdig kunnen zijn. IDRA werd bijgevolg positief onthaald door zowel onderzoeks- als industriële partners. Het volledige IDRA-framework is beschikbaar als open-source project op http://idraproject.net.

English Summary

The key to success in our modern society is information. Information is a business asset that is of key importance in today's competitive world. By obtaining the right information, it is even possible to create new commercial markets. Therefore, it comes at no surprise that information is increasingly accessible. By using wireless communication technologies, information in all its forms can be requested anywhere and anytime. As an example, consider the large number of productive hours that are saved due to the use of GPS devices with up-to-date traffic information. From a business point-of-view, it is clear that the advantages of freely-available information outweigh the risks in many situations.

In addition to an increasing amount of information, we are also surrounded by an increasing number of electronic devices. Due to recent advances in chip design, the cost and size of electronics has decreased drastically. As a result, cheap microchips are embedded in a large number of daily consumer items such as dishwashers or digital radios. Wireless sensor networks profit from this evolution by utilizing a large number of cheap sensor devices to monitor a large area. Sensor devices sample their environment, measuring information such as temperature or humidity. The collected information is exchanged hop-by-hop between the different sensor nodes, and finally collected in a central server that exposes the measured information to the outside world. The use of cheap sensor nodes allows sensor networks to make available interesting (environmental) information at a cost that is small enough to be commercially viable.

Wireless sensor networks can be used to enable a large number of application domains. Some examples are the following:

Environmental monitoring A large number of devices can be deployed to measure fine-grained information about an area. For example, when sensor devices detect an abnormal drought or a forest fire, fire fighters are automatically notified and water is automatically redirected to the affected area.

Wireless building automation By embedding sensor devices in household equipment, common household objects can be configured and controlled wirelessly. This way, wireless sensor networks enable applications such as intrusion detection, automatic control of household equipment or fine-grained control of the heating, air conditioning and ventilation of the building.

E-health Finally, by deploying sensor devices on a body, wireless sensor networks can be used to monitor the condition of top athletes or to detect dan-

gerous symptoms such as epileptic seizures.

Communication between the sensor nodes is handled by the communication stack. This communication stack consists of multiple network protocols responsible for sending and routing measured information to the correct destination. To keep their cost low, sensor devices have only a limited amount of ROM and RAM memory (in the order of a few kilobytes). The development of self-organizing network protocols for these resource-constrained devices is often difficult and time-consuming. To remedy this situation, this dissertation presents an alternative 'Information DRiven communication Architecture' or 'IDRA'. To simplify the design of network protocols, IDRA contains simple but useful functionality at an architectural level. For example, when an IDRA protocol wishes to exchange information with a remote node, the network protocol can simply hand over this information to the IDRA architecture. It is the responsibility of IDRA to encapsulate the information in a packet, to manage the packet buffers and to select the network protocols that should process the packets.

IDRA is also efficient in terms of memory requirements. Layered communication architectures do not make efficient use of the available memory, since a number of network functions are implemented at multiple network layers. Examples of these duplicate functions are packet creation, packet interaction, buffer provisioning and packet selection. In contrast, IDRA utilizes shared libraries that implement functionality that is not protocol-specific. By using an architecture which delegates these specific tasks to a central system, network protocols require up to a factor 10 less memory.

To easily deploy a new sensor network in remote areas, sensor devices are often battery powered. As such, these sensor nodes are not only limited in terms of memory capabilities, but also in terms of their available energy. To ensure that the batteries do not need frequent replacement, protocol designers need to limit the number of radio transmissions. To this end, the IDRA architecture can combine multiple information exchanges in a single packet. By using the in-built IDRA aggregation approach, the network lifetime increases by up to 30-50%. This architectural solution results in significantly more packet savings than aggregation approaches that combine packets of only a single network layer.

To utilize wireless sensor networks for next-generation applications, advanced communication requirements should be supported. For example, e-health applications require guarantees that all information is exchanged in a reliable way. IDRA enables next-generation applications by supporting Quality of Service (QoS), cross-layer optimizations and heterogeneity at an architectural level. Since the developed QoS and heterogeneity solutions are protocol-independent, they can transparently be combined with existing and new IDRA network protocols.

In the future, we expect that it will be possible to request measured sensor information using a large variety of communication technologies. This is already the case for many consumer items such as smartphones. Wireless devices use an increasing number of communication technologies to connect to the internet, such as USB, UMTS, bluetooth or Wi-Fi. To support these upcoming developments,

IDRA can transparently integrate wireless sensor networks with a variety of existing communication technologies. More specifically, IDRA is able to (i) automatically interpret different incoming packet types, (ii) convert outgoing packets to the correct format, (iii) select the correct network protocol to process the packets and (iv) manage multiple communication interfaces.

Finally, this dissertation describes a vision of the future in which networks are no longer configured manually. Instead, devices autonomously negotiate and cooperate with neighboring (wireless) devices. With this goal in mind, a methodology is described where devices automatically discover co-located (wireless) devices. Based on their network requirements, the devices negotiate with each other about optimal network settings and network configuration. By optimizing the network performance across multiple protocol layers and across network boundaries, the global performance of all involved devices can be optimized. This methodology is also implemented and evaluated using the IDRA framework.

All concepts from this dissertation have been successfully implemented in a single IDRA architecture. In addition, we have proven that our architecture is both efficient and versatile. IDRA received positive comments from both research and industrial partners and is freely available as open-source project at http://idraproject.net.

List of Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during my PhD research.

A1: Publications in International Journals

1 **E. De Poorter**, B. Latré, I. Moerman and P. Demeester, *Symbiotic networks: Towards a new level of cooperation between wireless networks*, Published in Special Issue of the Wireless Personal Communications Journal, Springer Netherlands, 45(4):479-495, June 2008

2 S. Bouckaert, **E. De Poorter**, B. Latré, J. Hoebeke, I. Moerman and P. Demeester, *Strategies and challenges for interconnecting wireless mesh and wireless sensor networks*, Published in the Wireless Personal Communications Journal, Springer Netherlands, 53(3):443-463, March 2010

3 **E. De Poorter**, S. Bouckaert, I. Moerman and P. Demeester, *Non-intrusive Aggregation in Wireless Sensor Networks*, Ad Hoc Networks, 9(3):324-340, online 10 August 2010, ISSN 1570-8705

4 **E. De Poorter**, I. Moerman and P. Demeester, *IDRA: a Flexible System Architecture for Next Generation Wireless Sensor Networks*, Accepted for the Wireless Networks Journal on 26 November 2010, In Press

5 Jeroen Hoebeke, **Eli De Poorter**, Stefan Bouckaert, Ingrid Moerman and Piet Demeester, *Managed Ecosystems of Networked Objects*, Wireless Personal Communications, Special Issue on Distributed and Secure Cloud Clustering (DISC), In Press

6 **E. De Poorter**, I. Moerman and P. Demeester, *Enabling direct connectivity between heterogeneous objects in the internet of things through a network service oriented architecture*, Submitted to EURASIP Journal on Wireless Communications and Networking on 9 November 2010

7 **E. De Poorter**, Pieter Becue, I. Moerman and P. Demeester, *A negotiation-based networking methodology to enable cooperation across heterogeneous co-located networks*, Submitted to Ad Hoc Networks journal

8 D. Plets, W. Joseph, **E. De Poorter**, L. Martens and I. Moerman, *Concept and Framework of a Self-Regulating Symbiotic Network*, Submitted to IEEE electronics letters

A2: Other International Journals

1 **E. De Poorter**, I. Moerman and P. Demeester, *Design of a Decoupled Sensor Network Architecture Based on Information Exchanges*, International Journal on Advances in Software, ISSN 1942-2628, vol. 2, no. 4, year 2009, pages 300-312, http://www.iariajournals.org/software/

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

1 B. Latré, **E. De Poorter**, I. Moerman and P. Demeester, *MOFBAN: A Lightweight Modular Framework for Body Area Networks*, Proceedings of the 2007 international conference on Embedded and ubiquitous computing, Springer Lecture Notes In Computer Science, pp. 610-622, Taipei, Taiwan, November 2007

2 P. De Mil, **E. De Poorter**, B. Latré, et al., *Definition and evaluation of local path recovery mechanisms in wireless sensor and actuator networks*, Third International Conference on Sensor Technologies and Applications (SENSORCOMM), pp. 358-365, Athens (Glyfada), Greece, June 2009

3 **E. De Poorter**, I. Moerman and P. Demeester, *An information driven sensornet architecture"*, Third International Conference on Sensor Technologies and Applications (SENSORCOMM), pp. 553-561, Athens (Glyfada), Greece, June 2009 (best paper award)

4 S. Bouckaert, **E. De Poorter**, P. De Mil, et al., *Interconnecting Wireless Sensor and Wireless Mesh Networks: Challenges and Strategies*, IEEE global communications conference (IEEE GLOBECOM 2009), Honolulu, Hawai, USA, 30 november - 4 december 2009

5 E. Troubleyn, **E. De Poorter**, P. Ruckebusch, I. Moerman, P. Demeester, *Supporting protocol-independent adaptive QoS in Wireless Sensor Networks*, The Third IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2010), Newport Beach, California, USA, June 7-9, 2010

6 J. Vanhie-Van Gerwen, **E. De Poorter**, B. Latré, I. Moerman and P. Demeester, *Real-life performance of protocol combinations for Wireless Sensor Networks*, The Third IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2010), Newport Beach, California, USA, June 7-9, 2010

7 **E. De Poorter**, Pieter Becue, I. Moerman and P. Demeester, *Exploring a boundary-less cooperation approach for heterogeneous co-located networks*, Accepted for the IEEE International Conference on Communications (ICC2011), Kyoto, Japan, June 5-9, 2011

Chapters in Books

1 **E. De Poorter**, B. Latré, I. Moerman and P. Demeester, *Universal Framework for Sensor Networks*, Chapter in the book *Sensor and Ad-Hoc Networks: Theoretical and Algorithmic Aspects*, Series: Lecture Notes Electrical Engineering, Vol. 7, edited by Makki S. Kami, X.-Y. Li, et al. ISBN: 978-0-387-77319-3, Springer, June 20, 2008

Publications in International Conferences

1 **E. De Poorter**, B. Latré, I. Moerman and P. Demeester, *Modular Architecture for Heterogeneous Sensor Networks*, 4th European conference on Wireless Sensor Networks (EWSN 2007), EWSN adjunct poster proceedings, pp. 21-22, Delft, Netherlands, 29-31 January 2007

2 B. Latré, **E. De Poorter**, I. Moerman, P. Demeester, *Symbiotic networks*, published in Proceedings of the 9th Strategic Workshop on International Mobile Telecommunications, Malaga, Spain, 30 May - 1 June 2007

3 **E. De Poorter**, B. Latré, I. Moerman and P. Demeester, *Universal Modular Framework for Sensor Networks*, International Workshop on Theoretical and Algorithmic Aspects of Sensor and Ad-hoc Networks (WTASA'07), Miami, Florida, June 28-29, 2007

4 E. Troubleyn, **E. De Poorter**, I. Moerman and P. Demeester, *AMoQoSA: Adaptive Modular QoS Architecture for Wireless Sensor Networks*, SENSORCOMM 2008, Cap Esterel, France, August 25-31, 2008

5 **E. De Poorter**, S. Bouckaert, I. Moerman and P. Demeester, *Broadening the concept of aggregation in wireless sensor networks*, SENSORCOMM 2008, Cap Esterel, France, August 25-31, 2008

6 L. Tytgat, B. Jooris, P. De Mil, **E. De Poorter**, et al., *Multichannel protocol for interference avoidance in wireless sensor networks*, KEIO and Ghent University 2nd G-COE Joint workshop for future network, pp. 55-58, Ghent, Belgium, 26-27 September, 2008

7 **E. De Poorter**, B. Latré, I. Moerman and P. Demeester, *Modular framework for sensor networks*, 3rd Gent University and Keio University Global COE Workshop, Ghent, Belgium, 23rd February, 2009

8 S. Bouckaert, **E. De Poorter**, B. Latré, J. Hoebeke, et al., *Global routing in heterogeneous wired/wireless ecosystems*, Strategic Workshop On Wireless Innovation for InterDynamic TecHnology (WIDTH) (SW), Rebild, Denmark, 15-17 May, 2009

9 S. Verstichel, **E. De Poorter**, T. De Pauw, P. Becue, B. Volckaert, F. De Turck, I. Moerman, P. Demeester, *Distributed ontology-based monitoring on the IBBT WiLab.t infrastructure*, International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom), Berlin, Germany, 18-20 May, 2010

10 **E. De Poorter**, I. Moerman and P. Demeester, *Incentive Based Network Cooperation in Next Generation Networks*, The 3th Ghent University - KEIO Joint Workshop on Future Networking, Ghent, Belgium, October 4, 2010

11 **E. De Poorter**, I. Moerman and P. Demeester, *Support for heterogeneous dynamic network environments through a reconfigurable network service platform*, Accepted for the 1st IEEE International Symposium on Access Spaces (IEEE-ISAS 2011), Yokohama, Japan, June 17-19, 2011

12 D. Plets, W. Joseph, K. Vanhecke, L. Martens, **E. De Poorter**, B. Jooris, and I. Moerman, *Development of a Dynamic Symbiotic Network Planner and Application to a Living Lab Testbed*, Accepted for the 2011 IEEE International Symposium on Antennas and Propagation (APS 2011), Spokane, Washington, USA, July 3-8, 2011.

Publications in National Conferences

1 **E. De Poorter**, I. Moerman, *Wireless Sensor Networks in Heterogeneous Environments*, 9e UGent-FirW Doctoraatssymposium, Ghent, Belgium, December 3, 2008

2 **E. De Poorter**, I. Moerman, *IDRA: a novel protocol architecture for networked embedded devices*, 11e UGent-FirW Doctoraatssymposium, Ghent, Belgium, December 1, 2010

Patents

1 *Node and Wireless Sensor Network Comprising the Node*, Application Number: US 12/817.722, Filing Date: June 17, 2010, 14 claims, Applicants: E. De Poorter, I. Moerman.

Table of Contents

1 Introduction

This chapter gives a general overview of wireless sensor networks (WSNs). The chapter describes the features of a typical WSN and illustrates the use of WSNs by giving an overview of existing sensor applications. Next, the hardware characteristics of a number of commercially available sensor devices are listed together with a description of the challenges that are involved to support networking between sensor nodes. Finally, the main contributions of this research are presented, together with an outline of the structure of this book.

1.1 An introduction to wireless sensor networks

Wireless sensor networks consist of embedded devices ('sensor devices') that are wirelessly connected with each other and are capable of sampling an environmental quantity of their surroundings. Sensor devices can measure environmental information such as temperature, light intensity, sound, humidity, vibration and pressure. Typically, all measured information is transmitted wirelessly to an application that is running on a remote server or 'sink', where actions are taken based upon the measured values. Typical applications for wireless sensor networks include habitat monitoring, wireless building automation, medical monitoring, security applications and asset tracking [1].

As an example, consider the situation where a wildlife park is wirelessly monitored using a WSN. By distributing a large amount of sensor nodes over the area, very accurate information regarding the area can be obtained. Park officials can

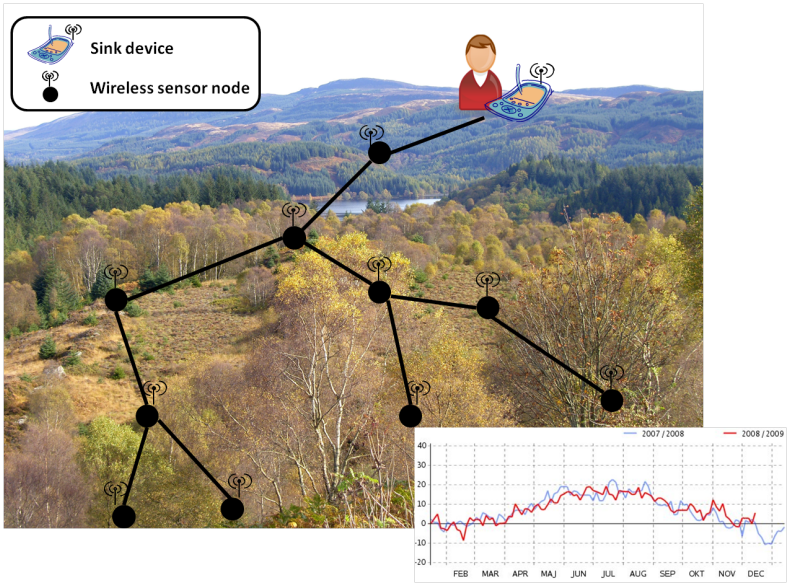

Figure 1.1: An example application for a wireless sensor network. Measured information (such as the temperature) is transmitted from one sensor node to the next one, until the information can be processed by an application on a remote sink device.

keep detailed statistics about monitored values such as temperature, humidity and the presence and behavior of tracked animals. If the measured humidity becomes too low, the park officials are automatically informed so that they can redirect water flows to the barren areas. In addition, if the measured temperature of a sensor node in the monitored area reaches a critical level, fire fighters are automatically dispatched to the correct location to prevent a bush fire from starting and/or spreading. When a large area is monitored, sensor nodes can not communicate with the application server directly. Instead, wireless sensor networks have to resort to multi-hop communication: packets are transmitted from one sensor device to the next, until the packet reaches the destination node (often referred to as the 'sink') (Figure 1.1).

To keep the cost of sensor nodes low, sensor devices are generally very simple. They consist of a small microprocessor (typically with a clock speed of 8MHz or less), a low-power radio (typically 250 kbps or lower), a sensing device and a small battery or energy harvesting supply (like a solar cell) to power these components. The use of batteries or energy harvesters eases the deployment of sensor devices, since no power lines or cables are required. This is especially useful for locations where infrastructure is lacking (nature environments, disaster areas, etc.), when introducing new cables and infrastructure is impractical (such as in existing buildings and enlisted monuments) or when in situ the network set-up time should be

minimized (such as in ad-hoc emergency applications).

Developing network solutions for resource-constrained devices often proves challenging. Due to the limited memory, processing power and energy provisions of sensor nodes, past WSN research has mainly focused on the design of networking solutions that minimize the energy and processing requirements [2, 3]. As such, the main challenges when designing network solutions for WNSs are the following:

- Networking solutions should be energy-efficient to maximize the network lifetime.
- Due to the limited availability of ROM and RAM memory, network protocols should have a very small memory footprint.
- The microprocessor of sensor devices is not suitable for complex algorithms and calculations.
- Finally, in direct contrast with the above requirements, networking solutions should be versatile enough to be used in a wide variety of application scenarios.

Due to these conflicting requirements, the optimal network solution for WSN applications is still missing. Instead, most network solutions are specifically designed and optimized for a single application scenario.

1.2 Application scenarios

The following section presents an overview with example scenarios. More example applications can be found in [1, 4].

1.2.1 Environmental monitoring

During the past few years, several sensor networks have been deployed to efficiently monitor large areas. Due to their self-organizing nature, a large amount of cheap sensor devices can be deployed even in the absence of infrastructure. Measured information is sent from one sensor node to the next, until the information reaches a remote sink device where the information is stored. Once deployed, the sensor network can be left out in the open field for several years, collecting data without requiring any human intervention. As such, WSNs can be a solution to make it economically feasible to accurately measure a phenomenum and to act in time when a disaster strikes or an unpredictable event occurs.

Initially, this 'deploy once and leave'em' feature made unattended wireless sensor networks especially interesting for risk-associated applications such as military applications. However, WSNs are also exceptionally suited to monitor environmental problems, such as global warming, the increasing rate of extinction of animal species or the occurrence of damaging forest fires. The following characteristics are typical for environmental monitoring applications: (i) all traffic is collected in a single monitoring device ('the sink'), (ii) the network needs to be self-organizing to minimize human intervention, (iii) to cope with failing devices, the network is robust and fault-tolerant.

Several example projects have been successfully deployed. For example, the ALERT system [5] is dedicated to reducing injuries, deaths, and property damage caused by floods in the US by utilizing WSNs for the automated real-time flood forecasting and early detection of flood conditions. As a second example, the GoodFood project [6] utilizes a number of distributed sensors to monitor the processes of food production and to detect the remainder chemical substances in products. As a last example, the Harvard university utilizes wireless sensor networks to monitor eruptions of active and hazardous volcanoes [7] .

1.2.2 Home, office, factory, emergency and industry applications

Recently, WSNs are used for increasingly complex 'next-generation' applications. For example, the applications described below are deployed in urban areas. As a result, a greater interactivity with the environment is required.

- Applications such as wireless building automation [8] require actuator devices, which are capable of acting upon their environment. Typical actuators are heating controllers, automatic window controllers and garage doors. The sensor devices communicate with these actuator devices to regulate common household functions (Figure 1.2). In addition to communication from sensor to sink, communication can also occur from sink to actuator and from sensor to actuator. As such, the communication patterns that occur in these use cases are more complex.

- In interactive museum exhibitions, information about the exhibition can be custom tailored to the preferences of the visitors. Items can respond to actions and presence of the visitors and exhibition items can be monitored. In addition, real-time cause-and-effect experiments can be used for educational purposes. To support fully interactive applications, a sensor network can be used to localize each visitor and exhibition item. Not only should interactivity be supported by the sensor network, it should also be easy to deploy additional sensor nodes and to wirelessly install new software when the exhibition changes.

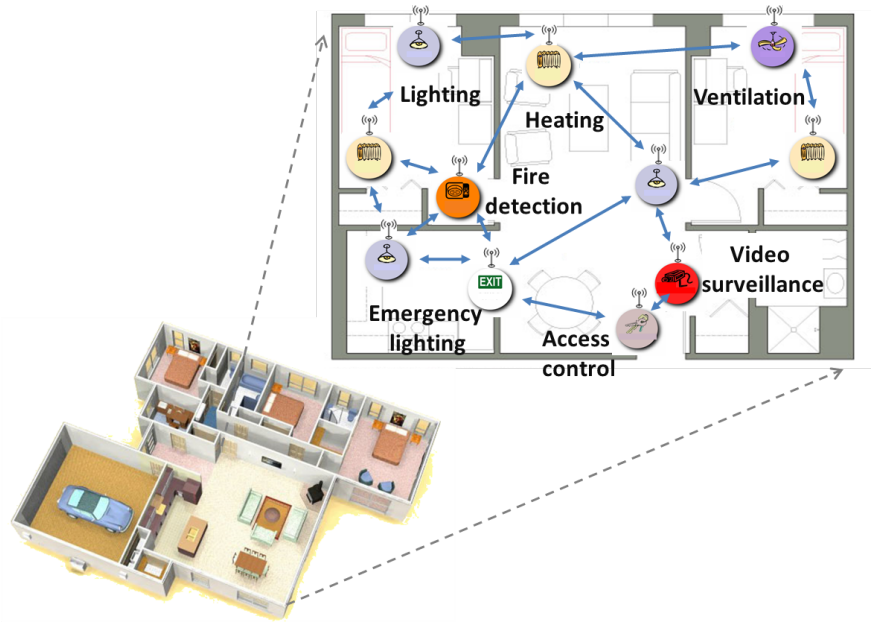

Figure 1.2: In a wireless building automation application, embedded devices automatically connect to each other to regulate household functions.

Next-generation WSN applications also have the potential to result in significant cost reductions. By accurately monitoring the distribution of temperature in each room of a building, the air flow and heating can be more finely controlled. According to [2], the installation of temperature monitoring WSNs in each building could correspond to a savings of $55 billion per year in the US, and a reduction of 35 million metric ton of carbon emissions.

1.2.3 Body area networks

When multiple sensor devices that are worn on the body are able to communicate with each other, the term Body Area Network (BAN) is used. Wireless body area networks are able to measure multiple physiological parameters of a person. For example, a BAN can be used to more accurately diagnose a person (Figure 1.3), to monitor the performance of athletes and sportsmen or to monitor the physical condition of a rescue worker such as a fire fighter. To minimize the health impact of BANs, the transmission power of the sensor devices is typically kept very low.

BANs are often used in e-health applications [9, 10] to preliminary act upon the detection of irregular medical symptoms. For example, a BAN can be used to detect the onset of a heart attack, or can be used to automatically inject a new dose of insulin through a pump when the glucose level of a diabetic patient is too

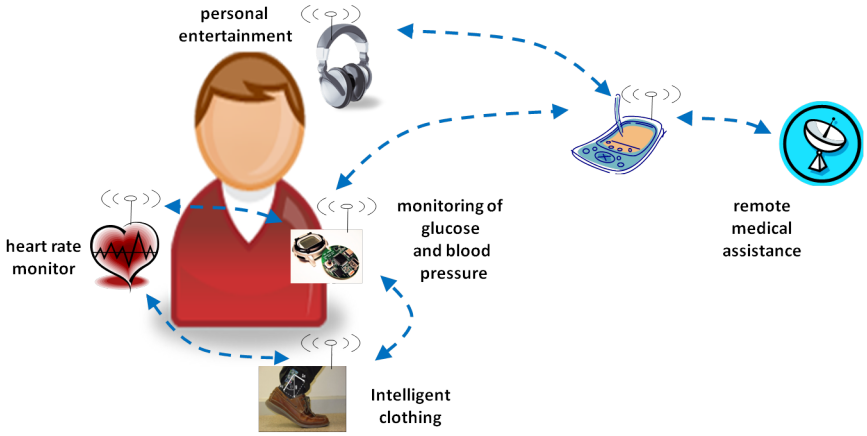

Figure 1.3: A body area network (BAN) can be used to monitor the life signs of patients so that a faster medical intervention is possible.

low. The use of sensors that are attached to patients or elderly people allows these persons greater freedom of movement, whilst potentially dangerous anomalies or situations are recognized earlier, resulting in better quality-of-life.

1.3 Hardware characteristics of a sensor node

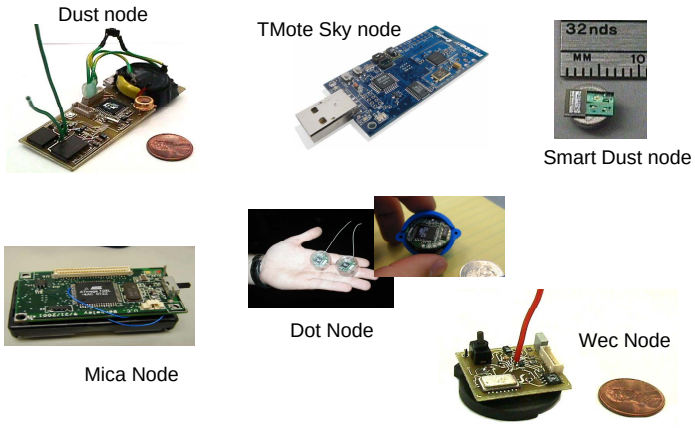

Figure 1.4: Typical sensor node hardware platforms

A large variety of sensor devices is currently available. Several example hardware platforms are shown in Figure 1.4. A comprehensive listing of current platforms is maintained by the Imperial College London [11] or listed in the Embed-

ded WiSeNts Platform Survey [12]. Depending on the platform, the exact hardware characteristics will differ. However, sensor devices do have the following common characteristics [11, 12].

- The *available memory* is a limiting factor when designing network solutions for sensor networks. For example, the TMoteSky sensor node has a memory of 48 kB ROM and 12 kB RAM [13].
- The *microprocessor* of an embedded sensor node typically has a processing speed of 8 MHz or less. Due to this low processing speed and the limited amount of memory that is available on a sensor node, it is not feasible to use a traditional operating system when programming sensor devices.
- The *radio* is designed for ultra-low power consumption. As a result, the bandwidth is typically limited: an IEEE 802.15.4 radio [14] has a physical bit rate of 250 kbps [15]. Depending on the environment, the radio has a maximum communication range of about 100 meter in outdoor environments, and typically far less in indoor environments [16].
- A sensor node is equipped with a *sensing device* that is able to sample its environment. The sensor devices can be integrated in the embedded device, or can be connected with the microprocessor through an external interface. The cost and energy consumption of the sensor depends on the type and purpose of the sensing device.
- Finally, sensor nodes are often *battery-powered*. An alternative is the use of energy harvesting [17] whereby energy is derived from environmental sources. Examples are the use of solar panels or motion converters.

As an example, Table 1.1 lists the hardware characteristics of a number of sensor devices that are currently in use. In the future, the available hardware platforms for sensor nodes will undoubtedly evolve. For the same cost, size and energy-consumption, next-generation sensor nodes will be available that are more powerful and are able to fulfill more complex calculations. However, at the same time, even smaller and less-intrusive sensor devices will be developed, such as implantable devices. As such, in the near future, there will still be devices that exhibit similar hardware limitations as present day sensor nodes.

1.4 Wireless sensor networking issues

Due to their limited processing power and memory, the networking stack of most WSNs is limited to the combination of a simple Medium Access Control (MAC) protocol and a simple routing protocol. On top of these two protocol layers, a

Feature	Imote (2003)	Mica2 (2003)	MicaZ (2004)	Telos/TMote Sky (2005)	IBCN-Rmoni (2010)
CPU type	32 bit ARM	8 bit Atmel	8 bit Atmel	16 bit TI	16 bit TI
Processor speed [MHz]	12	7.4	7.4	4	18
SRAM [kB]	64	4	4	10	16
FLASH [kB]	512	128	128	48	256
Radio	Bluetooth	Custom 300-900 MHz	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4
Bandwidth [kb/s]	720	38.4	250	250	250
CPU/Rx/Tx Current [mA]	15/24/24	8/10/27	8/20/18	1.8/20/18	1/18.5/25.8
Sleep current [μA]	1-250	19	27	6	2

Table 1.1: Hardware characteristics of wireless sensor nodes.

simple monitoring application is deployed. As such, existing network research mainly focuses on optimizing the lower layers of the OSI reference stack [21] in order to extend the lifetime of large-scale wireless sensor networks. Research regarding high-level WSN functionality, such as end-to-end reliability or support for mobility, is at the moment much less mature.

1.4.1 Networking definitions

Before describing the workings of typical WSN network protocols and behavior, this section first gives several important networking related definitions.

A *network protocol* is a formal description of the way networked devices interact and exchange information. The way devices exchange information is often formalized in the form of an algorithm. In addition, the description of a network protocol typically includes a formalization of the exchanged message formats.

A *protocol architecture* is a structured description of how network protocols interact on a single device. Most present-day protocol architectures use the OSI-reference protocol architecture [21]. This architecture uses a fixed number of protocol layers with well-defined functionality. The different layers do not interact with each other: only packets are passed from one layer to another.

A *communication framework* (or *communication system*) is a reusable set of li-

braries or classes for a software system. Whereas a protocol architecture represents a conceptual way to integrate multiple network protocols, a framework describes the actual implementation of the protocol architecture.

1.4.2 Characteristics of wireless transmissions

The use of a wireless communication medium has several advantages over wired technologies. Due to the absence of wires, the set-up cost and maintenance cost of the network is strongly reduced. In addition, wireless devices are less intrusive: they can be installed without damaging existing structures. Finally, in cases when mobility should be supported, wireless technologies are the only option. When designing network protocols that use a wireless communication medium, the following transmission characteristics need to be taken into account.

Shared medium. In wireless systems, multiple devices typically use the same channel or medium to transmit packets. To prevent packet collisions, sending nodes typically verify the absence of other traffic on the medium before transmitting a packet (i.e.: 'carrier sense'). To support multiple sending and receiving nodes on the same medium, the Carrier Sense Multiple Access (CSMA) protocol is frequently used.

Packet overhearing (Figure 1.5 a). Since all devices use the same communication medium, transmitted packets are received ('overheard') by all devices that are located in the range of the transmitter. When a device overhears a packet with a destination that does not match the address of the receiver, the overheard packet is ignored and discarded.

No collision detection. In wired communication system, collision detection (CD) systems are able to detect the occurrence of packet collisions, so that the transmission can be terminated as soon as a collision is detected. However, (i) wireless signals are strongly attenuated when they are transmitted over a large distance and (ii) a wireless radio can not receive and send information at the same time (i.e.: there is no support for full-duplex communication). As such, wireless communication systems can not guarantee that collisions at the receiving node are detected by the sending node. Wireless systems have to rely on collision avoidance (CA) techniques such as waiting a random back-off period when the medium is busy before sending packets.

Irregular packet reception areas (Figure 1.5 b). The transmission range of wireless sensor nodes are often represented as a circle with a fixed range. However, the signal strength of the transmitted signal typically degrades exponentially. In addition, the transmission range is highly dependent on the environment and the used hardware [16]. As a result, the packet reception

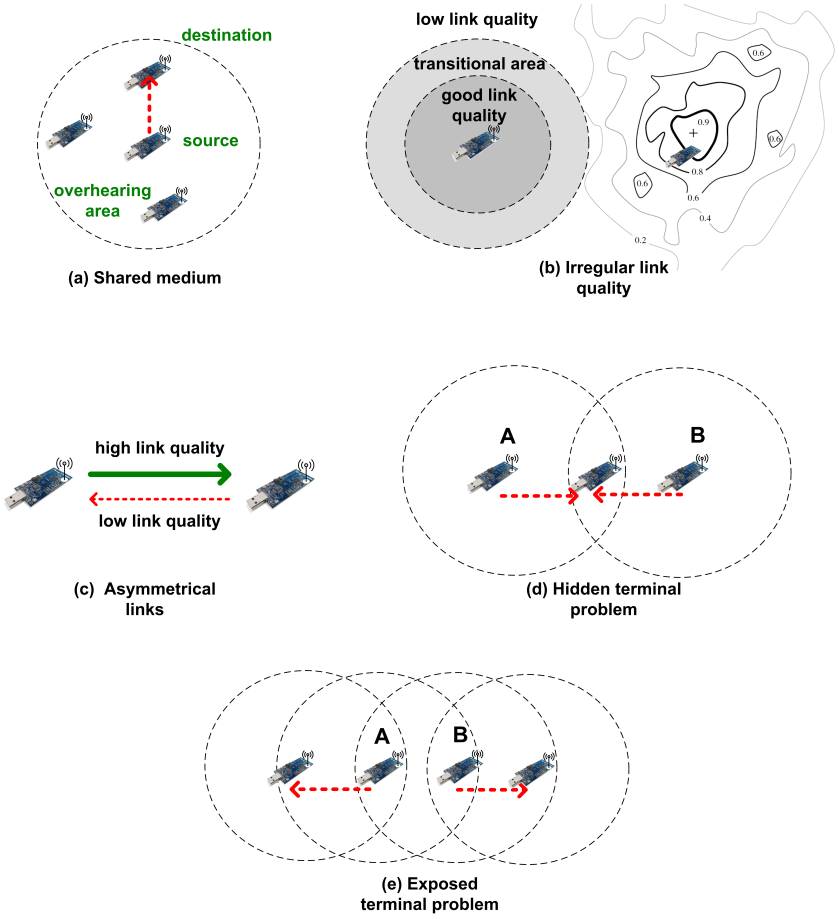

Figure 1.5: Characteristics of wireless communication systems. (a) Since a shared medium is used, transmitted packets can be overheard by other devices. (b) The contours that indicates the link quality based on the distance between two nodes can have very irregular forms. (c) The link quality between two devices can be asymmetrical. (d) Due to the hidden terminal problem, transmitting devices do not always know if packets have collided. (e) Due to the exposed terminal problem, transmitting devices can unnecessarily be prevented from sending packets.

contour formed by receptions at different locations from the same transmitter does not form a perfect circle [18]. Instead, there are three distinct regions of link quality: (i) a nearby connected region where packet reception rates (PRR) are consistently high, (ii) a transitional where the PRR are highly variant and (iii) an area where the PRR are consistently low.

Asymmetrical links (Figure 1.5 c). Based on experimental studies [19], it was observed that the link quality not only depends on the hardware and environment, but it also changes significantly over time. As a result, it is possible that the link between two devices is asymmetrical: the link in one direction has a higher link quality (higher PRR) than the link quality in the other direction.

Hidden terminal problem (Figure 1.5 d). A hidden node terminal problem can occur when two devices A en B transmit a packet to the same destination node. If devices A and B are out of each others transmission range, CSMA can not detect that a packet is already being transmitted to the destination. When both senders transmit simultaneously to the same destination, the packet collision can result in a scrambled packet. The IEEE 802.11 RTS/CTS standard partly solves this problem [20]: any node wishing to send data initiates the process by sending a Request to Send frame (RTS). The destination node replies with a Clear To Send frame (CTS). Any other node receiving the RTS or CTS frame is not allowed to send data for an indicated time.

Exposed terminal problem (Figure 1.5 e). Finally, it is possible that the carrier sense mechanism unnecessary prevents a device from sending a packet. The exposed terminal problem occurs if two transmitters are in each others range but their respective destinations are not part of the overlapping transmission area. In this situation, carrier sense will mistakenly conclude that the devices are interfering.

1.4.3 MAC protocols for sensor networks

A Medium Access (MAC) protocol is responsible for establishing a logical connection between neighboring nodes, so that they can communicate to each other. MAC protocols for wireless sensor networks differ strongly from MAC protocols used in traditional wireless media. MAC protocols, such as IEEE 802.11 [22], are designed for bidirectionial traffic, whereas MAC protocols for WSNs are often designed only for one-way traffic from sensor nodes to a sink. In addition, traditional MAC protocol for wireless networks focus mainly on increasing the throughput, lowering the delay and increasing the fairness [23]. In contrast, MAC protocols designed for sensor networks usually trade off performance (latency, throughput,

fairness) for lower energy cost [24, 25]. To reduce the energy consumption, MAC protocols aim to reduce the following sources of overhead:

- *idle listening* occurs when the radio is turned on but no packets are received destined to the listening node;
- *collisions* happen when two interfering nodes transmit a packet at the same time;
- *overhearing* occurs when a packet is received that was destined for another node;
- *protocol overhead* is required when the nodes exchange signaling information in the form of MAC headers or control packets (ACK, RTS, CTS);
- *over emitting* happens when a sender sends a packet to a receiving node that is not yet ready so that the packet must be sent again.

Since WSN applications typically do not require high data throughput, the amount of time spent on idle listening can be decreased by using sleep schemes, whereby the radio is regularly turned of. The efficiency of a MAC protocol for WSNs is often expressed in terms of its '*duty cycle*', which corresponds to the average percentage of time that the radio is turned on. When sleep schemes are used, precautions have to be taken to ensure that whenever a packet is sent to a neighboring sensor node, the radio of the receiving node is active. Three main approaches are possible.

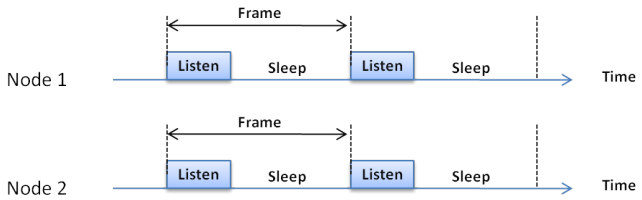

Figure 1.6: Illustration of a synchronized MAC protocol. All sensor devices use the same sleep schedule.

(i) When using *synchronized MAC protocols* such as S-MAC [26], all co-located nodes listen and sleep at the same moment (Figure 1.6). During the times that the radio is awake, devices can communicate to each other by contending for the medium. To match their sleeping schemes, the nodes regularly exchange synchronization information with their immediate neighbors. Typically, all neighboring nodes form a virtual cluster with a 'SYNC master node' that distributes the synchronization information. Occasionally, due to mobility or the network size, it is possible that a network is partitioned into multiple virtual clusters. To allow

communication between these clusters, the edge nodes should either (i) remain awake during the active period of multiple different sleeping schemes, or (ii) further distribute the SYNC messages to devices from the second cluster. Finally, to better cope with variable traffic loads, T-MAC [27] and DS-MAC [28] include mechanisms for dynamically adjusting the sleep schedule depending on the observed traffic.

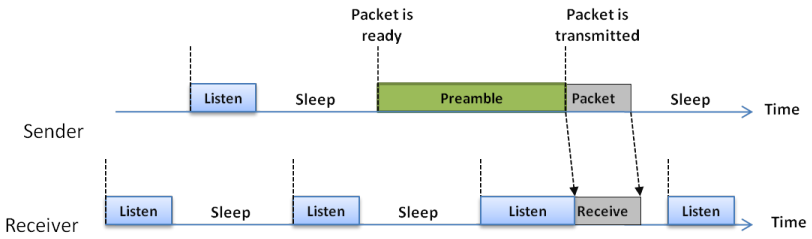

Figure 1.7: Illustration of a non-synchronized MAC protocol. Each sensor device has a different sleeping schedule.

(ii) In contrast, in *non-synchronized MAC protocols*, each node has its own sleeping scheme (Figure 1.7). Typical examples are the preamble-sampling MAC protocol [29] or low-power listening (LPL) [30]. Whenever a packet is exchanged to a neighboring node, the transmitting node first sends a preamble to notify receivers of the upcoming packet transfer, after which the packet is transmitted. To ensure that the preamble is correctly received by all destination nodes, the duration of the preamble should be at least as long as the sleep duration of the receiving nodes. Nodes receiving the preamble will stay awake until they receive the packet, only then can they reiterate their sleeping scheme. The X-MAC protocol [31] optimizes this scheme by including the intended receiver in the preamble so that other devices do not have to remain awake. In addition, a strobed preamble is used so that the receiver can interrupt the preamble to indicate that the packet can be transmitted. In addition, nodes do not go to sleep immediately after receiving a packet, but instead stay awake to offer neighbors the opportunity to sent additional packets. Finally, the WiseMAC [32] protocol reduces the energy consumption by deducing the wake-up schemes of neighbors whenever a packet is overheard. By taking into account the clock drift, the sending of the preamble is delayed until it is likely that the receiver is awake. Non-synchronized MAC approaches do not require synchronization overhead, but instead favor the use of additional energy when transmitting packets. As such, these approaches are mainly beneficial for low-traffic applications.

(iii) Finally, *slotted MAC protocols*, such as IEEE 802.15.4 [14], divide a time duration ('time frame') into different slots (Figure 1.8). An always-on master device assigns a slot to each neighboring 'slave' device during which the device has

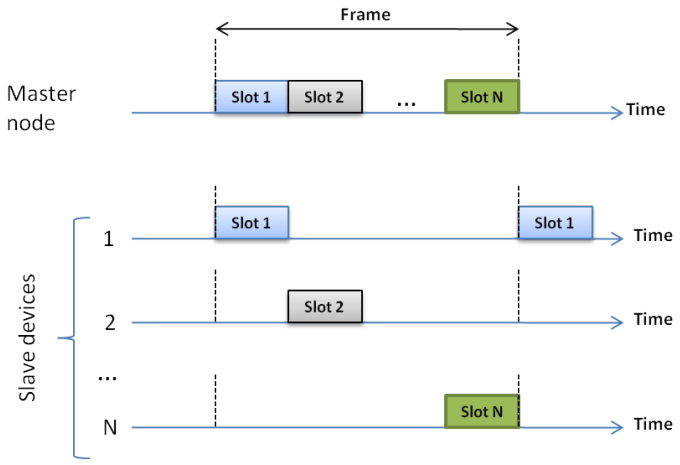

Figure 1.8: Illustration of a slotted MAC protocol. A master device assigns a wake-up slot to each neighboring slave device.

the sole right to transmit packets to the master device. As a result, all communication needs to pass through the master device. Several optimizations are possible. (i) The energy consumption can be spread out more evenly over the network by regularly selecting new master devices [33]. (ii) Multi-hop networks can be supported by also assigning slots to two-hop neighbors [34]. (iii) To cope with traffic fluctuations the number of slots that are assigned to each node can be dynamically adjusted [35]. Slotted MAC approaches typically require significant overhead in the form of accurate time synchronization, but due to the strict structure of each frame, nodes do not need to content for the wireless channel [36].

In conclusion, a large variety of energy-efficient MAC protocols has been proposed for specific network topologies and traffic flows. Unfortunately, there is currently no method to easily select the optimal MAC protocol for a specific application [37].

1.4.4 Routing protocol for sensor networks

Routing protocols are responsible for the end-to-end delivery of packets. The routing protocol delivers data from the source to the destination by relaying the packets across intermediary nodes of the network. The main performance criteria of routing protocols for WSNs is energy-efficiency. In addition, since a large number of sensor nodes can be deployed to observe a phenomena, the routing protocol should be able to support large-scale networks. Finally, the routing protocol should be robust enough to ensure that the failure of individual nodes may not harm the overall functioning of a sensor network.

In many monitoring applications, information is gathered by a sink node at a central location, where the information is processed or stored in a database for future use. As such, routing protocols for monitoring applications are typically designed for point-to-sink traffic patterns (sometimes called 'convergecast') whereby all information is gathered in a central sink. To this end, a large amount of point-to-sink routing protocols have been proposed. A comprehensive study can be found in [38]. The protocols can typically be categorized into one of the following classes.

In *hierarchical routing protocols*, sensor nodes are assigned different routing roles depending on their capabilities. High-energy nodes are often given a coordinating role: they are responsible for aggregating and routing information. For example, the LEACH routing protocol [33] organizes the nodes into local clusters, with one node acting as a coordinator or 'cluster head'. The cluster head can aggregate data from the different nodes and relay this data directly to the sink or to the next-hop (also a cluster-head) that is located closer to the sink. The role of the low-energy nodes is limited to sensing their environment and forwarding the measured data to the most nearby cluster head. When no high-energy nodes are available, the role as the cluster head can be rotated periodically to evenly distribute the energy load.

Data-centric routing protocols are designed specifically for information gathering applications. Instead of creating routes between addressable nodes, packets are processed based on the description of the content that is encapsulated. For example, the directed diffusion protocol [39] starts with one or more sinks that distribute a query indicating an interest for a specific type of information through the network. Devices that receive this query remember from which neighboring node the interest was received. As such, each intermediate device stores a list of next-hop addresses to which specific content can be transmitted. Nodes that are capable of generating information of the correct type with the correct granularity start sending the information to all neighbors from which an interest query was received. Once the sink receives response data from multiple neighbors, one or more of the available paths are reinforced and kept-alive, based on network characteristics such as delay or link qualities. Data-centric approaches can be optimized through the use of multi-path routing [40], more efficient query dissemination [41] or in-network data-aggregation techniques [42].

Finally, *geographic routing protocols* use location information to efficiently gather information. For example, the GEAR protocol [43] uses location information to limit the exchange of queries and information to a small region of interest. Other approaches, such as GAF [44], use the location information to deduce which nodes are redundant for routing purposes. GAF assigns a virtual grid structure over the monitored area. All sensor nodes that are located in the same grid cell are considered equivalent for routing purposes. By keeping only one node active in

each cell, the overall energy consumption can be reduced. To obtain location information, sensor nodes can use GPS (Global Positioning System). Alternatively, the distance between neighboring nodes can be estimated based on the strength of incoming signals.

In conclusion, routing in sensor networks is a well-studied research topic. In addition to existing routing protocols for ad-hoc Wi-Fi networks, the unique challenges of WSNs resulted in a number of innovative (but often application-specific) routing protocols. However, it remains a challenge to predict how these routing protocols perform in specific applications or in combination with a specific MAC protocol [37].

1.5 Outline and research contributions

Existing large-scale WSN deployments are mostly the result of application-specific research [33]. Current WSN solutions often have two significant shortcomings: (i) network protocols are often application-specific and as a result cannot easily be integrated in future deployments and (ii) current networking protocols focus on monitoring applications, rather than interactive, next-generation applications. To efficiently realize next-generation applications for WSNs, the following research challenges need to be solved.

QoS and mobility. When WSNs are used in emergency or tracking applications [10], support for *mobile nodes* should be added. In addition, applications such as e-health services can not be deployed without *QoS guarantees*.

Heterogeneous applications. It should be possible to easily *reuse and combine existing network protocols* that were developed for other applications [37]. In addition, when the applications requirements change, it should be possible to easily change the behavior of the network.

Heterogeneous devices. Many next-generation WSN applications use sensor nodes with very *diverging capabilities* [45]: they contain both simple nodes (such as light switches) and more complex nodes (such as heating controllers). Network solutions for WSNs should be small enough to implement on resource-constrained devices, but should be able to profit from the additional capabilities of more powerful sensor nodes.

Heterogeneous wireless technologies. When embedded devices are introduced in new environments, the number of co-located wireless communication technologies increases [46]. It should be possible to efficiently *integrate new sensor devices with co-located devices that use different (wireless) communication technologies*.

This dissertation argues that current networking approaches are not capable of solving these challenges on resource-constrained devices. Instead, a major paradigm shift is required towards the development of new protocol architectures that inherently cope with these next-generation WSN design criteria.

To this end, Chapter 2 introduces an 'information driven architecture' (IDRA) for wireless sensor networks. IDRA is designed to support next-generation sensor requirements such as heterogeneity, mobility, QoS and energy efficiency at an architectural level. To this end, the IDRA architecture provides shared libraries for network functionality, such as packet creation, packet interaction and buffer management, that is traditionally included in multiple layers of a protocol stack. This approach results in architecture-wide control of the packet behavior. In addition, by sharing these functionalities between different network protocols, the memory requirements of IDRA protocols are reduced by up to a factor two to ten. The performance analysis from Chapter 2 proves that IDRA is able to support the above application requirements, even when using resource-constrained embedded devices.

Chapter 3 discusses how the network lifetime can be increased by using the IDRA architecture. To this end, IDRA reduces the number of packet transmissions by automatically aggregating the information exchanges from *all network layers* into a single packet. In contrast, current data-aggregation approaches for WSNs typically combine only data from the application level [47, 48]. In addition, whereas many existing aggregation approaches are evaluated using simulations or using small-scale sensor networks, the effectiveness of the IDRA aggregation is experimentally evaluated in a wide-range of scenarios with up to 200 sensor nodes. The evaluation results show that IDRA outperforms existing approaches: the number of packet transmissions is reduced by a factor of more than two when compared to existing approaches, thus strongly increasing the network lifetime.

Chapter 4 discusses the role of IDRA in a future 'internet of things' [46]. As more and more sensor devices are installed in the same environment, an increasing number of (wireless) devices will be co-located with devices using different communication technologies. Traditionally, all communication between these devices goes through a remote gateway, which results in inefficient use of the network. To remedy this, this chapter discusses how IDRA can be used to optimize the network performance by enabling direct communication between heterogeneous devices that have multiple communication interfaces.

Finally, Chapter 5 focuses on cognitive networking. Traditionally, installed (sensor) devices are separated into different networks that do not cooperate with each other. As an alternative, this chapter proposes a methodology that takes into account the application and node preferences (called 'incentives') to partition devices into 'communities' of devices that have similar incentives. The described 'incentive-driven' negotiation methodology is designed to support efficient net-

work cooperation between heterogeneous devices through the use of cross-layer and cross-network optimizations. Different communities can cooperate with each other by making (software or hardware) network resources available to other communities, but only if cooperation is beneficial for all involved devices. Chapter 5 describes several possible negotiation and cooperation approaches and demonstrates the feasibility of the methodology through the evaluation of an experimental proof-of-concept implementation.

Finally, Chapter 6 gives the overall conclusions of this dissertation. It is worth noting that all contributions from this dissertation are demonstrated on low-resource sensor devices. It goes without saying that the developed solutions are also scalable towards a wide range of (more capable) devices. In fact, the typical WSN wireless challenges that are the starting point of this work are applicable to many (wireless) communication networks. As such, the concepts from this dissertation can be applied to a wide range of network and communication technologies.

References

[1] Carlos F. Garca-Hernndez, Pablo H. Ibargngoytia-Gonzlez, Joaqun Garca-Hernndez, and Jess A. Prez-Daz. *Wireless Sensor Networks and Applications: a Survey*. IJCSNS International Journal of Computer Science and Network Security, Vol.7 No.3, March 2007.

[2] I.F. Akyildiz, W. Su, Y. Skarasubramaniam, and E. Cayirci. *Wireless sensor networks: a survey*. Computer Networks, 38:393–422, 2002.

[3] Archana Bharathidasan and Vijay Anand Sai Ponduru. *Sensor Networks: An Overview*. Technical report, Technical report, Dept. of Computer Science, University of California at Davis, 2002.

[4] Ning Xu. *A survey of sensor network applications*. IEEE communications magazine, issue 8, Vol. 40:pp. 102, 2002.

[5] *ALERT flood detection, http://www.alertsystems.org/.*

[6] D. Guerri, M. Lettere, and R. Fontanelli. *Ambient Intelligence Overview: Vision, Evolution and Perspectives*. 1st GoodFood AmI Workshop, Florence, July 2004.

[7] Geoff Werner-Allen, Jeffrey Johnson, Mario Ruiz, Jonathan Lees, and Matt Welsh. *Monitoring Volcanic Eruptions with a Wireless Sensor Network*. Second European Workshop on Wireless Sensor Networks (EWSN'05), January, 2005.

[8] P. De Mil, T. Allemeersch, I. Moerman, P. Demeester, and W. De Kimpe. *A Scalable Low-Power WSAN Solution for Large-Scale Building Automation*. In Communications, 2008. ICC '08. IEEE International Conference on, pages 3130–3135, May 2008.

[9] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, R. Stoleru, S. Lin, and J. A. Stankovic. *An Advanced Wireless Sensor Network for Health Monitoring*. Transdisciplinary Conference on Distributed Diagnosis and Home Heathcare (D2H2), Arlington, VA, April 2-4, 2006.

[10] Hairong Yan, Youzhi Xu, and M Gidlund. *Experimental e-Health Applications in Wireless Sensor Networks*. volume 1, pages 563 –567, Jan. 2009.

[11] *Overview of current sensor platforms, Imperial College London, http://ubimon.doc.ic.ac.uk/bsn/m206.html.*

[12] *FP6 Embedded WiSeNts Project. Report 2.1: Critical evaluation of research platforms for wireless sensor network, http://www.embedded-wisents.org/studies/survey_wp2.html.*

[13] *TMoteSky Datasheet, http://www.snm.ethz.ch/Projects/TmoteSky*.

[14] *The IEEE802.15.4 Task Group, http://www.ieee802.org/15/pub/TG4.html*.

[15] Fredrik Osterlind and Adam Dunkels. *Approaching the Maximum 802.15.4 Multi-hop Throughput*. Technical report, Swedish Institute of Computer Science, SICS Technical Report T2008:05, ISSN 1100-3154, ISRN:SICS-T2008/05-SE, March 2008.

[16] Martin Wirz. *BTnode Application for Automated Link Measurements*. Master's thesis, ETH Zurich, 2007.

[17] S. Chalasani and J. M. Conrad. *A survey of energy harvesting sources for embedded systems*. In in IEEE Southeastcon, Huntsville, Ala, USA, pages pp. 442–447, April 2008.

[18] Andreas Meier, Tobias Rein, Jan Beutel, and Lothar Thiele. *Coping with Unreliable Channels: Efficient Link Estimation for Low-Power Wireless Sensor Networks*. In Proc. 5th Intl Conf. Networked Sensing Systems (INSS 2008), pages 19–26, Kanazawa, Japan, June 2008. IEEE.

[19] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. *Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks*. Wireless Sensor Networks. Technical Report CSD-TR 02-0013, UCLA, February 2002.

[20] Kaixin Xu, M. Gerla, and Sang Bae. *How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks*. volume 1, pages 72 – 76, Nov. 2002.

[21] Hubert Zimmermann. *OSI Reference Model The ISO Model of Architecture for Open Systems Interconnection*. IEEE Transactions on Communications, 28, no. 4:425 – 432, April 1980.

[22] *The IEEE802.11 Task Group, http://www.ieee802.org/11*.

[23] Sunil Kumar, Vineet S. Raghavan, and Jing Deng. *Medium Access Control protocols for ad hoc wireless networks: A survey*. Ad Hoc Networks, 4(3):326 – 358, 2006.

[24] I. Demirkol, C. Ersoy, and F. Alagoz. *MAC Protocols for Wireless Sensor Networks: a Survey*. IEEE Communications Magazine, 2005.

[25] G. P. Halkes, T. van Dam, and K. G. Langendoen. *Comparing energy-saving MAC protocols for wireless sensor networks*. Mobile Network Applications, Vol. 10(5):pages 783–791, 2005.

[26] W. Ye, J. Heidemann, and D. Estrin. *An Energy-efficient MAC Protocol for Wireless Sensor Networks*. In 21st Conference of the IEEE Computer and Communications Societies (INFOCOM), volume 3, pages 1567–1576, June 2002.

[27] T. van Dam and K. Langendoen. *An adaptive energy-efficient MAC protocol for wireless sensor networks*. In 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, CA, pages pp 171–180, November 2003.

[28] P. Lin, C. Qiao, and X. Wang. *Medium access control with a dynamic duty cycle for sensor networks*. IEEE Wireless Communications and Networking Conference, Volume: 3:Pages: 1534 – 1539, 21-25 March 2004.

[29] A. El-Hoiydi. *Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks*. In IEEE International Conference on Communications (ICC), New York, April 2002.

[30] J. Hill and D. Culler. *Mica: a wireless platform for deeply embedded networks*. IEEE Micro, 22(6):12–24, November 2002.

[31] M. Buettner, G. Yee, E. Anderson, and R. Han. *X-MAC: A Short Preamble MAC Protocol For Duty-Cycled Wireless Networks*. In SenSys06, pages 307–320, Boulder, CO, November 2006.

[32] A. El-Hoiydi and J.-D. Decotignie. *WiseMAC: An Ultra Low Power MAC Protocol for the Downlink of Infrastructure Wireless Sensor Networks*. Proceedings of the Ninth International Symposium on Computers and Communications, ISCC 2004, Vol.1:pp. 244–251, 28 June-1 July 2004.

[33] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. *An application-specific protocol architecture for wireless microsensor networks*. Wireless Communications, IEEE Transactions on, 1(4):660–670, Oct 2002.

[34] Injong Rhee, A. Warrier, M. Aia, Jeongki Min, and M.L. Sichitiu. *Z-MAC: A Hybrid MAC for Wireless Sensor Networks*. Networking, IEEE/ACM Transactions on, 16(3):511 –524, June 2008.

[35] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves. *Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks*. Proceedings of ACM SenSys 03, Pages:181 - 192, Los Angeles, California, 5-7 November 2003.

[36] L. van Hoesel and P. Havinga. *A lightweight medium access protocol (LMAC) for wireless sensor networks*. 1st International Workshop on Networked Sensing Systems (INSS 2004), Tokyo, Japan, June 2004.

[37] Jono Vanhie-Van Gerwen, Eli De Poorter, Benoît Latré, Ingrid Moerman, and Piet Demeester. *Real-Life Performance of Protocol Combinations for Wireless Sensor Networks*. International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 0:189–196, 2010.

[38] K. Akkaya and M. Younis. *A survey of routing protocols in wireless sensor networks*. Elsevier Ad Hoc Network Journal, Vol. 3, no. 3:pp 325–349, 2005.

[39] C. Intanagonwiwat, R. Govindan, and D. Estrin. *Directed diffusion: a scalable and robust communication paradigm for sensor networks*. In Mobile Computing and Networking, pages 56–67, 2000.

[40] R. Shah and J. Rabaey. *Energy aware routing for low energy ad hoc sensor networks*. Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Orlando, FL, March 2002.

[41] D. Braginsky and D. Estrin. *Rumor routing algorithm for sensor networks*. Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Atlanta, GA, October 2002.

[42] J. Gehrke and S.l Madden. *Query Processing in Sensor Networks*. IEEE Pervasive Computing, 3(1):46–55, 2004.

[43] Y. Yu, D. Estrin, and R. Govindan. *Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks*. UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023, May 2001.

[44] Y. Xu, J. Heidemann, and D. Estrin. *Geography-informed energy conservation for ad hoc routing*. Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom01), Rome, Italy, July 2001.

[45] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. *Exploiting heterogeneity in sensor networks*. In Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Infocom 2005), March 2005.

[46] *The Internet of Things*. ITU Internet Reports, 2005.

[47] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. *In-network aggregation techniques for wireless sensor networks: a survey*. Wireless Communications, IEEE, 14(2):70–87, April 2007.

[48] R. Rajagopalan and P.K. Varshney. *Data-aggregation techniques in sensor networks: a survey*. Communications Surveys & Tutorials, IEEE, 8(4):48–63, Fourth Quarter 2006.

2

IDRA: an Information DRiven Architecture for Wireless Sensor Networks

This chapter introduces the IDRA protocol architecture: an information driven architecture designed to support next-generation applications on wireless sensor networks. IDRA supports simple but useful optimizations at an architectural level. These include support for cross-protocol interactions, energy efficiency optimizations, QoS optimizations (packet priorities, dynamic protocol selection), mobility support and heterogeneous network support. By using an architecture that delegates specific tasks to a central system, the memory requirements of associated network protocols are decreased (at the expense of a bigger memory footprint of the IDRA system). Finally, a thorough experimental performance analysis demonstrates that IDRA is much more scalable than traditional system architectures in terms of memory requirements, energy requirements and processing overhead. More information, example code fragments and tutorials are available on http://idraproject.net, where the IDRA architecture is released as an open-source project.

2.1 Introduction

Wireless sensor networks were originally designed for large inaccessible areas. However, more recently, WSNs are used for more advanced applications such as wireless building automation, industrial process automation, security monitoring, disaster intervention and medical interventions. These applications benefit greatly from the flexibility and low deployment cost of WSNs. However, these next-generation applications impose many network requirements which are not found in traditional WSNs.

- In addition to point-to-sink traffic, *more complex communication patterns* (such as multicast and point-to-point traffic) must also be supported.
- Many future applications use sensor nodes with very diverging capabilities [1]. As a result, future WSNs will become *heterogeneous*, containing both simple nodes (such as light switches) and more complex nodes (such as heating controllers).
- Many commercial applications require mass-produced sensor nodes which are cheaper and even smaller, sometimes up-to-the point where sensor nodes can be implanted. As a result, new ways have to be found to ensure that network protocols have an *even smaller memory footprint* and *consume even less energy*.
- To provide sufficient end-user support, a WSN must be easy to update and maintain. *Run-time addition* of new services and network protocols should be supported.
- Sensor nodes can be used to monitor objects or persons. For these applications, *mobility* should be supported by the network protocols.
- Finally, *Quality-of-Service (QoS) requirements* can no longer be ignored [2]. Medical, security and surveillance applications require that each application has its own set of specific QoS requirements.

Due to these more and more challenging network requirements, developing network protocols for WSNs becomes an increasingly complex issue.

2.1.1 The need for new architectures

A protocol architecture is a structured way to allow the combination of -and interaction between- networks protocols. Even though a large number of WSN-specific network protocols have been developed, these network protocols largely

use the OSI-reference protocol architecture, which was not designed for resource-constrained devices. Rather than focusing on the design of optimal network protocols, we strongly believe that redesigning the protocol architecture is a much more promising approach. As stated by Culler et. al: "*the primary factor currently limiting progress in sensornets is not any specific technical challenge but is instead the lack of an overall sensor network architecture*" [3]. As such, there is a strong need for new architectures that inherently cope with the increasingly challenging network requirements of WSNs. The resulting architecture should ease the integration of network protocols, should support cross-protocol optimizations and have a very low implementation complexity to support even sensor nodes with very limited capabilities. At present, there is no architecture that supports all of these challenges.

Therefore, this chapter presents several architectural techniques that can be used to (i) reduce the complexity of developing new network protocols for WSNs, (ii) support advanced network requirements such as QoS and (iii) support heterogeneous networks. For each of the proposed architectural optimizations, experimental measurements are given that describe how the network performance is improved or, alternatively, which performance penalty is associated with the increase in network flexibility. Finally, this chapter evaluates the performance of a system in which all these individual optimizations are combined. Based upon our results, architecture designers should immediately be able to decide whether or not a certain optimization technique is suited for their network requirements. With this detailed overview of the advantages and costs of architectural improvements, we aim to persuade the research community that architectural design is equally important as purely protocol development.

2.1.2 Remainder of the chapter

Section 2.1 discussed how WSN characteristics complicate the design of applications and network protocols. Based upon this discussion, IDRA was designed with three main goals in mind. *(i)* Section 2.2 illustrates how IDRA simplifies the design of network protocols. *(ii)* Next, Section 2.3 discusses how IDRA is able to support the advanced network requirements that are needed to enable next-generation applications sensor applications, such as support for energy efficiency, for diverging application requirements, QoS and mobility. *(iii)* Finally, Section 2.4 demonstrates that the resulting architecture performs efficiently even in strongly heterogeneous networks. A comprehensive evaluation of all the presented techniques and optimizations discussed in this chapter can be found in Section 2.6. Finally, Section 2.7 compares IDRA with existing architectures for wireless sensor networks and Section 2.8 concludes the chapter.

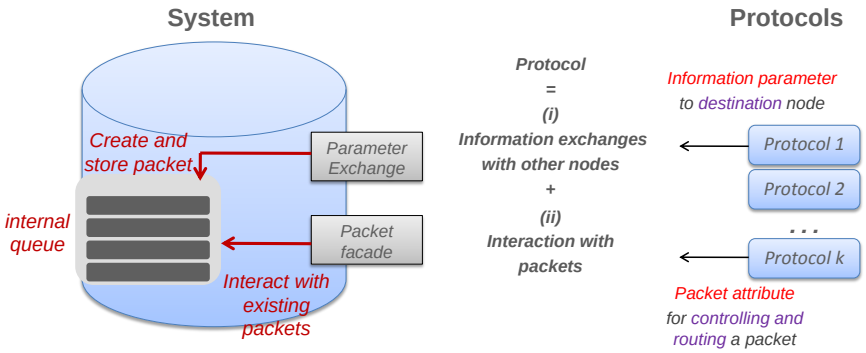

Figure 2.1: In the IDRA architecture, the role of a network protocol is simplified to two tasks: (i) exchanging information and (ii) interacting with packets.

2.2 Simplifying network protocols

Currently, implementing a network protocol is time-consuming and complex. Besides formulating a fully functional algorithm, many unrelated issues must be solved. More specifically, each protocol layer must *(i)* define a message format (including header and trailer fields), *(ii)* provide buffers to temporarily store packets and *(iii)* gather information from other nodes. We argue that this approach is very inefficient. The main responsibility of a network protocol is to ensure that information is relayed to the correct destination. It makes no sense that every individual protocol layer has to bear the burden of gathering information, providing buffers and implementing header manipulations. Such functions, which are repeated in each protocol layer, should be implemented in a single shared library.

In the IDRA architecture, protocol designers have to consider only the 'information exchanges' when implementing a network protocol. Other tasks, such as packet creation and buffer provisioning, are delegated to the architecture. As a result, network protocols are simpler and require less memory. In effect, the role of a network protocol is simplified to its 2 main tasks (see Figure 2.1): *(i) exchanging information* and *(ii) interacting with the relayed information*.

2.2.1 The exchange of information

Network protocols often exchange information with a remote node. Typical examples of exchanged information are:

- An application sends measured *data values*, such as the 'ROOM_TEMPERATURE', to a central monitoring node;
- a clustering protocol sends *status information* (e.g. 'ENERGY_REMAINING') to all neighboring nodes;

- or a routing protocol sends *control information* (such as a 'ROUTE_REQUEST') to a remote node.

Using our information driven approach, network protocols do not create a new packet to send these types of information to a remote node. Instead, they rely on the system to send and receive information. To *send information* to a remote node, the protocol hands over an information parameter to the system, together with the required destination. The system will transparently create a new packet, encapsulate the parameter into this packet and store the packet in a system-provided queue (Figure 2.1, interaction i). Whenever a packet arrives at its final destination, the system extracts the encapsulated information parameters from the packet and distributes them to the interested protocols and applications. When the packet contains no more information parameters, the empty packet is dropped by the system.

The main advantages of transferring the creation of packets to the system are: *(i)* the system can ensure that similar control information is sent only once; *(ii)* multiple interested network protocols can act upon the same exchanged information; *(iii)* protocols are simpler since they do not need to create packets and do not need to interact with packet buffers; and *(iv)* multiple information parameters can be combined into a single packet, so that the number of required packets decreases drastically (see Section 2.3.1).

2.2.2 Interacting with packets

Even when the system encapsulates the exchanged information in a packet format, network protocols must still interact with the forwarded packets. Traditionally, protocol layers do this by associating information with passing packets in the form of a (fixed size) packet header that precedes the packet payload. A packet header typically contains multiple header fields that contain control information. As pointed out in [4], this solution is inflexible, since information that is contained in the headers is not available for higher layers, which limits cross-layer optimization possibilities.

In our information driven architecture, protocols are not tasked with header creation or manipulation. To ensure that network protocols can interpret all incoming packets, network protocols use a 'Packet Facade' to interact with packets (Figure 2.1, interaction ii). Using this packet facade, protocols can associate packet attributes with a packet, such as 'source', 'destination', 'QoS ID' or 'time-to-live'. The protocols do not require any knowledge about the actual packet structure. Instead, the packet facade is responsible for the storage and retrieval of the packet attributes. Added packet attributes can be interpreted by any network protocol, not only the protocol that added the attribute. As a result, *the protocol logic and packet representation are effectively decoupled, and no information is hidden from higher layers*.

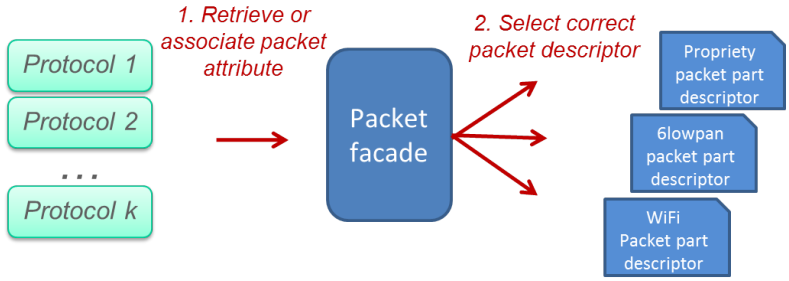

Figure 2.2: Through a packet facade, protocols interact with packets. Protocols do not require any knowledge about the actual packet format.

To correctly store and retrieve packet attributes, the packet facade should know how each packet is constructed. This information is stored in packet part descriptors (Figure 2.2) . *Packet part descriptors* describe how and where packet attributes are stored in a header (e.g: the header offset, the byte-ordering, the number of allocated bits, etc.). Examples of packet part descriptors are an IEEE 802.15.4 header, an IEEE 802.11 Wi-Fi header or an IP header.

New packet types can be created by combining multiple packet part descriptors. A *packet type* is defined as a unique sequence of one or more well-defined packet part descriptors. For example, an IEEE 802.15.4 packet part descriptor can be combined with a 6LoWPAN packet part descriptor to create IPv6 compatible packets. To create new packet types, developers can combine existing packet part descriptors, or develop new propriety packet part descriptors. Alternatively, a network designer can design a single highly optimized packet part descriptor that efficiently compresses all packet attributes that are used in his specific network scenario. Finally, it is important to note that network protocols are not limited to the use of (standardized) packet attributes. Packet attributes that are not recognized by any of the packet part descriptors are stored sequentially in the payload using a type-length-value (TLV) representation.

Using a packet facade to associate attributes with a packet has the following advantages: *(i)* protocol development is simplified, since there is no need to define headers or header operations; *(ii)* packet attributes have a system-wide significance: they can be inspected by the system or by any other protocol; *(iii)* multiple packet types can be supported: the transmitted packet type can transparently be changed without any changes to the protocols (e.g.: 6LoWPAN, IEEE 802.15.4 or a custom packet).

2.2.3 A system-wide shared queue

Finally, the system created packets must be stored. Layered networks use a 'store-and-process' approach, wherein each network layer stores its own packets. Each protocol requires a large enough internal queue to ensure that all received packets can be stored. Thus, the total amount of buffer memory increases linearly with the number of protocol layers (Figure 2.3).

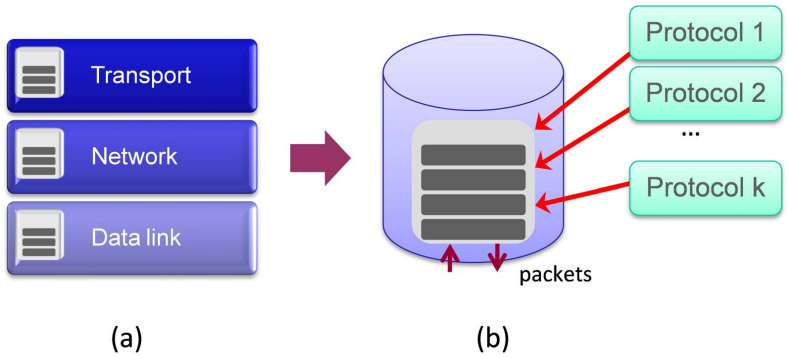

Figure 2.3: (a) In traditional layered architectures, each network layer allocates a packet buffer. (b) In a shared queue approach, only one single, system-wide shared packet buffer is allocated.

As part of the simplification process, the IDRA system is also responsible for storing created and incoming packets (Figure 2.1, internal queue). Arriving packets are stored once in a system-wide shared queue and remain there until processing is finished. This limits the total number of copy actions in the IDRA system. Network protocols can interact with any of the packets from the shared queue using the packet facade.

As stated by several authors [5, 6], the use of a shared, system-managed queue has several advantages: *(i)* protocols are simpler and smaller since they do not have to allocate queue memory; *(ii)* packets do not need to be copied between protocols, resulting in less processing overhead; *(iii)* since the queue occupation from all protocols is averaged, less total queue memory is required; and *(iv)* monitoring and managing the total number of packets in the system is simpler.

2.3 Advanced architectural optimizations

Next-generation WSN applications should not only be energy efficient, but they also require support for QoS and mobility. Moreover, as sensor networks become increasingly interactive, the application requirements can change from time

to time. This section discusses how the IDRA architecture efficiently copes with these next-generation WSN requirements.

These next-generation requirements can not be solved by adapting a single network protocol. On the contrary, efficient support for requirements, such as quality-of-service or mobility, requires the redesign of several network layers and requires advanced cross-layer cooperation. In an ideal situation, these features should be addressed independently from the MAC and routing strategy. This way, developed QoS and mobility solutions can be combined with any existing routing or MAC protocol. This separation is only possible when advanced WSN requirements are a part of the architectural design. It is well known that supporting additional features after the design phase of an architecture is increasingly difficult. In fact, *the lack of architectural support for energy efficiency, QoS, mobility and heterogeneity in existing WSN architectures is a major obstacle that hampers the deployment of many next-generation applications for WSNs*.

This section demonstrates how these next-generation applications requirements can be supported at an architectural level in our information driven system. Thus, our optimizations can be used to transparently enhance existing network protocols with a basic form of QoS and mobility. As a result, IDRA will facilitate the support of QoS and mobility without the need for changes to existing MAC and routing protocols.

2.3.1 Energy efficiency

In contrast to traditional networks, wireless sensor networks are typically battery powered. Even with a limited battery, a sensor network should have an operational lifetime of at least several years without the need for any manual intervention. In WSNs, most energy is spent when the radio is active. By periodically turning off the radio, the network lifetime increases significantly [7]. However, this approach is only feasible if the number of transmitted packets is very low. To reduce the number of packet transmissions, 'data aggregation' is often applied, which is a technique in which multiple measured data values are combined in a single packet.

Aggregation in WSNs is a well studied research topic [8, 9], on which many specialized aggregation protocols have been proposed. However, these are generally highly optimized for very specific types of traffic flows, and they often require complex fine-tuning to set-up optimal aggregation routes. To remedy this, IDRA contains an in-built aggregation function which can be activated when no other aggregation protocols are provided. This aggregation is part of the architecture and is 'non-intrusive': no fine-tuning of aggregation settings is required and no additional control messages are sent.

Our main assumption is that not all packet types need to be forwarded immediately. Control packets generated by protocols (e.g. routing, power management,

status information) often have a periodic character. Measurements, such as temperature or remaining battery capacity do not vary a lot between subsequent status updates. Therefore, it is reasonable to assume that these packets are not very time-sensitive, and can be delayed for a short amount of time before being sent.

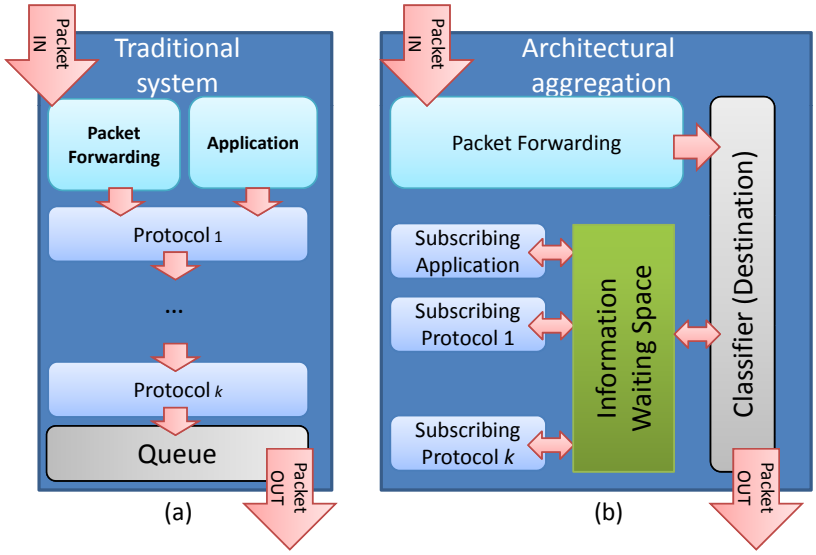

Figure 2.4: Extending the data aggregation concept. (a) Traditional architecture. (b) Architectural support for aggregation.

Whenever a protocol requests the sending of a parameter, the protocol also provides information regarding the maximum delay after which the parameters should be sent. Before encapsulating the parameters in a packet, the system collects the parameters in a central repository, called the *waiting space* (Figure 2.4). Whenever a packet is relayed through the node, all information parameters to the same 'next hop' or 'destination' address are added to the packet. Delay-tolerant parameters can remain in the waiting space for up to a per-parameter predefined period of time. If no data has been relayed within the allowed waiting time, the system generates a new packet which combines all parameters that are destined for the same node. To prevent the end-to-end delay from becoming too high, parameters are only delayed in the waiting space of the initial node: packets are not further delayed in intermediate nodes.

In contrast with traditional aggregation protocols, an architectural approach has three main advantages: (i) both application level information and network level control information can be combined, (ii) since aggregation is executed at an architectural level, the aggregation approach is compatible with any networking protocol, and (iii) this approach does not require any communication overhead

between different nodes.

2.3.2 Supporting diverging application requirements

Sensor networks are used for increasingly complex applications, from controlling thermostatic elements to security and health monitoring applications. These applications have very diverging network requirements in terms of QoS (reliability, maximum delay, etc.) and network characteristics. In addition, a sensor network is typically not an independent entity, but should interact with the outside world. As a result, network protocols for WSNs are becoming increasingly complex: they should support QoS requirements, they should interact with a diverse number of communication technologies, and at the same time remain simple enough to implement on a resource constrained sensor node.

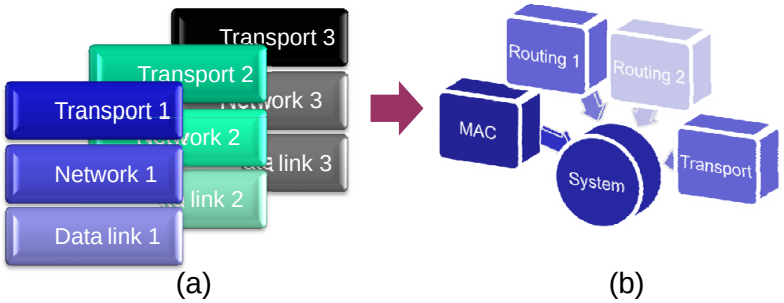

Figure 2.5: (a) Using a traditional layered approach, the order and types of network protocols are fixed. (b) In IDRA, new network protocols can dynamically be added per application requirement. The most optimal network protocols are automatically selected by the system.

IDRA uses an alternative approach in which the network designer is given the option to deploy multiple smaller and more specialized network protocols on a single node. Whenever a packet requires processing, the system is responsible for choosing and activating the most optimal network protocol (Figure 2.5). For example, a single node can contain: (i) an efficient broadcast protocol for disseminating information; (ii) a data-centric routing protocol for collecting measured data; (iii) a label switching protocol that creates virtual links between different nodes; and finally (iv) a protocol for routing packets to an external network. The architecture is able to dynamically change between these different routing or MAC protocols at run-time.

Currently, IDRA implements a simple filter-based solution to select the most optimal network protocol for each packet. Each IDRA protocol must register itself by adding one or more *filters* to the system. These filters describe the function

of the network protocol and indicate for which packets the protocol is optimized. Consider the following examples:

- A voice routing protocol adds a filter 'QoS_label>5'. All voice packets that require high QoS guarantees will be routed using this specialized protocol.
- A routing protocol implements an efficient broadcast algorithm. It registers itself using the filter 'address== BROADCAST_ADDR'.

Through the packet facade, the system checks if the attributes of the arriving packets match any of the registered filters. IDRA selects the network protocol with most matching filters to process the packet. When no filters match, a default routing or MAC protocol is chosen.

In layered architectures, network protocols are executed in a fixed sequence so that each layer can remove the packet header from incoming packets. In contrast, IDRA users can specify in which order protocols should be executed by defining 'call sequences'. The default IDRA implementation contains a simple, deterministic call sequence that will suffice for most networks. The default call sequence (application $\rightarrow$ routing protocol $\rightarrow$ MAC protocol) mimics very closely the behavior of traditional layered architectures. However, developers can define new call sequences to design more flexible systems. It is possible to execute network protocols in any order, and even change the execution sequence at run-time. Section 2.4 discusses how these advanced call sequences can be used to support heterogeneous networks.

Using this flexible protocol selection approach has several advantages: (i) smaller protocols that fulfill only a single function can be used; (ii) these smaller protocols are more suited for resource constrained devices: they are often more stable and are easier to maintain than monolithic protocols; (iii) the performance of the network can be optimized by switching between different network protocols, depending on the network circumstances [10].

2.3.3 Quality-of-Service

Before sensor networks can be used for critical and time-sensitive applications, WSNs should be able to deliver Quality-Of-Service (QoS) guarantees. IDRA is optimized for the design of transparent QoS solutions:

- Since all packets are stored in a shared packet queue, the system can monitor all available packets. As a result, the QoS module has a clear view on the number of packets, their current processing state and their expected delay.
- The QoS module can influence the order in which packets are processed and which packets should be transmitted first.

- Through the packet facade, the QoS module can read and modify the attributes of relayed packets at any processing stage. This information can be taken into account for intelligent packet selection and dropping strategies.
- Using dynamic protocol selection, packets with strict QoS requirements can be processed by specialized protocols.
- Analyzing which parameters can be aggregated with relayed packets results in additional processing delay. Therefore, the QoS module has the option to disable aggregation for high-priority packets. As a result, additional delay will only be introduced for low-priority traffic.

A single QoS module that is part of the IDRA system has control of all stored packets. By rewriting this module, new QoS solutions can be implemented. To implement new QoS logic, an interface is available through which the following commands can be given to the system:

- drop a specific (low-priority) packet (useful when the queue is full);
- select which packet should be processed first;
- put the processing of low-priority packets on hold (even when those packets are currently being processed by a protocol);
- activate the most suitable network protocol, depending on the characteristics of each packet;
- indicates which packet should be transmitted first;
- enable or disable aggregation on a per-packet basis.

Together, these commands can be used to design a wide range of possible QoS solutions. The developed QoS controller is *protocol independent*: it can transparently be combined with any IDRA network protocol. Of course, the developed QoS solution can also be combined with QoS aware network protocols for even more sophisticated results [11]. During the evaluation of IDRA, it will be demonstrated that these system commands suffice to transparently add simple QoS features to existing IDRA protocols.

2.3.4 Mobility support

Wireless sensor networks are also often used for localization or tracking purposes [12]. However, the presence of mobile nodes has a profound influence on the performance of the network. Whenever a node moves, all routes that involve this node have to be set-up again. In addition, the MAC protocol must ensure that communication is possible with all new nodes that arrive in the neighborhood.

IDRA provides the following features to facilitate the design of mobility-aware network protocols.

- A shared neighbor table is provided, which stores (network) statistics for each neighboring node.
- To discover new (or leaving) sensor nodes, the default call sequence can easily be extended with a neighbor discovery protocol (Section 2.3.2), that can be made responsible for updating the neighbor table and adding new neighbors.
- Whenever a neighbor is removed, all network protocols are informed about these changes. This way, the routing protocol knows that it should update all routes that involve this node.

The added neighbor discovery protocol can be either active or passive. *Active neighbor discovery protocols* regularly broadcasts information to all neighboring nodes. Newly discovered nodes are added to the neighbor table, whilst all nodes that do not respond are removed. A *passive neighbor monitoring protocol* typically does not exchange messages. Whenever a packet is received from a node, this node is added to the neighbor table. When no packets have been received from a node during a predetermined period, the neighbor is removed. As a result, passive neighbor discovery requires less communication overhead, but reacts more slowly to topology changes.

2.4 Towards heterogeneous networks

The previous sections presented several optimizations for (i) simplifying the development of network protocols and (ii) supporting advanced network requirements such as energy efficiency, dynamic protocol selection, QoS and mobility. These features are adequate for developing next-generation sensor applications. However, in the long term, WSNs will become increasingly heterogeneous:

- New, next-generation sensor nodes will be added to existing (legacy) WSNs.
- Advanced wireless sensor networks are often deployed on hardware with very diverging capabilities: from light switches to air-conditioning controllers [1, 13].
- Support for interaction with surrounding networks becomes increasingly important. An example is 'the internet of things' [14], which describes a vision in which any object is connected to any other object.

As a result, future sensor networks will be more diverse in regards to the capabilities of the sensor nodes. These sensor nodes can differ in terms of:

- *node capabilities* (diverging memory, processing or energy provisions);
- *communication methods* (different network protocols, packet types or radio technologies).

This section describes how IDRA can be used to facilitate this transition towards strongly heterogeneous networks.

2.4.1 Diverging node capabilities

Typical sensor nodes are too simple for complex tasks such as intrusion detection, equipment tracking or for controlling advanced machinery. When additional interaction with the environment is required, more capable nodes ('actuator nodes') are added to the WSN. These actuator nodes are sometimes connected to the power grid, and are often equipped with secondary communication interfaces (e.g. wired or Wi-Fi).

The availability of more capable nodes is often known at the design time of the network. As such, the network protocols can take into account the capabilities of the available nodes.

1 Non-essential protocols can be omitted from nodes with little capabilities. Typical functions that can be delegated to more capable nodes include data aggregation, position discovery and mobility detection.

2 In addition, the system can execute simpler protocol implementations on lightweight nodes. A typical example is the use of a clustered MAC protocol, in which advanced nodes ('clusterheads') calculate the optimal slot assignments and distribute these to the less capable nodes.

An example is shown in Figure 2.6. Based on their capabilities, four types of nodes are defined: lightweight, advanced, actuator and computing nodes. The lightweight nodes have very limited resources and only support basic functionalities, hence only the basic protocols are implemented. For example, the routing protocol can be as simple as forwarding sensed data to a more advanced node. The advanced nodes have more sophisticated functionalities. They implement functionality that is needed for a scalable and energy efficient WSN, such as advanced routing, clustering and the IDRA aggregation functionality. As such, advanced nodes typically fulfill the functions of relay nodes. Next, actuator nodes can be programmed to control individual sensor nodes. As such, they directly manage a number of sensor nodes. Finally, the computing nodes are the most powerful nodes and have a much larger battery capacity (e.g.connected to the power grid) and computing power. They offer additional services, such as mobility support and network monitoring, which are not required for data gathering and relaying, but

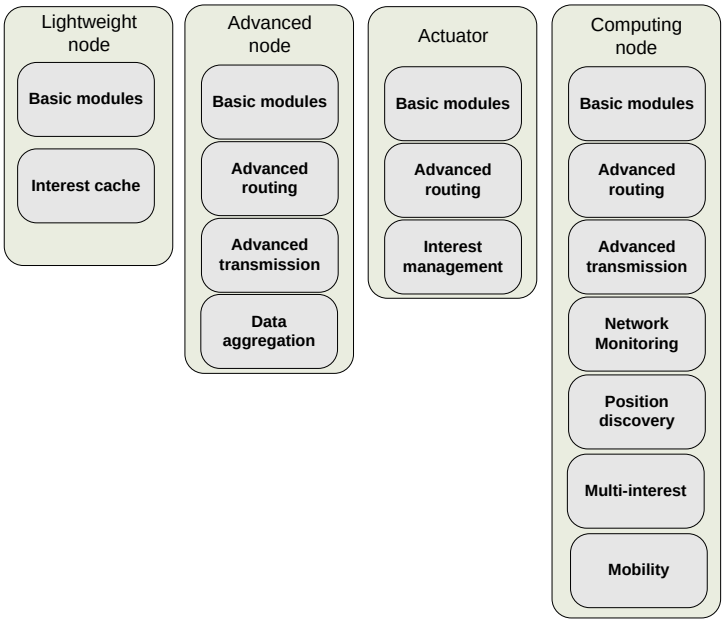

Figure 2.6: Depending on their capabilities, the number of protocols can be varied.

are necessary for more demanding applications. For example: a position discovery protocol can present valuable information for optimizing the routing protocol or for supporting mobility.

To enable such flexible systems, IDRA developers can design different execution sequences based on the capabilities of each node. By defining custom call sequences, very flexible systems can be implemented (see Section 2.3.2). In layered architectures, omitting network protocols would result in conflicts (for example, by not or incorrectly removing protocol headers). However, in IDRA, packet attributes remain associated with a packet even if network protocols are omitted at intermediate nodes. As such, the IDRA architecture can be customized for both high capacity and low capacity nodes.

2.4.2 Different communication technologies

Current wireless sensor networks typically consist of only a single network technology. However, new technologies are constantly being developed and integrated in existing networks. These co-located technologies will need to cooperate to perform efficiently. By allowing communication between (co-located) networks the network performance increases. For example, sensor packets can be routed over a co-located mesh network to obtain a shorter end-to-end delay for the WSN (Fig-

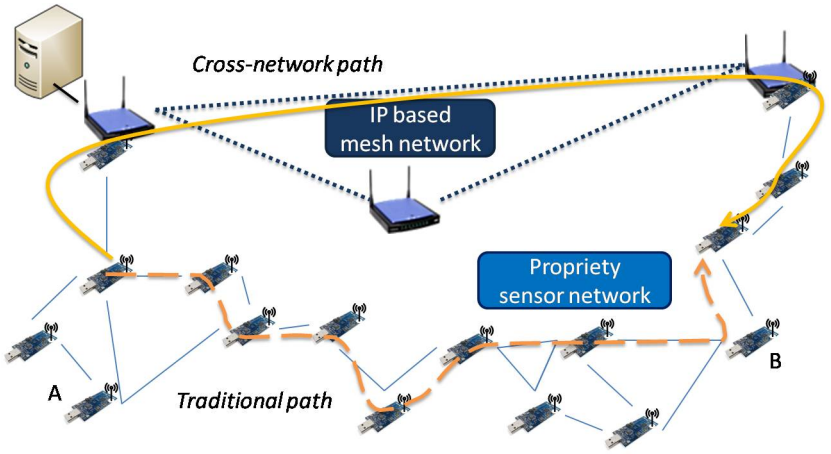

Figure 2.7: Routing a packet over multiple co-located network technologies can result in more efficient paths with shorter delays.

ure 2.7). Connecting different types of networks can also provide business advantages: when checking e-mails on a cell phone, rather than using an expensive 3G network, the cell phone can instead use bluetooth to connect with a body area network (BAN). The BAN, in turn, can make a connection with a nearby Wi-Fi gateway to provide cheap internet access.

IDRA includes a shared neighbor table that can be used to associate information with neighboring devices. For each neighbor, an entry can be made in the shared neighbor table that indicates the preferred packet type, routing protocol and MAC protocol. IDRA will automatically select the correct MAC protocol and sent the preferred packet type over the correct radio interface. More details about how IDRA can be used to transparently connect heterogeneous devices that use different communication technologies will be given in Chapter 4.

2.4.3 Porting legacy protocols to IDRA

Finally, it is possible to port existing network protocols to IDRA. Three changes need to be made to the internal logic of existing network protocols before they can be used in the IDRA framework. (i) Instead of creating packets to exchange information, protocols and applications hand over a parameter to the global aggregation architecture. The architecture will either create a packet to send the parameter, or add the parameter to a passing packet. (ii) Protocols and applications do not inspect the payload of received packets. Instead, the architecture extracts from received packets all the parameters that reached their destination and distributes them to all interested network protocols or applications.

2.5 IDRA implementation

The concepts discussed in the previous sections have been implemented using the TinyOS [15] operating system. Run-time addition of protocols is currently not supported, since TinyOS does not support dynamic code updates. First, the functionality of each component is described. Afterwards, the packet processing flow will be discussed in more detail.

2.5.1 Internal components

An overview of the IDRA implementation is shown in Figure 2.8. The IDRA implementation has the following internal components.

Parameter sending / dispatching: through this component, protocols and applications can distribute information parameters to other nodes, and receive information parameters from other nodes (cfr. Section 2.2.1). When information parameters do not need to be sent immediately, they are temporarily stored in an internal parameter queue so that they can be added to forwarded packets (cfr. Section 2.3.1).

Internal logic: this component manages the packet queue (cfr. Section 2.2.3). For each packet, status information is stored that indicates which protocols already processed the packet, whether or not the packet is ready for sending, etc.

QoS manager: at any moment, the QoS manager can drop a packet from the queue, change the priority of a packet, or indicate to the system which packet should be processed or transmitted first (cfr. Section 2.3.3).

Protocol Selection: this component is responsible for selecting which network protocols should process each packet. Protocols add filters to this component to indicate for which packet types the protocol is optimized (cfr. Section 2.3.2).

Packet Facade: the packet facade is used by network protocols to interact with packets (cfr. Section 2.2.2) and by the 'parameter sending' component to create new packets (whenever the acceptable delay of a stored information parameter has been exceeded).

Neighbor table: network protocols can use this table to store information about neighboring devices, such as link quality, battery status, etc.

Node information: this table is used as a general purpose 'whiteboard' where network protocols can store and retrieve status information.

System Settings: through this component, network protocols can read system information, such as the total number of transmitted or received packets, or update system settings, such as the node ID.

Transmission Settings: through this component, MAC protocols manage the sending of packets. It has provisions for (i) requesting how many packets from the shared queue are ready for sending, (ii) ordering the system to send a specific packet, and (iii) changing the radio settings.

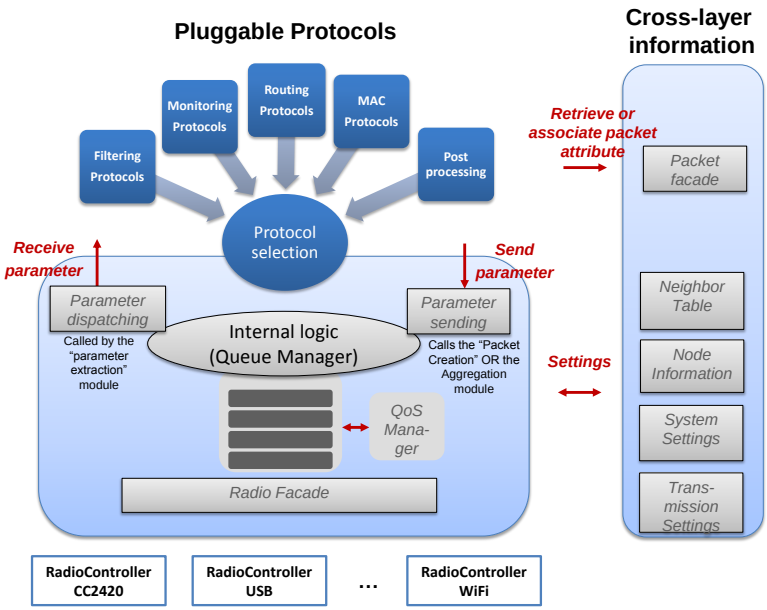

Figure 2.8: Overview of the IDRA implementation.

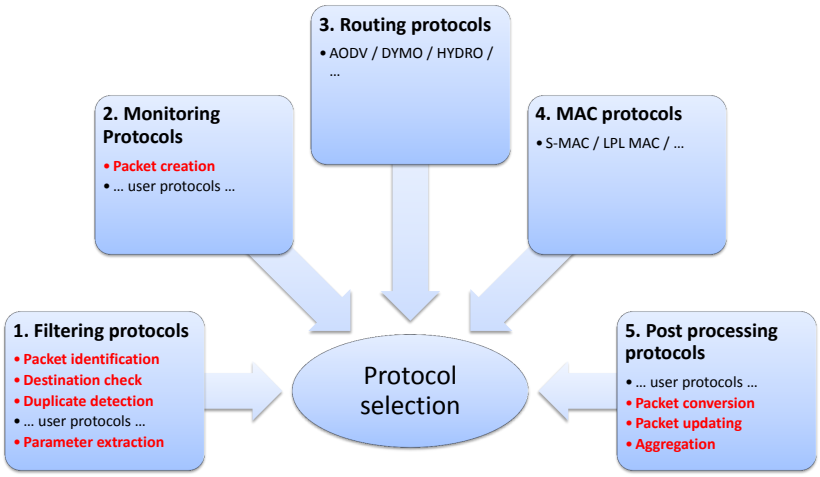

Figure 2.9: Protocol sequence of the IDRA implementation.

2.5.2 Processing flow

To better understand the internal logic of the IDRA implementation, this section describes how packets are processed.

1. *Storing incoming packets*. To limit the number of copy actions, the radio controllers can store incoming packets directly in the IDRA queue. To this end, the radio controller asks the radio facade at which queue entry the next incoming packet should be stored. Whenever a packet is copied to the queue, the radio facade is notified and provides the radio controller with a new queue entry to store the next incoming packet. After updating the packet metadata of the newly stored packet (RSSI, LQI, packet status), the radio facade indicates to the queue manager that the packet is 'ready for processing'.

2. *Selecting a packet for processing*. This step starts when a new packet is stored in the shared queue. This occurs when either (i) the queue manager ('Internal Logic') is notified by the radio facade that a new packet has arrived, or (ii) when the 'parameter sending' module creates a new packet to encapsulate an information parameter. If multiple packets are available, the queue manager is responsible for selecting which packet should be processed first. Unless otherwise notified by the QoS manager, the packet with highest priority is selected first for processing. If multiple packets with the same priority are present, precedence is given to the packet that arrived first.

3. *Processing the packets*. Next, the 'protocol selection' component is responsible for selecting which protocol should process the packet. Section 2.5.3 will describe in more detail how the protocol selection component decides which network protocols should be executed. Whenever a network protocol finishes processing a packet, the status of the packet is updated and the queue manager chooses which packet should be processed next. When the protocol selection component indicates that no more protocols remain to process the packet, the packet status of this packet is updated to 'ready for sending'. If multiple packets are available for sending, the QoS manager can indicate which packet should be transmitted first.

4. *Sending the packet*. The 'ready for sending' packet is stored until the MAC protocol uses the 'Transmission Settings' to indicate when and how (e.g. transmission power, radio channel, ...) the packet should be transmitted. The 'Radio Facade' selects which radio interfaces should be used to transmit the packet (cfr. Section 2.4.2) and forwards the packet to the selected radio controllers.

The next section describes in more detail step 3 of the processing flow, i.e. how does the 'protocol selection' component decides which protocols should process a packet.

2.5.3 Protocol selection

Whenever an IDRA protocol finishes processing a packet, the protocol signals one of the following return values to the system:

- SUCCESS The network protocol finished successfully; the next protocol can be executed.
- FAIL The network protocol can not process the packet; the packet should be dropped from the shared queue.
- BUSY The network protocol is not yet ready to process the packet; the protocol will be called again at a later time.

An IDRA protocol can implement either a complex network algorithm (such as a routing protocol), or a simple modules that requires no communication with other devices (such as duplicate detection). IDRA does not make a distinction between these types of functionality: both complex protocols and simple network functions register to the 'protocol selection' component. By default, the following protocol execution sequence is used (see Figure 2.9).

(i) First, **filtering protocols** are executed to filter away unwanted incoming packets. All filtering protocols are executed sequentially. The following filtering protocols are available.

- 'Packet identification' is responsible for identifying the packet type of incoming packets (see Section 4.5.1 of Chapter 4). Once the packet type is identified, the packet facade can be used to interact with the packet. If a packet is not recognized, it is dropped.
- The 'destination check' module drops all packets that are destined for different nodes.
- 'Duplicate detection' drops duplicate packets.
- Other user-created filtering protocols can be included. For example, 'link quality protocols' can be implemented that only accept packets from nodes that have a good link quality, or 'topology control protocols' that restrict from which neighbors packets can be received.
- Finally, the 'parameter extraction' module is executed. If the packet reaches its final destination, the encapsulated information parameters are extracted and distributed through the 'parameter dispatching' component. Afterwards, the packet is dropped[1].

[1] If the destination of the packet is equal to the BROADCAST or NEIGHBOR address, the parameters are also extracted but the packet is not dropped.

(ii) Next, the protocol selection implementation executes all relevant **monitoring protocols**. Monitoring protocols typically gather network information (for example, to fill in the neighbor table). All monitoring protocols are executed sequentially. When a new packet is created by the 'parameter sending' component, the execution schedule starts from the 'packet creation' component. Thus, newly created packets will not be dropped by any filtering protocols, but will be inspected by all monitoring protocols.

(iii) A **routing protocol** is used to determine the next hop address of the packet. The packet facade is used to update the next hop address or to add a path label to the packet. To set-up a network path, routing protocols typically send out 'route request' information parameters through the 'parameter exchange' component. If multiple routing protocols are available, only one will be selected.

(iv) Similarly, only a single **MAC protocol** is executed. The MAC protocol can update packet attributes, but can not yet order IDRA to sent the packet. Sending the packet will be possible afterwards through the 'Transmission Settings' components, once the packet has been fully processed.

(v) Finally, the **post processing protocols** prepare the packet for sending.

- The 'packet conversion' module can convert the packet to a different packet type. More implementation details are available in Section 4.5.1 of Chapter 4.
- The packet updating module updates the 'sender' packet attribute.
- The aggregation module checks if information parameter are available in the 'parameter sending' component. If possible, the information parameters are aggregated to the packet (cfr. Section 2.3.1).

Applications also register to the 'protocol selection' component. However, since applications do not process any packets, they are not part of the packet processing flow. It is possible for a protocol to register itself multiple times. For example, a MAC protocol can register itself to be executed both as a filtering protocol (to drop unwanted incoming packets) and as a monitoring protocol (to send packet acknowledges). Finally, by reconfiguring the protocol selection component, users can change the behavior of the protocol execution sequence.

2.6 System evaluation

For the performance evaluation of IDRA, the iLab.t wireless sensor test bed [16, 17] was used, which is located in the IBBT - Ghent University office building in Belgium. The iLab.t test bed consists of 200 TmoteSky sensor nodes, spread out over 3 floors. By setting the transmissions power of the sensor nodes to an output

power of -15 dBm, packets require 4 to 5 hops to be transmitted from one side of the building to the opposite side.

The following sections evaluate how each of the techniques described in Sections 2.2 and 2.3 influence the overall system performance of IDRA. For each of the techniques, a thorough qualitative and quantitative analysis in terms of processing time, memory overhead, throughput and behavior is provided.

2.6.1 Evaluation of the aggregation approach

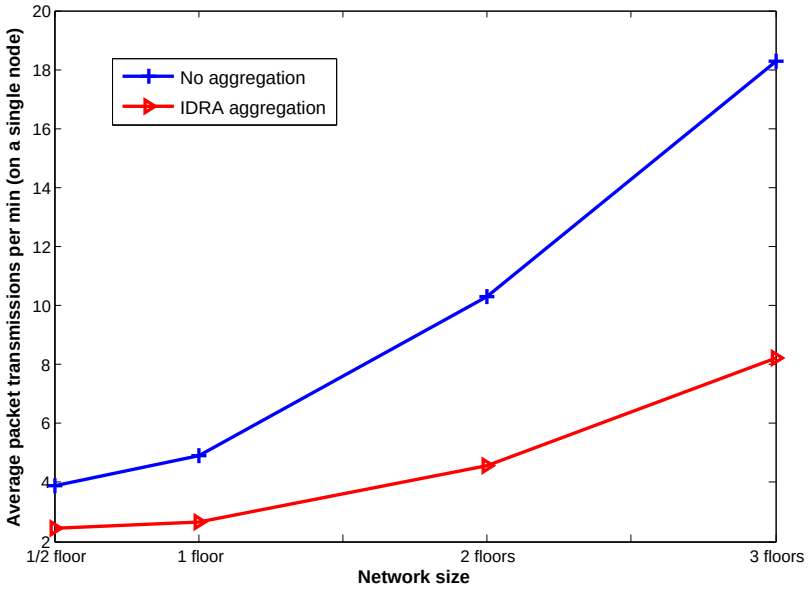

Figure 2.10: Performance of in-built IDRA aggregation in a point-to-point scenario. Each floor consists of $\pm$ 60 TMoteSky sensor nodes.

To evaluate the performance of the aggregation approach, all nodes of the test bed were configured to regularly broadcast status messages to their neighbors. In addition, 20% of the nodes were used as a source for point-to-point traffic: they contacted a random other node to which measured information was sent every 60 seconds. To set-up routes, the AODV [18] protocol was used. To compensate for the unreliability of the paths, the maximum lifetime of a AODV path is set to 10 minutes, after which a new path setup is executed. The maximum parameter delay of the information exchanges was set to 30 seconds.

Figure 2.10 shows that the resulting number of packet transmissions reduces by more than a factor two when architectural aggregation is enabled. The main reason for this significant reduction of packet transmissions is the fact that any

type of information (both application and network information) can be combined with information generated by any other protocol layer.

For a more in-depth comparison of the IDRA aggregation approach with existing techniques, we refer to Chapter 3, where it is demonstrated that our approach can increase the network lifetime by 30-50 percent, depending on the MAC protocol.

2.6.2 System processing overhead

Optimization		Avg execution time per packet	Percentage
Network Protocols	CTP & S-MAC	2.42 msec	37.99 %
Basic IDRA functionality	IDRA System Overhead	1.28 msec	20.09 %
	Information Management (Section 2.2.1)	0.59 msec	9.26 %
	Packet Facade and Packet Identification (Section 2.2.2)	0.97 msec	15.23 %
Advanced IDRA optimizations	Aggregation (Section 2.3.1)	0.58 msec	9.11 %
	Protocol Selection (Section 2.3.2)	0.02 msec	0.31 %
	QoS (Section 2.3.3)	0.51 msec	8.01 %
Total		6.37 msec	100 %

Table 2.1: Processing overhead of the architectural components of the IDRA system.

Table 2.1 shows the contribution of each technique to the average processing delay of a packet. For ease of comparison, the results from this section are given in msec as measured on a 8 MHz TMoteSky node. The estimated number of clock cycles can be derived by multiplying the resulting milliseconds by 8000. To get these results, the average execution time that was spent in each module was measured using the network scenario described in Section 2.6.1. The results in Table 2.1 give a broad indication of the contributions of each technique to the total processing overhead. Of course, these results depend strongly on the network topology and the number of information exchanges.

Based on Table 2.1, it is clear that most processing overhead is caused by the network protocols (almost 40%). The IDRA system overhead (about 20%) is caused by copying packets to the shared queue and for managing internal timers and tasks. The flexibility of the packet facade comes at an overhead of about 15% of the total processing overhead, which is a little less than 1 msec. The overhead of the advanced IDRA optimizations is limited to at most 10% of the total processing time.

Table 2.1 can be used by system developers to decide for each feature whether or not the advantages compensate for the additional processing overhead. However, even when all IDRA features are enabled, the processing delay is about 6-7

milliseconds on a TMoteSky sensor node. As such, the total processing overhead is negligible for most WSNs, since the duty cycle (sleep period) of typical MAC protocols for WSNs is typically 200 msec or higher.

2.6.3 System memory overhead

	Component	ROM	RAM
Radio Controller	CC2420	11196	493
Basic IDRA functionality	System Components	7016	909
	Information exchanges (Section 2.2.1)	1862	353
	Shared queue (Section 2.2.3)	1934	1858
	Packet facade (interpretor) (Section 2.2.2)	806	8
Advanced IDRA optimizations	Aggregation (Section 2.3.1)	1012	58
	Protocol selection (Section 2.3.2)	1564	86
	Quality-Of-Service (Section 2.3.3)	728	4
Total		27118	3769

Table 2.2: Memory footprint (in bytes) of the architectural components of the IDRA system.

The memory footprint of the different architectural components is shown in Table 2.2. The entry 'System Components' includes modules for duplicate detection, print statements and timer management.

The full architecture requires about 27kb ROM and 4 kB RAM memory (11 kB of these ROM requirements are required by the radio controller). As such, the total memory requirements are well under the limits of most sensor nodes. This demonstrates that IDRA can be used in most typical sensor networks. Moreover, Section 2.2 will demonstrate that this larger initial memory cost is compensated by the smaller size of the IDRA network protocols. For nodes with even more constrained memory limits, the advanced IDRA optimizations can be disabled.

2.6.4 Performance of the shared queue

Copying packets from one layer to another causes a significant processing overhead. According to [19], multi-hop throughput in WSNs is limited mainly by the number of times a packet needs to be copied. On the TMoteSky, the processing overhead for copying a single packet of 128 bytes corresponds to $\pm$1530 clock cycles, or 0.19 msec. Using a shared queue, only two copy actions are required: one to copy incoming packets from the radio to the queue and one to copy outgoing packets from the queue to the radio. In contrast, when using a layered 'store-and-process' architecture, packets traverse through each layer twice (once to remove all headers and once to process the packet). As a result, processing overhead increases by twice this amount for *each* protocol layer.

	Total queue entries		Packet drop ratio
Shared queue	1		56.4 %
	2		5.6 %
	3		0.1 %
Layered	Routing	MAC	
	1	1	47.8 %
	2	1	46.6 %
	1	2	4.8 %
	2	2	3.7 %

Table 2.3: Comparing the packet drop ratios (lower is better) for different queue sizes when using a shared queue versus a layered approach.

The optimal size of the shared queue depends on many factors, such as the average processing delay of the network protocols (including route set-up), the number of required control messages, the network characteristics and the application requirements. As an example, Table 2.3 shows the average number of packet drops for different buffer sizes. These results were obtained using the data collection scenario (without aggregation) from Section 2.6.1 on a single floor with 60 sensor nodes. Not only is the processing overhead of a shared queue lower, but the use of a shared queue *(i)* is significantly better in terms of packet drops (for a fixed number of queue entries) and *(ii)* makes it easier to determine the required number of queue entries.

2.6.5 Support for traffic streams of different priorities

This section demonstrates that the available QoS commands are sufficient to transparently add simple QoS features to existing IDRA protocols. To this end, a simple QoS module was developed for IDRA that implements the following two rules: *(i)* when the packet queue is full, the packet with the lowest priority is dropped; *(ii)* the packets with the highest priority are always processed and transmitted first.

To evaluate the effectiveness of the available QoS commands, the QoS module was added to a scenario whereby a collection tree protocol [20] was used to collect data. One node was configured as an always on sink node. The other nodes were generating data packets every 4 seconds but, due to their sleeping scheme, they could only transmit a packet every second. This way, an overloaded network was created where packets had to be dropped. Two types of traffic were generated, each having a different priority level. The high priority traffic flow was generated on only one node, while all the other nodes were generating a low priority traffic flow. Using the provided libraries, the implementation of these QoS policies required only 782 bytes ROM and a processing overhead of 4080 clock cycles (about 0.5

		Low-priority stream	High-priority stream
No QoS	Avg delay	14.69 sec	15.07 sec
	Avg packet reliability	8.90 %	15.67 %
Architectural QoS	Avg delay	17.59 sec	1.58 sec
	Avg packet reliability	3.76 %	98.22 %

Table 2.4: Average delay and the end-to-end reliability (percentage of successfully packet received) of packet streams with different priority level with and without QoS support. The QoS solutions are implemented at the architectural level (i.e.: the network protocols are not QoS-aware).

msec on a 8 MHz processor) per packet.

Table 2.4 shows the end-to-end delay and packet drop results from the high priority traffic flow compared with the average of the low priority traffic flows on the same hop level. The delay and packet loss for the high priority stream is significantly lower, even though the network protocols do not support any QoS at all. Thus, when using IDRA, it is possible to increase the global network performance by combining protocol-independent QoS solutions with any network protocol.

2.6.6 Performance of the packet facade

To decouple the packet structure from the protocol logic, the IDRA architecture relies on the use of a packet facade. This section investigates the cost that is incurred by using a packet facade to update packet attributes.

Table 2.5 shows the memory requirements that are used by different packet part descriptors to describe a packet structure. There is a clear correlation between the complexity of a packet part descriptor and its memory and processing overhead. For example, the 6LoWPAN packet part descriptor requires up to 588 bytes to describe the header structure. In contrast, the most simple packet part descriptor uses a Type-Length-Value (TLV) representation to sequentially store attributes. This TLV packet part descriptor requires only 34 bytes ROM and is used to store packet attributes in the packet payload. This ensures that all packet attributes can be stored, even when no other packet part knows how to process a certain packet attribute.

Table 2.6 compares the processing overhead for creating and manipulating packets with the overhead for traditional methods (e.g: using fixed C structures). Due to the low total processing delay, the overhead is expressed in clock cycles. To measure the packet update overhead, all packet attributes that are recognized by the packet part (for example: the sender, receiver and packet ID for a 802.15.4 packet part) were updated. Again, there is a clear correlation between the complexity of the packet structure and its processing overhead.

In general, *creating* a new packet using the packet facade does not require

Packet part descriptor	ROM footprint	Header size
Sequential storage (TLV)	34 bytes	Variable
IEEE 802.15.4	120 bytes	9 bytes
IPv6	396 bytes	Variable
6LoWPAN (HC4 spec.)	588 bytes	13 bytes

Table 2.5: Memory footprint (in bytes) and header size (in bytes) of multiple packet part descriptors.

Packet part descriptor	**Packet creation overhead**		**Packet update overhead**	
	Traditional approach	Packet part descriptor	Traditional approach	Packet part descriptor
Sequential storage (TLV)	n/a	376 cycles	n/a	$\pm$ 620 cycles per update
IEEE 802.15.4	288 cycles	456 cycles	172 cycles	940 cycles
IPv6	604 cycles	640 cycles	232 cycles	884 cycles
6LoWPAN (HC4 spec.)	1860 cycles	2040 cycles	604 cycles	1276 cycles

Table 2.6: Processing overhead (in clock cycles) of the available packet part descriptors.

significantly more processing than when using more traditional approaches. However, *updating* a header through the packet facade can require up to 5 times more processing cycles than directly assigning values to header fields. For most WSN networks this result is acceptable, since the total overhead of the packet facade is still significantly smaller than the overhead of the network protocols (see Table 2.1). At the cost of this additional processing overhead, the packet facade results in additional flexibility in terms of packet construction.

2.6.7 System throughput

For most WSNs, little emphasis is put on the maximal throughput of the system. Theoretically, an 802.15.4 network has a maximal physical bitrate of 250 kbit/sec. However, MAC protocols for sensor networks make heavy use of duty cycling in the form of sleep schemes. These sleep schemes create artificial bottlenecks in terms of throughput, resulting in a maximal throughput which is typically a factor 10 lower. However, as sensor networks are increasingly used for more demanding applications, such as camera surveillance networks, the importance of the maximum throughput might increase in the future.

Table 2.8 shows the measured single hop IDRA throughput for different packet types and payload sizes. Creating a packet using more advanced packet part descriptors result in smaller packets, since a higher number of packet attributes will have a fixed header location.

Using the notations from Table 2.7, the theoretical single hop throughput is

Variable	Meaning
T_{IDRA}	Per packet processing overhead of the IDRA architecture
T_{radio}	Processing overhead of the radio driver before a packet transmission
$T_{transmission}$	Time to transmit a packet of *PacketSize* kilobits

Table 2.7: List of symbols used in calculation of the theoretical throughput.

shown in Formula 2.1. It can be observed that a higher system processing overhead T_{IDRA} results in a lower effective throughput. In general, creating and processing a packet with a simple packet description requires about 6 msec, which results in a measured throughput of about 167 kbps. The use of a more complex 6LoWPAN packet type decreases the total packet size and increases the processing delay to 7 msec, which corresponds to a measured throughput of about 143 kbps.

$$\begin{aligned} Throughput &= \frac{PacketSize}{T_{IDRA} + T_{radio} + T_{transmission}} \\ &= \frac{PacketSize}{T_{IDRA} + T_{radio} + \frac{PacketSize}{Phys.bitrate}} \end{aligned} \tag{2.1}$$

These measurements correspond to an 'ideal' radio with no radio processing overhead ($T_{radio} = 0$). In practice, the radio controller also requires CPU cycles. When using a blocking radio, no calculations can be executed by the CPU while the radio is transmitting. Since the performance of the default TinyOS radio is very low, IDRA includes an optimized CC2420 radio controller, which blocks the CPU for only 4 to 6 milliseconds per packet transmission ($T_{radio} = 4$-$6\ msec$). Table 2.8 shows the resulting measured throughput. The combined IDRA and radio overhead for transmitting a packet is up to 12 msec. As a result, the resulting maximal throughput is about 95 kbps.

In contrast, the default TinyOS CC2420 radio controller blocks the CPU for up to 20 msec per packet transmission ($T_{radio} = \pm 20\ msec$). When using this default radio, the time required for sending a packet is significantly larger than the packet processing time of IDRA, which results in very low throughputs. The maximum throughput is limited to about 50 kbps.

Using Section 2.6.2, it is possible to *estimate the cost (in terms of throughput) of each proposed technique*. For example, as shown in Table 2.1, disabling QoS and aggregation lowers the processing overhead per packet by 1 msec. A decrease of processing time results in an increase of the throughput. Since the processing time will decrease by 17%, the maximum throughput will increase by a similar percentage from 167 kbps to 197 kbps. Using similar calculations, the throughput

Packet Part Type	Packet Size		Processing	Measured Throughput		
	Payload	Total packet	Overhead (T_{IDRA})	IDRA throughput	With blocking radio	With default TinyOS radio
Sequential storage (TLV)	100 bytes	128 bytes	6.06 msec	167.64 kbps	94.488 kbps	37.592 kbps
IEEE 802.15.4	100 bytes	126 bytes	6.75 msec	150.368 kbps	84.836 kbps	37.592 kbps
802.15.4 + 6LoWPAN (HC4)	100 bytes	122 bytes	6.80 msec	143.472 kbps	77.104 kbps	37.592 kbps
Sequential storage (TLV)	10 bytes	38 bytes	5.86 msec	60.09 kbps	40.128 kbps	18.74 kbps
IEEE 802.15.4	10 bytes	36 bytes	6.49 msec	46.816 kbps	34.504 kbps	12.92 kbps
802.15.4 + 6LoWPAN (HC4)	10 bytes	32 bytes	6.80 msec	37.632 kbps	29.184 kbps	9.504 kbps

Table 2.8: Measured throughput of the IDRA architecture (all features are enabled).

cost for new architectural techniques can be estimated before the actual deployment of the network.

To summarize, these results show that the maximum single-hop throughput is currently mainly limited by the radio controller, rather than the system. The use of a non-blocking radio controller would almost double the throughput. Other techniques to increase the throughput include: the use of a faster radio, increasing the CPU clock frequency, using simpler packet types or disabling architectural features such as QoS or aggregation.

2.6.8 Performance of the network protocols

	Traditional Protocol	ROM	RAM
MAC	TOS2.1 MAC [21]	11528	320
	X-MAC [22]	19854	876
	SCP-MAC [23]	21372	1056
Routing	Lunar [24]	5000	1518
	CTP [20]	7234	1198
	TYMO [25]	11404	482(+60 per route)

Table 2.9: Memory requirements (in bytes) of typical layered WSN network protocols.

	IDRA Protocol	ROM	RAM
MAC	LPL MAC [26]	822	176
	S-MAC [27]	1126	184
	FlexMAC	10210	858
Routing	CTP [20]	712	130
	AODV [18]	1836	158(+7 per route)
	HYDRO [28]	1924	692(+28 per route)
	DYMO [29]	5008	312(+18 per route)

Table 2.10: Memory requirements (in bytes) of different IDRA network protocols.

Section 2.2 demonstrated that it is easier to design network protocols for IDRA since packet creation, packet interaction and buffer provisioning are delegated to the architecture. As a result, network protocols also require significantly less memory. This is shown in Tables 2.9 and 2.10, where the memory requirements of typical layered protocols can be compared to those of different IDRA protocols.

When using a layered architecture, the total memory consumption increases linearly with the number of network protocols. As shown earlier in Table 2.2, using IDRA requires a significant initial memory investment (11.6 kB ROM and 3.1

kB RAM for the basic IDRA functionality). This initial cost is compensated by the lower memory requirements of IDRA protocols. In some cases, the memory footprint of IDRA network protocols is reduced by up to a factor 10. This shows that, using IDRA, it is indeed feasible to combine multiple routing and MAC protocols on a single node.

2.7 Related work

This section gives an overview of related existing WSN architectures.

2.7.1 A Sensor Network Architecture (SNA)

The sensor network architecture (SNA) [30] is based on 'functionality': the authors analyzed thoroughly which 'functions' or 'components' are often executed by protocols. They provided a modular MAC layer (Sensornet Protocol or 'SP') [5] and a modular routing layer (Network Layer Architecture or 'NLA') [3, 31]. A protocol designer can use the available modules to 'build' a custom network protocol. The components are 'glued together' using a cross-layer database that shares information such as a message pool (similar to the 'shared queue'), a link estimation table and an extensible neighbor table. As such, SNA can be regarded a collection of 'puzzle pieces' that can easily be combined to create new network protocols.

The SNA has several similar goals as IDRA, but differs in the following ways. *(i)* Rather than delegating tasks to a central system, their goal is to enable the quick development of protocol layers, using the provided components for each layer. *(ii)* Protocols need to define their own headers and must encapsulate packets from higher layers. *(iii)* Dynamic selection between protocols is not supported, and protocols can not view or reuse each others packet attributes. *(iv)* SNA has only limited support for energy-efficiency. Since their system can not extract meaningful parameters from packets, they combine full packets rather than only the relevant information. Additionally, they can not aggregate information to non-neighboring nodes. *(v)* Provisions for QoS or heterogeneity are not supported.

2.7.2 Mac Layer Architecture

A similar component-based architecture is the 'MAC Layer Architecture' (MLA) [32]. This architecture provides optimized, reusable components that implement common features that are shared by existing MAC protocols. Similar to the SNA architecture, the main focus of MLA is code reusability.

2.7.3 The Chameleon Architecture

The Chameleon architecture [6] is part of the Contiki operating system [33]. Similar to the packet facade presented in this chapter, the chameleon architecture uses packet attributes which are transformed into packets by header transformation modules. The architecture includes example header transformation modules for IEEE 802.15.4, UDP/IP and TCP/IP packets. In addition, an interesting transformation module is included that automatically calculates the optimal packing of attributes into a packet.

The main differences with the IDRA architecture are the following. *(i)* IDRA packet attributes are stored directly at (and read directly from) the correct header offset, which results in minimal processing overhead. In contrast, the Chameleon architecture dismantles every incoming packet and stores each packet attribute at a separate location. As such, the Chameleon approach requires additional buffer spaces and copy operations to process each incoming packet. *(ii)* The Chameleon header transformation modules are implemented in the higher network layers above the MAC protocol. As a result, the MAC header does not profit from the decoupling of protocol logic and packet structure. As discussed in Section 2.4.2, IDRA is able to support multiple packet recognition approaches. *(iii)* The chameleon architecture provides only a single approach to identify incoming packets (a unique 'channel' identifier which is added to each transmitted packet). *(iv)* In the Chameleon architecture, part of the MAC protocol logic needs to be implemented in the Chameleon header transformation module. As a result, the Chameleon packet structure modules are significantly larger than the IDRA packet descriptors. *(v)* No solutions are provided to support energy efficiency, QoS, mobility or heterogeneity at an architectural level.

2.7.4 A declarative sensornet architecture

The declarative sensor network architecture (DSN) [34, 35] aims to facilitate the programming of sensor nodes, using a declarative language (called Snlog). This language provides a high level of abstraction: protocols describe what the code is doing, but not how it is doing it. Algorithms are implemented using predicates, tuples, facts and rules.

The compiler represents all this information as tables. Rules are converted to dataflow plans, using database operations (Join, Select, Aggregate and Project). Execution of the dataflow plans is triggered by the associated predicates. Finally, the intermediary operators are compiled into a nesC program.

DSN is especially suited for recursive protocols, such as tree construction (which requires only 7 lines of code). Additionally, protocol interoperability can be supported using database scheme matching techniques on the packets. However, the architecture currently has several disadvantages: *(i)* complex data struc-

tures are not supported, *(ii)* total memory size increases (up to a factor 3) and *(iii)* no fine grained radio control is supported (which makes the language unsuited for low-level MAC protocols).

2.7.5 Modular architectures

One of the limitations of a layered architecture is that it is difficult to incorporate new cross-layer services, since interfaces are explicitly embedded in each layer. An alternative is to completely discard the layered structure. Instead of using protocol layers, all responsibilities of a protocol layer are divided over separate modules [4] with a well-defined function. For example, a complex MAC layer can be divided into a neighbor management module, a sleep management module, a channel monitoring module and a retransmission module.

The use of a modular architecture has several advantages:

- duplication of functionality is prevented;
- when developing a new network protocol, existing modules can easily be reused;
- cross-layer information can be exchanged, supporting the development of energy-efficient protocols;
- depending on the node capabilities or network conditions, it is easy to add or adapt a single module.

IDRA is designed to support both layered, modular or hybrid approaches. To prevent a large number of dependencies between the different modules, IDRA protocols do not interact with each other directly. Instead, communications between modules go through a cross-layer database repository [36]. To support modular approaches, developers can define new call sequences (Section 2.3.2) that determine the order in which the modules should be executed. To support system-wide (cross-layer) cooperation between protocols a shared neighbor table and information repository is provided. Thus, IDRA is not only specifically designed to be suitable for both layered and layerless approaches, IDRA can also be configured for backwards compatibility with any existing layered or layerless architecture (see Section 2.4).

2.7.6 Performance comparison of IDRA with existing architectures

Giving a performance-based comparison with existing architectures is not an easy task:

- each related work architecture implements different functionalities and focuses its evaluation on different design goals;
- several of the conceptual architectures still lack a practical implementation (see further);
- some performance metrics, such as ease-of-use, are subjective and can not easily be measured;
- finally, the performance metrics used to evaluate the architectures are often strongly different.

Keeping these limitations in mind, this section aims to compare the performance of IDRA with the performance of the architectures discussed in the related work section.

(i) The sensor network architecture (SNA) does not analyze the architectural behavior, but instead evaluates the performance of the implemented network protocols. The memory requirements of the implemented routing protocols varies from 6140 bytes ROM and 1862 bytes RAM for MintRoute to 9060 bytes ROM and 1889 bytes RAM for BVR routing [31]. As shown in Table 2.10, the memory requirements of IDRA routing protocols are significantly smaller. On the other hand, the processing overhead of SNA varies from 0.8 msec (for mintrouting) to 3.9 msec (for BVR routing), which is in the same order of magnitude as the IDRA protocol overhead. No advanced network requirements, such as QoS, are evaluated in the SNA architecture.

(ii) The MAC Layer Architecture (MLA) also evaluates the behavior of a number of implemented MAC protocols. According to [32], the typical overhead of a MAC protocol that uses MLA is about 20kB ROM and 1 kB RAM. Again, as shown in Table 2.10, the memory requirements of IDRA routing protocols are significantly smaller (up to a factor 10). The processing overhead of the MAC protocols is calculated whilst using sleeping schemes. As such, the delay is in the order of several 100 milliseconds. No results are given about the architectural processing overhead.

(iii) The header transformation modules from the Chameleon architecture [33] have a similar function as the IDRA packet facade. However, Chameleon header modules also contain protocol logic. For example, the UDP/IP module also includes the ARP protocol. The memory requirements of the Chameleon header modules vary from 475 bytes for a TLV-representation to 6042 bytes for the TCP/IP representation. Since the IDRA packet part descriptors do not contain any protocol logic, the IDRA header descriptors are smaller by more than a factor 10 [2]. The

[2]Even though the IDRA packet part descriptors do not contain protocol logic, it is possible in IDRA to associate packet-dependent protocol logic to existing packet types. For example, a ARP protocol can register itself to be executed before the transmission of a UDP/IP packet (see Section 2.3.2).

processing overhead of the Chameleon header transformation modules is similar to the packet processing overhead from IDRA, which is shown in Table 2.6.

(iv) Similar to SNA and MLA, the evaluation of the declarative network architecture [34, 35] focuses on protocol behavior, rather than architectural behavior. According to the paper, the high level of abstraction offered comes at a high memory cost. For example, the implemented CTP routing protocol requires 48.8 kB ROM and 3.2 kB RAM, which is more than the available memory on many sensor nodes.

(v) Finally, no performance comparison can be made with the heap based architecture from [4] since this architecture was not implemented in a proof-of-concept.

To conclude, the performance analysis of existing architectures mainly focuses on demonstrating that the architecture can be used to implement new network protocols. It is unfortunate that these papers do not evaluate the influence of their design choices: this information would have been very valuable for designers of future protocol architectures. However, when comparing the performance of the implemented network protocols, the IDRA architecture typically outperforms existing architectures, even those that offer less functionalities.

2.8 Conclusions

In this chapter, the IDRA architecture was presented and evaluated. IDRA is designed for next-generation sensor applications, such as e-health, wireless building automation, asset tracking or emergency rescue operations. These next-generation applications impose very challenging network requirements which are difficult to support on resource constrained nodes. Since existing traditional layered architectures are not designed with these constraints in mind, new and innovative system architectures are required.

To solve this problem, the IDRA architecture was created. This architecture has three main goals: (i) the simplification of network protocols and (ii) support for advanced requirements such as energy efficiency, flexibility, QoS and mobility at an architectural level and finally (iii) support for an easy transition towards fully heterogeneous network environments.

The first goal, the simplification of network protocols, is realized by delegating common operations to the system. More specifically, the system is responsible for queue provisioning, packet generation and all packet interactions. By providing a separate packet facade for packet interactions, protocols are not tied to a specific packet structure. Thus, IDRA network protocols only worry about exchanging information. The system is responsible for managing all information exchanges.

By intelligently manipulating the exchanged information, the system can fulfill the second goal: supporting advanced network requirements for next-generation sensor applications. (i) *Energy efficiency* is supported by intelligently combin-

ing multiple information exchanges in a single packet. (ii) *Flexibility* is improved since multiple network protocols can be designed that are optimized for a specific function. The system will dynamically select the optimal network protocol on a per-packet base. (iii) System wide *quality-of-service (QoS)* is enforced by intelligently managing the shared queue. Moreover, the developed QoS solutions can transparently be combined with any IDRA network protocol. And finally, (iv) to support *mobility*, a shared neighbor table notifies all network protocols of arriving or leaving nodes.

With regard to the third goal: it was argued that the IDRA architecture can be used to facilitate a future transition towards strongly heterogeneous networks. To cope with different capabilities, network protocols can be added to a node according to its resources. The system supports transparent communication with co-located networks that use different communication technologies by intelligently managing multiple radio interfaces and automatically converting outgoing packets to the correct packet type.

Finally, the overhead of each of these techniques was thoroughly evaluated. As part of the evaluation, it was shown that the built-in aggregation mechanism can more than double the network lifetime, that the single hop throughput of our system is up to 4 times higher than the throughput of the default TinyOS radio implementation and that IDRA network protocols require significantly less memory, at the cost of a larger initial architecture memory cost. Moreover, it was demonstrated that the processing and memory overhead of each architectural technique is small enough to be implemented on resource constrained embedded devices. In addition, to help users decide which techniques are suitable for their applications needs, the chapter includes detailed overviews that give a quantitative analysis of the overhead of each technique.

To conclude this chapter, we are convinced that future applications for WSNs will be very demanding for the network in terms of flexibility, reliability and adaptivity. We claim that support for these network requirements should be part of the architectural design, rather than being added as an afterthought. As such, innovative architectural techniques that support these requirements will be indispensable for the successful development of next-generation network architectures.

References

[1] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. *Exploiting heterogeneity in sensor networks*. In Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Infocom 2005), March 2005.

[2] Evy Troubleyn, Eli De Poorter, Ingrid Moerman, and Piet Demeester. *AMoQoSA: Adaptive Modular QoS Architecture for Wireless Sensor Networks*. SENSORCOMM 2008, Cap Esterel, France, August 25-31, 2008.

[3] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, , and I. Stoica. *A unifying link abstraction for wireless sensor networks.* SenSys '05, San Diego, CA, USA,, pages pp. 76–89, Nov. 2005.

[4] Robert Braden, Ted Faber, and Mark Handley. *From protocol stack to protocol heap: role-based architecture*. SIGCOMM Comput. Commun. Rev., 33(1):17–22, 2003.

[5] D. Culler, P. Dutta, C.T. Eee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. *Towards a Sensor Network Architecture: Lowering the Waistline*. In In Proceedings of the Tenth Workshop on Hot Topics in Operating Systems (HotOS X), 2005.

[6] Adam Dunkels, Fredrik Österlind, and Zhitao He. *An adaptive communication architecture for wireless sensor networks*. In SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems, pages 335–349, New York, NY, USA, 2007. ACM.

[7] I. Demirkol, C. Ersoy, and F. Alagoz. *MAC Protocols for Wireless Sensor Networks: a Survey*. IEEE Communications Magazine, 2005.

[8] R. Rajagopalan and P.K. Varshney. *Data-aggregation techniques in sensor networks: a survey*. Communications Surveys & Tutorials, IEEE, 8(4):48–63, Fourth Quarter 2006.

[9] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. *In-network aggregation techniques for wireless sensor networks: a survey*. Wireless Communications, IEEE, 14(2):70–87, April 2007.

[10] J. Hoebeke, I. Moerman, B. Dhoedt, and P. Demeester. *Towards adaptive ad hoc network routing*. International Journal of Wireless and Mobile Computing, vol. 1:pp. 18, 2005.

[11] Troubleyn Evy, De Poorter Eli, Ruckebusch Peter, Ingrid Moerma, and Demeester Piet. *Supporting Protocol-Independent Adaptive QoS in Wireless Sensor Networks*. In Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on, pages 253 –260, 2010.

[12] Muneeb Ali, Thiemo Voigt, and Zartash Afzal Uzmi. *Mobility management in sensor networks*. Workshop Proceedings of 2nd IEEE DCOSS, 2006.

[13] I. Akyildiz and I. Kasimoglu. *Wireless sensor and actor networks: Research challenges*. Ad Hoc Networks Journal (Elsevier), 2(4):351–367, October 2004.

[14] *The Internet of Things*. ITU Internet Reports, 2005.

[15] Philip Levis, David Gay, Vlado H, Jan hinrich Hauer, Ben Greenstein, Martin Turon, Jonathan Hui, Kevin Klues, Cory Sharp, Robert Szewczyk, Joe Polastre, Philip Buonadonna, Lama Nachman, Gilman Tolle, David Culler, Adam Wolisz, Technische Universitt Berlin, Crossbow Inc, and Arched Rock Corpration. *T2: A second generation OS for embedded sensor networks*. Technical report, Technical Report TKN-05-007, Telecommunication Networks Group, Technische Universitat Berlin, 2005.

[16] *The IBBT W-iLab.t wireless sensor testbed. http://ilabt.ibbt.be/.*

[17] L. Tytgat, B. Jooris, P. De Mil, B. Latré, I. Moerman, and P. Demeester. *Demo abstract: WiLab, a real-life wireless sensor testbed with environment emulation*. published in European conference on Wireless Sensor Networks, EWSN adjunct poster proceedings (EWSN), Cork, Ireland, 11-13 February 2009.

[18] *Ad hoc On-Demand Distance Vector (AODV) Routing. Networking Group Request For Comments (RFC): 3561, http://tools.ietf.org/html/rfc3561*, July 2003.

[19] Fredrik Osterlind and Adam Dunkels. *Approaching the Maximum 802.15.4 Multi-hop Throughput*. Technical report, Swedish Institute of Computer Science, SICS Technical Report T2008:05, ISSN 1100-3154, ISRN:SICS-T2008/05-SE, March 2008.

[20] *Collection Tree Protocol (CTP) for tinyOS 2.x., december 2008. http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html*.

[21] *TinyOS operating system*. http://www.tinyos.net/.

[22] M. Buettner, G. Yee, E. Anderson, and R. Han. *X-MAC: A Short Preamble MAC Protocol For Duty-Cycled Wireless Networks*. In SenSys06, pages 307–320, Boulder, CO, November 2006.

[23] W. Ye, F. Silva, and J. Heidemann. *Ultra-Low Duty Cycle MAC with Scheduled Channel Polling*. In SenSys06, pages 321–334, Boulder, CO, November 2006.

[24] *LUNAR - Lightweight Underlay Network Ad hoc Routing, december 2008. http://cn.cs.unibas.ch/projects/lunar/.*

[25] *Tymo source code repository. Tymo: DYMO implementation for TinyOS, December 2008. http://tymo.sourceforge.net.*

[26] J. Hill and D. Culler. *Mica: a wireless platform for deeply embedded networks*. IEEE Micro, 22(6):12–24, November 2002.

[27] W. Ye, J. Heidemann, and D. Estrin. *An Energy-efficient MAC Protocol for Wireless Sensor Networks*. In 21st Conference of the IEEE Computer and Communications Societies (INFOCOM), volume 3, pages 1567–1576, June 2002.

[28] A. Tavakoli and D. Culler. *HYDRO: A Hybrid Routing Protocol for Lossy and Low Power Networks*. IETF Internet Draft, http://tools.ietf.org/html/draft-tavakoli-hydro-01, September 10, 2009.

[29] Ian D. Chakeres and Charles E. Perkins. *Dynamic MANET On-demand Routing Protocol (DYMO)*. IETF Internet Draft, draft-ietf-manet-dymo-12.txt, http://ianchak.com/dymo/draft-ietf-manet-dymo-12.html, February 2008 (Work in Progress).

[30] Arsalan Tavakoli, Prabal Dutta, Jaein Jeong, Sukun Kim, Jorge Ortiz, David Culler, Phillip Levis, and Scott Shenker. *A modular sensornet architecture: past, present, and future directions*. SIGBED Rev., 4(3):49–54, 2007.

[31] C.T. Ee, R. Fonseca, S. Kim, D. Moon, A. Tavakoli, D. Culler, S. Shenker, and I. Stoica. *A Modular Network Layer for Sensornets*. In the Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006), Seattle, WA, November 2006.

[32] Kevin Klues, Gregory Hackmann, Octav Chipara, and Chenyang Lu. *A component-based architecture for power-efficient media access control in wireless sensor networks*. In SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems, pages 59–72, New York, NY, USA, 2007. ACM.

[33] *Contiki - The Operating System for Embedded Smart Objects - the Internet of Things . http://www.sics.se/contiki/.*

[34] Arsalan Tavakoli, David Chu, Joseph M. Hellerstein, Phillip Levis, and Scott Shenker. *A declarative sensornet architecture*. SIGBED Rev., 4(3):55–60, 2007.

[35] David Chu, Joseph M. Hellerstein, and Tsung te Lai. *Optimizing declarative sensornets*. In SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems, pages 403–404, New York, NY, USA, 2008. ACM.

[36] Tommaso Melodia, Mehmet C. Vuran, and Dario Pompili. *The State of the Art in Cross-layer Design for Wireless Sensor Networks*. Wireless Syst./Network Architect., LNCS 3883, page 7892, 2006.

3

Non-intrusive Aggregation in Wireless Sensor Networks

This chapter focuses on the energy efficiency of the IDRA architecture. Since energy is scarce in sensor nodes, wireless sensor networks try to limit the number of radio transmissions. To achieve this goal, sensor protocols often aggregate measured data from multiple sensor nodes into a single packet. The previous chapter introduced a 'non-intrusive' aggregation approach that is capable of combining information exchanges from all network layers. In this chapter, it is shown both mathematically and experimentally that this architectural aggregation approach outperforms existing non-intrusive aggregation techniques in a wide range of scenarios.

3.1 Introduction

3.1.1 The need for aggregation

Due to the large number of sensor nodes in a single network, developers aim for a functional lifetime of at least several years before battery replacement. As shown in Figure 3.1, the radio is one of the main sources of energy consumption in a wireless sensor node [1]. To save energy, and thus obtain a longer network lifetime, MAC protocols of wireless sensor networks use sleep schemes which alternately turn on and off their radio interface [2].

To ensure that the radio can be switched off regularly, the number of packet

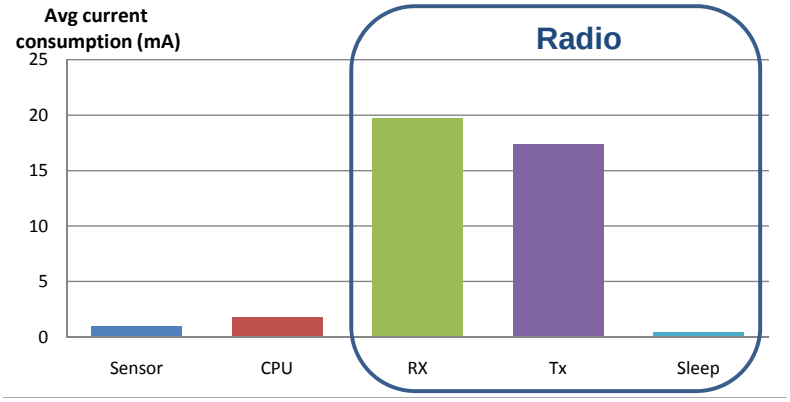

Figure 3.1: Average energy consumption for the TMoteSky sensor node.

transmissions must be low. To this end, data-aggregation protocols have been proposed, which combine measured information from multiple sources in a single packet at intermediate nodes. Since less packets need to be sent, the radio can be put into an energy saving state for a longer time. Moreover, because the number of packets is reduced, the total amount of interference and network contention is decreased.

However, as will be shown in Section 3.2, current data-aggregation approaches often assume that (i) the network does not contain mobile nodes, (ii) the used communication patterns are predictable (all information is sent to a limited number of sink nodes), (iii) information is gathered in predictable intervals, (iv) the overhead of the routing and MAC protocols are negligible compared to the application overhead and (v) only a single application is deployed on the sensor network. In addition, for efficiency reasons, existing data-aggregation approaches are *tightly integrated into a specific routing protocol*. As a result, it is generally not possible to combine the aggregation approach with the routing protocol of your choice.

3.1.2 Beyond the state-of-the-art

In contrast to traditional sensor applications, next-generation sensor network applications are neither predictable nor static [3]. As of recently, WSNs have been used for advanced applications, such as wireless building automation [4], e-health applications [5] or voice over sensor networks [6]. The nodes in these scenarios are typically more heterogeneous: nodes with more capabilities can act as 'actuators' that interact with their environment [7]. Communication patterns are also more complex: information is no longer gathered by a single sink, but can be sent to any other sensor node. Moreover, in contrast with special purpose networks, multiple services are often deployed on a single network [3]. As a result, the above

data-aggregation assumptions are no longer applicable.

Therefore, in this chapter, an alternative 'non-intrusive' aggregation approach is proposed and evaluated. To support next-generation sensor applications, the proposed aggregation scheme overcomes the issues of existing aggregation schemes. More specifically, the proposed approach:

- also copes with monitored events that occur with a non-predictable frequency;
- can be used with complex traffic patterns such as point-to-point communication;
- combines information exchanges from any network layer (rather than only application data);
- aggregates efficiently when multiple applications are deployed on a single WSN;
- is not hindered by networks that contain mobile or failing nodes, which traditionally break fixed aggregation paths; and
- can be used when the network developer lacks time or knowledge to manually fine-tune the aggregation settings.

3.1.3 Remainder of this chapter

In the remainder of this chapter, the performance of a new non-intrusive global aggregation scheme is theoretically and experimentally analyzed. Section 3.1 argued that current data-aggregation approaches are only applicable in static, predictable networks that use point-to-sink communication. Section 3.2 supports this statement by giving a survey of existing aggregation techniques and methods. In order to enable aggregation for a wider range of scenarios, Section 3.3 presents the non-intrusive aggregation architecture that is part of the IDRA architecture but that can also be applied to other architectures for next-generation sensor networks. The proposed aggregation approach is theoretically evaluated in Section 3.4, based on a closed-form ILP formulation. In addition, Section 3.5 experimentally evaluates the proposed aggregation scheme on a large-scale sensor testbed. A wide range of test scenarios is used to validate how our non-intrusive aggregation method performs against existing aggregation methods. Finally, after listing future directions in Section 3.6, the chapter is concluded in Section 3.7.

3.2 Related work

To reduce the number of packets, three main approaches are used.

1 In IEEE 802.11 Wi-Fi networks, the most popular approach is the '*packet combination*' paradigm. Since the number of hops is limited in wireless infrastructure LANs, this approach is designed for small-scale networks with mainly single-hop information exchanges.

2 In contrast, wireless sensor networks most often use '*data-aggregation*' approaches. These approaches assume that packet transmissions are mainly used for the sending of measured information to a far-away sink node. As such, these approaches are often optimized for large-scale point-to-sink networks.

3 Finally, to simultaneously reduce the number of packet transmissions from multiple network layers, wireless networks sometimes resort to the *joint design of multiple network layers*.

This section describes the concepts behind these aggregation paradigms and discusses the main limitations of each approach.

3.2.1 Packet combination in Wi-Fi based LANs

Wi-Fi based LANs are suffering from a large MAC and PHY control overhead. The payload of a IEEE 802.11 Wi-Fi packet is encapsulated in a MAC and PHY header. Regardless of the data rate at which the MAC frame is sent, the corresponding PHY header is transmitted at basic rate only, leading to a sub-optimal channel usage. The RTS/CTS and ACK mechanisms add additional overhead to each packet being sent.

In an attempt to decrease this overhead, several authors have investigated the use of aggregation for wireless LANs. The authors of [8] present an analytical framework for estimating the performance of WLAN aggregation schemes. They list four different types of aggregation techniques. Firstly, there are the IEEE 802.11e [9] and similar block ACK schemes, in which an ACK is only sent after a group of data frames is received correctly instead of on a per-packet basis. Secondly, some techniques expand the block ACK scheme by reducing the IEEE 802.11 short inter-frame spacing (SIFS), allowing frames to be aggregated over less time. A third aggregation technique combines MAC frames by separating the packet payload from its MAC headers, creating a new, larger packet with a single compressed MAC header. This technique is suitable for MAC frames destined to a single receiver. A fourth technique combines IEEE 802.11 frames at the PHY layer while retaining the original MAC headers. While the first two techniques are specifically designed for use with IEEE 802.11 networks, MAC and PHY aggregation can also be used in sensor networks.

The recent IEEE 802.11n standard [10] describes two options to combine frames: Aggregated MAC Service Data Unit and Aggregated MAC Protocol Data

Unit. The Aggregated MAC Service Data Unit (A-MSDU) technique collects ethernet frames transmitted to the same destination and wraps them in a single IEEE 802.11n frame. Since a single aggregated MAC frame header is created, all original MSDUs must have the same source, destination and priority class. The Aggregated MAC Protocol Data Unit (A-MPDU) option collects ethernet frames with the same destination, but wraps each frame in a separate 802.11n frame. As a result, the aggregated frames can have different priority levels and the frames can be acknowledged and retransmitted on an individual base.

Wi-Fi packets are in general considerably larger than packets sent by sensor networks. As a larger packet size increases the chance of transmission errors and collisions, the benefits of Wi-Fi packet combination largely depend on the channel conditions. Authors such as [11] propose the use of adaptive schemes, which optimize the aggregated packet size depending on the channel conditions. However, since the maximum packet size for wireless sensor networks is generally smaller, WSNs offer less possibilities to alter the packet size.

Packet combination approaches in Wi-Fi networks typically have the following limitations: *(i)* packets are only combined, the information they contain is not inspected nor processed; *(ii)* the concept is limited to communications with neighboring nodes; *(iii)* packet combination relies on the MAC/PHY protocol, and as such cannot be combined with different MAC/PHY layers; *(iv)* several techniques assume broadcast packets are overheard by all neighboring nodes, which is not the case in sensor networks using asynchronous sleeping schemes.

3.2.2 Data-aggregation in WSNs

Traditional WSN applications, such as environmental monitoring applications, consist of a network where a large amount of nodes gather information and send this information to one or more sinks. The measurements from different nodes are typically highly correlated, especially when the nodes are densely deployed. Data-aggregation protocols for WSNs strive to exploit this correlation by processing the *measured data* from different sensor nodes locally, before sending the resulting packet to a remote sink node.

The different data-aggregation techniques are often categorized according to their networking approach [12, 13]: how can the data be processed in intermediate nodes.

(i) *Cluster-based data-aggregation* approaches, such as LEACH [14] and COUGAR [15], select cluster head nodes which collect the measured data from surrounding neighbors. The cluster head performs local aggregation and sends the digest to the sink. To prevent the cluster head from running out of battery power, the role of cluster head is rotated regularly. Implementations differ in the way routing is executed (single-hop or multi-hop clusterheads) and in the way cluster

heads are selected and rotated.

(ii) Other aggregation approaches, such as Directed Diffusion [16] and TAG [17], construct *aggregation trees*. For each node, a predetermined path is setup towards the sink. All these paths form a directed tree: the measuring nodes are located at the leaves and the sink is located at the root. Data packets traverse the directed tree towards the root. Aggregation is executed at the location where the different branches merge.

(iii) A third category of aggregation approaches uses *multiple aggregation paths*. Since a tree topology is not robust against node and communication failures, schemes such as Synopsis Diffusion [18] send the aggregated result over multiple paths towards the sink.

(iv) *Hybrid approaches*, such as Tributaries and Deltas [19], combine the advantages of cluster based and tree based data-aggregation. Depending on the network conditions, the aggregation scheme in the different regions of the network is adjusted.

(v) A different approach is taken in PEGASIS [20], which uses *chain based aggregation*. Each data packet is received from and transmitted to the nearest neighbor. By selecting the nearest neighbors, the transmission power (and energy consumption) of these transmissions is very low. Gathered data moves over the 'chain' of nodes: the measured data gets fused at every intermediate node and is eventually transmitted to the sink once the end of the chain is reached.

(vi) *Suppression based aggregation techniques* [21] refrain from sending information if the measured data has not changed from the last measured value, or send less data packets by exploiting the spatial correlation of the measured values.

(vii) And finally, *location-aware approaches*, such as [22], use spatial knowledge to optimize aggregation trees.

Based upon this overview, it can be concluded that current *data-aggregation* approaches in sensor networks have the following concepts in common: *(i)* Aggregation is not implemented as a dedicated aggregation protocol, but is tightly coupled with the routing protocol. As such, the concepts cannot directly be reused with new routing approaches. *(ii)* Data-aggregation is limited to the measured '*data*'. It does not include other exchanged information such as control messages. *(iii)* Data on the same node that originates from different applications is not aggregated in a single packet. *(iv)* Data-aggregation schemes for WSNs are designed for monitoring applications with point-to-sink traffic pattern. As such, they cannot be used for fully interactive WSN applications with point-to-point traffic.

3.2.3 Joint design of several layers

A third paradigm to reduce the number of packet transmissions is the joint design of several network layers. By combining the control overhead of different network

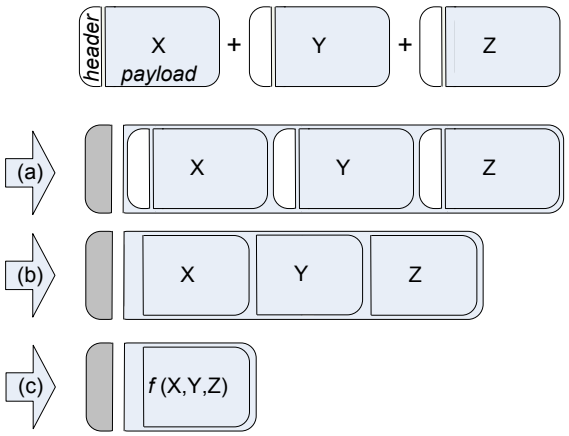

Figure 3.2: Original packet sequence and new packet structure after applying (a) packet combination, (b) packet fusion, or (c) information merging.

layers, the number of control packets can be reduced. For example, in [23], a 'final destination address' field is added to the RTS and CTS frames, thus combining the routing and MAC algorithms. This way, joint design can be used to define common packets that can be used by more than one layer. Many authors agree that sensor networks can profit strongly from cross-layer design [24].

However, joint design also introduces several disadvantages. *(i)* Dependencies between the different layers are introduced, which complicates protocol design. *(ii)* Jointly-optimizing network layers results in large, monolithic code blocks that are difficult to maintain. *(iii)* It is not possible to reuse the developed protocols and aggregation methods in different protocol combinations.

3.2.4 Packet reduction techniques

From the previous sections, it can be concluded that current aggregation paradigms typically use one of the following options to combine packets.

Packet combination: this approach combines entire packets (including their headers) into a new aggregated packet. The aggregated packet (with a new header) is the 'carrier packet' for the inner packets (Figure 3.2a).

Packet fusion: this approach combines only the payload from the packets (without any headers), resulting in smaller packets. A new, common header must be created (Figure 3.2b). To reduce the complexity of this common header, this approach is best suited for networks with multiple traffic flows towards a single destination node.

Information merging: an even higher compression ratio can be obtained by

not combining the payload but only the *information* it contains. This way, similar information coming from different protocols or nodes can be processed and merged together, resulting in a higher compression ratio (Figure 3.2c). However, this approach requires that the aggregation technique knows (i) from which application the information originates, (ii) which function should be used to merge the information, and (iii) that the information is similar enough to be merged. The merging function can be a very simple mathematical function, such as max, min or $average$, or it can be a more complex algorithm which is either *lossy* (not all the original information can be reconstructed) or *lossless* (retaining all original information).

3.3 Non-intrusive aggregation

To overcome these limitations of traditional aggregation approaches, we argue that future aggregation approaches should be non-intrusive. To be considered non-intrusive, an aggregation approach should be (i) independent of the network protocols, (ii) independent of the information source, and (iii) easy-to-use.

1 In our vision, the aggregation protocol should be **protocol-independent**. The aggregation mechanism should not influence the behavior of the network protocols, nor should it make any assumptions about the traffic patterns or the inner workings of the network protocols. This ensures that the aggregation mechanism can be used in combination with any network protocol.

2 In addition, an efficient aggregation scheme should **aggregate information from any source**. This includes control and application information from all layers of the stack and from any node. When all information exchanges are considered, the number of transmissions can be reduced more strongly than when considering aggregation of measured data only.

3 Finally, non-intrusive aggregation approaches should be **easy-to-use**. Application developers should not be forced to fine-tune aggregation specific parameters, such as the optimal aggregation path refresh rate. Optimizing these parameters should be part of the proposed aggregation scheme. Thus, non-intrusive aggregation should work out of the box for any network scenario.

In which situations are non-intrusive aggregation approaches more suitable than traditional aggregation approaches? To answer this question, we envisage two types of future wireless sensor networks: custom made sensor networks and general purpose sensor networks (see Table 3.1).

	Custom network	General purpose network
Traffic pattern	Point-to-sink	Any
Events	Predictable	Unpredictable
# of applications	1	Varying (≥ 1)
Packet overhead	Mainly data	Any
Development cost	High	Low
Suggested aggregation	Traditional	Non-intrusive

Table 3.1: Traditional data-aggregation approaches are better suited for custom high-cost, high-performance point-to-sink sensor networks. Non-intrusive aggregation approaches are best suited for dynamic, adaptive or low-cost sensor networks.

(i) *Custom sensor networks* are specifically designed to support one or more functions. Thus, these networks are predictable and can be highly optimized in terms of energy consumption or desired QoS. A drawback is that, due to the custom design, these networks are expensive to develop and serve only a single purpose. As such, little value is given to compatibility with other protocols. This type of network will be used for applications with stringent requirements. For these networks, traditional aggregation approaches are a justifiable solution.

(ii) *General purpose networks* are more adaptable to changing network conditions and will support a wide range of applications that are unaware of underlying network conditions. These networks will have to support multiple tasks in one network. For applications such as wireless building automation, nodes with new functionalities can be added after deployment. As such, great care should be taken to allow interoperability with existing protocols and to support changing traffic patterns. To promote the use of off-the-shelf WSNs, both the hardware and the deployment cost of these networks should be as low as possible. For these types of networks, non-intrusive aggregation approaches are better suited, being both more flexible and easier to deploy. In the next section, one such non-intrusive architecture, called 'global aggregation' is described.

3.3.1 An architecture for global aggregation

System architectures based on the OSI reference model, as illustrated in Figure 3.3a, are based on the assumption that packets should be sent as fast as possible, as every delay in the send buffer results in an unwanted increase of the end-to-end delay of the packet. Packets which are generated at different protocols are passed down to an output queue and are transmitted as soon as the physical medium is available.

However, in wireless sensor networks, not all packet types need to be forwarded immediately. Control packets generated by protocols (e.g. routing, power management, status information) often have a periodic character. Measurements,

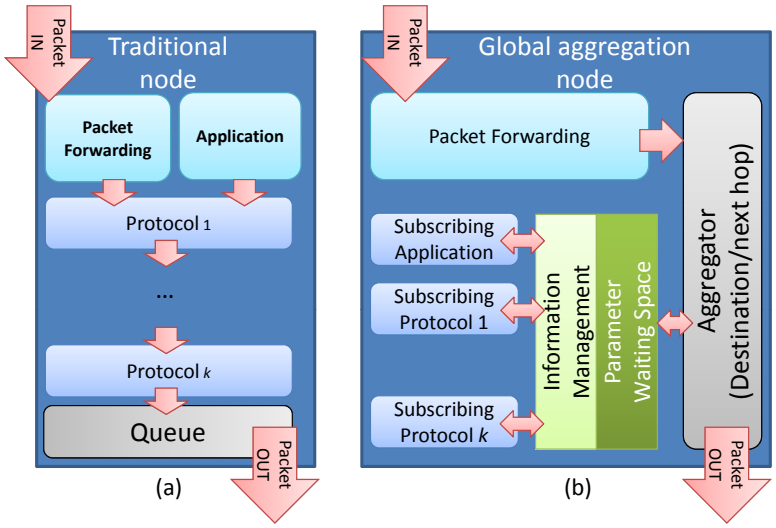

Figure 3.3: (a) Protocol stack based on the OSI reference model. (b) The 'global aggregation' architecture which supports both single-hop and multi-hop non-intrusive aggregation.

such as temperature or remaining battery power, typically do not vary a lot between subsequent status updates. Therefore, it is reasonable to assume that these packets are not very time-sensitive, and can be delayed for a short amount of time before being sent. Similarly, low-priority monitoring packets generated by an application often show a high tolerance for delay.

Using our global aggregation approach, depicted in Figure 3.3b, protocols and applications subscribe to an information management system. To exchange information, the protocols and applications do not generate packets. Instead, the information management offers a uniform API for exchanging information with a remote node. To send information, the protocol or application hands over a parameter to the information management system. The information management system is responsible for the encapsulation of the parameter: the information management system will either (i) add this parameter to a passing packet or (ii) create a new packet that encapsulates the parameter. Typical examples of parameters are:

- measured data values, such as the local temperature;
- status updates, such as the remaining battery capacity;
- or control information such as a route-request or packet acknowledgments.

The example code in Table 3.2 shows how the API can be used to send a sink notification parameter to remote nodes.

```
call InformationExchange.sendParameter (
        SINK_NOTIFICATION,      // Parameter type
        sizeof(parameter),      // Size of the parameter
        &parameter,             // Location of the parameter
        BROADCAST_ADDR,         // Destination of the parameter
        QoS_level,              // Priority of the parameter
        ACCEPTED_DELAY);        // Maximum acceptable delay (msec)
                                // before packet creation
```

Table 3.2: Example code: the provided aggregation API is used by a sink node to send a sink notification parameter to all other nodes.

When using our aggregation architecture, each parameter has a unique parameter type. Uniquely identifying network information has greatly aided the development of uniform network management solutions for IP based networks [25]. Similarly, standardization solutions can be used to better integrate diverging applications from different sensor networks. This uniqueness should be enforced network wide, either (i) by manually assigning each parameter a unique type, (ii) through the use of descriptive ontologies, (iii) by using standardization forums, or (iv) by using hashing methods that generate a unique parameter type based on a description.

When handing over a parameter to the information management system, the parameter priority is given, as well as an indication of the *maximum acceptable delay*. The maximum acceptable delay indicates how much delay a specific parameter can tolerate on the initial node before it should be encapsulated in a packet and sent over the network. All parameters are collected in a central repository, called the *waiting space*. Parameters that do not need to be sent immediately can remain in the waiting space for up to the per-parameter predefined maximum period of time. As the acceptable delay increases, so does the probability that the parameter can be added to a passing packet. To prevent the end-to-end delay from becoming too high, parameters are only delayed in the waiting space of the source node: packets are not further delayed in intermediate nodes.

Algorithm 1 shows the pseudocode for processing incoming packets.[1]

- For each incoming packet, the architecture will first extract all parameters which are destined for this intermediate node. The extracted parameters are distributed through the information management system to all registered protocols and applications.

- If the packet contains no more parameters in the payload, it is dropped by

[1]The implementation evaluated in Section 3.5 also supports broadcast destinations.

```
1:  // Phase 1: check all encapsulated parameters
2:  for all encapsulated parameters do
3:     if parameter destination == local address then
4:        a. Copy the parameter to the parameter space
5:        b. Notify all interested protocols
6:        c. Remove the parameter from the packet
7:     end if
8:  end for
9:
10: // Phase 2: drop empty packets
11: if number of remaining parameters == 0 then
12:    Drop the packet
13: else
14:    Route the packet (determine the next hop address)
15:
16:    // Phase 3: add new parameters to the packet
17:    Sort all parameters in the waiting space according to their deadline, starting
       with the most urgent parameter.
18:    for all the parameters in the sorted list do
19:       if the destination of the parameter equals the next hop address of the
          packet or the destination address of the packet then
20:          if the size of the parameter ≤ unused payload size of the packet then
21:             Add the parameter to the packet payload
22:          end if
23:       end if
24:    end for
25: end if
26: Forward the packet
```

algorithm 1: Pseudocode for processing incoming packets.

the system. Otherwise, the routing protocol processes the packet and sets the next hop address of the packet.

- The system checks if the destination of any of the parameters from the waiting space corresponds to the destination address or the next hop address of the packet. All matching control parameters are added sequentially to the payload of the relayed packet, starting with the parameter with the nearest deadline. This process continues until the maximum packet size is reached.

A similar algorithm is executed when the maximum delay of any of the parameters from the waiting space is reached. First, a new packet is created which encapsulates the corresponding parameter. This packet is then routed to the correct next hop address, after which phase 3 of algorithm 1 is executed. This way, the

system ensures that time-sensitive protocol parameters are delivered timely.

3.3.2 Implementation

For the implementation of these aggregation algorithms, the IDRA framework was used, which was presented in the previous chapter of this dissertation. Thanks to the packet facade from IDRA, any incoming packet can be interpreted, as long as the correct packet descriptor is available [26]. Thus, the packet facade approach ensures that our aggregation architecture can request the next hop and destination address of any packet that passes through the system. As a result, our aggregation approach does not require any specific knowledge about the format of relayed packets (for example: IEEE 802.15.4 (zigbee), 6LoWPAN or propriety packets). This way, our implementation is not only protocol-independent, but also packet-independent. Of course, the proposed algorithms can also be implemented on different frameworks on the condition that the next hop address and destination address of each relayed packet can be located.

3.3.3 Available packet aggregation options

The IDRA implementation of global aggregation has the following options to combine information exchanges into a single packet.

1 By default, all aggregated parameters are stored sequentially in the payload. This is an efficient solution in case no information merging is required or in case multiple protocols and applications exchange different information types.

2 Alternatively, the protocols or applications can indicate to the system that the *parameters in the waiting space* may be overwritten so that they contain only the most recent information. This option is useful when several protocols at different layers of a single node wish to send identical information to another node. For example, multiple network protocols might send a parameter that contains information regarding the remaining node energy. In this case, the outdated energy information will be discarded and only the most recent energy information will be transmitted.

3 Finally, applications can intelligently merge gathered information at intermediate nodes. To realize this, the application can use the packet facade to retrieve, update or remove a specific parameter from the payload of the packet. Using this retrieved information, the application developer can implement application-level aggregation solutions.

Variable	Meaning
T_j	Time message type j is generated for the first time
ΔT_j	Time between two generated messages of type j
AD_j	Maximum *Acceptable Delay* for message of type j

Table 3.3: List of symbols used for the ILP formulation.

Protocol	Message	Frequency	Size
MAC	Synchronization	15 sec	10 bytes
Routing	Route reinforce	3 min	15 bytes
Routing	Location information	30 sec	25 bytes
Clustering	Clusterhead signaling	60 sec	10 bytes
Clustering	Energy update	10 min	4 bytes
Application	Data measurements	Variable	Variable

Table 3.4: Typical messages resulting in periodical exchanges between neighbors

3.4 Mathematical analysis

In this section, a mathematical model is constructed that can be used to quantify the benefits of global aggregation. Section 3.5 compares the mathematical results with the results of a real-life performance study.

The packet savings will first be mathematically evaluated for single-hop aggregation. For simplicity, in this section, only periodic information sent in discrete time intervals is considered (every ΔT seconds).

3.4.1 Definition of variables

Table 3.3 lists the symbols that are used in the following analysis. Assume that the first information exchange is generated at T_1. This information is repeated every ΔT_1 time units. It is assumed that the information must not be sent immediately: it is allowed to remain in the data buffer until its acceptable delay AD_1 is reached. However, once this moment is reached, a new packet must be sent, and all other stored information parameters from other protocols are added and sent together with it. This ensures that the information is combined in an optimal way, so that the least amount of packets needs to be sent.

Assumption:

$$\forall \Delta T_i : AD_i < \Delta T_i$$

Example values of typical periodic message intervals for different protocols are given in Table 3.4.

3.4.2 ILP formulation for multi-protocol optimization

Using these variables, it is possible to derive an ILP formulation of the problem which can be applied to any K number of applications.

Consider:

$$x_i, \qquad i = 0...RP-1 \tag{3.1}$$

$$\Delta T_j, \qquad j = 1...K \tag{3.2}$$

In Equation (3.1), $RP = lcm(\Delta T_1, ..., \Delta T_K)$ is the repetition period for all K protocols. It is defined as the amount of time between two identical sending patterns of the considered protocol messages, that is, the amount of time units before the initial situation reoccurs. This is illustrated in Figure 3.4. The repetition period is 20 time units, which is the lowest common multiple of 4 and 5. After this repetition period, the moment of information generation is indistinguishable from the initial situation.

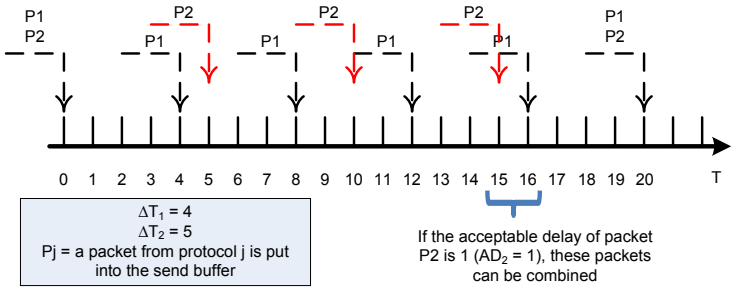

Figure 3.4: Illustration of the repetition period for 2 network protocols.

These protocols are each characterized by an inter packet time $\Delta T_j, j = 1..K$. The x_i variables are binary variables, each representing a timeslot in which a packet can be sent. They are defined as follows:

$$x_i = \begin{cases} 1 & \text{if a packet is sent during timeslot } i; \\ 0 & \text{if no packet is sent during timeslot } i. \end{cases}$$

Then minimize:

$$\sum_{i=0}^{RP-1} x_i \tag{3.3}$$

Satisfying, $\forall j = 1...K, \forall l(j) = 0...\sigma_j$

$$\sum_{i=S_j^l}^{E_j^l} x_i \geq 1 \tag{3.4}$$

Minimize $\sum_{i=0}^{5} x_i$ (Formula (3.3))

With regards to Formula (3.4):

The following equations ensure that the packets from the first protocol are sent in time:

$$\begin{cases} x_0 + x_1 & \geq 1; \\ x_2 + x_3 & \geq 1; \\ x_4 + x_5 & \geq 1; \end{cases}$$

The following equations ensure that the packets from the second protocol are sent in time:

$$\begin{cases} x_0 + x_1 & \geq 1; \\ x_3 + x_4 & \geq 1; \end{cases}$$

Table 3.5: Example ILP formulation for $\Delta T_1 = 2$ and $\Delta T_2 = 3$. $AD_1 = AD_2 = 1$

where

$$\sigma_j = \frac{RP}{\Delta T_j} - 1 \tag{3.5}$$

$$S_j^l = l(j) \cdot \Delta T_j \tag{3.6}$$

$$E_j^l = S_j^l + AD_j \tag{3.7}$$

This formulation can be understood as follows: equation (3.5) specifies the number of information parameters of protocol j that are generated in the RP interval. Equation (3.6) indicates the timeslot when information parameter l from protocol j is available (starting to count from 0). Equation (3.7) indicates the latest timeslot that information should be sent. Equation (3.4) then assures that during this span of timeslots at least one packet is sent, and this for every protocol j and every information parameter to be sent. The ILP formulas are illustrated with an example in Table 3.5.

By minimizing equation (3.3), the amount of packets sent is minimized. The number of *packets per time unit* (PPT) can be obtained by dividing equation (3.3) by the RP. This way, the number of required number of packets per time can be obtained for any number of protocols.

A reduction of the number of transmitted packets occurs when *(i)* the acceptable delay of applications increases, *(ii)* the information exchange interval of different applications are multiples of each other, or *(iii)* both these situations happen at the same time. The average number of packets per time unit can decrease up to a minimum of $\frac{1}{\Delta T_x}$, with ΔT_x the lowest information interval. Or, in other words, the maximum possible transmission savings are limited by the lowest information generation interval. The influence of the acceptable delay is illustrated in Figure 3.5, where the ILP formulation is solved for a variable number of protocols.

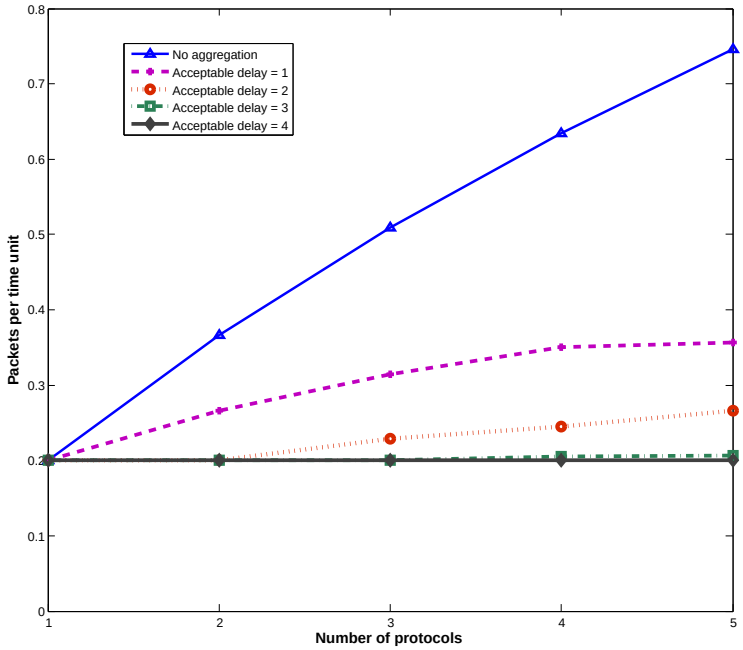

Figure 3.5: Required number of packets per time unit when using our non-intrusive aggregation scheme with multiple protocols ($\Delta T_i = i + 4$).

3.4.3 Advanced ILP formulation

The linear program can be extended to also take into account the maximum packet size. To this end, Table 3.6 introduces several new constants that will be used in the advanced linear program.

Variable	Meaning
$S_{(i,j)}$	Size of the messages generated by application j during timeslot i
S_{max}	Maximum size of a single packet

Table 3.6: List of symbols used for the advanced ILP formulation.

The goal of the linear program does not change. Using the binary variables x_i:

$$x_i = \begin{cases} 1 & \text{if a packet is sent during timeslot } i; \\ 0 & \text{if no packet is sent during timeslot } i. \end{cases}$$

Then minimize:

$$\sum_{i=0}^{RP-1} x_i \tag{3.8}$$

However, to keep track of which application sends a message during which timeslot, additional binary variables are introduced:

$$x_{(i,j)} = \begin{cases} 1 & \text{if application j sends a message in timeslot } i; \\ 0 & \text{if application j does not send a message in timeslot } i. \end{cases}$$

For the advanced ILP formulation, three boundary conditions are required.

(i) First, the following conditions guarantees that the packets from each application are sent in time.

$\forall j = 1...K, \forall l(j) = 0...\sigma_j$

$$\sum_{i=S_j^l}^{E_j^l} x_{(i,j)} \geq 1 \tag{3.9}$$

where

$$\sigma_j = \frac{RP}{\Delta T_j} - 1 \tag{3.10}$$

$$S_j^l = l(j) \cdot \Delta T_j \tag{3.11}$$

$$E_j^l = S_j^l + AD_j \tag{3.12}$$

Equation (3.10) specifies the number of information parameters of protocol j that are generated in the RP interval. Equation (3.11) indicates the timeslot when information parameter l from protocol j is available (starting to count from 0). Equation (3.12) indicates the latest timeslot that information should be sent. Equation (3.9) then assures that during this span of timeslots at least one packet is sent, and this for every protocol j and every information parameter to be sent.

(ii) The following conditions ensures that the maximum packet size is taken into account.

$\forall i = 0...RP - 1$

$$\sum_{j=1}^{K} x_{(i,j)} * S_{(i,j)} \leq S_{max} \tag{3.13}$$

(iii) Finally, the following conditions indicate that a slot x_i is used if at least one protocol sends a message in timeslot i. To express this condition, a disjunctive constraint is used that requires the introduction of new binary variables y_i.

$\forall i = 0...RP - 1$

$$\sum_{j=1}^{K} x_{(i,j)} \leq (1 - y_i) * K \tag{3.14}$$

$$x_i \geq 1 - y_i \tag{3.15}$$

This condition can be understood as follows. Formula (3.14) forces y_i to be equal to zero in timeslots i when at least one application uses this timeslot to send a message. Formula (3.15) then ensures that x_i is equal to one when y_i is zero.

3.4.4 Applying the formulas to more complex scenarios

Up until now, the mathematical analysis assumed that all applications sent information exchanges to a single sink node. However, the situation becomes more complex if applications, such as building automation, send information parameters to different destinations and next hop addresses. If this is the case, the mathematical analysis must be applied once for every set of applications that exchanges information with a specific destination. In addition, the analysis does not account for scenarios where packets must be routed over multiple intermediate hops. In this situation, the total number of transmitted packets needs to be modified, depending on the size of the network and the characteristics of the applied routing protocol. The upper limit of the total number of required packets can be obtained by multiplying the results of the ILP formula (3.3) by the average hop count. The number of packet transmissions will be lower in case the intermediate nodes generate information to the same destination, since these information exchanges can also be added to relayed packets.

In the next section, the performance of our aggregation scheme will be experimentally evaluated in real-life scenarios. In addition, the performance of our aggregation will be compared to the performance of traditional aggregation schemes in varying network conditions.

3.5 Real-life performance evaluation

Network protocols for wireless sensor networks are often evaluated using simulation software (ns2, glomosim, j-sim, matlab, etc.). The use of simulation software has two major benefits: *(i)* experiments are repeatable and *(ii)* large-scale networks are easy to simulate. However, simulation software uses many simplifications that are not found in real-life deployments. As such, simulations often return non-realistic results, which can be very different from the results of real-life deployments. Characteristics like clock drift, failing nodes or fading and reflection of transmissions, are not accounted for. In order to get realistic results, it is necessary to use a real-life testbed [27]. Before giving the results of the performance evaluation, the experimental setup is described.

3.5.1 Experimental setup

Testbed description
For our experiments, the iLab.t wireless sensor testbed [28, 29] was used, which is located in the IBBT - Ghent University office building in Belgium. The iLab.t test bed consists of about 200 TmoteSky sensor nodes, spread out over three floors. Some of the sensor nodes are situated in ventilation shafts between the floors. As such, these ventilation shafts provide a connection corridor through which communication is possible between different floor levels. Figure 3.6 shows the location of the sensor nodes on the three floors. The sensor nodes are marked with a circle, the ventilator shafts are indicated with a rectangle. Large-scale multi-hop experiments are created by setting the transmissions power of the sensor nodes to an output power of -15 dBm. Using these settings, packets require 4-5 hops to be transmitted from one side of the building to the opposite side. Each test case has a duration of 1 h, and is executed five times. The results are averaged to remove outliers.

Information exchanges
To evaluate the effectiveness of our aggregation scheme, traffic must be generated by the sensor nodes. As stated in [2], most traffic in wireless sensor networks consists of two communication types: information exchanges to a remote destination (measured data) and information exchanges with direct neighbors (to exchange local status information).

Depending on the exact application, multiple types of data are often gathered [3, 30], for example temperature, air pressure, humidity or webcam images. For our evaluation, the number of information types on each node were limited to (a) two types of data traffic and (b) two types of control information between direct neighbors. The applications on the nodes are not synchronized: they start generating information at a random start-up time. The following information is exchanged (see also Table 3.7):

- Each node has two separate applications that send measured information to a sink, which is located on floor 3 at the edge of the building (Figure 3.6).

- In addition, whilst the network is operational, networking protocols send notification messages to directly neighboring nodes. These information exchanges contain status information, such as the remaining energy or the signal strength. This information is typically used by neighbor discovery, slot assignments, link quality estimations or synchronization protocols.

In case more types of information exchanges are used, the benefits of our global aggregation approach will increase accordingly.

Figure 3.6: The ilab.t wireless sensor testbed contains about 200 nodes spread over 3 floors. Each floor measures 15 by 90 meter. The sensor nodes are indicated with a dot. Elevator shafts, indicated with a rectangle, provide connectivity between different floors.

Information exchange	destination	# of exchanges	ΔT	Acceptable delay (AD)
Sensor data	unicast to the sink	2	0 to 60 sec.	0 to 50% of ΔT
Control traffic	broadcast to the direct neighbors	2	0 to 60 sec.	0 to 50% of ΔT

Table 3.7: Information exchanges in the monitoring scenario.

Communication patterns
Wireless sensor networks typically use point-to-sink communication for monitoring applications and point-to-point communication for interactive applications such as wireless building automation. When evaluating point-to-sink scenarios (Sections 3.5.3-3.5.5), communication paths are setup in advance by broadcasting a message from the sink to all other nodes. In the case of point-to-point communication (Section 3.5.6), a random destination node is chosen for each traffic flow during the experiments.

3.5.2 Evaluated aggregation paradigms

In the next sections, the performance of global aggregation is compared to the performance of several traditional aggregation approaches. First, a short description is given of all aggregation approaches that have been evaluated using the real-life testbed.

No aggregation: When no aggregation is used, packets are never combined. This situation is used as a reference scenario.

Packet combination: This approach is the most commonly used aggregation solution for Wi-Fi networks (see related work Section 3.2.1). Created packets are delayed for a short time at the MAC or PHY layer before they are sent over the wireless network. If multiple packets to the same next hop address are delayed in this way, they are combined in a single MAC or PHY frame.

Traditional data-aggregation: This approach describes the most common non-intrusive aggregation approach for wireless sensor network (see related work Section 3.2.2). Each application generates information which is encapsulated in a packet. In intermediate hops, the packet is decapsulated and the payload is offered to the application. The application can choose to fuse its own measured data with the received information. Afterwards, the packet is further forwarded to its destination. Data packets coming from different applications are not aggregated.

Joint-application data-aggregation: This approach depicts the approach whereby all applications are jointly designed (see related work Section 3.2.3). The result is similar to traditional data-aggregation, but the resulting application can combine data packets from any application. The evaluated joint-design approach includes only the application levels, thus control packets cannot be combined.

Global aggregation: This is our non-intrusive approach which was described in section 3.3. Applications can be maintained by different developers, but their data is combined as if using joint-application data-aggregation. In addition, control

messages are also aggregated, without the need for introducing any dependencies between the network layers.

In the following sections, the performance of these different aggregation approaches is evaluated under the following conditions: (i) changing network size, (ii) using different time intervals of control traffic versus data traffic, (iii) using different acceptable delay values, and (iv) using point-to-point traffic patterns. In addition, the average processing overhead, queue occupation and energy savings of the evaluated aggregation approaches are studied.

3.5.3 Influence of network size

According to the vision of 'the internet of things', a future office building could exist of several thousands of sensor nodes. As such, it is important to have detailed knowledge about the scalability of different aggregation methods. Therefore, our first analysis investigates which aggregation methods are most scalable for large networks in terms of the number of saved transmissions. For this analysis, the data and control information intervals are fixed, but the network size is varied from half a floor (40 nodes) to three floors (about 200 nodes). The resulting average packet transmissions per minute per node are shown in Figure 3.7.

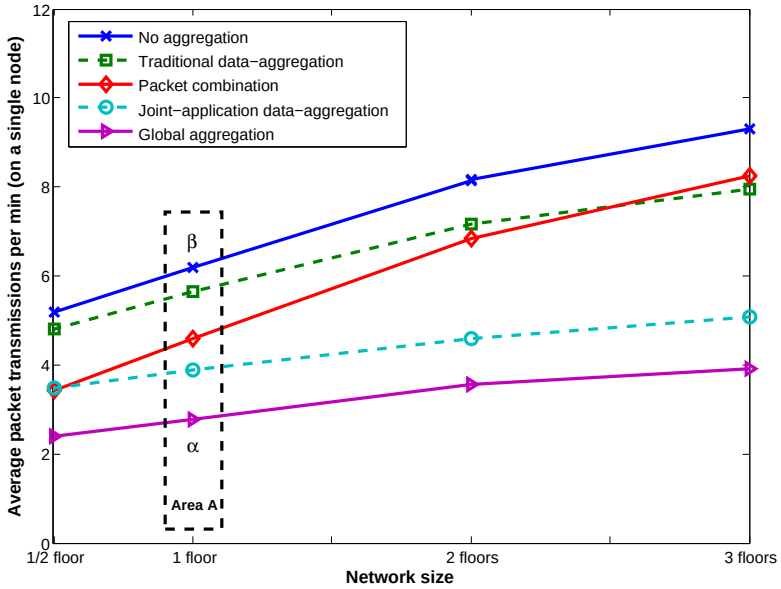

Figure 3.7: Influence of the network size on aggregation method (ΔT_{data} = $\Delta T_{control}$ = 60 sec; AD_{data} = $AD_{control}$ = 30 sec). Each floor consists of $\pm$ 60 TMoteSky sensor nodes.

Packet combination is mainly advantageous in small-scale scenarios, with many single-hop information exchanges. In these networks, many packets are destined to the same next hop address. As the network size increases, packet combination becomes limited to combining control messages to neighboring nodes. As a result, using the number of packet exchanges from Table 3.7, the average netto savings in our scenario converge towards a fixed reduction of 1 packet transmission per minute.

In contrast, *traditional data-aggregation* becomes more useful in large-scale networks. As the network size increases, more intermediate nodes can add their information to passing data packets, each time resulting in a saved packet transmission.

In small, single-hop networks, the use of *joint-application data-aggregation* results in the same number of packet transmissions as the packet combination approach. After all, when using single-hop networks, aggregation based on next hop addresses results in the same packet combinations as aggregation based on destination addresses. However, when the network size increases, the efficiency of joint-application data-aggregation becomes much more noticeable. In multi-hop networks, the reduction of a single packet means a packet transmission less for each intermediate hop towards the sink. This shows that *(i)* aggregating multi-hop packets has a much larger influence than aggregating single-hop packets and *(ii)* packets should be aggregated as soon as possible, preferably on the node where the data originates.

Finally, *global aggregation* combines the advantages of both approaches: both the destination and the next hop field of each packet are checked for combining packets. Thus, the number of packet reductions of this approach are similar to those of joint-application aggregation, together with the (fixed) netto saving of 1 packet/s from the packet combination approach.

3.5.4 Ratio of control traffic versus data traffic

Depending on the application, the time interval (ΔT) between sensor measurements can vary widely, from a few seconds for fire-detection applications to several hours for the monitoring of long-term nature phenomena. Similarly, unreliable networks with mobile nodes or frequently occurring node failures require more frequent status updates than static sensor networks. Since existing aggregation protocols are often optimized for data traffic, the ratio of data traffic versus control traffic has a profound influence on the performance of the aggregation protocol.

In a realistic scenario, the ratio of control traffic versus data traffic depends on many factors, such as:

- protocol design, for example: reactive routing versus proactive routing;

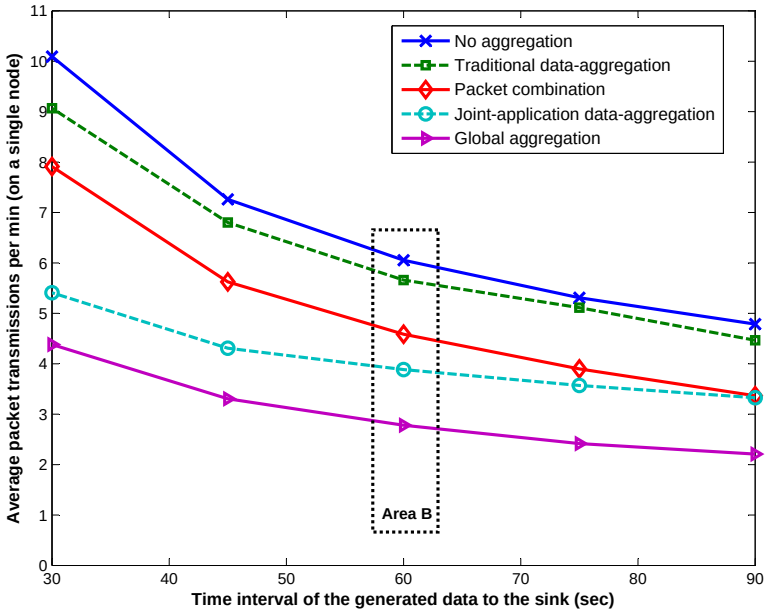

Figure 3.8: Influence of the traffic frequency on aggregation methods (1 floor; ΔT_{data} = variable; $\Delta T_{control}$ = 60 sec; AD_{data} = $AD_{control}$ = 30 sec).

- network stability: a network with many node failures requires regular 'alive' messages between neighbors; and

- network mobility: mobility requires the use of regular discovery and registration messages.

In the previous situation, both control information and measured data were sent every 60 seconds. In the next scenario, the number of control messages is kept constant, but the time interval of the data traffic to the sink is varied. The results are shown in Figure 3.8. In Area B of Figure 3.8, the data interval is 60 s, which corresponds to the results from Area A of the previous figure.

Section 3.5.3 demonstrated that traditional data-aggregation does not perform well in small-size networks. Since this experiment uses only a single floor (about 80 nodes), the performance of traditional data-aggregation is accordingly also very poor. Traditional data-aggregation will only result in profound transmission savings if the number of data exchanges is several times higher than the number of control exchanges.

For a network with an equal amount of data and single-hop control traffic (Area B of Figure 3.8), the use of *joint-application data-aggregation* results in more packet savings than *packet combination*, since each data packet requires additional

packet transmissions in intermediate hops towards the sink. From this, it can be concluded that for networks with data traffic higher or equal to the amount of control overhead, a destination based aggregation approach is the better choice. In contrast, for networks where the major transmission overhead consists of control messages to direct neighbors (the far right side of Figure 3.8), the use of a packet combination becomes a better approach.

Finally, *global aggregation* again has the best performance of the compared aggregation methods. The number of packet transmissions is always reduced by over 50% when compared to the reference scenario.

3.5.5 Influence of acceptable delay

For the previous experiments, information could be delayed for up to 30 s. In this experiment, the influence of the acceptable delay is evaluated. Using the notation from Table 3.8, the probability that information from the waiting space can be combined with another packet is the following:

$$P[pckt\_reduction] = P[info\_available] * P[pckt\_passing]$$

Variable	Meaning
P[pckt_reduction]	Probability that a packet transmission is avoided.
P[info_available]	Probability that information is waiting in the waiting space.
P[pckt_passing]	Probability that a packet passes through the system to which the information can be added, or a local packet is created to encapsulate another information parameter to the same destination.

Table 3.8: Significance of the symbols used to calculate the number of packet reductions

Figure 3.9 shows the resulting average packet transmissions per minute per node in a network of 1 floor when the acceptable delay is varied. When both applications on each node generate information with the same time interval, an acceptable delay of 50% (Figure 3.9, point α) ensures that the information is delayed long enough to guarantee that the information from the first application can be combined with the data from the second application. Similarly, all control packets of the first network protocol can be combined with the control packets of the second network protocol. This situation corresponds with point α of the previous Figure 3.7.

Once the acceptable delay reaches zero, information will no longer be delayed on the first node (point β of figure 3.9). In this situation, our aggregation scheme

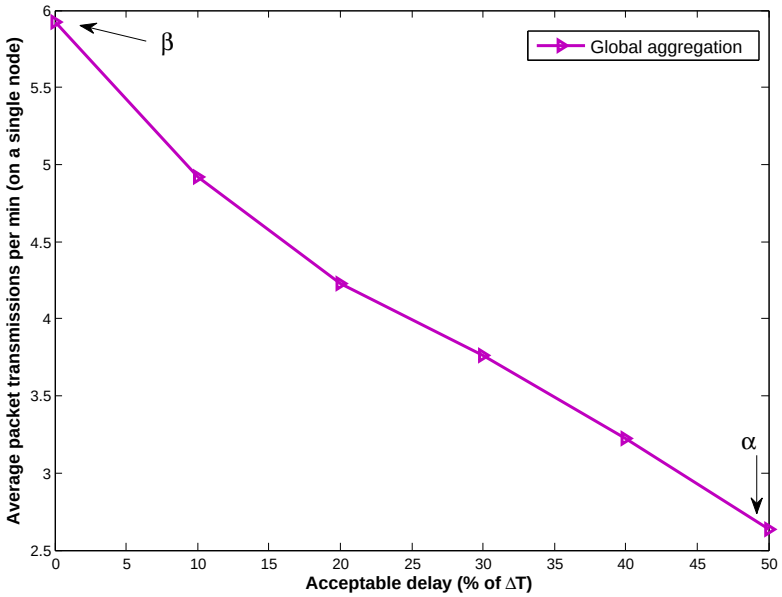

Figure 3.9: Influence of the acceptable delay on aggregation methods (1 floor; $\Delta T_{control}$ = $\Delta T_{control}$ = 60 sec; AD_{data} = $AD_{control}$ = variable).

does not result in transmission savings, since no parameters can be combined. As such, this situation corresponds to the results from the experiments without aggregation (point β of previous Figure 3.7).

For the described network, linearly decreasing the acceptable delay results in a linear increase of the packet transmissions. The graphs describing the influence of acceptable delay for the other aggregation schemes can be deduced similarly based on previous figures, by drawing a line from the required number of packet transmissions of a specific aggregation approach, to the required number of packet transmissions in the same scenario without aggregation.

3.5.6 Point-to-point communication patterns

This section evaluates how efficient different aggregation methods support next-generation sensor applications that use complex traffic patterns. In the following scenario, all nodes exchange regular status messages with their neighbors. However, 20% of the nodes are used as a source for point-to-point traffic: they contact a random other node to which they send their measured data. To setup routes, the AODV [31] protocol is used. The AODV implementation generates a limited amount of control overhead: every 10 min the maximum lifetime of an AODV path is reached and a new path setup is executed. In Figure 3.10, the resulting average

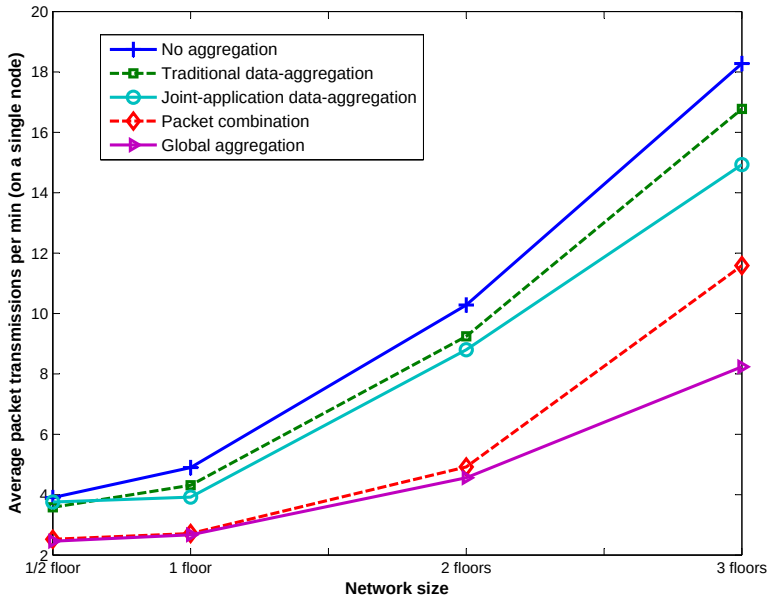

Figure 3.10: Performance of different aggregation methods in a point-to-point scenario (ΔT_{data} = $\Delta T_{control}$ = 60 sec; AD_{data} = $AD_{control}$ = 30 sec). Each floor consists of $\pm$ 60 TMoteSky sensor nodes.

number of packet transmissions is shown for different aggregation methods.

The number of packet transmissions when using *no aggregation* increases more quickly than when using the previous point-to-sink scenario. This is mainly due to the additional broadcast messages that are required for route-setups.

Destination-based aggregation schemes (traditional and single application data-aggregation) have a positive effective on the number of packet transmissions. However, this effect is rather limited, since the probability of having multiple nodes sending information to the same destination is small.

In comparison, *packet combination* approaches perform better: *(i)* control messages to direct neighbors can be piggybacked on the route-requests that are broadcast and *(ii)* due to the random selection of destination nodes, routes are shorter on average when compared to the single sink scenario which favors the packet combination approach.

Global aggregation and packet combination perform similarly for small networks. This demonstrates the fact that in many small networks, the packet reduction from destination based aggregation methods consists mostly of control messages that have both the same next hop and the same destination address. When the number of hops becomes larger, global aggregation scales even better than packet combination in terms of packet transmissions, reducing the total amount of packet

transmissions up to 45%.

In conclusion: destination based approaches do not perform well in point-to-point scenarios, due to *(i)* the increase of broadcast control packets (used for path-setups), *(ii)* the decreased probability of having many nodes with the same destination, and *(iii)* the decrease of the average path length (due to the random selection of a destination node). Packet combination or global aggregation still result in a significant reduction of the number of transmitted packets.

3.5.7 Processing overhead

During the experiments, the same simple information merging function was used for all aggregation approaches: all aggregated parameters are copied sequentially to the payload. Thus, the average processing overhead of the individual aggregation method depends mainly on the number of addresses that should be inspected when relaying a packet, e.g.: checking the next hop address of passing packets, checking the destination address or checking both.

Aggregation type	Aggregated Addresses	Avg processing delay
No Aggregation	None	5.54 msec
Traditional Data-Aggregation	Destination	6.53 msec
Packet Combination	Next Hop	7.15 msec
Joint-Application Data-Aggregation	Destination	6.52 msec
Global Aggregation	Destination & Next Hop	8.21 msec

Table 3.9: Typical processing overhead of the different aggregation techniques.

Table 3.9 shows the delay that is required until a received packet is fully processed (e.g. it is ready for sending) using different aggregation methods.[2] It is clear that the use of more complex aggregation methods results in additional processing delay. However, a large processing delay is generally not problematic for WSNs: a MAC protocol for WNSs typically has a sleeping interval that is larger than the processing delays from Table 3.9. Thus, the delay caused by the MAC protocol will typically become the bottleneck with regard to total throughput. In non-WSN deployments, the delay can be lowered by disabling aggregation on a per-packet basis, depending on the required QoS guarantees of the packet.

3.5.8 Queue occupation

Wireless sensor nodes are usually limited in terms of memory. Lowering the size of the packet queues results in additional memory for other purposes. However,

[2]This result is obtained using a microprocessor with a clock speed of 8 MHz.

using a small packet queue is only possible when the number of packets in the system ('the average queue occupation') is low. The average queue occupation depends on *(i)* the average number of packets that need to be transmitted; *(ii)* the duration that a packet remains in the packet queue (i.e. the processing delay from Section 3.5.7); and *(iii)* the packet size (which is less relevant for WSNs, since most sensor nodes use queues with a fixed packet size).

Aggregation type	Avg queue occupation (%) (1)	Avg queue occupation (%) (2)
No Aggregation	0.317	13.775
Traditional Data-Aggregation	0.374	13.581
Packet Combination	0.301	13.124
Joint-Application Data-Aggregation	0.102	12.900
Global Aggregation	0.080	11.688

Table 3.10: Average queue occupation of the different aggregation techniques when using: (1) CSMA/CA and (2) S-MAC [32] with a sleeping period of 200 msec. (max queue occupation = 10 packets).

In Table 3.10, the average queue occupation is shown for the results from the experiments depicted in Figure 3.8, with a data interval of 90 seconds. The column with the first results shows the average queue delay when using a CSMA/CA MAC protocol. This type of MAC protocol sends packets as soon as the wireless medium is free. Note that the average queue occupation when using 'no aggregation' is lower than the average queue occupation when using 'traditional data-aggregation', even though more packets are transmitted. This can be explained by the higher processing overhead of the latter approach, which is not compensated by a significant decrease of packet transmissions. Thus, when the queue size is limited, it is important to use an aggregation scheme that is efficient in both terms of processing overhead and packet reductions.

The column with the second results from Table 3.10 shows that, when a MAC protocol with sleeping schemes is used [32], the aggregation delay becomes negligible compared to the sleep duration of the radio. Due to the fixed delay of each packet, the queue occupation depends only on the number of packet transmissions and not on the processing complexity of the aggregation protocol.

In summary, for energy-efficient networks that use a MAC protocol with sleeping schemes, the additional processing delay is generally not an issue. The fact that the average queue occupation is lower has two possible positive effects: *(i)* less packets are lost due to packet overflows and *(ii)* memory can be saved by assigning smaller queues for the packets.

3.5.9 Energy savings

By intelligently turning on and off the radio, the amount of time spent in low-power sleep modus is increased. However, the sleep time is limited by the total throughput that the MAC protocol should support. Thus, if less packets need to be sent, the sleep duration can be increased without causing network congestion.

Throughput (bits / sec)	S-MAC	B-MAC
25	4 mW	5 mW
50	7.5 mW	7 mW
75	11.5 mW	9 mW
100	15 mW	11 mW
150	23 mW	13 mW
200	31 mW	16 mW

Table 3.11: Estimated average energy consumption required for a given throughput when using the S-MAC [32] or B-MAC [33] protocols (mW = mJ / second).

As an example, Table 3.11 shows the expected energy consumption (estimated from [33]), associated with different throughputs for the popular S-MAC and B-MAC protocols. In the previous sections it was demonstrated that our aggregation protocol can reduce the number of packet transmissions with more than 50%, which in turn increases the network lifetime by 30-50%, depending on the MAC protocol.

3.5.10 Conclusions from the real-life performance evaluation

The results from these real-life benchmarks show that the choice of an optimal aggregation scheme depends on many external factors. Traditional data-aggregation, while often used in current sensor networks, is mainly suited for large multi-hop networks with a single application that generates point-to-sink traffic. Joint-application aggregation performs better than traditional aggregation, both in small and large-scale networks. However, its main use is still limited to scenarios with point-to-sink traffic. In contrast, packet combination performs well in small-size networks, for scenarios that require point-to-point traffic and in networks where control traffic to direct neighbors is dominating the data traffic.

In addition, it was shown that, in order to obtain the best results, aggregation should be executed as soon as possible, preferably on the node where the data originates. Finally, it was shown that our non-intrusive approach combines the advantages of both Wi-Fi packet combination and joint-application data-aggregation. As a result, our proposed aggregation approach is currently the best non-intrusive solution for reducing the number of packet transmissions in a multitude of scenarios.

3.6 Open research directions

During the mathematical and experimental performance evaluation, it was concluded that the optimal (traditional) aggregation approach depends on the network situation. Using this analysis as a basis, new adaptive or hybrid aggregation protocols can be developed that change their behavior depending on the network conditions.

Current state-of-the-art data-aggregation approaches often ignore Quality-of-Service (QoS) constraints [3]. To cope with different QoS classes in our architecture, global aggregation can be configured to:

- disable aggregation for certain QoS classes; and
- automatically update the QoS field of outgoing packets to the QoS class of the parameter with highest QoS requirements.

As there is as of yet no agreement on the necessary QoS provisions for next-generation applications for WSNs, it is too early to evaluate if this solution suffices for future networks.

Finally, in current aggregation approaches, it is generally assumed that the most energy-efficiency is obtained by aggregating as much information as possible in a single packet. This assumption is plausible, considering the small size of data in sensor networks. However, using big packets may not be most optimal in terms of reliability and delay. As such, the definition of an optimal packet size for WSNs depending on the network conditions can be an interesting research topic, similar to research which has been done for Wi-Fi networks in [11].

3.7 Conclusions

Data-aggregation approaches reduce the number of packet transmissions in data-gathering applications. However, many existing aggregation approaches need to be custom fine-tuned for use in specific scenarios. In addition, they are typically tightly integrated with the routing protocol and work only for predictable sensor deployments (in terms of communication patterns and/or event periodicity).

To remedy this situation, this chapter presented a non-intrusive aggregation architecture in which information exchanges from all layers are aggregated and combined to reduce the number of wireless transmissions. In contrast to traditional data-aggregation protocols, our architecture can be used for networks that (i) generate regular but unpredictable events, or (ii) have unpredictable communication patterns, or (iii) require rapid deployment without prior information about the network characteristics. In addition, the presented approach is protocol-independent: the aggregation approach can be combined with any other network protocol.

Even though the aggregation approach is included in the IDRA architecture from chapter 2, the concepts can also be used in different architectures for WSNs. The aggregation approach uses a very simple queuing system without complex calculations or aggregation functions, making it suitable for both Wi-Fi and sensor networks. Nevertheless, even a simple architecture such as the one described results in a profound reduction of the wireless transmissions. This was demonstrated through a thorough mathematical analysis, which showed that the number of packets per time unit can decrease up to a minimum of $\frac{1}{\Delta T_x}$, with ΔT_x the lowest information interval.

In addition to the mathematical analysis, the performance of different existing aggregation schemes was compared in various network conditions. The main conclusion is that currently no existing single solution is suited for a wide range of applications and network scenarios. However, by combining several existing techniques, our aggregation method proves to be superior in all tested use cases.

Aggregating measured data is currently considered essential for obtaining a long network lifetime for wireless sensor networks. Throughout the chapter it was shown that the same will hold true for the extension of data-aggregation towards aggregation of information in general, both control and data information. As such, we strongly believe that aggregation research should be tackled with as few dependencies as possible with existing network protocols.

References

[1] *CC2420 Radio Datasheet*. http://focus.ti.com/docs/prod/folders/print/-cc2420.html, March 2007.

[2] G. P. Halkes, T. van Dam, and K. G. Langendoen. *Comparing energy-saving MAC protocols for wireless sensor networks*. Mobile Network Applications, Vol. 10(5):pages 783–791, 2005.

[3] Xia Feng. *QoS Challenges and Opportunities in Wireless Sensor/Actuator Networks*. Sensors 2008 (2), pages pp. 1099–1110, 21 February 2008.

[4] P. De Mil, T. Allemeersch, I. Moerman, P. Demeester, and W. De Kimpe. *A Scalable Low-Power WSAN Solution for Large-Scale Building Automation*. In Communications, 2008. ICC '08. IEEE International Conference on, pages 3130–3135, May 2008.

[5] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, R. Stoleru, S. Lin, and J. A. Stankovic. *An Advanced Wireless Sensor Network for Health Monitoring*. Transdisciplinary Conference on Distributed Diagnosis and Home Heathcare (D2H2), Arlington, VA, April 2-4, 2006.

[6] R. Mangharam, A. Rowe, R. Rajkumar, and R. Suzuki. *Voice Over Sensor Networks*. RTSS 2006. Proc. of the 27th IEEE International Real-Time Systems Symposium, Dec. 2006.

[7] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. *Exploiting heterogeneity in sensor networks*. In Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Infocom 2005), March 2005.

[8] R. R. Choudhury, A. chen, and S. Emeott. *An Analytical View of Data Aggregation in IEEE 802.11 LANs*. In Proceedings of Globecom06, San Francisco, USA, November 2006.

[9] *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, 802.11E-2005*.

[10] *IEEE 802.11n-2009 - Amendment 5: Enhancements for Higher Throughput, 29 October 2009, http://standards.ieee.org/findstds/standard/802.11n-2009.html*.

[11] R. Riggio, Francesco De Pellegrini, N. Scalabrino, Pan Li, Yuguang Fang, and I. Chlamtac. *Performance of a Novel Adaptive Traffic Aggregation*

Scheme for Wireless Mesh Networks. In Proceedings of MILCOM07, Orlando, Florida, USA, October 2007.

[12] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. *In-network aggregation techniques for wireless sensor networks: a survey*. Wireless Communications, IEEE, 14(2):70–87, April 2007.

[13] R. Rajagopalan and P.K. Varshney. *Data-aggregation techniques in sensor networks: a survey*. Communications Surveys & Tutorials, IEEE, 8(4):48–63, Fourth Quarter 2006.

[14] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. *An application-specific protocol architecture for wireless microsensor networks*. Wireless Communications, IEEE Transactions on, 1(4):660–670, Oct 2002.

[15] J. Gehrke and S.l Madden. *Query Processing in Sensor Networks*. IEEE Pervasive Computing, 3(1):46–55, 2004.

[16] C. Intanagonwiwat, R. Govindan, and D. Estrin. *Directed diffusion: a scalable and robust communication paradigm for sensor networks*. In Mobile Computing and Networking, pages 56–67, 2000.

[17] S. Madden, M. J. Franklin, J. M. Hellerstein, and Wei Hong. *TAG: a Tiny AGgregation service for ad-hoc sensor networks*. SIGOPS Oper. Syst. Rev., 36(SI):131–146, 2002.

[18] S. Nath, P. B. Gibbons, S. S., and Z. R. Anderson. *Synopsis diffusion for robust aggregation in sensor networks*. In SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, pages 250–262, New York, NY, USA, 2004. ACM.

[19] Amit Manjhi, Suman Nath, and Phillip B. Gibbons. *Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams*. In in ACM SIGMOD, pages 287–298. ACM Press, 2005.

[20] S. Lindsey, C. Raghavendra, and K.M. Sivalingam. *Data gathering algorithms in sensor networks using energy metrics*. Parallel and Distributed Systems, IEEE Transactions on, 13(9):924–935, Sep 2002.

[21] A. Silberstein, R. Braynard, and J. Yang. *Constraint chaining: on energy-efficient continuous monitoring in sensor networks*. In In Proceedings of SIGMOD, page pp. 157168, 2006.

[22] Muhammad Umer, Lars Kulik, and Egemen Tanin. *A Location Based Aggregation Algorithm for Selective Aggregate Queries in Sensor Networks*. 2nd international conference on geosensor networks, october 1-3, 2006.

[23] Changsu Suh, Young-Bae Ko, , and Dong-Min Son. *An Energy Efficient Cross-Layer MAC Protocol for Wireless Sensor Networks*. H.T. Shen et al. (Eds.): APWeb 2006, LNCS 3842, page 410419, 2006.

[24] T. Melodia, M. C. Vuran, and D. Pompili. *The State of the Art in Cross-Layer Design for Wireless Sensor Networks*. In Proceedings of EuroNGI Workshops on Wireless and Mobility. Springer Lecture Notes in Computer Science 3883, Como, Italy, July 2005.

[25] *Distributed Management Task Force, Inc., Common Information Model (CIM) Standards, http://www.dmtf.org/standards/cim/.*

[26] Eli De Poorter, Ingrid Moerman, and Piet Demeester. *An Information Driven Sensornet Architecture*. SENSORCOMM 2009 The Third International Conference on Sensor Technologies and Applications, Athens/Glyfada, Greece, pages 553–561, June 18-23, 2009.

[27] C.J. Sreenan, S. Nawaz, T.D. Le, and S. Jha. *On the Sensitivity of Sensor Network Simulations*. In Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd, volume Volume 3, pages Page(s):1043 – 1047, 7-10 May 2006. Digital Object Identifier 10.1109/VETECS.2006.1682993.

[28] *The IBBT W-iLab.t wireless sensor testbed. http://ilabt.ibbt.be/.*

[29] L. Tytgat, B. Jooris, P. De Mil, B. Latré, I. Moerman, and P. Demeester. *Demo abstract: WiLab, a real-life wireless sensor testbed with environment emulation*. published in European conference on Wireless Sensor Networks, EWSN adjunct poster proceedings (EWSN), Cork, Ireland, 11-13 February 2009.

[30] Ning Xu. *A survey of sensor network applications*. IEEE communications magazine, issue 8, Vol. 40:pp. 102, 2002.

[31] *Ad hoc On-Demand Distance Vector (AODV) Routing. Networking Group Request For Comments (RFC): 3561, http://tools.ietf.org/html/rfc3561*, July 2003.

[32] W. Ye, J. Heidemann, and D. Estrin. *An Energy-efficient MAC Protocol for Wireless Sensor Networks*. In 21st Conference of the IEEE Computer and Communications Societies (INFOCOM), volume 3, pages 1567–1576, June 2002.

[33] Joseph Polastre, Jason Hill, and David Culler. *Versatile low power media access for wireless sensor networks*. In SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, pages 95–107, New York, NY, USA, 2004. ACM.

4

Connecting Heterogeneous Internet of Things Objects through a Network Service Oriented Architecture

In a future internet of things, an increasing number of every-day objects are connected with each other. These objects can be very diverse in terms of the used network protocols and communication technologies, which leads to a wild growth of co-located networking technologies. However, at the moment, communication between these devices is only possible through the use of gateway nodes, resulting in inefficient use of the wireless medium. To remedy this situation, this chapter discusses how the IDRA architecture can be used to facilitate the integration of devices into a single internet of things by supporting efficient direct connectivity between heterogeneous objects.

4.1 Introduction

New communication technologies are introduced and deployed on a regular basis. Even common everyday objects nowadays come equipped with (wireless) communication possibilities. As a result, many sources have described an 'internet of things' view of the future, in which every object is connected with every other object [1] (Figure 4.1). By connecting these different objects, intelligent next-generation applications such as wireless building automation applications [2] or

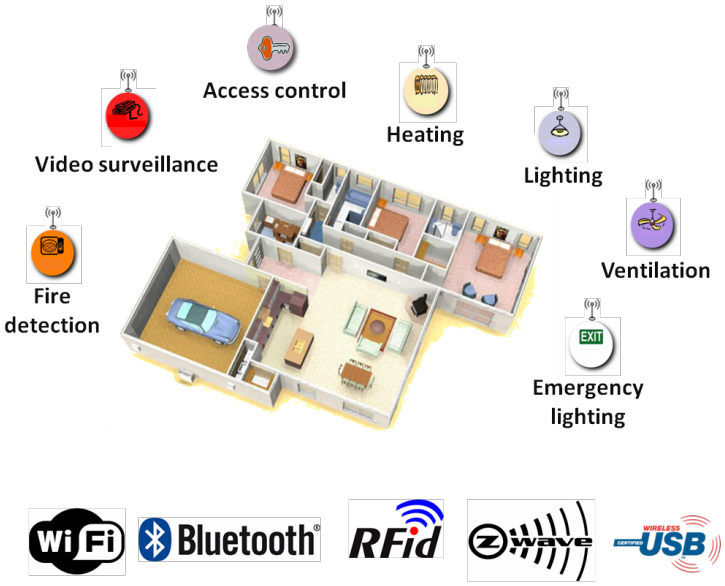

Figure 4.1: In the vision of the internet of things, everyday objects will all become interconnected using a variety of communication technologies. These objects can be used in intelligent applications such as wireless building automation or e-health scenarios.

e-health applications [3, 4] become possible.

However, as the number of communicating objects increases, so does the number of co-located communication technologies. When multiple networks operate in the same geographical environment, co-located networks overhear transmissions from multiple networks. Most often, overhearing these transmissions results in harmful interference and performance degradation, since the overheard transmissions can not be interpreted by devices that are not part of the originating network. This is especially a problem in 'last mile' home and office networks. A typical example is the interference in the free license ISM band, which is used by a variety of communication technologies such as IEEE 802.11 (Wi-Fi), car alarms, baby monitors, IEEE 802.15.1 (bluetooth), cordless DECT phones and IEEE 802.15.4 (zigbee) personal body area networks.

Even when co-located devices use the same radio technology, direct communication between devices is not always supported. For example, existing sensor and actuator networks often use propriety network technologies that are incompatible with technologies from other vendors, even though the devices use the same radio chip. To enable communication between networks from different vendors, or between devices that use different network protocols, each network is connected to a different vendor-specific translation gateway. This translation gateway ter-

minates the connection from one network and sets up a new connection to a second network. However, translation gateways break the end-to-end communication paradigm and are inherently complex to design, manage and deploy [5, 6]. In addition, forcing all communication through the gateway results in additional packet overhead, which in turn leads to increased interference, lower throughput and a lower network lifetime for battery powered devices. To remedy this situation, this chapter describes how the IDRA architecture, which was previously implemented in [7], can be used to enable efficient direct connectivity between heterogeneous wired and wireless devices using different communication technologies.

The remainder of the chapter is structured as follows. Section 4.1 gave an introduction on the vision of the internet of things, and argued that current devices are typically not able to efficiently connect with co-located devices that use different communication technologies. Section 4.2 discusses this topic further by giving a thorough overview of the requirements that should be solved to realize a more efficient internet of things. Related work is given in Section 4.3 where existing architectures are listed and the advantages and disadvantages of each of these approaches are discussed. In Section 4.4, a discussion is given on how the main IDRA concepts fit in the vision of an internet of things. Afterwards, Section 4.5 describes how IDRA can be used to support two typical internet of things use cases: (i) supporting backwards compatibility with legacy networks and (ii) bridging networks using different communication technologies. The economic viability of introducing IDRA in existing networks is discussed in Section 4.6. Finally, Section 4.7 concludes the chapter.

4.2 Requirements of a future internet of things

Several sources already listed a large amount of challenges that must be overcome to support an all encompassing connectivity between objects [8–10]. Amongst the listed internet of things requirements, the following four requirements can typically be found: *providing network connectivity*, *supplying content*, *easily managing the network* and *being extensible*.

4.2.1 Network connectivity

The first and foremost requirement of the internet of things is to provide connectivity between any type of object: from machine to machine, from person to machine or from machine to person. The involved objects differ in terms of both *communication technologies* and *capabilities*.

- Co-located devices that wish to exchange information often use *different communication technologies*. Any architecture suitable for an internet of things must be able to efficiently support communication between devices,

even if they use different protocol stacks, different radio frequencies, different communication technologies and different packet types. In addition, many objects will be equipped with *multiple communication interfaces* such as a IEEE 802.15.1 bluetooth interface and a IEEE 802.11 Wi-Fi interface.

- In addition, the devices will have *different hardware and software capabilities*. Internet of things devices range from high-end PC devices to low-end battery powered embedded devices. Even networks that use only a single communication technology can consist of heterogeneous nodes. For example, a networks used for wireless building automation or industry monitoring used both resource-constrained embedded devices and high-end controller PCs. Using the traditional OSI reference stack, this heterogeneity is difficult to support: each communicating device requires exactly the same protocol stack. However, the communication stack of the powerful devices should not be limited by the capabilities of the most restrictive objects.

4.2.2 Content and context

The internet of things will also become increasingly content oriented [11]. Users expect to be able to retrieve any content, from any device. This includes content that is part of the public domain (dictionaries, transportation information, etc), but also private content such as e-mails, personal media and home information such as the current home temperature. Some of the challenges that need to be overcome are the following.

- Location awareness is increasingly important. This includes awareness of the personal surroundings, the tracking and positioning of objects, as well as support for user and object mobility. For example, in applications that require vehicle-to-vehicle communication all networked objects are mobile.
- The context associated with information is also increasingly important. Future devices require mechanisms to easily associate metadata with content, such as the originating location, information about the producer of the content and the content description. This metadata should also be included at the network level. For example, by associating metadata with packets, the location of the packet destination can be added to packets to facilitate geographic routing.
- Media, security and emergency content often has strict real-time requirements. As such, mechanisms are required that provide quality-of-service solutions that span several networks.
- Finally, mechanisms that control access to information, and that provide privacy, security and anonymity should be supported over several network

layers and physical network boundaries.

4.2.3 Network management

To be commercially viable, a future internet of things should be easy to set up and use even for non-network experts [9].

- Self configuration [12, 13] solutions are required that automatically set up and configure devices. This includes solutions for automatic address allocation and automatic detection of configuration inconsistencies.

- The internet of thing can be fully autonomous: in the absence of human intervention the network should be able to take its own decisions by detecting potential (network) problems and proposing solutions based on artificial intelligence algorithms.

- Finally, to ease network management, underlying network solutions should become more 'invisible'. To be able to reuse network solutions in different contexts, underlying communication interfaces should be presented in an abstract and ubiquitous way. However, this abstraction should not hinder the collection of detailled metadata (such as the radio frequency) that is associated with the used technology.

4.2.4 Network extensibility

Finally, a sustainable internet of things architecture should not only be robust, but needs to cope with continuously changing application requirements and changing hardware capabilities.

- It should be easy to install new software so that new applications can be deployed on previously installed devices.

- Networks should become more 'service-like', where network services can be added and reconfigured according to the applications needs.

- In addition, to support ongoing innovation, it should be possible to change any protocol characteristics such as the addressing schemes (for example from IPv4 to IPv6), the used packet types, the communication technology or the security mechanisms without making any changes to the network protocols themselves. Ideally, these changes should be possible at run-time, without breaking the active communication between devices.

4.3 Related architectures

There is a need for new protocol architectures that inherently support these requirements, such as reconfigurability and support for heterogeneity, over all network layers [14]. This related work section gives a non-exhaustive overview of architectures that are designed to support *direct network connectivity* between *heterogeneous networks*. Two main approaches are discussed: (i) incremental 'evolutionary' architectures and (ii) clean slate 'revolutionary' architectures.

4.3.1 Evolutionary internet of things approaches

Advocates of an evolutionary approach to a future internet of things create new architectures by reusing as many components as possible from existing networking solutions. In their vision, the current internet should 'evolve' into an architecture that is more suited for an internet of things. A first approach is to gradually improve the existing communication stacks, replacing one function at a time, whenever the need arises. A typical example is the introduction of IPv6 addresses to replace current IPv4 addresses. For this approach to be successful, architectures should be easily extensible. Otherwise, this approach results in a difficult adoption of new technologies, as shown by the problematic transition into IPv6 we are witnessing at the moment.

An alternative evolutionary approach is the use of virtualized network components. Network virtualization [15] is used to present underlying network layers in a uniform way towards a high level application. Different devices are connected by forming virtual networks on top of existing networks: logical links are created between distributed systems using native internet routing and standard IP addresses. Well known examples are Virtual Private Networks (VPN) [16] or peer-to-peer applications [17]. The FP7 4WARD project [18] considers virtual networks to be a fundamental part of the design of future internet devices. The project includes virtualized network solutions for in-network management, generic connectivity and content-centric information objects [19]. Similarly, the MAGNET project [20, 21] offers network virtualization at both layer 2 and layer 3, whereas the ITEA2 usenet project [22] focuses on network virtualization for machine-to-machine communication. One well-developed solution is VPAN [23], in which self-organizing and self-maintaining overlay networks are created that provide a shielded and trusted environment for networked applications that share a common context.

In the context of an internet of things, network virtualization can be viewed in two ways [24]. First, these techniques can be used as a tool for evaluating new disruptive architectures on a large scale using existing networks. Secondly, virtualization can be regarded as a fundamental part of next-generation architectures, whereby multiple 'overlay' networks coexist by creating different logical networks for communication purposes [25]. However, for directly connecting heterogeneous

networks (such as described in our vision of the internet of things), the use of virtualization techniques has the following disadvantages. (i) Network virtualization is not yet proven to be highly scalable, since setting up an overlay network is often difficult and time-consuming. (ii) Virtualization techniques are often too complex and inefficient to be implemented on resource-constrained embedded devices. And (iii) virtualization techniques are often too high in the protocol stack to efficiently bridge networks that use different communication technologies.

4.3.2 Revolutionary internet of things approaches

Opponents of the evolutionary approach emphasize the need for a redesigned, clean slate architecture that inherently copes with next-generation network challenges [14, 26], sometimes even abandoning IP based addressing in favor of different addressing schemes. Clean slate initiatives are not always meant to be used directly in new devices, but can be used to sketch a revolutionary new perspective, which can then be brought into existing networks. Several approaches have been proposed.

(i) *Database centric architectures* hide the heterogeneity of underlying networks by only allowing access to network information using database operations. For example, the SENSEI project [27] solves the inaccessibility of low-resource end devices by collecting all data from the end devices and making it available in a centrally accessible database. Unfortunately, this approach often results in significant network overhead.

(ii) *Content centric architectures* focus on describing the information that is exchanged between networks. For example, the SemsorGrid4Env project [28] focuses on the development of a semantic middleware layer. At the network layer, network protocols are implemented semantically using a 'descriptive language' [29] that focuses on functionality rather than implementation. Unfortunately, support for directly connecting different networks at the lower network layers is still lacking.

(iii) *Cloud computing approaches* try to offload resource intensive tasks to more capable nodes. Typically, cloud computing can offer infrastructure, platforms or software as a service to less capable devices [30, 31]. Since cloud computing is regarded as a high layer service, this approach does not solve connectivity challenges.

(iv) *Service oriented architectures* (SOAs) use loosely coupled software entities that implement a single software function. These software services are dynamically combined to form ad hoc applications. In regards to the internet of things, SOAs have two main disadvantages [32]: (i) SOAs focus mainly on higher layers rather than solving network issues and (ii) the technologies used to realize service oriented architectures, such as ML, SOAP, Web Services or BPEL, are often not

suited for use in resource-constrained devices.

(v) *Modular approaches* have also been proposed, whereby the protocol stack is divided into different modules which can be combined to create new network protocols with different functionalities. As such, these approaches can easily integrate new network technologies by updating a single module. Modular frameworks, such as SNA [33], can be used to design new network layers. However, most existing modular frameworks compile these distinct modules into a static network layer. In addition, current modular approaches do not focus on supporting connectivity in heterogeneous environments. Thus, although promising, existing modular approaches offer no additional run-time flexibility when compared to traditional layered approaches. In contrast, the NewArch project [34] discusses how a flexible internet architecture can be created whereby different roles can dynamically be combined at run-time to form 'heaps' [35] which can be adapted to the needs of the network. Unfortunately, the project did not result in a practical proof-of-concept implementation.

4.3.3 The need for new architectures

As shown in the previous sections, several research projects are currently involved with the design of new network architectures. However, an economically viable solution might still be a long way off:

- Though several research projects are currently involved with future internet research, most of these efforts focus on the design of a (high-speed) future internet backbones. These solutions are not suitable for use in resource-constrained environments.
- As cited in [36] 'too many future internet proposals are just extensions of existing protocols or architectures'. As such, these proposals lack the innovation to cope with specific internet of things challenges.
- Finally, too many proposals remain 'paperware': there is a definite lack of implemented prototypes [14, 37].

As such, more practical implementations are needed before a decision can be made regarding the feasibility of an all-encompassing internet of things solution. Especially for resource-constrained devices, there is still room for several improvements. More specifically there exists not yet a simple architecture that

- enables optimized communication **at a network and link level** between co-located heterogeneous networks without the use of complex translation gateways;
- has been implemented and evaluated as a prototype in a large scale experimental setting;

- is compact enough to fit even on low resource embedded devices;
- is fully clean slate, but is also backwards compatible with legacy networks;

In the following section, we will discuss how our IDRA architecture fills this gap.

4.4 IDRA as an enabler of the internet of things

As discussed in Chapter 2, IDRA [7] was originally designed to support next-generation applications on wireless sensor networks. Each year, the WSN research community develops new and optimized hardware devices, communication technologies, network protocols and applications, resulting in a strongly heterogeneous and varying environment. To cope with such a varying environment, IDRA has built-in solutions to support heterogeneous devices (in terms of hardware and communication technologies) and to support evolving services and applications. As such, there are many similarities between the requirements of WSNs and those of a more general internet of things.

This section discusses which IDRA design choices can also be useful in the context of the internet of things.

4.4.1 Network protocols as services

The OSI reference architecture [38] uses a layered protocol stack whereby all network functionality is assigned to a specific protocol layer. For example, the routing layer includes functionality for providing reliability, for duplicate detection, for retransmissions, etc. In contrast, in IDRA it is possible to implement these different network functions (such as addressing, naming, CRC calculation, routing, etc.) each in a simple, standardized, technology-independent component. These components can implement either a full network protocol (such as routing) or a simple function (such as duplicate detection). Each component implements the same interface and functions independent, without direct interaction with other components. To indicate that these components are part of the packet processing flow and influence the network behavior, they are called ‘network services’ in the remainder of this dissertation.

New network services can be added whenever there is a need for them. For example, a localization service can be added when an application is run that requires location information. This way, more advanced network services can be composed by combining elementary network services according to the needs of the network (Figure 4.2c). A default ‘call sequence’ is included that indicates the order in which the network services should be executed. By adding new network services or by changing the call sequence, the network behavior can be changed. Some examples of network services are:

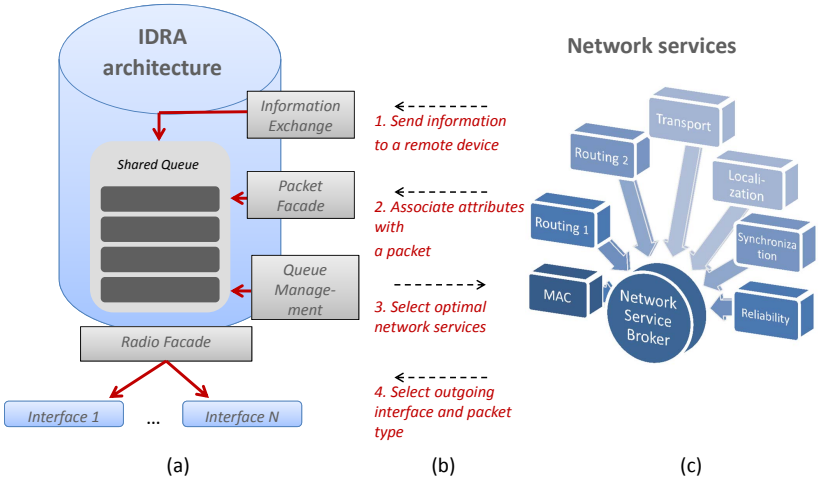

Figure 4.2: Conceptual presentation of the IDRA architecture. (a) The IDRA system is responsible for packet creation, packet storing and packet interactions. (b) Interactions between the network services and IDRA are mainly descriptive in nature. (c) Network services can be added dynamically according to the needs of the device.

- a localization service inspects the RSSI of received packets so that it can provide the network with accurate location information;
- a MAC service is responsible for controlling the timing and sending of packets;
- a topology service decides from which neighboring devices packets can be received;
- a synchronization service delivers a network wide reference clock;
- a reliability service is responsible for the retransmission of packets;
- a management service collects and makes available network statistics such as the background noise level and the number of failed transmissions.

Initially, such a network service oriented approach is compatible with a layered approach: fully implemented network layers can register themselves as network services. A transition towards a network-service oriented architecture can occur gradually by standardizing a further decomposition of the protocol layer into multiple well-defined network services. In the context of the internet of things, a network-service oriented network architecture has the following advantages.

Pervasiveness Separating network functionality into different services results in a lower memory-footprint, since (i) network services are only added on a per-need base and (ii) functionality is not duplicated in several protocol layers. As such, this approach is well suited for networks that contain resource-constrained devices.

Extensibility and maintenance A second advantage of a service oriented network architecture is the ease with which the network copes with future developments. New applications are supported at a network level by plugging in the appropriate network services (for example: an advanced encryption service can be added to process all packets from an online banking application).

Transparency Rather than directly interacting with a multitude of different communication technologies, such as IEEE 802.15.4 (Zigbee), IEEE 802.15.1 (Bluetooth), IEEE 802.11 (Wi-Fi), LAN or UMTS, the 'radio facade' from Figure 4.2a translates the communication capabilities of the underlying communication technology in terms of the services they can provide, such as average reliability, energy cost, etc.

4.4.2 Information driven network services

To limit the dependence of network services on specific technologies, network services in IDRA are made technology independent. Rather than creating technology-dependent packets to exchange information, network services hand over to the IDRA system any information they wish to send (Figure 4.2b, interaction 1). Example information exchanges are a route request, a web page request or a packet acknowledgment. IDRA creates the actual packet, encapsulates the information in the payload and stores the resulting packet in a system wide queue. IDRA can be configured to create or interpret any packet type (see Section 4.4.3). Thus, instead of packet-based sending, IDRA offers *information based communication*.

Similarly, all interactions with the communication interfaces are descriptive. For example, a MAC protocol can describe when and how each packet is allowed to be transmitted (e.g: the maximum tolerated background noise, the scheduled sending time, the radio frequency, etc) and how the communication interfaces should be configured (listening frequency, power state, etc.). A conflict resolution scheme is used to detect conflicting settings and inform the MAC service of undesired behavior. As a result, multiple MAC services can reside on the same node, each with one or more associated communication interfaces.

Delegating all technology related functions, such as packet creation, packet manipulation, packet sending and buffer provisioning, to the IDRA architecture has the following advantages.

Hardware heterogeneity Tasks which are typically duplicated in several network layers (ie: packet creation, packet manipulation, packet sending and buffer provisioning) are delegated to the system, thus avoiding code redundancy. As a result, the overall code size is reduced, making it possible to support a large number of services even on resource constrained devices.

Ease of use Since network services need to consider only 'information exchanges', the development of network services is simplified.

Transparency The same information exchange mechanism is used for all network services, independent of which packet type will be created and which communication interface will be used. The complexity of low-level operations (buffer management, packet construction, etc) are thus hidden from the network services. This way, network services are not technology dependent, which promotes reuse of network services in different contexts.

4.4.3 Decoupling of protocol logic and packet representation

Since the network services do not create the packets, they have no knowledge about the header structure of received packets. To retrieve information about created or received packets, network services interact with packets through a 'packet facade' (Figure 4.2b, interaction 2). Through this packet facade, standardized *packet attributes* (metadata) can be added, updated or requested. This metadata can represent header fields such as 'destination', 'quality-of-service' or 'time-to-live', but can also be used to describe extra context information such as the packet origin or destination location, the packet owner or additional descriptive information about the packet.

To interpret the structure of created or received packets, the packet facade uses one or more 'packet descriptors'. These packet descriptors describe at which header offset each packet attributes should be stored (Figure 4.3). This way, any packet type can be generated or interpreted, as long as the correct packet descriptor is available. Packet attributes that do not have a fixed location in the packet header are stored sequentially in the payload in the form of type-length-value (TLV) elements. Since there is no direct interaction between network services and packet descriptors, the packet type can be changed without any changes to the used network services. Decoupling the protocol logic from the packet structure has several advantages.

Packet heterogeneity Since any packet type can be interpreted as long as the correct packet descriptor is available, it is possible for multiple packet types to reside on a single node, transparent for the network services.

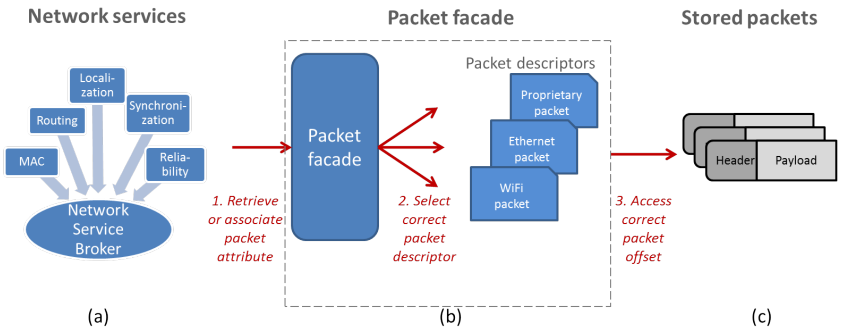

Figure 4.3: Network services can transparently interact with any packet type. (a) Network services can associate metadata with, or retrieve metadata from, stored packets using the packet facade. (b) Only the packet facade requires knowledge about the packet format. As long as the correct packet descriptor is available, the packet facade knows how and where metadata is stored. (c) Finally, the packet facade accesses the correct header offset or the packet payload.

Legacy support By providing the packet descriptors of legacy devices, IDRA services can interpret packets from legacy networks and interact with legacy packet types.

Context awareness Any type of context information can be associated with a packet in the form of a packet attribute. This promotes the development of new network services which rely on advanced packet information such as ownership or visibility rights. Metadata can also be used to facilitate mobility solutions [39].

Hardware heterogeneity Using a packet facade, network services do not need to strip the protocol headers from received packets to interact with packets. Thus, when non-essential network services are omitted from devices with low resources, the remaining network services can still interpret the received packets. In addition, packet attributes remain associated with a packet even if network protocols are omitted at intermediate nodes. Thus, when light-weight nodes are provided with simpler versions of the network services, these simpler services can inspect the packet attributes that were added by more advanced network services.

Future proof Network services can be implemented independently from the representation of the packets. As a result, reuse of network services is promoted. To change the packet structure or support new packet structures, only the packet descriptor needs to be updated. All other network services remain unchanged.

4.4.4 Queue management

Depending on the required network performance, different queuing systems can be used in IDRA. To reduce the memory footprint of the queues, the current implementation of IDRA uses a single, system-wide queue for storing packets. Arriving packets are stored once in the shared queue and remain there until processing is finished. This approach limits the number of copy actions of the packets. Network protocols can interact with any of the packets from the shared queue using the packet facade. Since network services are not responsible for queue management, the complexity and memory footprint of the network services is reduced. The advantages of using a single, system wide queue are the following.

Simplicity In layered architectures, each network layer requires overprovisioning of its provided buffers to ensure that all packets can be stored. Using a shared queue approach, this overprovisioning is required only once. As a result, network services are simpler and have a small memory footprint.

Network management Using a single queue ensures that the system can monitor all available packets. This eases the gathering of real-time network statistics.

QoS management The shared queue has an associated QoS module. This QoS module has a global view on the number of packets, their current processing state and their expected delay. The QoS module can drop packets and selects which packets are processed or transmitted first. Since the QoS module only interacts with the queue, it can provide basic, protocol independent QoS which can transparently be combined with any IDRA network service.

4.4.5 Network service broker

Network services can be dynamically added, removed or updated according to the needs of the network. Rather than having a strict execution order (such as in layered networks), network services are activated only when they are needed. Currently, IDRA implements a simple service broker. Each network service registers itself using 'filters' which describe the function of the network service (e.g: routing, localization, etc). In addition, the filters specify for which types of packets the network services can be used (Figure 4.2b, interaction 3). For example, a QoS aware routing service can register itself for routing high priority packets ('QoS attribute higher than 5'), a georouting service can be used for routing when location information is available ('location attribute is available') and a multicast routing service registers itself for routing packets to multicast addresses.

In high end devices, intelligent service discovery mechanisms can be used to detect the capabilities of the network services, and to automatically select and configure the appropriate network services depending on the application requirements. For example, to support a fire alarm or emergency reporting service, the

network broker can disable energy-efficient routing in favor of an optimized low-delay routing service. Or a key-distribution service can be activated before a device is allowed to join an existing network. As such, a dynamic network service broker promotes a more flexible internet of things.

Self configuration Devices can change their own behavior by plugging in new network services when required. Intelligent self configuring networks can use the service broker for self-adaptation and for automatic network upgrades.

Hardware heterogeneity Devices with less capabilities can be configured to use simplified execution sequences which contain less network services. Alternatively, they can negotiate with neighboring nodes to use simplified versions of the required network services.

Legacy support When interacting with legacy neighboring devices, the service broker can be configured to execute legacy network services in a typical layered order (e.g: MAC, routing, transport, application). This way, network service oriented devices can coexist with traditional layered devices.

4.4.6 System wide aggregation

Many information exchanges (such as status updates, low-priority monitoring information or delay tolerant measurements) between different devices do not need to be transmitted immediately. In the IDRA system, network services can include information about the maximum tolerated delay when handing over information exchanges to the IDRA system. Time sensitive parameters are immediately encapsulated in a packet, but all other parameters are temporarily stored in a central repository. Whenever a packet is relayed through the node, all information parameters with the same 'next hop' or 'destination' attribute are added to the packet. Delay-tolerant parameters can remain in the waiting space for up to a per-parameter predefined period of time. If no data has been relayed within the allowed waiting time, the system generates a new packet which combines all parameters that are have the same destination. As a result, information exchanges from all network services using IDRA can be combined. To avoid high end-to-end delays, the current implementation only delays the parameters in the waiting space of the initial node: packets are not further delayed in intermediate nodes. Since the aggregation is part of the IDRA architecture, the aggregation approach can be changed or optimized depending on the network requirements, without any changes to the network services. The advantages of aggregation at an architectural level are the following.

Reduced interference By limiting the number of transmissions, the overall wireless interference decreases, resulting in more optimal use of wireless bandwidth.

Increased throughput It has been shown that the use of small packets has a negative influence on the maximum throughput of transmission systems, in particular for wireless networks. By combining multiple information exchanges in a larger packet when possible, the overall throughput increases.

Increased energy efficiency Finally, for devices powered by small batteries or by energy scavenging, the use of a radio for transmitting packets results in a serious decrease in network lifetime. By limiting the number of transmissions, the time before battery replacement increases.

4.5 Advanced IDRA use cases

The previous section gave a high level overview of different IDRA concepts and discussed their relevance in the context of the internet of things. This next section describes in more detail on how IDRA can support two important internet of things use cases: (i) supporting backwards compatibility with existing networks and (ii) transparently bridging a diverse number of (co-located) wired and wireless communication technologies.

Backwards compatibility Before an all encompassing internet of things exists, there will be a need for a transitional period, whereby internet of things devices transparently coexist with existing (legacy) networks. As an example, consider a scenario in which an existing corporate Wi-Fi mesh network is extended with new internet of things Wi-Fi devices that use a next-generation protocol stack. Most clean slate solutions solve this by setting up a new network that is fully separated from the existing legacy network (Figure 4.4). Communication between the legacy nodes and the state-of-the-art devices typically requires the development of a complex gateway device, in which all network protocols are translated. In contrast, when IDRA is used on the new devices, nodes can communicate directly with any existing Wi-Fi node resulting in an optimized network performance.

Heterogeneous networks In the future, the internet of things will be accessible using a large number of communication technologies. As an example, consider an industry building where the following communication technologies are used: a wireless entrance and security system is installed, wireless internet access is provided through Wi-Fi access points, the wireless company phone network uses DECT, a company LAN network is used for high

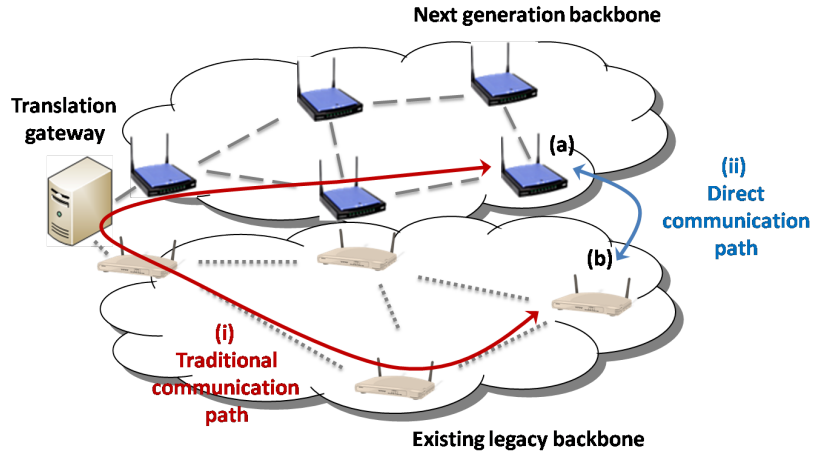

Figure 4.4: The coverage of an existing legacy network is expanded by installing an additional next-generation backbone. (i) Using existing technology, all communication must pass through a translation gateway, resulting in suboptimal use of the network. (ii) A next-generation IDRA network can converge with existing networks and use direct communication paths, thus prolonging the operational lifetime of legacy networks.

speed connections and finally an expensive UMTS connection is used to provide connectivity to remote parts of the industry terrain. When a resource-constrained Body Area Network (BAN) is introduced to monitor the health of the employees, the BAN nodes should be able to connect directly to all existing co-located networks, without the use of a remote gateway (Figure 4.5).

When using IDRA, it is not necessary to install a full protocol stack for each of these diverse communication technologies. As such, IDRA can implement 'always best connected' (ABC) solutions even on resource constrained internet of things devices.

To realize these use cases, IDRA has built-in features that are able to cope with the following network challenges: (i) heterogeneous (legacy) devices can use different packet types, (ii) heterogeneous (legacy) devices can use conflicting medium access mechanisms and (iii) heterogeneous (legacy) devices can use different higher layer network protocols

4.5.1 Connecting devices that use different packet types

In a heterogeneous environment, multiple packet types can transparently reside on the same node at the same time. The IDRA packet facade is able to interpret any in-

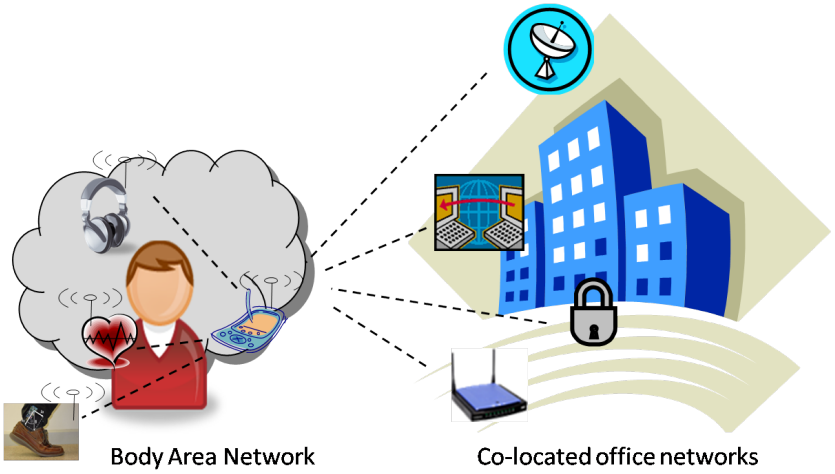

Figure 4.5: A resource-constrained personal Body Area Network (BAN) monitors the health of an employee. For efficient communication, the BAN should be able to communicate directly with all co-located network technology such as wireless entrance and security control, UMTS, Wi-Fi and DECT.

coming packet type, as long as the correct packet part descriptors are available. To this end, IDRA includes a packet identification service that indicates which packet descriptors should be used to interpret incoming packets. To identify *incoming packets*, one of the following identification services can be used.

- Networks that utilize multiple communication technologies can associate a packet type with each interface (e.g: an 802.11 packet type for the Wi-Fi interface, an 6LoWPAN packet type for the IEEE 802.15.4 interface, etc).
- Alternatively, in case multiple packet types can arrive on the same interface, a publicly available, standardized packet type can be added as a unique packet identification field to each outgoing packet.
- If the radio offers hardware address recognition features, the address of the sending node can be identified. In this case, the neighbor table is used to describe the expected packet type of each neighboring node. This approach is not possible for networks that use non radio-compliant MAC headers or networks that include address-free communication interfaces (such as USB interfaces).
- Finally, a last option is to compare incoming packets with the descriptors of existing packet descriptors using bitmap operations.

This wide range of identification methods ensures that network designers can always choose the most optimal method for identifying incoming packets. The

IDRA system automatically drops all packets that are not recognized by any of these packet identification services.

To select the correct *outgoing* packet type, a configurable shared neighbor table is used. For each of its neighbors, an entry is available in the shared neighbor table that indicates the preferred packet type, routing protocol and MAC protocol. IDRA will automatically select the correct MAC protocol and sent the packet over the correct radio interface. This shared neighbor table can be configured at run-time or at compile-time.

In heterogeneous networks, the outgoing packet type might be different from the incoming packet type. *Packet conversion* occurs when an outgoing packet must be transmitted to a neighbor that is associated with a different packet type. When packet conversion is required, the packet facade is used to create a new packet of the correct type. Next, the packet facade extracts all packet attributes from the original packet (thus dismantling the original packet). Finally, the packet facade is used to add all extracted packet attributes to the newly created packet. The conversion process is fully transparent for the network protocols: the network protocols can not distinguish the new packet from the original packet.

4.5.2 Connecting devices with different MAC protocols

IDRA can also support communication with (legacy) devices that use different MAC protocols. To this end, both the legacy and the new MAC service can be registered to manage the same network interface. Using the shared neighbor table, the IDRA service broker will automatically use the correct MAC service when sending packets to the legacy nodes. Also, using packet filters incoming packets can be directed towards the correct MAC protocol based on their packet type or other packet attributes. Finally, IDRA includes several simple algorithms for resolving MAC conflicts that occur when multiple MAC protocols want to manage the same radio interface. For example, the radio will only be disabled when all registered MAC protocols have requested a low power radio state.

4.5.3 Connecting devices which use different routing protocols

To cope with different routing protocols, the legacy routing protocol can be installed on the IDRA device as an additional routing service. The dynamic network service broker can be configured to use this routing protocol for any packet that goes to (or comes from) a legacy device. In addition, by providing the correct packet descriptor, IDRA nodes can interpret legacy headers to retrieve the source and destination of each (legacy) packet. As such, IDRA devices can route legacy packets to their destination using a new state-of-the-art routing protocol, without changing the packet structure. This way, next-generation IDRA devices can be used to transparently route packets from legacy nodes.

4.6 Feasibility of the concepts

According to [14, 40], the development of actual prototypes is crucial to prove the merit of future protocol architectures. To prove the feasilibily of the IDRA concepts, IDRA has been tested in several network deployments.

4.6.1 Proof-of-concept implementations

The concepts above have all been implemented in the DEUS project [41] using a large scale testbed of 200 TMoteSky nodes and two real-life network deployments (the arts center 'Vooruit' and a home for the elderly). Devices were installed that use different routing protocols (DYMO, HYDRO or AODV) [42]. Depending on their neighbors and the packet metadata, the nodes automatically selected the appropriate routing protocol, thus enabling direct connectivity between these different devices. As a result, even on such a large scale, devices running IDRA are capable of efficient direct communication by using packet conversion and a dynamic network service broker on each intermediate hop.

IDRA currently includes several network services such as routing, MAC, topology control, duplicate detection, packet identification, packet ownership, quality-of-service and localization services, which can all be combined as required by the network. A full list of experimental IDRA deployments can be found at http://idraproject.net/content/experimental-validation.

4.6.2 Business aspects

Apart from technical aspects, one of the main drivers behind innovation are marketable results. In regards to IDRA, we identified the following economic advantages.

- Due to its low memory and processing requirements, IDRA can reduce the hardware costs of involved devices.
- Low cost end devices can be included in IP networks since IP packets can be generated and processed even without a full protocol stack.
- By activating the built-in aggregation service, increased wireless throughput can be provided and battery powered devices have a longer functional lifetime.
- IDRA transparently supports an 'always best connected' strategy between different technologies at all network levels. Network cooperation can save the consumer money. For example: when watching videos on a cell phone, rather than using an expensive 3G network, the cell phone can connect with a body area network (BAN) using the bluetooth connection. The BAN, in

turn, can make a connection with a nearby Wi-Fi gateway to provide cheap internet access.

- The architecture supports the concept of dynamically plugging in new network services whenever required. This paves the road for pay-per update services and stimulates the development of companies supporting network services.
- Finally, IDRA can be used to deploy next-generation networks while still supporting existing legacy networks.

These advantages can be exploited by business innovators even before the commercialization and roll-out of a large scale internet of things.

4.7 Conclusions

Our everyday environment is equipped with an increasing number of communication technologies, from high speed internet backbones to 'last mile' access and cable replacement technologies such as Wi-Fi, bluetooth or UMTS. This trend will likely not change, prompting the need for internet of things architectures that inherently cope with this diversity. This chapter gave an overview of existing promising architectural approaches. However, at the moment, none of these offers a solution to enable efficient direct communication between co-located resource-constrained devices that use different communication technologies.

This chapter argues that this gap can be filled by the IDRA architecture. To motivate this, a broad overview of the IDRA architecture was given, and it was motivated how the discussed concepts fit within the vision of the internet of things. The following contributions of IDRA towards a future internet architecture were discussed in greater detail. (i) IDRA is a clean slate architecture, but is backwards compatible with legacy networks. (ii) The complexity of IDRA is low enough to implement even on resource-constrained devices. (iii) IDRA enables direct communication between co-located networks without the use of complex translation gateways. In addition, (v) IDRA copes with changing network and application requirements by introducing a dynamic service broker responsible for selecting the most optimal network services. And finally, (iv) IDRA has fully been implemented and evaluated, which proves the feasibility of the proposed concepts. As such, the IDRA architecture is a promising candidate to connect heterogeneous next-generation networks in a straightforward way, while supporting many of the requirements of the internet of things at an architectural level.

References

[1] *The Internet of Things*. ITU Internet Reports, 2005.

[2] P. De Mil, T. Allemeersch, I. Moerman, P. Demeester, and W. De Kimpe. *A Scalable Low-Power WSAN Solution for Large-Scale Building Automation*. In Communications, 2008. ICC '08. IEEE International Conference on, pages 3130–3135, May 2008.

[3] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, R. Stoleru, S. Lin, and J. A. Stankovic. *An Advanced Wireless Sensor Network for Health Monitoring*. Transdisciplinary Conference on Distributed Diagnosis and Home Heathcare (D2H2), Arlington, VA, April 2-4, 2006.

[4] Hairong Yan, Youzhi Xu, and M Gidlund. *Experimental e-Health Applications in Wireless Sensor Networks*. volume 1, pages 563 –567, Jan. 2009.

[5] A. Dunkels and JP Vasseur. *"IP for Smart Objects". White Paper*, September 2008.

[6] K. Mayer and W. Fritsche. *IP-enabled wireless sensor networks and their integration into the internet*. In Proceedings of the First international Conference on integrated internet Ad Hoc and Sensor Networks. InterSense '06. Nice, France, vol. 138, May 30 - 31, 2006.

[7] Eli De Poorter, Ingrid Moerman, and Piet Demeester. *An Information Driven Sensornet Architecture (best paper award)*. In The Third International Conference on Sensor Technologies and Applications (sensorcomm 2009), Athens/Glyfada, Greece, June June 18-23, 2009.

[8] Peter Stuckmann and Rainer Zimmermann. *European Research on Future Internet Design*. IEEE Wireless Communications Magazine, October 2009.

[9] Sung-Su Kim, Mi-Jung Choi, Hong-Taek Ju, Masayoshi Ejiri, and James Won-Ki Hong. *Towards Management Requirements of Future Internet*. Lecture Notes in Computer Science: Challenges for Next Generation Network Operations and Service Management, Volume 5297:pp 156–166, October 16, 2008.

[10] Frank-Uwe Andersen, Hendrik Berndt, Henrik Abramowicz, and Rahim Tafazolli. *Future Internet: From Mobile and Wireless Requirements Perspective*. eMobility Technology Platform Whitepaper, 2007.

[11] Kideok Cho, Jaeyoung Choi, Dong il Diko Ko, Taekyoung Kwon, and Yanghee Choi. *Content-Oriented Networking as a Future Internet Infrastructure: Concepts, Strengths, and Application Scenarios*. Proc. of International Conference on Future Internet Technologies (CFI) 2008, Seoul, Korea, June 2008.

[12] *FP7 Self-NET project: Self-Management of Cognitive Future InterNET Elements*. https://www.ictselfnet.eu/.

[13] BOUTABA Raouf, OMARI Salima, and AJAY PAL SINGH VIRK. *SELFCON: An architecture for self-configuration of networks*. Journal of communication and networks, ISSN 1229-2370, vol. 3, no 4:pp. 317–323, 2001.

[14] Anja Feldmann. *Internet clean-slate design: what and why?* SIGCOMM Comput. Commun. Rev., Vol. 37, No. 3:pp. 59–64, 2007.

[15] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. *A survey of network virtualization*. Computer Networks, 54(5):862 – 876, 2010.

[16] S. Khanvilkar and A. Khokhar. *Virtual Private Networks: An Overview with Performance Evaluation*. IEEE Communication magazine, Vol. 42(10):pp. 146–154, October, 2004.

[17] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. *A Survey and Comparison of Peer-to-Peer Overlay Network Schemes*. IEEE Communications Surveys & Tutorials, pages pp. 72–93, Second Quarter 2005.

[18] *The FP7 IST 4WARD project, Architecture and Design for the Future Internet*. http://www.4ward-project.eu/.

[19] *FP7 IST 4WARD project, D2.1 Technical Requirements*.

[20] *The FP6 MAGNET and MAGNET beyond project, http://magnet.aau.dk/*.

[21] Ramjee Prasad, editor. *My personal Adaptive Global NET (MAGNET), ISBN: 978-90-481-3436-6*. Springer, Signals and Communication Technology, 1st Edition (22 Dec 2009), XXXIV, 435 p., Hardcover.

[22] *The ITEA2 usenet project, https://usenet.erve.vtt.fi/*.

[23] Jeroen Hoebeke, Gerry Holderbeke, Ingrid Moerman, Bart Dhoedt, and Piet Demeester. *Virtual Private Ad Hoc Networking*. Wireless Personal Communications, 38:125–141, 2006. 10.1007/s11277-006-9021-1.

[24] N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. *Network Virtualization: State of the Art and Research Challenges*. IEEE Communications Magazine - IEEE ComSoc,, no. 47, no. 7:pp 20–26, Jul. 2009.

[25] N. Niebert, I. El Khayat, S. Baucke, R. Keller, R. Rembarz, and J. Sachs. *Network Virtualization: A Viable Path Towards the Future Internet*. Wireless Personal Communications, vol. 45(4):pp. 511–520, June 2008.

[26] James Roberts. *The clean-slate approach to future Internet design: a survey of research initiatives*. Annals of Telecommunications, Volume 64, Numbers 5-6, June, 2009.

[27] *FP7 Sensei project*. http://www.ict-sensei.org/.

[28] FP7 SemSorGrid4Env project. *http:// www.semsorgrid4env.eu/*.

[29] David Chu, Joseph M. Hellerstein, and Tsung te Lai. *Optimizing declarative sensornets*. In SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems, pages 403–404, New York, NY, USA, 2008. ACM.

[30] M. D. Dikaiakos, D. Katsaros, G. Pallis, A. Vakali, and P. Mehra. *Guest Editors Introduction: Cloud Computing*. IEEE Internet Computing, Vol. 12(5), Sep. 2009.

[31] Luis M. Vaquero et al. *A Break in the Clouds: Toward a Cloud Definition*. ACM SIGCOMM Computer Communication Review, ISSN:0146-4833, Volume 39, Issue 1:Pages 50–55, January 2009.

[32] M. Pistore, P. Traverso, M. Paolucci, , and M. Wagner. *From software services to a future internet of services.* Towards the Future Internet: A European Research Perspective. IOS Press, Amsterdam, The Netherlands, page pages 183192, 2009.

[33] Arsalan Tavakoli, Prabal Dutta, Jaein Jeong, Sukun Kim, Jorge Ortiz, David Culler, Phillip Levis, and Scott Shenker. *A modular sensornet architecture: past, present, and future directions*. SIGBED Rev., 4(3):49–54, 2007.

[34] David Clark et al. *NewArch Project: Future-Generation Internet Architecture*. http://www.isi.edu/newarch/, 2003.

[35] Robert Braden, Ted Faber, and Mark Handley. *From protocol stack to protocol heap: role-based architecture*. SIGCOMM Comput. Commun. Rev., 33(1):17–22, 2003.

[36] *NSF FIND (Future Internet Design) research program. FIND Informational Meeting: Lessons from FIND 2006*. http://www.nets-find.net/Meetings/FirstPIMeeting/FirstInfoMeeting.ppt, November 7, 2006.

[37] *NSF FIND research program. Observer Panel Report 2009. Vint Cerf, Bruce Davie, Albert Greenberg, Susan Landau and David Sincoskie*. http://www.nets-find.net/FIND_report_final.pdf, April 9, 2009.

[38] Hubert Zimmermann. *OSI Reference Model The ISO Model of Architecture for Open Systems Interconnection*. IEEE Transactions on Communications, 28, no. 4:425 – 432, April 1980.

[39] D. Clark, R. Braden, A. Falk, and V. Pingali. *FARA: Reorganizing the Addressing Architecture*. Proc. ACM SIGCOMM FDNA Workshop, Karlruhe, August 2003.

[40] *NSF GENI project - Global Environment for Network Innovations*. http://www.geni.net/.

[41] *The DEUS project: Deployment and Easy Use of wireless Services. http://www.ibbt.be/project/DEUS*.

[42] *DEUS project leaflets: Wireless Sensor Network. https://projects.ibbt.be/deus*.

5

A Negotiation-Based Networking Methodology to Enable Cooperation Across Heterogeneous Co-located Networks

In a future internet of things, an increasing number of every-day objects will become interconnected with each other. To cope with these increasingly large and heterogeneous networks, this chapter presents an 'incentive-driven' networking approach that aims to (i) increase the network performance of co-located devices through cross-network cooperation and (ii) simplify the configuration and setup of networks for the end-users. To this end, a negotiation approach is presented that takes into account the network preferences of individual devices. The incentive-driven negotiation methodology enables efficient network cooperation between heterogeneous devices through the use of cross-layer and cross-network optimizations. The feasibility of the methodology is demonstrated through an experimental proof-of-concept implementation that optimizes the network performance of two networks of resource-constrained embedded devices. Finally, to show that the proposed methodology is applicable to a wide range of marketable applications, the chapter describes a large number of example use cases that can benefit from incentive driven networking.

5.1 Introduction

In the future, an increasing number of objects will be (wirelessly) connected with each other [1, 2]. The rising popularity of wireless car ports, televisions, radios, rolling shutters and different types of environmental sensors demonstrates that even every-day household objects will come equipped with (wireless) communication possibilities. Supporting connectivity between these fixed and mobile objects enables an increased interactivity with our environment, which in turn enables wireless next-generation applications such as wireless building automation, automated e-health solutions, interactive museum exhibitions and personalized entertainment systems [3, 4].

Nowadays, supporting connectivity between these co-located devices is supported by manually grouping together the different devices in separate subnets based on their network technology. Regardless of the characteristics of the devices, the same network configuration and network policies are used for all the devices of a single subnet.

However, this manual approach is complex and inefficient [5].

- Due to the sheer amount of co-located devices, a manual configuration approach is very time-consuming and expensive, especially in large-scale networks.

- Manual or static configuration approaches do not take into account dynamically changing network requirements such as networks that change over time, networks that use mobile devices or networks that are formed spontaneously after impromptu encounters [6].

- Devices from different subnets typically ignore each other, resulting in harmful interference [7] and missed opportunities for cooperation.

- Finally, current networking approaches most often do not take into account the heterogeneity of next-generation networks. Different devices typically have (i) different network preferences and (ii) different hardware and software capabilities [8].

As such, there is a need for network solutions that efficiently support at runtime cooperation between devices from different subnets while taking into account the diverging goals and capabilities of the networked objects. To fill this gap, this chapter presents a networking approach called 'incentive driven networking' that aims to both (i) *increase the network performance* of co-located devices through cross-network cooperation and (ii) *simplify the configuration* and setup of networks for the end-users.

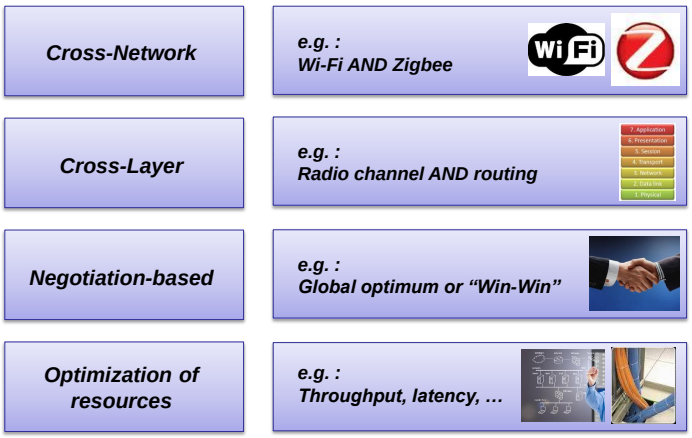

Figure 5.1: The characteristics of incentive driven networking.

Incentive driven networking describes a *cross-layer, cross-network negotiation methodology* for optimizing network resources such as throughput or latency (Figure 5.1). Using our methodology, devices (even from different owners) can engage in efficient cooperation with co-located devices that have different network preferences and capabilities, ultimately resulting in an overall increase in the network performance.

The remainder of this chapter is as follows. Section 4.1 argued that current networks are not designed to support interconnected objects that differ in terms of behavior and network requirements. As a possible solution, this chapter introduces incentive driven networking. Section 5.2 defines the terms and concepts that are used in incentive driven networking. Afterwards, Section 5.3 presents the incentive driven methodology that is used to realize the negotiation based network optimization. The feasibility of the proposed concepts is defended in Section 5.4, where the performance of an experimental implementation is evaluated. Next, Section 5.5 gives an overview of related network approaches. Afterwards, Section 5.6 gives an overview of potential marketable applications that can be realized using the proposed incentive driven networking concepts. Finally, Section 5.8 concludes the chapter.

5.2 Terminology

Before exploring the methodology, the key components of the cooperation schemes are described in more detail.

5.2.1 Incentives

Each device involved in incentive driven networking has a number of well-defined incentives that describe the preferred high-level network behavior. An incentive can either (i) describe behavioral aspects of the network (i.e: 'limit the battery consumption'); or (ii) express the need for additional functionality (i.e: 'get internet access'); or (iii) give an indication of the expected performance network metrics (i.e: 'support video streaming'). Example incentives are the following:

- HIGH_THROUGHPUT, HIGH_RELIABILITY or LOW_DELAY (to obtain better QoS guarantees)
- HIGH_NETWORK_LIFETIME (to prevent frequent battery replacement)
- HIGH_COVERAGE (to reach more clients)
- LOW_EXPOSURE (due to health regulations)
- GET_PUBLIC_ACCESS (to get internet connectivity)

Incentives describe the 'reasons for cooperation': devices will only engage in cooperation with other devices when this cooperation is beneficial for the incentives of the participating nodes. The incentives of a device are typically set by the application, or configured manually by a network administrator.

5.2.2 Communities

A community is defined as a set of nodes that have derived common incentives ('network goals'). As such, a community describes a set of co-located nodes that have the same network behavior and the same network goals: they are similar in terms of capabilities (such as available services) and incentives. All devices of a single community should be able to communicate with each other (either directly, or through intermediate devices that are part of the same community). As an example, the devices of an office building can be divided into the following three separate communities: Wi-Fi enabled devices that are battery powered, Wi-Fi enabled devices that are plugged into a power line and UMTS capable devices.

Devices require a trust relation with all other community members before joining a community. Devices that belong to the same owner are implicitly assumed to trust each other. Otherwise, a trust relationship can be established through the use of certificates which are issued by a (remote) trusted certification authority.

5.2.3 Network services

An incentive can be realized using a large number of networking techniques. For example, the reliability incentive can be improved by using retransmission

Network service	Description	Expected influence on			
		Through-put	Delay	Relia-bility	Network Life-time
Shared routing	Allows cooperating communities to interpret and route packets from other communities	+	+	+	±
Interference avoidance	Communities cooperate by selecting the transmission frequencies which are least harmful for each other	+	±	+	±
Coordinated sleeping schemes	To conserve energy, the devices from the involved communities use matching sleep schemes	-	-	±	+
Packet aggregation	To reduce the number of transmissions, multiple information exchanges are aggregated into a single packet	+	-	-	+

Table 5.1: Example list of network services and their influence on community incentives (+: positive influence, -: negative influence, ±: variable or no influence).

schemes, by increasing the transmission power or by using advanced error correction codes. These optimization techniques, that influence one or more of the incentives, are called network services. Thus, whereas incentives indicate network goals, network services are the means to realize these goals.

Table 5.1 lists several example network services and their influence on the incentives. A network service is not crucial for the correct working of the individual communities, but can be activated or deactivated in a community depending on the required incentives of the communities. For example, activating retransmissions will positively influence the reliability, at the cost of a lower network lifetime.

5.2.4 Negotiation profiles

To enable negotiation, the characteristics of each community are described in negotiation profiles. A negotiation profile should contain at least the following information.

- A timestamp (time of last update).
- A certificate guaranteeing that the community can be trusted.
- The community ID and priority.

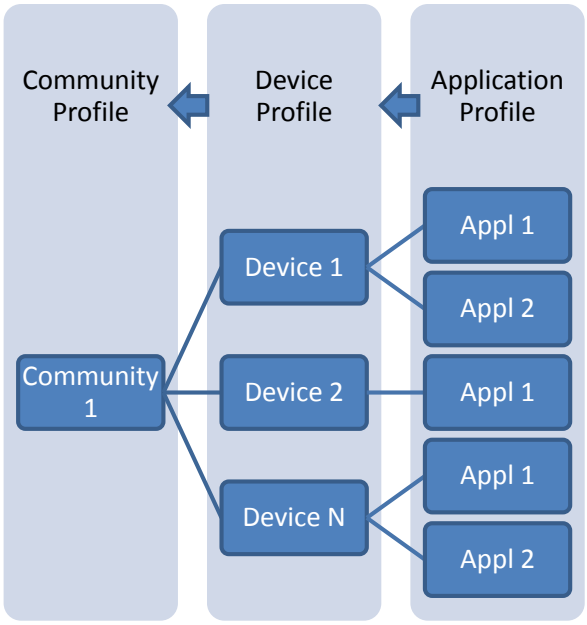

Figure 5.2: Profiles are constructed in a hierarchical manner. One or more application profiles are combined in a single device profile. Similarly, a community profile is generated based on the profiles of all participating devices.

- A list of incentives and their associated importance for the community.
- A list of available network services.
- A description of the configurable settings (transmission frequencies, available packet types, etc).

Profile information can be represented using new or existing standardized XML schemes [9]. Alternatively, resource-constrained networks can utilize more efficient binary formats to represent the profile information.

Negotiation profiles are constructed in a hierarchical manner (Figure 5.2). The incentives of the applications are described in an *application profile*. If only a single application is deployed on a device, a direct conversion from application incentives to device incentives is possible. However, if multiple applications are deployed on the same device, the application profiles are merged into a *device profile*. In the case of conflicting incentives, different application priorities can be used to prioritize certain incentives. Similarly, the *community profile* represents a merged representation of the device profiles of all participating nodes (see Section 5.3.1).

5.2.5 Incentive driven networking

For optimal network performance, it is important to make intelligent decisions about which network services should be activated. Since incentives describe high-level network requirements, an incentive can be improved using different network services. For example, the incentive 'reliability' can be improved by utilizing better error correction codes, by activating packet acknowledgments or by using reliable routing protocols. To identify which set of network services should be activated, it is important to note that a single network service often influences multiple incentives. For example, the additional transmissions required for packet acknowledgments influence the network lifetime of battery-powered devices, and the use of reliable routing protocols might increase the end-to-end delays.

Incentive driven networking is defined as the selection and activation of the optimal set of network services in each community with the goal to optimize the incentives of each participating community. Based on this definition, the distinction between incentives and network services can be understood as follows: a network service can be activated or deactivated, whereas an incentive indicates a high-level application or management objective.

5.3 Incentive driven networking methodology

As stated before, our incentive driven networking approach aims to globally optimize network resources through negotiation based cross-layer and cross-network optimizations. Figure 5.3 gives a general overview of the discussed incentive driven network methodology. The methodology to support incentive driven networking consists of the following 5 phases, all of which will be discussed in more detail:

1 First, *communities* of similar devices *are created*.

2 The communities use varying communication technologies to *discover* each other.

3 After discovery, the communities *negotiate* about the optimal set of network services.

4 This is followed by the actual *activation* of the services.

5 Finally, the communities *monitor* if all services are actually deployed and if the communities behave correctly.

Figure 5.3: The 5 phases of the incentive driven network methodology

5.3.1 Phase 1 - Community Creation

Initially, devices are deployed with a simple (standardized) MAC and routing protocol. They can communicate with co-located devices, but no network services are yet activated. After deployment of the devices, the nodes first find out if they can form a community with similar co-located devices. Joining a community has both benefits and disadvantages for a device. When a device joins a community, the community incentives might differ from the incentives of the individual device. In this situation, the community will optimize towards incentives that are suboptimal for the joining device. On the other hand, by joining a community, the device enters a stronger negotiation position, since a community can negotiate on behalf of a large group of nodes.

The partitioning of devices into separate communities can occur either at run-time or at design-time. To cope with dynamically changing network conditions, as well as to avoid complex and time-consuming manual network configuration, a non-manual approach is preferred. The end result of this phase is a partitioning of the devices into communities of directly connected devices. As an example, the outline of a simple at run-time partitioning protocol is presented in Algorithm 2. The algorithm assumes that the incentives, services and settings of each community are described using a standardized community profile, which is transmitted over a predetermined radio frequency. The algorithm starts when each individual device creates a device profile based on the requirements of its high-level appli-

Require: initial condition: all devices form their own community
1: **repeat**
2: Each device single-hop broadcasts its profile P_0 over all available communication interfaces.
3: From the received profiles P_i, select the one that best matches your own profile P_0
4: Propose to P_i to be part of the community
5: **if** P_i accepts **then**
6: // create common community profile
7: $P_0 \leftarrow merge(P_0, P_i)$
8: **else**
9: Ignore P_i from now on
10: **end if**
11: **until** No new devices can be added to the communities

algorithm 2: Outline of a simple algorithm for the partitioning of devices into different communities

cations. During the algorithm, devices search for co-located devices which are similar enough to be part of the same community. When no more compatible co-located devices are found, the algorithm is finished. In environments that are strongly heterogeneous, a community might be as small as a single device.

Of course, alternative approaches are possible. Whichever method is used, the end-result of this step is that all co-located devices are divided over different communities consisting of devices with similar services and incentives. Since communities are independent entities, it is now possible to optimize the performance of each individual community. However, optimizing at this phase of the methodology might not always be beneficial. (i) It is not yet known how the network services will influence the incentives of neighboring communities. To cooperate with other networks, the negotiation output might require that some of these network services are disabled again, thus resulting in unnecessary set-up and configuration overhead. (ii) Additionally, some optimization techniques complicate the discovery and negotiation process. For example, utilizing encryption schemes or channel hopping schemes might prevent different communities from detecting each other.

As such, depending on the complexity and length of the 'community discovery' and 'community negotiation' phases, it can be beneficial to delay some (or all) network optimizations until after the negotiation process.

5.3.2 Phase 2 - Community Discovery

In the previous phase, devices were partitioned into separate communities with similar incentives and services. Now, the different communities find out if they are co-located with other communities capable of incentive driven networking.

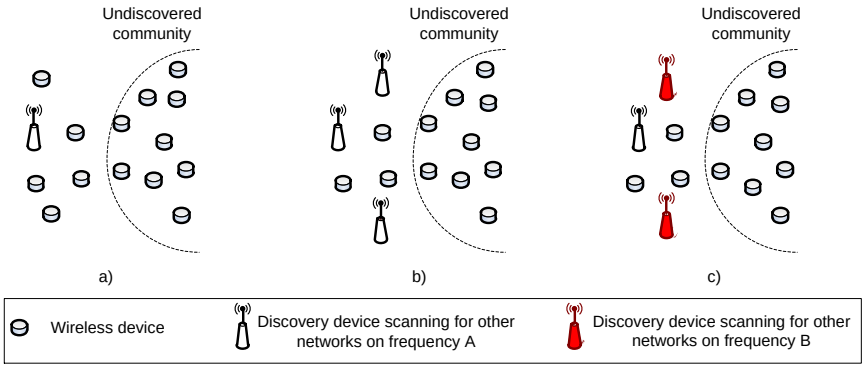

Figure 5.4: Distributed Commmunity Discovery. a) A subset of the discovery devices is used for detecting other communities. b) Adding more discovery devices increases the probability of successful detection. c) Multiple discovery devices can transmit discovery beacons in parallel on different frequencies to ensure a timely detection of co-located communities.

Community discovery consists of the following steps:

1 *Assignation of discovery nodes*. Each community decides on the optimal number of devices that are needed to detect co-located communities. To bear minimal impact on the network performance, a subset of discovery nodes can suffice (see Figure 5.4).

2 *Community discovery*. Next, the discovery devices are used to detect the co-located communities. Community detection can be *passive* (i.e.: discovery devices passively scan for recognized packets on multiple frequenties in order to overhear existing communities [10]) or *active* (i.e.: by broadcasting community advertisement messages over multiple frequencies containing information about the network settings that should be used to contact the advertising community). The end result of this step is that single hop communication between the discovery nodes of different communities is possible.

3 *Profile exchange*. Afterwards, the discovered communication settings are used to exchange the community profiles between the discovery nodes.

4 *Forward the received profile*. Finally, received community profiles are forwarded to the 'negotiation' entity of each community (see next section).

5.3.3 Phase 3 - Community Negotiation

By now, the co-located communities have exchanged profiles which describe the incentives of each community. The next steps investigate if cooperation between different co-located communities (in the form of activating cross-network services such as interference avoidance) is beneficial. To be able to participate in this step of incentive driven networking, each participating community should have a negotiation entity. This negotiation entity is either a single, central manager that is trusted by both communities or an entity that is distributed over several nodes of each community. Negotiation consists of the following phases:

1 *Announcement of negotiation entity.* The negotiation entity of each community regularly announces its presence to all nodes of the community by broadcasting 'negotiation advertise' messages.

2 *Collection of community profiles.* All received community profiles are forwarded to the nearest negotiation entity where the negotiation process is initialized.

3 *Determine an influence rating for each service.* For each available network service, the negotiation manager determines how the activation of the available service will influence the incentives of each community. For example, enabling aggregation can increase the network lifetime incentive by 30% [11]. To agree on estimated influence of network services, results can be used from (i) existing literature, from (ii) network simulators or from (iii) network monitoring agents.

4 *Calculate optimal set of network services.* Based upon these influence ratios, the negotiation entity calculates the optimal selection of services that should be activated.

To calculate the optimal set of network services, several negotiation approaches are possible based on methods such as game theory, self-learning approaches or mathematical formulas. As an example, below a heuristic ILP formulation of the negotiation process is derived which can be applied to any number N participating networks, using the notations from Table 5.2.

Assuming:

$$\sum_{i=0}^{I} IW_{i,a} = 1, \qquad \forall\ a = 0..N \tag{5.1}$$

Then maximize:

$$\sum_{a=0}^{N} CP_a * profit_a \tag{5.2}$$

Symbol	Meaning
CP_a	The priority of community a. Under normal operations, the priority of each community equals one. However, the performance of certified emergency networks can be improved by giving them a higher priority, at the cost of a lower network performance of the other participating communities.
$profit_a$	The profit function of community a. This objective function should be maximized for each network to optimally profit from incentive driven networking.
$IW_{i,a}$	The weight factor that is given to incentive i in community a.
$SI_{i,a;s,b}$	The percentage by which incentive i from community a is improved when service s is activated in community b. These values can be configured at design-time, or monitoring agents can use learning techniques to intelligently monitor and change these percentages at run-time.
$SA_{s,a}$	A binary variable (0 or 1) that indicates if service s is activated in community a. These variables are determined as the end result of the linear program.

Table 5.2: List of variables used during the negotiation process.

subject to:

$$profit_a = \sum_{i=0}^{I} \left[IW_{i,a} * \left\{ 1 + \sum_{s=0}^{S} \sum_{b=0}^{N} (SA_{s,b} * SI_{i,a;s,b} \right\} \right] \tag{5.3}$$

$$profit_a \geq 1, \qquad \forall\ a = 0..N \tag{5.4}$$

$$SA_{s,b} = \begin{cases} 1 & \text{if service s is activated in network b;} \\ 0 & \text{if service s is deactivated in network b.} \end{cases} \tag{5.5}$$

With:

$$\begin{cases} N = & \text{the total number of communities participating in the negotiation process.} \\ I = & \text{the total number of incentives.} \\ S = & \text{the total number of available services.} \end{cases}$$

Formula (5.1) enforces that the sum of the incentive weights of each community is normalized to one. For example, consider the situation where two applications are running on node a. One application requires a maximal throughput, whereas the other requires a long network lifetime. Assuming both applications are equally important, the incentive weights would be divided equally amongst both applications, resulting in $IW_{throughput,a} = IW_{lifetime,a} = 0.5$.

Formula (5.2) specifies that the total profit function should be maximized. The total profit is calculated as the sum of the profits of all N communities weighted by their priority.

The profit of each individual community a is calculated in Formula (5.3). The formula evaluates how each service influences the incentives of community a when activated. Since initially no services are used ($SA_{s,b} = 0, \forall s = 0..S, \forall b = 0..N$) the profit without incentive driven networking equals one. When new services are added, these services increase or decrease the value of the incentives (for example: using sleep schemes might improve the 'network lifetime incentive' by 60%). Depending on the weights of the incentives, the profit function of a community will favor different incentives.

The condition described in Formula (5.4) ensures that the performance of none of the participating communities is degraded after cooperation. If no solution is found that results in better performance for a community a, this community will not participate in the cooperation. Optionally, condition (5.4) can be omitted when a community agrees to accept a decreased performance (for example to support nearby emergency networks).

Finally, the last condition in Formula (5.5) indicates that $SA_{s,b}$ are binary variables.

After solving the ILP formulation, the binary variables SA indicate which services should be activated to maximally increase the objective function of all involved communities. Based on the description of the network services, additional constraints can be added. For example: if a network service needs to be activated over both communities (such as when using frequency hopping), the condition $SA_{freqhop,a} = SA_{freqhop,b}$ is added. Similarly, if network service a requires the activation of network service b, the condition $SA_{service,a} \leq SA_{service,b}$ is added.

In some cases, optimal incentive driven networking might require that the communities do not merge, but try to avoid each other through the use of interference avoidance mechanisms. Finally, in case no neighboring communities are found, or if the negotiation process with the neighboring communities fails, the same linear program is used to optimize the performance of only a single community (by ignoring the incentives of neighboring communities).

5.3.4 Phase 4 - Enabling of the incentive driven cooperation

After selecting the optimal set of services in each community, the next phase selects and activates the services in all involved communities. In addition, the settings of both networks are configured such that communication between the different networks is possible.

1 *Propagation of the proposition.* The negotiation entity forwards the decision about the service selection to the discovery nodes of the community. These

devices know how to contact the other communities and relay the proposition to the co-located communities.

2 *Additional negotiation (optional).* Depending on the negotiation approach, the conclusions about the optimal set of services reached by both communities might differ. In this case, additional negotiation is required to find a set of services that both communities can agree on.

3 *Confirmation and distribution.* Once all involved communities agree on the service selection, the negotiation server distributes the chosen set of activated services and network settings in both communities.

4 *Service migration (optional).* If devices are missing certain network services, these services can be exchanged or installed remotely.

5 *Activate the settings and services.* Once both communities have received the optimal settings and services, the communities simultaneously switch to the selected configuration. It is possible that some services are activated only for specific packets. For example: a QoS service can be activated to process only packets that contain real-time information.

5.3.5 Phase 5 - Policy enforcement

Finally, communities will want to check if all other communities are 'playing by the rules', i.e.: are not cheating. For example, a monitoring agent can be used to (i) investigate if the selected services are actually activated and performing as expected and (ii) monitor the actual influence ratios of the activated services. If the set of services changes, or the measured influence ratios differ greatly from the influence ratios used in the negotiation, a new negotiation process is started.

5.4 Proof-of-concept implementation

This section experimentally measures the benefits of incentive driven networking. To this end, a proof-of-concept implementation of the cooperation approach was implemented on resource-constrained TMoteSky sensor nodes [12].

5.4.1 Experimental setup

For our experiments, the iLab.t wireless sensor testbed [13, 14] was used, which is located in the IBBT - Ghent University office building in Belgium. Figure 5.5 shows the location of the TMoteSky nodes.

For the proof-of-concept, temperature monitoring sensor nodes (A) were installed in multiple rooms. These devices send a temperature report every 10 seconds to the HVAC control unit ('sink device A'), so that the heating, ventilation

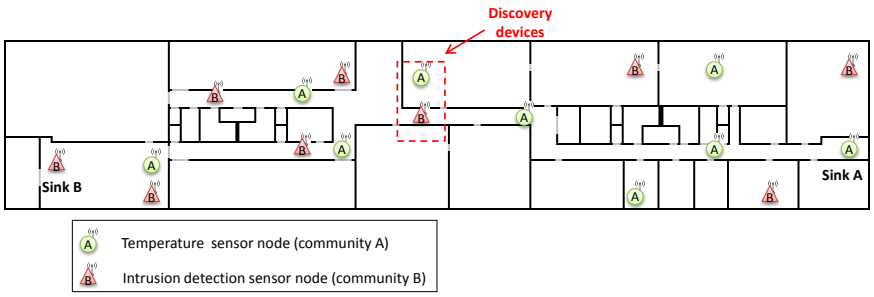

Figure 5.5: The network used in the proof-of-concept demonstrator. Two types of nodes are deployed: battery-powered temperature monitoring devices (A) and reliable intrusion detection security nodes (B).

and air conditioning system (HVAC) can be optimized to reduce its energy consumption. To prevent frequent battery replacements, the main incentive of these battery-powered nodes is the network lifetime ($IW_{lifetime;a} = 0.7$), with reliability as a secondary incentive ($IW_{reliability;a} = 0.3$).

Due to recent burglaries in the area, the proof-of-concept owner decides afterwards to also purchase a wireless anti-theft system, to be installed and operated by an external security firm. Intrusion detection sensor devices (B) are installed at key locations in the building. Since this network has a more critical function, these sensor devices are powered by high-capacity batteries. Every 10 seconds a security report is forwarded to a monitoring PC ('sink device B') which can be accessed remotely by the external security firm. These high-priority information exchanges represent information such as status updates, intrusion alerts or static images from a webcam. To ensure a timely reaction in emergency situations, the devices have stringent delay and reliability incentives ($IW_{delay;b} = 0.5; IW_{reliability;b} = 0.5$).

Multi-hop experiments are created by setting the transmission power of the sensor nodes to an output power of -15 dBm. Using these settings, packets require maximum 4 hops to be transmitted from one side of the building to the opposite side. The AODV protocol [15] is used to route all exchanged packets.

5.4.2 Overhead of the community discovery process

When the communities are deployed, they initially function independent from each other at different radio frequencies. On each floor, one device of each community was manually assigned the role of discovery device (indicated on Figure 5.5). The discovery algorithm is illustrated in Figure 5.6. The discovery devices send out a COMMUNITY_ADVERTISEMENT message every ADVERTISE_INTERVAL time units, which is transmitted sequentially on all available radio frequencies F_n. Discovery nodes from neighboring communities use the information from

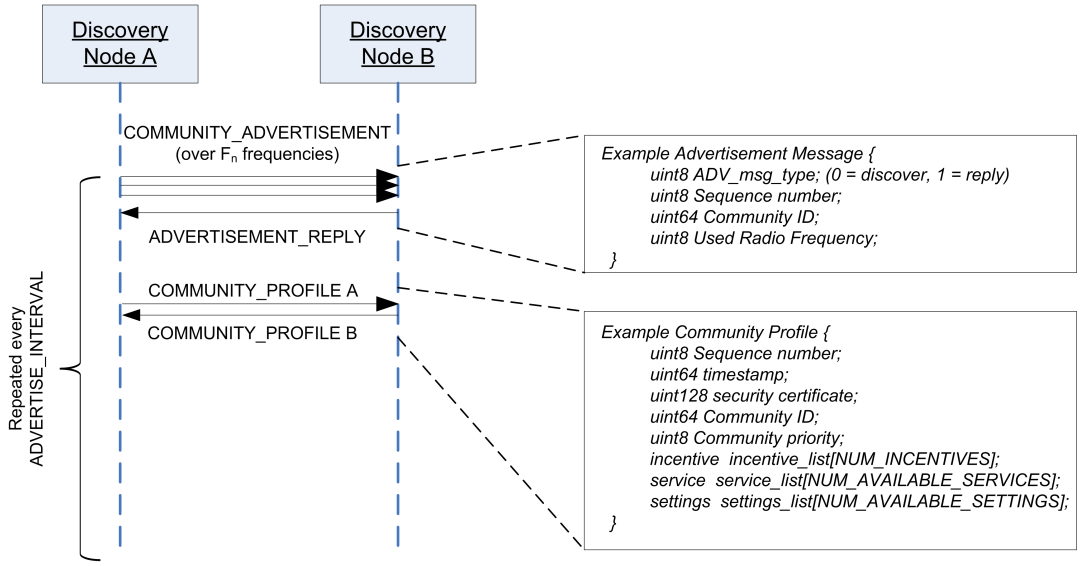

Figure 5.6: Sequence diagram of the community discovery process from the proof-of-concept demonstrator.

the COMMUNITY_ADVERTISEMENT to send an ADVERTISEMENT_REPLY using the radio frequency used by the original discovery node[1]. Once this connection is established, the COMMUNITY_PROFILE is exchanged between the discovery nodes.

The total number of packets per time unit (PPT) required for the discovery process in community C can be calculated as follows.

$$PPT_{(Discovery;\ Community\ C)} =$$

$$(F_n * DN_C + \sum_{j=1}^{DN_C} \sum_{i=1;i\neq C}^{C_j} DN_{i,j}) * \frac{1}{ADV_C} \tag{5.6}$$

$$+ \sum_{j=1}^{DN_C} * \sum_{i=j;i\neq C}^{C_j} \frac{2 * DN_{i,j}}{ADV_i} \tag{5.7}$$

With:

$$\begin{cases} DN_C = & \text{The \# of discovery nodes in community C} \\ DN_{i,j} = & \text{The \# of discovery nodes in community i} \\ & \text{that are within reach of discovery node j.} \\ C_j = & \text{The \# of communities that are in reach of} \\ & \text{discovery node j.} \\ ADV_C = & \text{The advertise interval of community C.} \end{cases}$$

The first part of Formula 5.6 calculates the total number of advertisement messages that are transmitted by the discovery nodes of community C per time unit ($F_n * \frac{DN_C}{ADV_C}$). The second part of formula 5.6 calculates the number of community profiles that are sent in response to advertisement replies from neighboring discovery nodes. Part 5.7 of the formula expresses that an advertisement reply and a community profile is transmitted in response to each community advertisement that is received from a neighboring community.

In the proof-of-concept implementation, the ADVERTISE_INTERVAL is set to 5 minutes, and F_n equals 16 (all available IEEE 802.15.4 channels). Using these settings, the discovery overhead of our proof-of-concept network is limited to 3.8 packets per minute. For dynamic networks, the advertisement interval should be set to a low value, whereas energy constrained networks or networks that interact rarely would prefer a much higher value to reduce the energy consumption. In large networks, the overhead of the discovery process can further be reduced by intelligently choosing the location of the discovery devices; or by implementing more intelligent discovery algorithms[2].

[1] A short delay is introduced to give the original node time to finish transmitting its advertisements over all frequencies.

[2] For example: a discovery node can choose to send only a single reply message if it is in reach of multiple discovery nodes from the same community.

5.4.3 Overhead of the negotiation process

The negotiation process is illustrated in Figure 5.7. Whenever the discovery node receives a new or updated community profile, this profile is forwarded to the negotiation entity (the sink) of each community. The linear program from Section 5.3 is implemented to automatically calculate the optimal set of network services. The negotiation entity will only calculate new service proposals if a new profile is detected, or if the profile information of one of the communities has changed.

After the calculation of the optimal services, service negotiation messages (in the form of SERVICE_PROPOSALS and SERVICE_REPLIES) are exchanged between the negotiation entities to reach a common decision. Each service negotiation messages includes a transaction ID to keep track of the negotiation process. The type of the negotiation message (proposal, reply, etc.) is indicated by the negotiation code of the negotiation message (see Figure 5.7). Each message type can also include one or more options. For example, a SERVICE_REPLY message can have one of the following options: PROPOSAL_ACCEPTED, PROPOSAL_REFUSED, PROPOSAL_COUNTEROFFER. Finally, the list with services and settings may be omitted from the service negotiation message if they remain unchanged from the previous message with the same transaction ID.

Finally, once both communities reach an agreement on which network services should be activated, a SERVICE_DISTRIBUTION message is broadcast by the negotiation entities of each community (see Figure 5.7). These messages inform the individual devices of each community of the selected set of services and settings. If settings can not be activated, a device may respond with a SERVICE_UNAVAILABLE message. If this occurs, the conflicts must be solved (by installing the missing service, by removing the device from the community, or by renegotiation) before the SERVICE_ACTIVATION message can be broadcast. Once every intermediate device has received the activation message the network settings will be changed.

The overhead for the negotiation process (in number of packets) can be calculated as follows.

$$Packets_{(Negotiation;\ Community\ C)} =$$

$$5 * \sum_{j=1}^{DN_C} \sum_{i=1;i\neq C}^{C_j} D_j \tag{5.8}$$

$$+ 2 * (Nodes_C - 1) \tag{5.9}$$

With:

$$\begin{cases} D_j = & \text{The distance (\# of hops) from} \\ & \text{discovery node j to the negotiation entity.} \\ Nodes_C = & \text{The \# of nodes in community C} \end{cases}$$

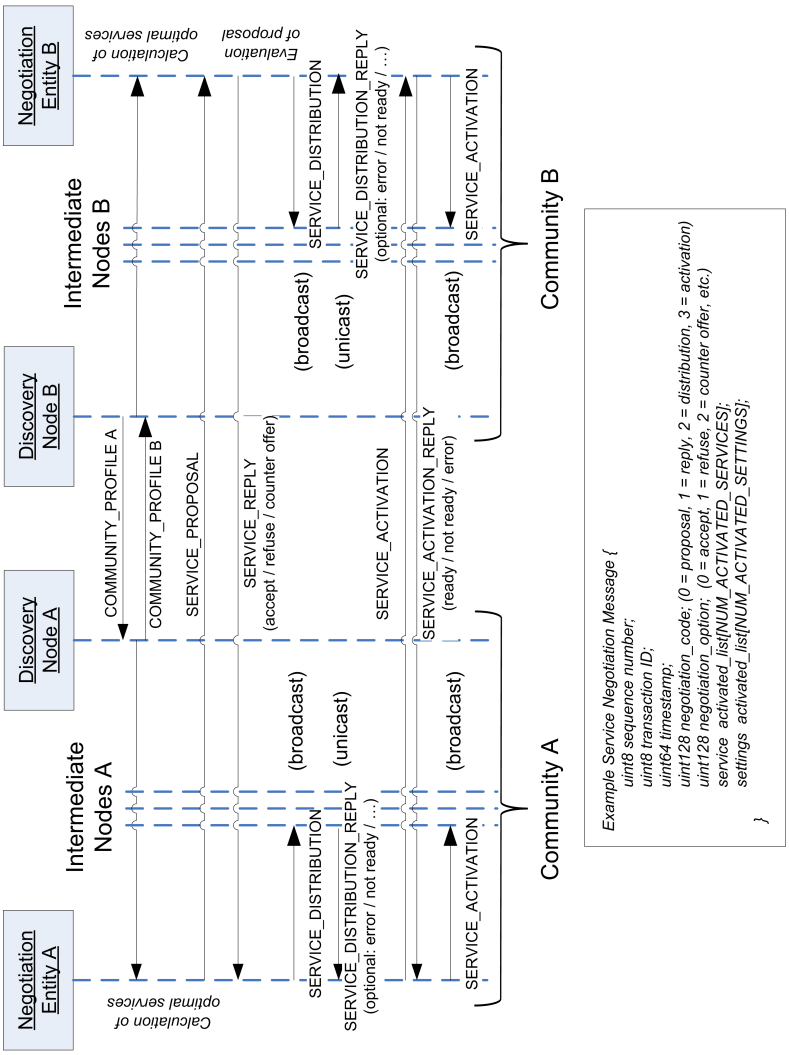

Figure 5.7: Sequence diagram of the negotiation process from the proof-of-concept demonstrator.

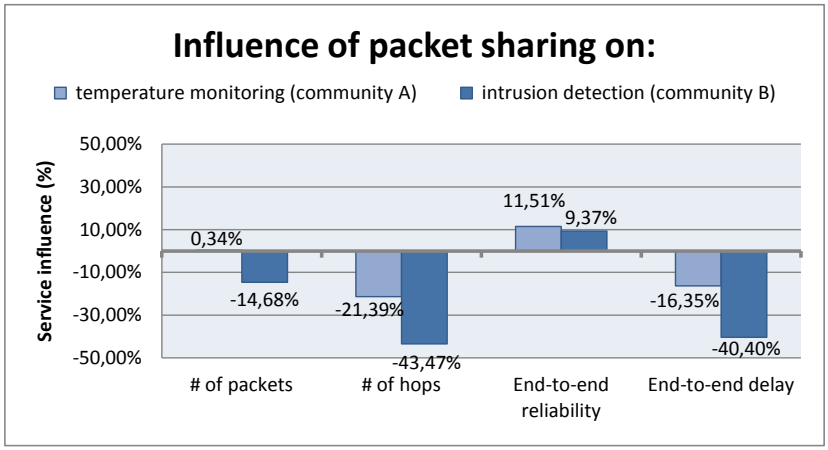

Figure 5.8: Influence of activating packet sharing on the network performance.

It is assumed that each discovery node is capable of filtering duplicate profiles (for example, if the discovery device is in range of multiple discovery devices of a neighboring community). Formula 5.8 calculates the number of packet transmissions between the negotiation entity and the discovery nodes of the community. Part 5.9 of the formula adds the service distribution and service activation overhead.

In the proof-of-concept, $D_a = 2$ and $Nodes_a = 9$. Using the above formulas, the total overhead of a single negotiation round is 26 packets for each community. Re-negotiation occurs whenever (i) a new neighboring community is discovered; (ii) a neighboring community is no longer available for cooperation; or (iii) the profile of one of the participating communities changes. To account for failing nodes, the proof-of-concept implementation performs a new negotiation process once every hour. As such, the negotiation overhead corresponds to ± 2.9 packets per hour for each node. In most scenarios, this overhead is negligible when compared to the amount of traffic generated by the application(s).

5.4.4 Evaluation of the available network services

The sensor devices from the proof-of-concept are capable of activating two types of network services. (i) Activating '*packet sharing*' enables devices to interpret incoming packets from different communities. In addition, packets can be transmitted to any of the available sinks. (ii) The second network service is the aggregation services described in [11], which reduces the number of packet transmissions by aggregating information exchanges from multiple network layers into a single packet.

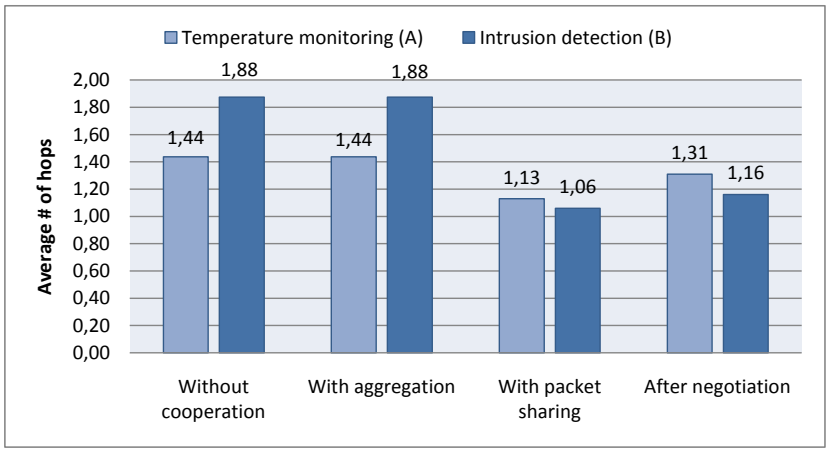

Figure 5.9: The average number of hops in communities A and B in the following situations: without cooperation between the communities, with packet sharing active in both communities, with aggregation active in both communities, and finally with optimal service selection after negotiation (packet sharing active in A and B, aggregation active in A)

For the negotiation process, information about how these network services influence the incentives is required. Figure 5.8 describes the influence of activating *packet sharing* in both communities. As expected, packet sharing reduces the average number of packet transmissions because (i) nodes can select more optimal paths and (ii) two sinks are now available, thereby reducing the average distance to the sink. However, whereas community B indeed shows a reduction in the number of packet transmissions, community A instead shows a small increase in the number of packet transmissions, even though the two communities have a very similar topology. This can be explained by the better link quality of community A, which causes traffic from the intrusion detection community B to be off-loaded to the temperature monitoring community A. The average number of hops, the average end-to-end reliability and the average end-to-end delay are improved in both networks, as expected. The main conclusion from Figure 5.8 is that, while it is often easy to predict the general influence of a network service on the network performance (e.g: positive, negative, ...), it is sometimes difficult to calculate the exact influence rates for an actual network deployment.

Finally, the performance of the aggregation service is similar to the results from Chapter 3 and will be discussed in more detail in the next section.

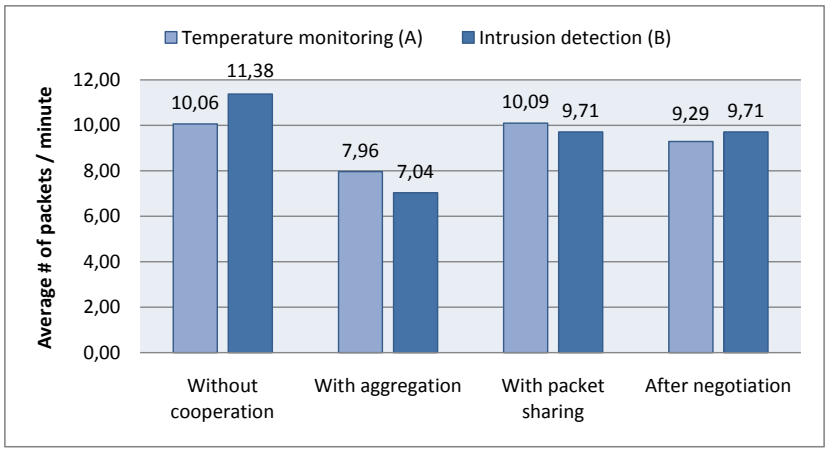

Figure 5.10: The average number of packet transmissions in communities A and B in the following situations: without cooperation between the communities, with packet sharing active in both communities, with aggregation active in both communities, and finally with optimal service selection after negotiation (packet sharing active in A and B, aggregation active in A)

5.4.5 Performance of the proof-of-concept

To calculate the optimal set of services in the communities, the linear program from Section 5.3.3 was implemented. The temperature monitoring community A had the following incentives: $IW_{lifetime;a} = 0.7$ and $IW_{reliability;a} = 0.3$. The incentive weights of community B were $IW_{delay;b} = 0.5$ and $IW_{reliability;b} = 0.5$. Based on the influence rates from the previous section, the negotiation entity concluded that the best network performance is obtained by: (i) activating the packet sharing service and the aggregation service in the temperature monitoring community, (ii) activating packet sharing in the intrusion detection community, and (iii) using the same radio frequency for both communities.

The network performance for both communities was evaluated in the following situations: without cooperation between the communities, with packet sharing active in both communities, with aggregation active in both communities, and finally with optimal service selection after negotiation (packet sharing active in A and B, aggregation active in B).

Figure 5.9 shows the average hops that are required to reach the destination. Since aggregation does not result in different communication paths, the aggregation service has no influence on the average number of hops from the nodes to the sink. The number of hops is important however, as the number of hops directly influences the number of packet transmissions, the reliability and the delay of information exchanges, which in turn influence the incentives of the communities

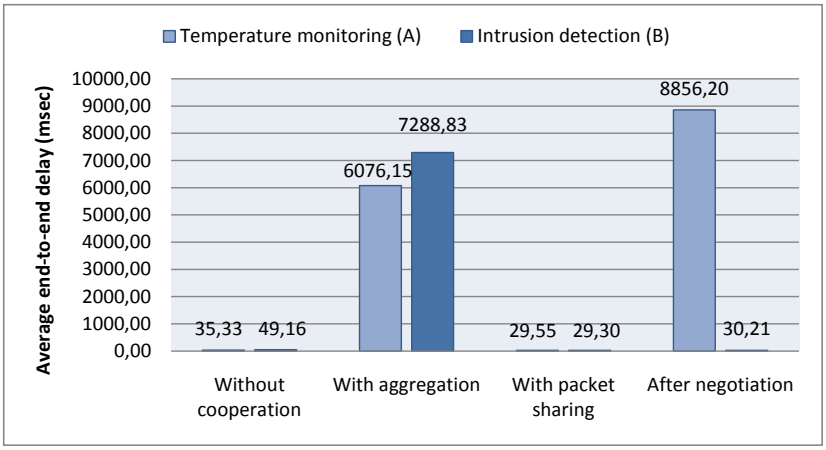

Figure 5.11: The average delay in communities A and B in the following situations: without cooperation between the communities, with packet sharing active in both communities, with aggregation active in both communities, and finally with optimal service selection after negotiation (packet sharing active in A and B, aggregation active in A)

(see below).

The average number of packet transmissions is shown in Figure 5.10. Activating aggregation significantly reduces the number of packet transmissions (by 20-40%). To a lesser amount, activating packet sharing also results in a lower number of packet transmissions since less hops need to be traversed. However, it is worth noting the following. Even though both services lower the number of packet transmissions, the number of packet transmissions after negotiation (when both services are active at the same time in community A) is *higher* than when aggregation is the only active service. The reason is as follows: since less intermediate hops are used, the opportunities to aggregate information becomes more limited. As such, in situations where the only incentive is 'obtaining a high network lifetime', aggregation should not be activated together with packet sharing.

The average end-to-end delay is shown in Figure 5.11. The aggregation service temporarily stores information in buffers in order to aggregate multiple information exchanges. As a result, the delay increases significantly up to a (preconfigured) value of maximum 10 seconds. For this reason, aggregation is not activated in community B which requires low delay incentives. It is interesting that the delay caused by aggregation is larger when packet sharing is also activated, since less aggregation opportunities means that the information is stored longer before aggregated packets can be transmitted.

Finally, Figure 5.12 shows the average end-to-end reliability. Activating the aggregation service decreases the reliability by about 3%, since a single lost packet

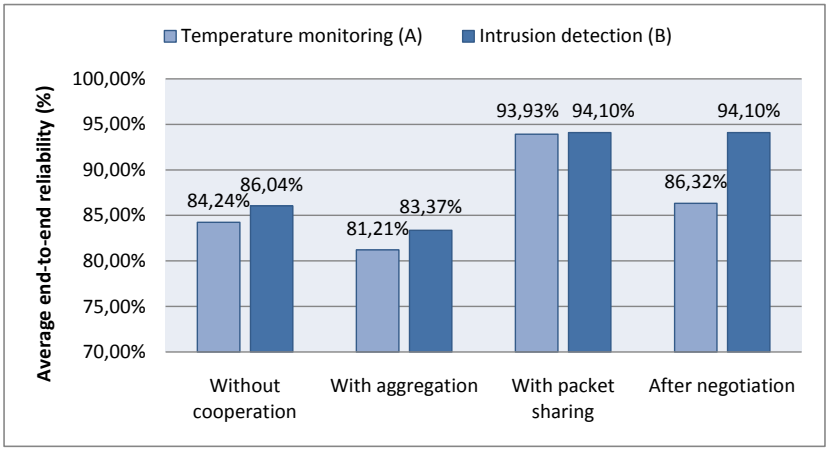

Figure 5.12: The average reliability in communities A and B in the following situations: without cooperation between the communities, with packet sharing active in both communities, with aggregation active in both communities, and finally with optimal service selection after negotiation (packet sharing active in A and B, aggregation active in A)

can now result in the loss of multiple information exchanges. In contrast, activating packet sharing increases the reliability by almost 10% since less intermediate packet transmissions are required. The drop in reliability that results from activating the aggregation service is offset in community A by activating the packet sharing service in both communities after negotiation (see Figure 5.12).

To summarize, the performance of both communities after negotiation is as follows. The packet transmissions and reliability of community A improve by respectively 7.7% and 2.4%. These improvements correspond closely to the requested distribution of the incentive weight factors from community A (that is, improving the network lifetime by twice as much as the reliability). The delay and reliability of community B improve by respectively 14.5% and 11.7%. This distribution matches closely the requested distribution of the incentive weight factors from community B (that is, equally improving the network lifetime and the reliability). As such, even when only a limited amount of network services are available, comunities can improve the performance of their incentives by negotiating and cooperating with each other.

5.4.6 Conclusion of the proof-of-concept

The lessons learned from this proof-of-concept are the following. The influence of network services can vary strongly, depending on the network topology and the used communication technologies [16]. In addition, the influence of network

services also varies over time, especially in wireless environments [17]. Finally, networks services can behave differently when other network services are activated at the same time. These facts illustrate that accurate monitoring and policy enforcement solutions are needed to efficiently support dynamic network negotiation.

The implemented demonstrator serves to (i) demonstrate the feasibility of the discovery and negotiation strategies and to (ii) experimentally verify and measure the benefits of incentive driven cooperation. As shown in this section, incentive driven networking can result in better network performance for all participating devices. The gain in performance depends largely on the network topology, the incentives of the devices and the available network services. As long as the influence rates give a correct indication of the influence of the network services, incentive driven networking will always result in a network performance that is at least as good as having different independent networks.

5.5 Related work

This section gives an overview of related network cooperation approaches that are designed for a closer collaboration between different wireless networks and discusses the differences with the presented approach.

Network planning tools aim to optimize network criteria such as coverage or throughput by calculating the optimal placement and transmission power of devices. Network planning is very efficient in static and predictable network deployments [18, 19]. However, network planning solutions can not be used in networks that (i) dynamically change network topology (such as ad-hoc networks), (ii) have network requirements that change over time, or (iii) in mobile environments, such as when portable devices such as PDAs, body-area-networks and laptops are frequently moved around. Even though network planning solutions are limited to static networks, planning tools can be used in combination with incentive driven network methodologies. For example, existing planning tools can be used to estimate the influence of network services on the incentives, which can be used as input for the negotiation phase. Figure 5.13 illustrates several of these related work approaches.

The use of a *cognitive radio* [20, 21] enables devices to autonomously reconfigure their transmission parameters based on the environment in which they operate. This allows the devices to reuse unused licensed spectrum without interfering with licensed users or to support an always best connected (ABC) paradigm [22]. When parameters of the higher network layers are optimized based on changes in the network environment, the term *cognitive networking* [23] is used. A cognitive network is capable of perceiving current network conditions and use this information to plan, learn, and act according to end-to-end goals [24]. Both cognitive

approaches are focused on the optimization of a single protocol layer or a single device and do not usually involve negotiation or cooperation mechanics.

Whereas cognitive networking is designed for parameter optimization, in *cooperative networks* multiple devices work together towards reaching specific goals. For example, in [25] two MAC protocols are presented that use a relay node to store the packets that failed transmissions in previous time slots and attempts to retransmit them in an empty time slot. Depending on the network layer that is optimized, different approaches are possible. However, barring some exceptions such as [26], most cooperation approaches have mainly considered one layer at a time.

Opportunistic or delay-tolerant networking [27] can occur when part of the infrastructure is not fixed but exists of mobile devices or exists in an environment in which devices often appear and disappear. Data exchanges can take place using the connection opportunities that arise due to impromptu encounters with other devices: nodes can forward data from the source to the destination by using connections with temporary neighbors. Opportunistic Networks enable users to communicate in disconnected environments, in which islands of connected devices appear, disappear, and reconfigure dynamically. Opportunistic networking can be regarded as a special form of cooperative networking.

Finally, *service oriented architectures* support automatic discovery of web services on newly discovered devices. Standards such as OWL-S [28] facilitate the automation of web service tasks including automated web service discovery, execution, interoperation, semantic descriptions, composition and execution monitoring. Unfortunately, these approaches are not designed to optimize networks at the lower network levels. However, incentive driven networking shares several concepts with service-oriented architectures (SOAs) [29].

- Service composability: network services can be combined to reach a specific goal (i.e: optimize the incentives of the communities).
- Standardized service contract: network services expose a standardized interface that is used to activate and configure them.
- Service loose coupling: network services can be activated independently and do not have any dependencies between each other.
- Service abstraction: network services hide their implementation logic, they only describe how they influence the incentives.
- Service reusability: network services can be reused for several topologies and communication technologies.
- Similar to SOAs the service broker from SOAs, a negotiation entity (see Section 5.3.3) decides which network services should be activated/deployed on which devices.

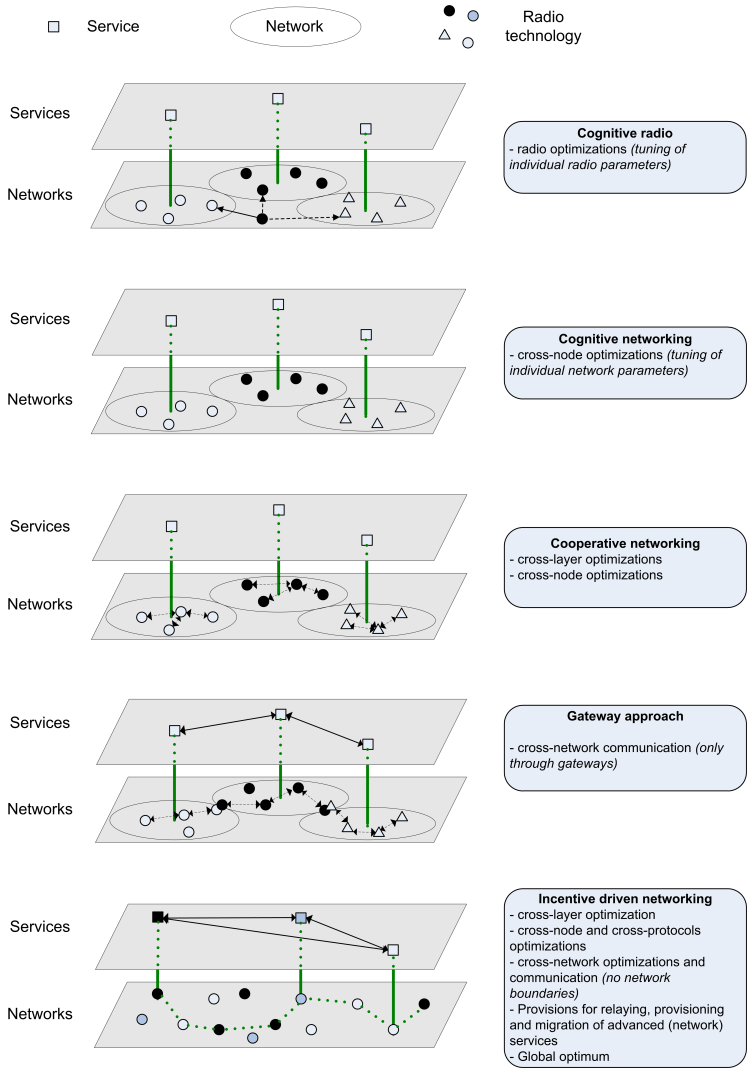

Figure 5.13: Comparison of incentive driven networking with related work approaches

As such, the incentive driven approach could be extended to also include negotiation about high-level services.

In conclusion, even though many network optimization techniques exist, they typically have one of the following disadvantages: (i) optimizations must typically be included at the design phase and thus do not take into account dynamically changing environments, (ii) existing optimizations are often limited to optimizations in a single network layer and (iii) most solutions only allow interaction between different independent networks through translation gateways at fixed locations. In contrast, our approach (i) takes into account the incentives of each individual device, (ii) is not limited to a single network layer, (iii) is designed to cope with heterogeneous devices and (iv) can cross network boundaries that are traditionally fixed. For optimal network coexistence, our approach can be combined with some of the mentioned techniques that are complementary to ours.

5.6 Commercial opportunities

Finally, it is worth noting that incentive driven networking does not aim to replace all traditional networking solutions. Instead, it offers a complimentary approach suited for application domains that can not efficiently be solved using traditional network solutions. More specifically, applications that exhibit one or more of the following characteristics profit most from incentive driven networking.

1 *Multi-party applications* often benefit from negotiation. When devices or parties with different network requirements are co-located, the network performance of the individual parties can increase by making opportunistic use of each others resources.

2 *Dynamic applications*. That is, applications that deploy a variable number of devices, are set-up in an ad-hoc fashion without any planning, or that exhibit a variation in the number of users, traffic flows and traffic requirements.

3 *Heterogeneous networks*. Especially, multiple networks that are co-located, heterogeneous networks that consist of devices with different hardware characteristics and capabilities, and heterogeneous services with different incentives.

4 Applications that engage in, or benefit from, *multi-hop* behavior such as mesh networks, wireless sensor networks or ad-hoc networks.

Table 5.3 describes in more detail several example use cases that exhibit several of these characteristics. Based on this (non-exhaustive) overview, it is clear that incentive driven networking can be used in a wide range of applications.

Type of interaction	Example use cases	Characteristics
Ad-hoc relay between persons or objects	*Emergency services*: optimize the coverage of emergency networks (such as TETRA) by using collaboration (shared routing, power management, auto channel selection, etc.) between the ad-hoc networks from different emergency services present in a disaster area. (e.g. the fire brigade, the police, etc.), as well as by making opportunistic reuse of available deployed infrastructured networks such as corporate and consumer Wi-Fi access points.	*Multi-party*: yes (forced co-location) *Dynamic*: yes (# nodes, # users, ad-hoc) *Heterogeneous*: yes (multi-network, multi-technology, multi-service) *Multi-hop*: yes (ad-hoc)
	Traffic jam: improve network accessibility in densely populated situation such as traffic jam by having individuals connecting ad-hoc with each other in order to reach internet or mobile phone network.	*Multi-party*: yes (forced co-location) *Dynamic*: yes (# nodes, # users, ad-hoc) *Heterogeneous*: yes (multi-network) *Multi-hop*: yes (to circumvent crowded areas)
Coexistence of sensor networks	*Industrial environments*: allow easy configuration and growth of multiple sensor networks used to control machinery in a plant environment by dynamically optimizing radio and routing capabilities across these networks.	*Multi-party*: no *Dynamic*: yes (# nodes, # users) *Heterogeneous*: yes (multi-network, multi-service) *Multi-hop*: yes
Opportunistically offload of traffic	*Telecom offloading*: data and voice traffic from mobile network can be offloaded to the Wi-Fi Network when the user is in the range of a Wi-Fi access point.	*Multi-party*: variable *Dynamic*: yes (# users, # traffic) *Heterogeneous*: yes (multi-network, multi-technology) *Multi-hop*: no

	BAN monitoring: instead of mobile networks, available Wi-Fi networks can be used to backhaul medical information from body area networks (BAN) in order to reduce the transmission cost.	*Multi-party*: yes (opportunistic) *Dynamic*: yes (# traffic) *Heterogeneous*: yes (multi-network, multi-technology) *Multi-hop*: no
Sharing of information and capabilities between vehicles	*Vehicle to vehicle*: allow the sharing of sensed information (e.g. ice detection, break detection, warning of accident) between vehicles in the near environment. Guarantee the reliability of transmissions between road equipment and vehicles by dynamically selecting the best radio configuration between the different vehicles and infrastructures.	*Multi-party*: yes (opportunistic, forced co-location) *Dynamic*: yes (# users, # traffic) *Heterogeneous*: yes (multi-service) *Multi-hop*: yes
Smart dynamic radio planning across services and buildings	*Home environment*: improve coexistence of the different wireless networks present in a home environment (alarm system, Wi-Fi, DECT, home automation, ...) by dynamically optimizing the radio settings. Allow collaboration and improve coexistence of the different wireless networks across flats or offices in the same building.	*Multi-party*: yes (forced co-location) *Dynamic*: yes (# users, # traffic) *Heterogeneous*: yes (multi-network, multi-technology, multi-service) *Multi-hop*: no
Temporary installation of multi-service networks	*Fairs and festivals*: allow easy setup of ad-hoc festival telecom infrastructure by minimizing interference and optimizing usage of resources and coverage. Dynamically optimization of Wi-Fi parameters (channel, power).	*Multi-party*: yes (forced co-location) *Dynamic*: yes (# users, # traffic) *Heterogeneous*: yes (multi-network, multi-technology, multi-service) *Multi-hop*: yes
	Construction site: allow connection sharing across construction site infrastructure used for security (typically meshed wireless network with sensors and cameras) and communication devices	*Multi-party*: yes (forced co-location) *Dynamic*: yes (# nodes, # users) *Heterogeneous*: yes (multi-network, multi-technology, multi-service) *Multi-hop*: yes

Table 5.3: Example use cases that can benefit from incentive driven networking.

5.7 Research opportunities

Finally, it is clear that each step of the proposed methodology can be custom-tailored towards a specific application domain. These custom-tailored implementations can lead to interesting research opportunities or patentable network algorithms:

- Network monitoring algorithms can be developed that are capable of estimating the (real-time) influence of cross-network services on the network performance.
- New negotiation approaches can be developed based on game-theory or machine learning.
- Heterogeneous network discovery can include methods for deducing and translating the network settings of neighboring communities, such as the type of MAC protocol, the structure of the supported packet types, the used routing protocols and the used communication settings.
- New dynamic addressing schemes can be designed that cope with dynamically created communities.

5.8 Conclusion

This chapter introduced incentive driven networking: a cross-layer, cross-network networking approach that supports cooperation between heterogeneous networked devices. Incentive driven networking aims to (i) simplify the configuration and setup of networks for the end-users and (ii) increase the network performance of co-located devices. Rather than using manually configured (and time-consuming) fixed network boundaries, network creation and negotiation is based on the concept of 'network incentives' or 'device goals'. The methodology comprises the following steps:

- Devices cluster together with other devices that have similar incentives, thus forming communities of like-minded, interconnected objects.
- Different communities broadcast their existence to each other.
- The communities exchange profiles which describe their available network services, their incentives and their network settings.

- A negotiation entity determines the optimal set of network services so that each participating community benefits from cooperation.
- Finally, the selected services are activated, so that the incentives of each participating community are improved.

Depending on the device incentives, incentive driven networking results in better use of the scarce spectrum, better scalability, more efficient energy consumption, lower radio emissions, sharing of service capabilities (such as GPS, processing power or internet connectivity) and/or better QoS guarantees.

The proposed methodology was validated in the form of a proof-of-concept implementation. Experimental results showed that the main requirement for successful negotiation is the accurate estimation of the influence of network services on the incentives of co-located devices. However, as long as the influence rates give a correct (broad) indication of the influence of the network services, incentive driven networking will always result in a network performance that is at least as good as having different independent networks. Indeed, after cooperation, both networks in the proof-of-concept benefited from cooperation in the form of having better reliability, delay and/or longer network lifetime.

It is clear that the methodology is not limited to the example proof-of-concept scenario. In general, networks that exhibit one or more of the following characteristics are likely to profit from incentive driven networking: (i) multi-user networks, (ii) dynamic networks, (iii) heterogeneous networks, and/or (iv) multi-hop networks. To prove this fact, the chapter gave an overview of a large number of potential marketable applications that can be implemented using our incentive driven cooperation methodology.

To conclude, the incentive driven networking paradigm is applicable to a wide range of applications domains and can ultimately lead to an improved coexistence of co-located networked devices.

References

[1] *The Internet of Things*. ITU Internet Reports, 2005.

[2] S. Buljore, H. Harada, S. Filin, P. Houze, K. Tsagkaris, O. Holland, K. Nolte, T. Farnham, and V. Ivanov. *Architecture and Enablers for Optimized Radio Resource usage in Heterogeneous Wireless Access Networks: The IEEE 1900.4 Working Group*. IEEE Communications Magazine, January 2009.

[3] A. Wheeler. *Commercial Applications of Wireless Sensor Networks Using ZigBee*. Communications Magazine, IEEE, 45(4):70 –77, apr. 2007.

[4] Carlos F. Garca-Hernndez, Pablo H. Ibargngoytia-Gonzlez, Joaqun Garca-Hernndez, and Jess A. Prez-Daz. *Wireless Sensor Networks and Applications: a Survey*. IJCSNS International Journal of Computer Science and Network Security, Vol.7 No.3, March 2007.

[5] N Wakamiya, S Arakawa, and M. Murata. *Self-Organization Based Network Architecture for New Generation Networks*. First International Conference on Emerging Network Intelligence, pages pp.61–68, Oct. 2009.

[6] Tracy Camp, Jeff Boleng, and Vanessa Davies. *A survey of mobility models for ad hoc network research*. Wireless Communications and Mobile Computing, Special Issue: Mobile Ad Hoc Networking Research, Trends and Applications, Vol. 2, issue 5:pp 483–502, August 2002.

[7] Razvan Musaloiu-E. and Andreas Terzis. *Minimising the effect of WiFi interference in 802.15.4 wireless sensor networks*. International Journal on Sensor Networks, Vol. 3:pp. 43–54, December 2008.

[8] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. *Exploiting heterogeneity in sensor networks*. In Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Infocom 2005), March 2005.

[9] *Device description structures*. http://www.w3.org/TR/dd-structures/.

[10] T Yucek and H. Arslan. *A survey of spectrum sensing algorithms for cognitive radio applications*. IEEE Communications Surveys & Tutorials, Vol. 11:pp.116–130, First Quarter 2009.

[11] Eli De Poorter, Stefan Bouckaert, Ingrid Moerman, and Piet Demeester. *Non-intrusive aggregation in wireless sensor networks*. Ad Hoc Networks, 9(3):324 – 340, 2011.

[12] *TMoteSky Datasheet, http://www.snm.ethz.ch/Projects/TmoteSky*.

[13] *The IBBT W-iLab.t wireless sensor testbed. http://ilabt.ibbt.be/.*

[14] L. Tytgat, B. Jooris, P. De Mil, B. Latré, I. Moerman, and P. Demeester. *Demo abstract: WiLab, a real-life wireless sensor testbed with environment emulation*. published in European conference on Wireless Sensor Networks, EWSN adjunct poster proceedings (EWSN), Cork, Ireland, 11-13 February 2009.

[15] *Ad hoc On-Demand Distance Vector (AODV) Routing. Networking Group Request For Comments (RFC): 3561, http://tools.ietf.org/html/rfc3561*, July 2003.

[16] Martin Wirz. *BTnode Application for Automated Link Measurements*. Master's thesis, ETH Zurich, 2007.

[17] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. *Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks*. Wireless Sensor Networks. Technical Report CSD-TR 02-0013, UCLA, February 2002.

[18] Y. Wu, P.A. Chou, Qian Zhang, K. Jain, Wenwu Zhu, and Sun-Yuan Kung. *Network planning in wireless ad hoc networks: a cross-Layer approach*. Selected Areas in Communications, IEEE Journal on, 23(1):136 – 150, 2005.

[19] Y. Wu, P. Chou, Q. Zhang, K. Jain, W. Zhu, and S.-Y. Kung. *Network planning in wireless ad hoc networks: a cross-layer approach*. IEEE Journal on Selected Areas in Communications, Vol. 23:pp. 136 150, 2005.

[20] S. Haykin. *Cognitive radio: brain-empowered wireless communications*. Selected Areas in Communications, IEEE Journal on, 23(2):201 – 220, feb. 2005.

[21] Friedrich K. Jondral. *Software-Defined Radio - Basics and Evolution to Cognitive Radio*. EURASIP Journal on Wireless Communications and Networking, vol. 2005, no. 3:pp. 275283, 2005.

[22] I. Akyildiz, W. Lee, M. Vuran, and S. Mohanty. *NeXt Generation/Dynamic Spectrum Access/Cognitive Radio Wireless Networks: A Survey*. Elsevier Computer Networks, Vol 50:pp. 2127–2159, 2006.

[23] Ryan W. Thomas, Daniel H. Friend, Luiz A. Dasilva, and Allen B. Mackenzie. *Cognitive networks: adaptation and learning to achieve end-to-end performance objectives*. Communications Magazine, IEEE, 44(12):51 –57, dec. 2006.

[24] C. Fortuna and M. Mohorcic. *Trends in the development of communication networks: Cognitive networks*. Computer Networks, 2009.

[25] A. K. Sadek, K.J.R. Liu, and A. Ephremides. *Collaborative Multiple-Access Protocols for Wireless Networks*. ICC 2006.

[26] J. Garcïa-Vidal, M. Guerrero-Zapata, J. Morillo, and D. Fustï. *A Protocol Stack for Cooperative Wireless Networks*. Wireless Systems and Mobility in Next Generation Internet, In Lecture Notes in Computer Science (LNCS), Volume 4396:pp 62–73, 2007.

[27] L. Pelusi, A. Passarella, and M. Conti. *Opportunistic networking: data forwarding in disconnected mobile ad hoc networks*. Communications Magazine, IEEE, vol.44, no.11:pp.134–141, November 2006.

[28] *OWL-S: Semantic Markup standard for Web Services, http://www.w3.org/Submission/OWL-S/.*

[29] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

6 Overall Conclusion

6.1 Summary of the chapters

During the last few years, research about wireless sensor networks (WSNs) has become increasingly popular. Whereas 10 years ago sensor research focused on military applications, current WSN applications are used in a wide-range of commonplace applications. Sensor networks can monitor valuable habitats, they can automate the control of buildings in order to reduce their energy consumption, or they can provide wireless health care for athletes, elderly people and people with chronic diseases. To keep the cost of wireless sensor networks low, sensor devices are typically limited in terms of bandwidth, energy provisions and computational power. Designing network protocols and applications that take into account these constraints is far from trivial. As a result, the development costs of todays sensor networks are often quite high.

Most researchers agree that WSN software should be both simple (to fit in resource-constrained devices) and energy-efficient (to obtain a long network lifetime). However, there is not yet a consensus about which network requirements will be important for next-generation applications. At the moment, WSN network protocols are quite diverse in terms of their support for mobility, adaptivity, energy efficiency, heterogeneity and quality-of-service. Even when a network protocol is available that suits the network requirements of a specific application, there are no guarantees that other network protocols will be compatible. As a result, when faced with a new application domain, WSN network developers are often forced

to design and/or implement a different network protocol for each new deployment. In turn, these custom-designed network protocols are again difficult to reuse since (i) they support the network requirements required by a single application domain and (ii) the protocols can often only be used in combination with specific hardware and/or software. Thus, these custom-design approaches are expensive in terms of development cost.

Due to the resource-constrained nature of wireless sensor networks, it is at this moment impossible to design a one-size-fits-all network protocol that is compact enough to fit in current-day sensor nodes. As an alternative, this dissertation introduced the IDRA architecture. IDRA supports next-generation WSN requirements at an architectural level (i.e.: separately from the routing and MAC protocols). Compared to single-layer protocol optimization, architectural optimization has two major advantages: (i) the development complexity of new network protocols is simplified and (ii) innovative solutions that support next-generation WSN requirements can transparently be combined with existing network algorithms.

First of all, the IDRA architecture fulfills the basic needs of wireless sensor networks: IDRA exhibits a *low memory overhead* and *reduces the energy consumption*. (i) By delegating duplicate network functionality such as packet creation and packet interaction to the architecture, IDRA network protocols are simpler and smaller in terms of memory requirements (up to a factor 10). (ii) In addition, by delegating packet creation to the IDRA architecture, IDRA can combine information exchanges from all network protocols into a single packet. The reduction in packet transmissions increases the network lifetime by 30-50%, depending on the MAC protocol. This reduction in the number of packet transmissions is far greater than the results that are obtained by designing and optimizing only a single network layer.

The above optimizations typically suffice for applications that gather low-priority information from an area. However, in time, next-generation sensor network applications, such as process automation, medical monitoring, tracking of goods and wireless building automation will become increasingly popular. These next-generation applications for WSNs exhibit a more dynamic behavior. For example: health and security applications deploy multiple services on a single network, they require additional reliability guarantees and the sensor nodes can be mobile. To support these next-generation applications, IDRA includes support for cross-protocol interactions, QoS optimizations (packet priorities, dynamic protocol selection), mobility support and heterogeneous network support. In addition, a large number of small but specialized network protocols can be installed on a single device: based on the packet characteristics, IDRA automatically selects the optimal network protocol to process each packet. As such, IDRA supports several next-generation network functionalities at an architectural level, so that they can be combined with both existing and new network protocols.

Furthermore, as stated in the vision of the 'internet of things', networked objects will become increasingly ubiquitous. The performance of sensor networks can be improved by supporting communication between sensor nodes and co-located devices that use different communication technologies. Using 'always best connected' strategies, sensor nodes can connect to the internet using co-located devices and access technologies that best suit the needs of the application (i.e.: cheapest cost, best connection, etc.). To enable these strategies, IDRA allows network protocols to cope with heterogeneous packet types and diverging communication interfaces in a protocol-independent way. To this end, IDRA provides a configuration table that describes which packet structure and which network protocols should be used to communicate with a neighboring device. Based on the information from this table, the IDRA architecture is capable of (i) interpreting different types of incoming packets, (ii) converting packets to different packet types, (iii) selecting the optimal network protocol to process packets and (iv) selecting the outgoing communication interface. In a proof-of-concept implementation, it was shown that these features can be used to utilize Wi-Fi nodes as intermediate hops while routing WSN traffic.

Current homogeneous Wi-Fi and sensor networks are typically separated into different networks. However, as time progresses, the number of co-located network technologies will increase, up to a point where heterogeneous networks are the default situation. In environments with an ever-increasing number of co-located wireless devices, a better spectral efficiency is obtained when devices from different networks and/or with different network requirements are able to cooperate directly with each other, regardless of which network they belong to. To enable this innovative vision, this dissertation introduced an 'incentive-driven' networking approach. Based on their application goals, devices autonomously form 'communities' consisting of devices with similar network preferences. Communities can cooperate with each other by activating and sharing network services, but only if this cooperation is beneficial for all involved devices. Experimental results show that the main requirement for successful negotiation is the accurate estimation of the influence of network services on the incentives of co-located devices. As long as the influence rates give a correct (broad) indication of the influence of the network services, incentive driven networking will always result in a network performance that is at least as good as the network performance of independent networks.

To summarize, wireless sensor networks grew from a relatively unknown and specialized type of networks into a research topic that supports a wide range of innovative next-generation applications. This ever-increasing number of sensor applications makes the sensor network research field a truly fascinating subject. Whilst custom-designed network solutions can offer a highly-optimized performance for a single WSN application domain, these network solutions are expen-

sive to develop and serve only a single purpose. As such, custom-designed network solutions are only suitable for applications with very stringent network requirements. In contrast, whilst not as optimized as custom-designed network solutions, the IDRA architecture is compact, easy-to-use, flexible, well-tested and supports most next-generation sensor requirements. As such, IDRA is a future-proof architecture that is well-suited for dynamic next-generation networks that fulfill multiple purposes or networks whose purpose might change after deployment.

6.2 Outlook and future work

The previous section summarized the findings of my PhD research. This section describes the impact of my PhD research and describes the direction IDRA will take in the future.

Chapter 2 demonstrates that it is possible to support protocol-independent quality-of-service solutions using the IDRA architecture. During the PhD research of my colleague Evy Troubleyn, these IDRA features will be used to design advanced quality-of-service algorithms. The developed algorithms will be capable of (i) intelligently prioritizing and/or dropping packets in the IDRA queue, (ii) influencing the behavior of network protocols at run-time, (iii) dynamically replacing network protocols with protocols that better fit the requested quality-of-service requirements and (iv) supporting QoS-aware information aggregation. The PhD research of Evy Troubleyn will be finalized during 2013.

In addition, Chapter 2 describes the IDRA protocol selector. The protocol selector is responsible for selecting the optimal network protocol to process each packet. Currently, the selection algorithm is rule-based: based on the number of matching packet attributes the optimal protocol is selected and executed. During the PhD research of my colleague Jono Vanhie-Van Gerwen, automated benchmarking protocols, capable of evaluating the network performance at run-time, will be added to the IDRA architecture. The output of these benchmarking protocols can be used to make more intelligent decisions regarding which network protocols should be activated. This approach enables the design of dynamic, self-learning protocol selection algorithms that select optimal network protocols based on environmental influences and application requirements.

To support heterogeneous networks, IDRA includes a configuration table that describes for each neighboring device which network protocol, which packet structures and which communication interface should be used. At the moment, this table is configured manually. However, for more complex environments, it is possible to write intelligent plug-ins that automatically configure the neighbor table based on measured environmental information. To this end, the FP7 CONSERN and the IWT SymbioNets projects will, amongst others, (i) integrate ‘sensing devices’ (in the form of cognitive radio devices) with the IDRA architecture and (ii)

develop distributed sensing algorithms capable of identifying heterogeneous networks and communication technologies. Moreover, efforts are underway to run IDRA on a variety of hardware platforms, including Unix environments. As a result, IDRA can be used to manage a wide variety of communication technologies: besides IEEE 802.15.4 (zigbee) interfaces, also IEEE 802.11 Wi-Fi, ethernet and bluetooth interfaces will be supported.

The incentive driven methodology proposed in Chapter 4 enables spontaneous cooperation between otherwise independent networks. The proposed methodology was shown to be more efficient than the use of independent networks. However, it is worth noting that these methodologies are not designed to suddenly replace existing networking approaches, but can instead be gradually introduced in existing networks. For example, during a transition phase, community discovery and negotiation solutions can be added to existing devices in the form of small downloadable software drivers, or they can be installed as intelligent hardware plug-ins such as USB or ethernet connectors. This way, existing networks can be optimized by deploying new sensing components that can negotiate with each other. A central negotiation server can gather the sensed information, make cooperation decisions and change the characteristics of deployed devices through existing management interfaces. Additional negotiation and cooperation algorithms will be designed and implemented in IDRA in the FP7 CONSERN and IWT SymbioNets projects.

To conclude, the IDRA architecture demonstrates that innovative protocol architectures can be both efficient (in terms of memory requirements, processing power, network lifetime, etc.) and easy to use. We are surrounded by an increasing number of wireless resource-constrained devices that have network requirements similar to those of sensor nodes. Devices such as PDAs and notebooks aim to optimize their network lifetime, throughput and quality-of-service guarantees whilst using as few resources as possible. As such, the lessons learned in sensor networks can be applied to a wide range of research topics in related fields, such as ad-hoc networks, cyber-physical systems, vehicle-to-vehicle communication, delay-tolerant networks, thin client architectures and the future internet.

6.3 Validation of the IDRA architecture

Finally, this section gives an overview of the projects that successfully utilized the IDRA architecture (as of February 2011).

Education:

- 2008-2009: The Bachelor thesis ('Vak Overschrijdend Project'): Design of an Automated Parking Lot.
- 2009-2011: Master thesis - Design of a Modular MAC protocol for Wireless Sensor Networks by Jellen Vermeir.

- 2010-2011: Master thesis Automated Configuration of Networked Sensor Objects by Alberto Ceballes.
- 2010-2011: Wireless Sensor Networks Lab Session, as part of the course Mobile and Broadband Access Networks, given by prof. Ingrid Moerman.

Research projects:

- PhD research of Eli De Poorter (IWT grant, 2007 - 2010), Ghent University.
- PhD research of Evy Troubleyn (IWT grant, 2008 - 2013), Ghent University.
- PhD research of Yann-Aël, Vrije Universiteit Brussel.
- The IBBT DEUS project: Deployment and Easy Use of wireless Services for wireless building automation (2008-2010).
- The FWO acoustic sensor networks project: signal processing and network design for wireless acoustic sensor networks (2008-2012).
- The IBBT ISBO NGWINET project (2009-2013).
- The IWT SBO SymbioNets project (2009 2013)
- ICON MoCo: Monitoring of Containers (2010-2012).
- The FP7 SPITFIRE project (2010-2014).
- The FP7 CONSERN project (2010-2014).

IDRA is available as an open-source architecture at http://idraproject.net.

List of Figures

List of Tables

List of Symbols and Acronyms

Symbols

σ_j	Number of information parameters of protocol j that are generated in the RP interval.
$\Delta \mathrm{T}_j$	Time interval the generation of messages from type j.
AD_j	Maximum acceptable delay for message of type j.
ADV_C	The advertise interval of community C.
C_a	The # of communities that are in reach of discovery node a.
CP_a	Priority of community a.
D_a	The distance (# of hops) from discovery node a to the negotiation entity.
DN_C	The # of discovery nodes in community C.
$DN_{C,a}$	The # of discovery nodes in community C that are within reach of discovery node a.
E_j^l	Latest timeslot when information parameter l from protocol j should be transmitted.
F_n	Number of available radio frequencies.
I	Total number of incentives.
$IW_{i,a}$	The weight factor from incentive i in community a.
K	Number of applications running on a single device.
N	Number of communities capable of incentive driven networking.
$Nodes_C$	The # of nodes in community C.
P_i	Device profile from device i.
$P_{info\_available}$	Probability that information is waiting in the waiting space.
$P_{pckt\_passing}$	Probability that a packet passes through the system to which the information can be added, or a local packet

	is created to encapsulate another information parameter to the same destination.
$P_{pckt\_reduction}$	Probability that a packet transmission is avoided.
$profit_a$	Profit function of community a.
RP	Repetition period, i.e: the number of time units before an initial situation reoccurs.
S	Total number of services.
S_j^l	Timeslot when information parameter l from protocol j is available.
$SA_{s,a}$	Binary variable indicating if service s is active in community a.
$SI_{i,a;s,b}$	Service influence: the percentage by which incentive i from community a is improved when service s is activated in community b.
T_{IDRA}	IDRA time duration overhead to process a single packet.
T_j	Time message type j is generated for the first time
T_{radio}	Time duration overhead of the radio driver before a packet can be transmitted.
$T_{transmission}$	Transmission duration to send a packet.
x_i	Binary variable representing a timeslot in which a packet can be sent.

ACRONYMS

3G	3rd Generation International Mobile Telecommunications
6LoWPAN	IPv6 over LoW Power wireless Area Networks

A

A-MPDU	Aggregated MAC Protocol Data Unit
A-MSDU	Aggregated MAC Service Data Unit
ABC	Always Best Connected
ACK	Acknowledgment
AD	Acceptable Delay

AODV	Ad hoc On-Demand Distance Vector routing protocol
ARP	Address Resolution Protocol

B

BAN	Body Area Network
B-MAC	Berkeley MAC
BPEL	(Web Services) Business Process Execution Language
BVR	Beacon Vector Routing

C

CA	Collision Avoidance
CD	Collision Detection
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CPU	Control Processing Unit
CRC	Cyclic Redundancy Check
CTP	Collection Tree Protocol

D

DC	Duty Cycle
DECT	Digital Enhanced Cordless Telecommunications
DEUS	Deployment and Easy Use of Wireless Services
DS-MAC	Medium access control with a dynamic duty cycle for sensor networks
DSN	Declarative Sensor Network architecture
DYMO	Dynamic MANET On-demand Routing

G

GAF	GeograPhy-inFormed energy conservation for ad hoc routing
GEAR	Geographical and Anergy Aware Routing
GPS	Global Positioning System

H

HVAC	Heating, Ventilation and Air Conditioning system
HYDRO	A Hybrid Routing Protocol for Lossy and Low Power Networks

I

IBBT	Interdisciplinary Institute for Broadband Technology
ID	Identification
IDRA	Information DRiven Architecture
IEEE	Institute of Electrical and Electronics Engineers
iLab.t	IBBT Test Labs
ILP	Integer Linear Programming
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISM	Industrial, Scientific and Medical radio bands

K

kbps	kilobit per second

L

LAN	Local Area Network
LEACH	Low Energy Adaptive Clustering Hierarchy
LPL	Low Power Listening MAC

M

MAC	Medium Access Control
MAGNET	My personal Adaptive Global NET
ML	Modeling Language
MLA	Mac Layer Architecture

N

NLA	Network Layer Architecture
NS-2	Network Similator 2

O

OSI	ISO reference model for Open Systems Interconnection
OWL-S	Web Ontology Language for the Semantic Web

P

PC	Personal Computer
PDA	Personal Digital Assistant
PEGASIS	Power Efficient Gathering in Sensor Information System
PER	Packet Error rate
PhD	Doctor of Philosophy
PHY	Physical Layer
PPT	Packets Per Time
PRR	Packet Reception rate

Q

QoS	Quality of Service

R

RAM	Random-access memory
ROM	Read-only memory
RP	Repetition Period
RSSI	Received Signal Strength Indicator
RTS/CTS	Request to Send/Clear to Send
Rx	Receiver

S

SCP-MAC	Ultra-Low Duty Cycle MAC with Scheduled Channel Polling
SIFS	Short Inter-Frame Spacing
S-MAC	Sensor MAC
SNA	Sensor Network Architecture
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SP	Sensornet Protocol
SYNC	Synchronization

T

TAG	Tiny AGgregation service for ad-hoc sensor networks
TCP	Transport Control Protocol
TDMA	Time Division Multiple Access
TETRA	Terrestrial Trunked Radio

TLV	Type-length-value representation
T-MAC	An Energy Efficient MAC Protocol with adaptive duty cycle
Tx	Transmitter

U

UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus

V

VPN	Virtual Private Network

W

Wi-Fi	Wireless Fidelity
W-iLab.t	Wireless IBBT Test Lab
WLAN	Wireless Local Area Network
VPN	Virtual Private Network
VPAN	Virtual Private Ad hoc Network
WSN	Wireless Sensor Network
WSAN	Wireless Sensor and Actuator Network

X

X-MAC	A Short Preamble MAC Protocol For Duty-Cycled Wireless Networks
XML	Extensible Markup Language