# Autonomous Cooperation in The Internet of Things

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 12 oktober 2015 om 12:30 uur

door **Yunus Durmuş**
Master of Science, Computer Engineering
geboren te Denizli, Turkije.

This dissertation has been approved by the
promotor: Prof. dr. K.G. Langendoen
copromotor: Dr. Ertan Onur

*Composition of the doctoral committee:*

| | |
|---|---|
| Rector Magnificus | voorzitter |
| Prof. dr. K.G. Langendoen | TU Delft, promotor |
| Dr. Ertan Onur | METU, copromotor |

*Independent members:*

| | |
|---|---|
| Prof. dr. ir. Dick H. J. Epema | TU Delft, TU/e |
| Prof. dr. Sonia Heemstra de Groot | TU/e |
| Dr. ir. F.T.H. den Hartog | TNO |
| Prof. dr. ing. Paul Havinga | UTwente |
| Prof. dr. ir. Reginald L. Lagendijk | TU Delft |

ISBN/EAN: 978-94-6259-850-8


Author email: yunus@yanis.co

Cover design by: Yunus Durmuş, Dicle Hasdemir Durmuş, and Ceylan Çölmekçi Öncü.
Printed in the Netherlands by: Ipskamp Drukkers.

This thesis was funded by the Trans-sector Research Academy for complex Networks and Services (TRANS) project.

# Contents

# Summary

The Internet of Things (IoT) represents the concept of cognitive networked devices that measure their environment and act on it intelligently. For instance, health sensors monitor vital human signs and inform their owner; smart meters measure the energy consumption and relay the information in real time to energy providers and consumers; and smart thermostats optimize heating while reducing costs. Though most IoT devices are designed to work alone, collective operation advances their capabilities. In a smart building application, for instance, several devices from temperature and presence sensors to heating and lighting appliances, cooperate to maximize energy efficiency and comfort. From the application perspective, presence sensors feed lighting and heating appliances with information; from the networking perspective, all these sensors and actuators relay each other's traffic for connectivity (if the medium is wireless). Without cooperation context awareness fails and wireless multi-hop networks collapse.

Unfortunately, when the altruistic act of cooperation is costly, devices become selfish. For a battery-powered device, forwarding a neighbor's packet increases its energy consumption and consequently, decreases its lifetime. Therefore, that device does not cooperate and refrains from forwarding foreign packets. When all nodes in a wireless network follow the same reasoning, none of the packets are relayed, and consequently the network gets disconnected. In this thesis, first, we investigate the mechanisms and incentives for cooperation and reveal that social relations such as family and friendship are crucial. Then, we automate cooperation mechanism for devices based on social relations.

Advancing "smart" IoT devices by making them "social" is becoming a hot topic in IoT research. It is argued that social devices can share their data and assist each other without requiring human intervention and consequently, improve their management. But, what is the meaning of a social device? Being a social device does not necessarily mean assisting all others by sharing data and forwarding packets. A social device has its own identity and social profile such that it is aware of its owner. The criterion of assisting others is its owner's preferences, which are embedded in social relations. As we prove in the thesis, consumers desire to know to whom they assist, suggesting that peers should be inside the circle of trusted social relations.

Social relations are crucial for cooperation, now the question is: how can we automate cooperation decisions based on social relations? Without automation, consumers cannot manage all their devices' interactions. The reason is that IoT imposes the challenge of scaling up to billions of devices such that each person will be equipped with tens of devices. Our solution is a decentralized architecture where every device is identified by a URI that points to the social profile of that device. Ownership relations are declared in this social profile. When a resource server (e.g., light bulb, temperature sensor) receives a request from a client device (e.g., smartphone), the resource server crawls the client's and its owner's social profile. If the resource server discovers a social relation that grants access, it responds positively to the client's request.

Unlike centralized approaches, our decentralized proposal protects privacy, provides end-to-end security, and can operate without an Internet connection. The drawback of our approach is the complexity of searching decentralized social profiles especially for indirect relations such as friend-of-a-friend. For unconstrained devices, we limit the search space based on proximity. In an access point (AP) scenario, the AP overhears WiFi beaconing messages from clients to discover their existence. For constrained devices, the whole search operation is delegated to a more resourceful cloud service.

Our solutions for social network integration depend on secured identity information. Unfortunately, highly constrained devices that have less than 20 KBs of memory cannot be protected from identity-related attacks. These constrained devices can neither punish their defector neighbors nor reward only cooperators. They either cooperate always and are exploited by free-riders or defect always and disrupt network traffic. In this thesis, we offer adaptability to these devices via meta-strategies that only require local information. Devices overhear the traffic in their neighborhood and practice the best local strategy (defection or cooperation). We show that even if free-riders change their identities, meta-strategies protect them against exploitation while still promoting cooperation throughout the network.

All in all, in this thesis we make a few stepts towards the goal of autonomous cooperation in IoT; and in particular we show that

- social relations are crucial in cooperation decisions,
- decentralized social-device networks (proposed in this thesis) can automate cooperation and provide secure-by-default IoT systems,
- constrained devices that are vulnerable to identity-change attacks can protect themselves by observing the traffic in their neighborhood.

# Samenvatting

Het Internet of Things (IoT) symboliseert het concept van cognitieve netwerken waarin apparaten hun omgeving waarnemen en daarop slim reageren. Bijvoorbeeld, medische sensoren die vitale lichaamsfuncties monitoren en daarover aan hun drager rapporteren; slimme energiemeters die het verbruik in real-time doorgeven aan gebruikers en energieleveranciers; en slimme thermostaten die het comfort verhogen en de kosten reduceren. Alhoewel de meeste IoT apparaten zelfstandig opereren, kunnen ze door samen te werken hun toepassingsmogelijkheden aanzienlijk verruimen. In intelligente kantoor- of huissystemen, bijvoorbeeld, werken temperatuursensoren, aanwezigheidsmelders, verwarmingselementen en lichtbronnen samen om een optimaal comfort te creëren tegen minimale kosten. Vanuit het toepassingsperspectief gezien sturen aanwezigheidssensoren de verwarming en verlichting aan. Vanuit het netwerkperspectief bezien, werken alle apparaten samen middels het doorsturen van elkaars (draadloze) berichten om zo tot de benodigde informatie-uitwisseling te komen. Zonder zulke samenwerking vervalt de mogelijkheid om context informatie te gebruiken en wordt multi-hop communicatie onmogelijk.

Helaas, wordt het altruïstische model van samenwerking ernstig op de proef gesteld als er hoge kosten mee gemoeid zijn, apparaten gaan dan egoïstisch gedrag vertonen (net als mensen). Mocht een apparaat door een batterij gevoed worden, dan zal deze (veel) eerder leeg raken als er ook berichten voor anderen doorgestuurd moeten worden. Daarom zal er dan niet meegewerkt worden en zullen berichten niet doorgestuurd worden. Als elk apparaat deze afweging maakt dan wordt er geen enkel bericht meer doorgestuurd en valt het hele communicatienetwerk in duigen. In dit proefschrift onderzoeken we allereerst welke mechanismen ten grondslag liggen aan samenwerking, en hoe dit gestimuleerd kan worden. We tonen aan dat sociale verbanden, zoals familie en vriendschappen, cruciaal zijn in deze. Vervolgens gebruiken we deze kennis om ook apparaten autonoom te laten samenwerken.

Het “socialiseren” van intelligente apparaten, om ze beter te laten functioneren, is een trending topic aan het worden in de IoT onderzoeksgemeenschap. Het idee is dat sociale apparaten hun data en informatie delen en elkaar kunnen helpen zonder tussenkomst van de mens, en zo het beheer aanzienlijk vereenvoudigd kan worden. De vraag rijst dan wel “wat is een sociaal apparaat?”. Het betekent niet noodzakelijkerwijs dat een apparaat met elk willekeurig ander apparaat zal moeten

samenwerken. Nee, een apparaat zal een eigen identiteit hebben en weten wie de eigenaar is, om zo op basis van diens voorkeuren en sociaal netwerk te kunnen beslissen of er wel/niet samen gewerkt moet worden. Een belangrijk element, aangetoond in dit proefschrift, is dat gebruikers willen weten met wie ze te doen hebben alvorens tot samenwerking over te gaan. Dit impliceert dat partners (c.q. apparaten) uit vertrouwde sociale kringen dienen te komen.

Nu we weten dat sociale verbanden essentieel zijn voor onderlinge samenwerking rijst de vraag "hoe kunnen we apparaten automatisch laten samenwerken?". Zonder automatisering wordt het praktisch gezien onmogelijk een groot aantal apparaten en hun interacties te hanteren, en dat terwijl in de IoT visie er in de nabije toekomst miljarden apparation zijn, zodat ieder mens binnenkort van tientallen apparaten voorzien zal zijn. Ons voorstel is gedecentraliseerde architectuur waarin elk apparaat voorzien is van een uniek label (URI - Uniform Resource Identifier) dat wijst naar een sociaal profiel op het Web. Dit profiel zal informatie bevatten over de eigenaar(s). Als een dienstverlener, bijv. een lamp of temperatuursensor, een verzoek ontvangt van een client device, zeg een smartphone, dan kan er gezocht worden m.b.v. deze profielen naar een relatie tussen de twee apparaten en vastgesteld worden of de gevraagde actie gerechtigd is of niet.

In tegenstelling tot een gecentraliseerd systeem, kan onze gedecentraliseerde architectuur de privacy en veiligheid waarborgen, en werken zelfs als er (tijdelijk) geen Internet toegang aanwezig is. Een keerzijde is wel dat het zoeken naar sociale verbanden een tijdrovende bezigheid kan zijn, i.h.b. voor indirecte relaties zoals "een vriend van een vriend". Dit nadeel kan ondervangen worden door het zoeken te beperken tot alleen de apparaten (vrienden) in de directe omgeving. In geval van een draadloos access point bijvoorbeeld, kunnen de apparaten in de omgeving eenvoudig geïdentificeerd worden door de WiFi beaconing berichten af te luisteren. Voor heel kleine apparaten met minimale (reken-) capaciteiten, zogeheten constrained devices, kan de zoekactie compleet gedelegeerd worden naar een service op het web.

Een fundamentele pilaar onder onze "gesocialiseerde netwerken" is de aanname dat elk apparaat een vaste, geverifieerde identiteit heeft. Helaas vereist dit cryptografische rekenkracht die niet op constrained devices met minder dan 20 kB aanwezig is. Deze apparaten kunnen hun misbruikers niet bestraffen, noch hun samenwerkingspartners belonen; ofwel ze werken altijd mee wat wordt misbruikt, of ze zien af van elke samenwerking waardoor het netwerk van goedwillende apparaten geschaad wordt. Als laatste bijdrage in dit proefschrift laten we zien dat er toch een uitweg is door de acties in de directe omgeving in ogenschouw te nemen. Als de (meeste) berichten doorgestuurd worden, dan kan men besluiten dit ook te doen, zo niet dan ziet men af van samenwerking. We presenteren twee zulke meta strategieën, en laten

zien dat zelfs als misbruikers frequent van identiteit wisselen ze niet de constrained devices kunnen uitbuiten en dat samenwerking door het hele netwerk mogelijk blijft.

Concluderend kunnen we stellen dat het onderzoek beschreven in dit proefschrift enkele wezenlijke stappen gezet heeft om automatische samenwerking door IoT apparaten mogelijk te maken. De belangrijkste constateringen zijn dat

- sociale verbanden beslissend zijn in de overweging om al dan niet samen te werken
- de voorgestelde architectuur van sociale apparaten inderdaad samenwerking laat automatiseren op een veilige manier (secure by default)
- constrained devices zichzelf kunnen beschermen tegen uitbuiting door freeriders door ze niet op hun woord (identiteit) te geloven, maar hun daden (traffic forwarding) te beoordelen

# 1

# Introduction

*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.*

Mark Weiser

INTERNET-connected devices exist for decades, while recently these devices have permeated into our lives and are popularly conceptualized as the Internet of Things (IoT). IoT is composed of physical objects equipped with "mostly" constrained hardware providing some computing and networking support. The very first Internet-connected device was deployed in 1982 at Carnegie Mellon University[1], even before the creation of the world wide web. The device was a coke machine that could be queried for its inventory–you could even locate the coldest coke. Then, it took a decade to name it. Mark Weiser's seminal 1991 paper, "The computer for the 21st century", described the technology as *ubiquitous computing* [97], which has evolved into IoT over time. Weiser envisioned that computation would be pervasive; devices would surround us and operate seamlessly. Up to now, 2015, we have witnessed the proliferation of ubiquitous technologies, from cellular phones to smart phones, from desktop PCs to laptops, and to tablets, from wired sensors to wireless sensors, and to wearables. Presently, we have devices such as remotely-controlled

[1] https://www.cs.cmu.edu/~coke/history_long.txt

1

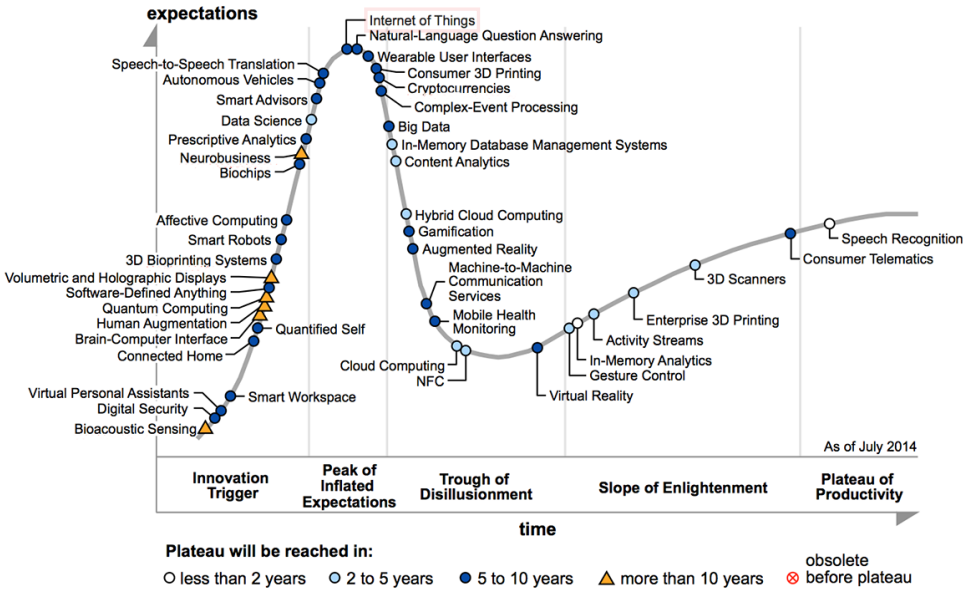



Figure 1.1: Gartner's hype curve as of 2014 [33]. Gartner suggests that IoT is at its peak of inflated expectations.

door locks, intelligent lights and smart thermostats, health status trackers, and wearable location trackers.

Why it took two decades after Weiser's vision to realize IoT? Moreover, Gartner's hype index for 2014 [33], depicted in Figure 1.1, proclaims that IoT is at its peak of inflated expectations. That is, IoT has not reached its productive age, yet. Then, what makes IoT challenging? The most significant reason may be the fact that the challenges are spread over many research fields. Among many, we have determined four main technical challenges to realize the initial products as well as one socio-technical challenge to emphasize human factor. We advocate that the biggest gap is in socio-technical one while the solutions for technical challenges have reached some maturity.

The four main technical challenges and their example solutions are as follows: **(i) Low-cost hardware:** Adding computational power to all devices require affordable and tiny hardware with adequate resources to run a networking stack. Popular do-it-yourself micro-controllers like Arduino and micro-PC Raspberry-PI are the outcomes of related efforts. **(ii) Communication:** As its name suggests, IoT devices should be remotely reachable and connected to Internet. Wireless communication is especially preferred since wiring is mostly inconvenient–even infeasible–such as mobile

communication. Researchers have advanced many communication technologies such as 802.11, 802.15.4, Bluetooth, RFID, NFC, and recently visible light. **(iii) Cognition:** As Weiser also pointed out, inferring the context and surrounding events are the key values of IoT. Data science, originated from artificial intelligence, paved the way for information processing. **(iv) Lifetime:** Energy is a scarce resource in IoT since most devices are battery powered. In a deployment with hundreds of devices whose batteries last in days or weeks, maintenance becomes a frequent and consequently, a demanding task. As a solution, many micro-controllers include low-power modes where both the computation and communication units are duty cycled (i.e., operate periodically).

The socio-technical challenge is **cooperation**, which is defined as "the process of groups of organisms working or acting together for their common/mutual benefit, as opposed to working in competition for selfish benefit" [2]. What makes it really different from other challenges is that cooperation is not only a technical one, but also a social challenge. After all, humans, the owners of devices, are the ones who cooperate, devices only leverage the decisions of their owners.

Cooperation is fundamental in enhancing the potential of IoT. For instance, wireless communication with battery-powered devices cannot reach long distances (km) while with cooperation multi-hopping techniques can increase the coverage. Another example is context inference. With only one temperature sensor a smart thermostat can provide target temperature only in the room that it is installed while other rooms may be cooler, warmer or heated redundantly even if no one is inside. On the other hand, heating costs can be lowered and a better user experience can be achieved if room level and human presence based heating were offered. It is possible when several sensors cooperate such as room level temperature and activity recognition sensors. These examples show that cooperation enhances the capabilities of individual devices and increases the value of applications. Unfortunately, we still observe standalone products in the market, which do not interact with other devices. One challenge has been the interoperability and some third party products, like *The Thing System*[3], are now available to fulfill the gap. However, the main challenge, which remains to be solved, is incentivizing cooperation among devices whose goals, concerns and especially owners differ.

Throughout this thesis, we focus on the cooperation of devices and reveal the incentives of humans for cooperation. *Our main research area is wireless networks, particularly constrained-node networks.* Packet forwarding and network access control are the examples of the aspects that we concentrate on. Our approach is unique

[2] https://en.wikipedia.org/wiki/Cooperation
[3] http://thethingsystem.com/

in the sense that we have carried out social studies to comprehend human perspective in cooperation. After we have identified the key social mechanisms of cooperation, we have proposed technologies that enable devices taking cooperation decisions on behalf of their owners. In Section 1.3, we explain our vision and in Section 1.4 we summarize the challenges that we address. Before that, first we investigate why cooperation has to be incentivized or enforced.

## 1.1. Selfishness and Rationality in Nature

Natural selection is a competition for scarce resources and only the fittest, who acquires the largest portion, survives. To obtain the largest portion an agent should behave selfishly that it should only be concerned by itself regardless of others [61]. Moreover, selfish behavior is also a rational choice that agents act to maximize their utility.

*Game Theory*, which John Nash introduced in 1951 [65], explains the outcome of rationality and selfishness in a famous game, called the *Prisoner's Dilemma*. In the simplest two-player game, two criminals, A and B, have been arrested by the authorities and are being questioned separately without knowing the action of the other. They have two choices either *cooperate* and deny that they have been involved in the crime or *defect* and blame their peer. If both of them *defect* and blame the other they share the outcome, 5 years of jail for each as shown in the lower right corner of the punishment matrix shown in Table 1.1. If only one cooperates, for instance A cooperates, and B defects, then A is sentenced to 10 years, while B is released immediately. On the other hand, for both of them there is a safe resort which is cooperation. If they both cooperate, due to the lack of evidence, they are both charged for only 1 year. Cooperation of both sides thus leads to the best overall outcome, 2 man-years of jail, but Nash proved that the equilibrium of this system is defection, leading to a total of 5+5 man-years of imprisonment. The reason is that cooperate-cooperate state is not stable. Rationality dictates that individually each criminal can do better than cooperating by defecting, reducing the jail time from 1 year to zero. However, in the defect-defect state, none of the criminals alone can opt to a better outcome. As a consequence, rationality leads criminals to defection. We can claim that the outcome is good for society since we capture criminals, whereas the same dilemma keeps countries from taking action against the climate change [28].

Table 1.1: The punishment matrix for the prisoner's dilemma. Punishments for criminals **A** and **B** are given in $P_A, P_B$ format–also black and gray–, respectively.

| | | **A** | |
|---|---|---|---|
| | | Cooperate | Defect |
| **B** | Cooperate | 1, 1 | 0, 10 |
| | Defect | 10, 0 | 5, 5 |

## 1.2. Why Cooperate?

Selfishness is a rational choice that is promoted by natural selection and it contradicts cooperation. Nevertheless, selfishness has not prevented cooperation in nature. Individual cells cooperate to build multicellular organisms, insects build societies such as ant colonies and bee hives, humans build towns, cities, and states. By cooperating, all these agents unite to build something greater than their own. Then the question is: how do we still observe cooperation in nature? Nowak suggests five mechanisms [70]:

- *Kin-Selection*: Individuals cooperate if they are genetic relatives of each other. Note that in this thesis we expand kin-selection to include friends as well to cover all interpersonal relations.
- *Direct Reciprocity*: Cooperation may emerge among unrelated individuals if there is a possibility of future interaction, that is a game such as prisoner's dilemma is played repeatedly. Then, due to expected punishments in the future, defectors switch to cooperation.
- *Indirect Reciprocity*: Punishments do not necessarily arise from the directly interacting individuals. Identity of defectors can be distributed inside a society, then every individual of that society isolates defectors. In this case, defectors have to switch to cooperation to benefit from the indirectly related individuals.
- *Network Reciprocity*: In a network, the members of a cooperative cluster may have higher average fitness than that of a defective cluster. After all, there is no altruism inside a defective cluster. Then, a defector may switch to cooperation when it detects the advantage of cooperative clusters by its local observations. Under certain conditions (See Chapter 5), these cooperative clusters may gradually enlarge and spread out the whole network.
- *Group Selection*: Competition also exists beyond individuals, such that the groups that they establish compete for resources. Group selection is concerned

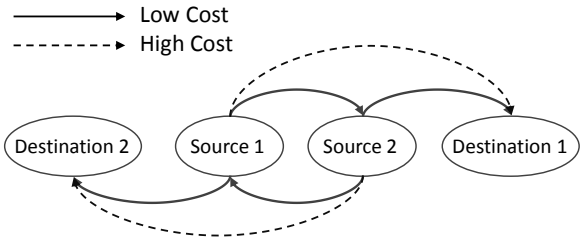



Figure 1.2: Simple relaying scenario where direct link depicted as a dashed arrow may not exist or leads to higher energy consumption. If the nodes cooperate they can use each other as a hop, whereas if they defect they have to use the direct links.

with this multi-level cooperation scheme. Groups with higher benefits, which are cooperator ones, have higher rate of growing. On the other hand, inside a group defectors have more advantage and they may easily invade the whole group.

The prisoner's dilemma of wireless networks is the *Forwarder's Dilemma*, where nodes are selfish and do not relay each other's packets. That is, without cooperation no packet can reach to its destination that is multiple hops away as shown in Figure 1.2. Among the above mechanisms, direct and indirect reciprocity have been investigated extensively to deal with the forwarder's dilemma [7, 11, 30, 59, 62]. If there is a possibility of future interaction, direct reciprocity is employed, otherwise indirect reciprocity is performed. *In this thesis, we advance the research on cooperation by focusing on kin-selection (social relations) (See Chapter 4) and network reciprocity (See Chapter 5).*

The motivation for kin-selection is our survey on consumer preferences where social relations have been the most influential cooperation mechanism (See Chapter 2). In the literature, kin-selection is mostly restricted to only genetically related individuals, whereas we generalize it for all types of familiarities such as family, cowork, and friendship due to the outcome of our survey. Among the five mechanisms of Nowak, we believe that kin-selection fits better for social relations. The reason is that social relations certainly covers kin-selection while they are partially correlated to (in)direct reciprocity. Reciprocity mechanisms are active among strangers who are not in the same social network, while kin-selection necessitates social network relation. Moreover, we should note that other researchers also extend kin-selection beyond biological similarities by involving social bondings such as attachment and nurture kinship [44].

For network reciprocity, the motivation have been its identity oblivious nature. Kin selection and (in)direct reciprocity mechanisms have a common drawback, iden-

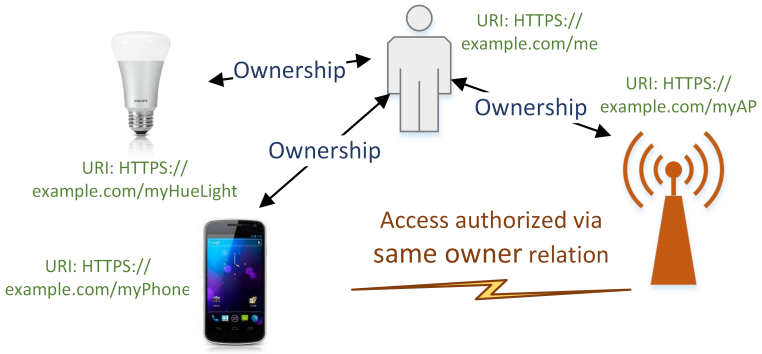



Figure 1.3: Devices share their resources based on their social relation to each other. Devices are identified by URIs.

tities of devices and their owners must be known. Otherwise, punishments and bad reputation can be avoided by changing identity. Unfortunately, in wireless networks spoofing an identity is as easy as changing Media Access Control (MAC) or Internet Protocol (IP) addresses. Encryption techniques are required to ensure the identity. However, the IETF-CoRE (Constrained RESTful Environments) working group aims to provide a secure network "only" for Class-1 and above devices [55]. Many IoT devices, however, are Class-0 devices, which have less than 10KBs of memory, cannot employ security protocols. Due to the lack of true identity information, these computationally constrained devices cannot be sure of which neighbors to punish or reward. Therefore, they are hardwired to either cooperate always and consequently get exploited or defect always and lead to a collapse in the network. With network reciprocity, we have made devices adaptive to their environment without relying on identities.

Before proceeding, we do acknowledge that rationality and selfishness assumptions are also debatable. Many researchers have indicated irrational and non-selfish (i.e., altruistic) behavior in society [14, 18]. Irrational behavior is suggested to happen in case of confusion and error, while selfish behavior is claimed to be influenced by social preferences. For instance, some people are concerned more about inequality and social efficiency [6]. However, in wireless networks research we assume that devices are programmed by professionals who eliminate irrational behavior [19, 88].

1

## 1.3. Vision: Cooperation without Human Intervention

Cooperation is crucial for enhancing the capabilities of IoT devices. For collective operation, each device should discover its neighbors with their capabilities and share its resources, regardless of their owners. Moreover, according to Mark Weiser's vision, in all these interactions humans must be kept out of the loop. For instance, when a consumer buys a product, the only action to install it should be powering on. The product should attach itself to available network interfaces and advertise its existence. When other devices need assistance, such as forwarding a packet in a multi-hop network, the product should make decisions by itself.

Human intervention should also be minimized while securing a device such that default settings should be secure enough. A study on the security of IoT devices by HP in 2014 revealed that 80% of IoT devices fail to require passwords of sufficient complexity and length [45]. Consumers choose weak passwords, reuse existing ones or even keep the defaults because strong passwords are hard to remember and one password is not enough, there are too many IoT devices per person. Instead of blaming consumers for using weak passwords, we need a new perspective on securing IoT devices, where companies take over the responsibility. Without any complicated setup and strong password requirement, even with default settings, a system should be secure enough.

In this thesis, we claim that when a device knows its owner(s) with her social network, firstly, cooperation can be promoted autonomously without any human intervention, and secondly, we can create systems that are secure-by-default. Inspired by the web of things, we assign a uniform resource identifier (URI) to each device. A URI points to the social profile of a device. In these social profiles, the ownership relations are defined. When devices interact, they securely share these identifiers and infer the social relations between their owners, such as same-owner, same-family, direct-friend or friend-of-a-friend. Combined with access control rules, a device can decide with whom to cooperate. For instance, in Figure 1.3, a smartphone joins the home network by using the same-owner relation. The access point does not require a password, instead it checks the social relation. Thus, a user does not have to determine a strong password for the access point, joining the network occurs seamlessly.

## 1.4. Challenges

Sustaining cooperation in IoT requires a social analysis as well as addressing several technical challenges. The social analysis is to comprehend human involvement in cooperation. The technical challenges derive from the need to get humans out of the loop with minimal computational tools, which is due to two main characteristics of IoT: abundance of devices per person and constrained computational resources. In the next subsections we further explain our social analysis and based on its outcome, we detail technical challenges.

**Analysis of cooperation mechanisms.** Among the five mechanisms of cooperation, researchers are not aware of which one is the most significant for a consumer. Most of the literature in wireless networks have concentrated on direct and indirect reciprocity without involving *human perspective*. Our analysis reveals that familiarity has the utmost importance in cooperation. The rest of the challenges are shaped according to this preliminary analysis.

**Social network integration.** Based on the outcome of our analysis: if we can integrate social networks with IoT, devices can leverage the social relations for cooperation amongst each other. This integration is not straightforward since it cannot be accomplished via centralized solutions. The scale of IoT, intermittent Internet communication and privacy demand a decentralized approach.

**Autonomy and security.** IoT increases the number of devices per person that a user cannot handle due to lack of expertise and time. Dealing with periodic maintenance and control is cumbersome and devalues a product. Moreover, users may not secure their devices properly. Therefore, all the operations of cooperation including securing the network and access control should be autonomous. Devices should be aware of their owners' preferences and make decisions on behalf of them.

**Discovery.** We expect that IoT devices will spread into the fabric of daily life such that humans will not even recognize their existence. In that case manual search and discovery for available services will not be feasible. Moreover, current discovery methods are not flexible to accommodate new services and they are not capable of searching for devices of a specific person. Therefore, we need new discovery technologies that are more expressive and able to evolve over time.

**Computational Constraints.** IoT is composed of constrained nodes, for instance a Class-0 device has less than 10 KBs of RAM, which is five orders of magni-

tude lower than an ordinary desktop computer with 1 GB RAM. Therefore, all the solutions to previous challenges must be devised explicitly for constrained devices. If computationally demanding operations cannot be avoided, they must be delegated to unconstrained devices.

## 1.5. Contributions of the Thesis

In the following sections, we present each chapter with the contributions thereof.

**Consumer Perspective on Cooperation — Chapter 2.** In this chapter, we identify the priority of cooperation mechanisms according to consumers. As an example application, we have chosen mobile tethering that enables sharing the cellular data connection of a smartphone with other devices over WiFi, Bluetooth or USB. For instance, roaming subscribers may connect to other tethering capable devices for Internet connectivity. Firstly, we have presented a complete picture of data connection sharing with real world tests on energy and bandwidth consumption. Secondly, *with a conjoint analysis questionnaire, we have investigated why and in what conditions people are willing to share their mobile data connection?* Social familiarity has surfaced as the most significant criterion for cooperation, while security has been flagged as the biggest concern.

- Constantinescu, M.; Durmus, Y.; Onur, E.; Nikou, S.; Reuver, M.; Bouwman, H.; Djurica, M.; Glatz, P.M. *Mobile tethering: overview, perspectives and challenges*, info 2014, Vol. 16 Iss: 3, pp.40 - 53
- Constantinescu, M. M.; Durmus, Y.; Onur, E.; Bouwman, H.; Djurica, M.; Reuver, M. *Cooperative networks: the mobile tethering game*, In Proceedings of the seventh ACM international workshop on Mobility in the evolving internet architecture, MobiArch '12, Pages 41-43, Istanbul, 2012

**Service Knowledge Discovery in Smart Machine Networks — Chapter 3.** In this chapter, we address resource discovery with the aim of improved interoperability and enabling owner-based resource search. Today's operational service discovery protocols carry simple text-based uniform resource identifiers that are not expressive enough. Machines cannot comprehend the meaning of a new service that is not in their knowledge base or *cannot request services based on its owner*. In this chapter, we propose the Smart Discovery Protocol (SDP) that extends the operational service discovery protocols with three main features: *(i)* more expressive semantic representation of the services–including identity, *(ii)* operating in the network layer to deal with diversity, and *(iii)* unifying existing service discovery protocols. SDP represents services with ontologies and further enhances the success

of semantic representations by creating a unified platform that can carry legacy discovery services.

- Durmus, Y. and Onur, E. *Service Knowledge Discovery in Smart Machine Networks*, Wireless Personal Communications, Springer US, Volume 81, Issue 4, pp 1455-1480, April 2015.

**Secure-by-default IoT via a Decentralized Social Device Network — Chapter 4.** In this chapter, we propose an autonomous and decentralized kin-selection mechanism for devices. We introduce decentralized social networks to IoT authentication and authorization. A certificate-based The WebID standard, using X509v3 certificates, is the main building block for credential distribution. Devices have their own social profiles where they define their properties and owners. An authenticator crawls those distributed profiles on the web to discover social relationships. This discovery has a quadratic complexity for indirect friend relationships (i.e., friend of a friend). To decrease the search complexity, we have incorporated context information. The pool of direct friends who bridge the authenticator to indirect friends are sorted and even bounded by their existence in the vicinity. As an example application, we have created a social access point, which captures WiFi probe requests to sense the direct friends. Real-world experiments indicate that the search duration for indirect relationships can be reduced by 82% in a social network of neighbor degree four.

Moreover, we enhance our architecture of decentralized social device networks for constrained devices. The lack of computational resources of constrained IoT devices necessitates an external help for social network search. Therefore, we have employed a delegation-based architecture by modifying existing security standards such as Datagram Transport Layer Security. We have evaluated the delegation-based system on real sensor nodes and presented the computational requirements. A constrained device should have at least 20 KBs of RAM, of which 10% is contributed by our modifications. Moreover, acceleration hardware for public key operations is crucial for decreasing the duration of cryptographic operations.

- Durmus, Y.; Langendoen, K., *WiFi authentication through social networks — A decentralized and context-aware approach*, Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on, pp.532-538, March 2014.
- Durmus, Y.; Erkin, Z.; Onur, E.; Langendoen, K. *Secure-by-default IoT via a Decentralized Social Device Network*, Elsevier Computer Communications SI:IoT Challenges, under Review.

**An Identity-Oblivious Evolutionary Approach to the Forwarder's Dilemma — Chapter 5.** Constrained devices with at least 20 KBs of RAM are capable of

maintaining secure identities. However, there are even more constrained devices, that cannot support cryptographic operations. Without a secured identity, incentives and punishments for (in)direct reciprocity and kin selection cannot be employed. As a consequence, it is easy to trick such highly constrained devices and exploit their resources. In this chapter, we have provided identity-agnostic meta-strategies to discover the locally best strategy in the neighborhood and prevent the exploitation of cooperators. We have modified two meta-strategies from evolutionary game theory, Win-Stay Lose-Shift (WSLS) and Stochastic Imitate Best Strategy (SIBS), for wireless ad hoc networks. Simulations and real-life experiments have proved that both WSLS and SIBS are able to discover the locally best strategy, while they are robust to fake identities. Moreover, we have analyzed and experimented the effects of local decisions on the evolution of the network. While WSLS promotes cooperation up to half of the network, SIBS achieves full network cooperation. To summarize, in the absence of identity information, these meta-strategies protect the nodes against exploitation by free-riders and still favor the spread of cooperation.

- Durmus Y.; Loukas A.; Langendoen K.G. and Onur E. *Sybil-Resistant Meta Strategies for the Forwarder's Dilemma.* In 8th IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems, pp. 90–99, London, UK. SASO 2014
- Durmus Y.; Loukas A.; Langendoen K.G. and Onur E. *An Identity-Oblivious Evolutionary Approach to the Forwarder's Dilemma*, Elsevier Ad Hoc Networks, in Preparation.

Finally, Table 1.2 presents a summary of addressed challenges in each chapter.

Table 1.2: Contributions of the chapters to each challenge

| | Ch. 2 | Ch. 3 | Ch. 4 | Ch. 5 |
|---|---|---|---|---|
| Analysis of cooperation mechanisms | • | | | |
| Social Network Integration | | • | • | |
| Autonomy and Security | | | • | • |
| Discovery | | • | | |
| Computational Constraints | | | • | • |

# 2

# The Consumer Perspective on Cooperation Case Study: Mobile Tethering

In this chapter we analyse the consumer's perspective on cooperation over the case study of mobile tethering. Mobile tethering represents an interesting feature that enables sharing the cellular data connection of a smartphone with other devices over WiFi, Bluetooth or USB. For instance, roaming subscribers may connect to other tethering-capable devices for Internet connectivity. In this way, the coverage of mobile operators enlarges. However, users should cooperate and share their connection with others, maybe even with total strangers. *With a conjoint analysis questionnaire, we investigate why and in what conditions people are willing to share their mobile data connection.* We complete the picture of data connection sharing with real-world tests on energy and bandwidth consumption.

Our results reveal that although energy, bandwidth and security are important technical challenges, users are mainly concerned about social aspects, such as with whom the connection will be shared, rather than monetary issues. In general, mobile tethering is a viable cooperative service, only when users are familiar with the person with whom the data connection is being shared.

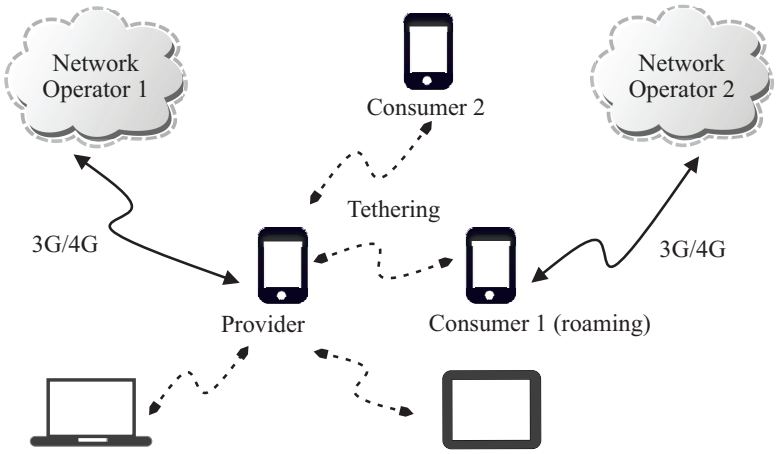



Figure 2.1: Mobile tethering converts smartphones in "mobile hotspots", a gateway for the packets forwarded by other devices through its interfaces.

## 2.1. Introduction

Mobile devices are rapidly becoming part of peoples' daily routines and ways of interacting in society. Connectivity to the Internet is becoming a basic need for consumers to interact with others, receive information and conduct transactions. The concept of mobile tethering enables devices without a mobile broadband connection to access the Internet through nearby devices. The nearby device creates a wireless local area network and relays the packets via its mobile broadband connection. Mobile tethering can be attractive for users without a mobile broadband connection, in regions with underdeveloped or missing infrastructure, in crisis situations or to simply avoid high expenses in the case of roaming. For operators, mobile tethering might reduce network congestion and allow offloading traffic from an overburdened access route. Operators can also apply mobile tethering to transform mobile devices into femtocells, in order to enhance the performance of the network, indoor coverage and capacity.

Despite these advantages, mobile tethering is still largely unknown by the generic public and involves several technological and social challenges. Regarding technological challenges, the person sharing the mobile device for tethering may experience increased power consumption and reduced quality of service due to sharing bandwidth. The person receiving the connectivity faces privacy threats as the person sharing the connectivity has access to the traffic of the user, which are security and trust-related issues. Regarding social challenges, there should be incentives to share a data connection with others, which could be monetary compensations, reputation

mechanisms or other means to ensure that sharing will be reciprocated in the future. How users make decisions regarding mobile tethering may also depend on who they share a data connection with. These technological and social challenges may interact, for instance the severity of privacy, trust and security issues depends on the familiarity of the person with whom a connection is being shared [94].



This chapter analyses the potential of mobile tethering (See Figure 2.1) from both the technological and social perspective. We do so by exploring technological issues through experimenting a mobile tethering application on a test-bed and by evaluating the importance of technological and social issues through a conjoint analysis among consumers. On a theoretical level, we contribute by developing a model for acceptance and use of mobile technologies that take techno-economic as well as social aspects into account. On a practical level, we contribute to understanding the issues regarding mobile cloud computing applications, which is an emerging field that deals with how users can share and pool resources of local mobile devices [25, 31].

In Section 2.2, we describe mobile tethering in more detail. Section 2.3 deals with technological challenges, including the test of a mobile tethering application in practice. Section 2.4 discusses social issues regarding mobile tethering, which are analyzed in conjoint analysis. Discussion and conclusion is provided in Section 2.5.

## 2.2. Mobile Tethering Applications and Gaps

Tethering refers to connecting devices together using available interfaces. In the context of mobile technologies, tethering is the only available option to allow sharing the data connection with others. Tethering involves forwarding of the traffic from one network interface to another, bridging the 3G/4G interface with the WiFi, Bluetooth or USB (See Figure 2.1). Most modern smartphones and tablets provide tethering capabilities in their firmware. For other devices, specific applications allow to tether the data connection. These applications might need root access to bypass software or hardware limitations and security mechanisms in order to allow privileged access to the operating system. Obtaining root access requires a complex procedure that might discourage inexperienced users and, even though legal, it might void the warranty of the device.

The most common, currently used, tethering mechanism involves an Internet Protocol (IP) gateway solution, smartphones act as an IP router and gateway for the Local Area Network (LAN), forwarding IP packets between LAN and Wide Area Network (WAN). Some other existing tethering techniques, like modem gateways, application layer proxies or port forwarding provide only limited connectivity and do

Table 2.1: Existing mobile tethering applications and their characteristics (advantages, disadvantages)

| Characteristic/Application | Portable WiFi Hot Spot | WiFi Tether | Open Garden |
|---|---|---|---|
| Availability | Android Firmware | Google Market | Google Market |
| Rooting | No | Yes | Yes |
| Mode | Infrastructure | Ad-Hoc (Infrastructure kernel) | Mesh |
| Interfaces | WiFi, BT | WiFi, BT | WiFi, BT |
| Connectivity | WiFi Authentication | WiFi Authentication | BT Authentication |
| Security | WPA2 | WEP | WPA2 |
| Multi-Hops | No | No | Yes |
| Path Choice | No | No | Automatic |

not allow simultaneous use of voice and data services [83]. Technically, IP gateways for tethering are implemented using Network Address Translation (NAT), so in this way the mobile device acts as an IP router with NAT for LAN clients, forwarding their IP packets through the provider's network (e.g., GPRS tunnel). Deployment of NAT has some technical implications. Transforming forwarded IP traffic from private LAN to public IP, results in a modified traffic pattern. A consumer cannot be directly reached from the wireless network and since the NAT is designed to be transparent payloads are transmitted unmodified.

Several applications enable tethering next to standard Android Tethering and Portable WiFi Hotspots. Applications like Android WiFi-Tether, Wireless Tether for Root Users or Open Garden WiFi Tethering [2][3] can be found on Google Market (Play). These applications enable tethering through the WiFi, Bluetooth or USB interfaces for rooted handsets running Android, providing a standard IP gateway with Domain Host Configuration Protocol (DHCP) and NAT. Clients can connect using the WiFi interface (ad-hoc mode) and get access to the data connection using the 4G, 3G, or 2G mobile connection which is established by the handset.

Android engineers prefer WiFi Direct and ad-hoc has slipped in priority in favor of other solutions keeping power constraints and security in mind. WiFi Direct might be a better solution in the future, even though ad-hoc mode is a well known technology supported by many devices and used for a couple of years. Moreover, WiFi Direct is not yet available on all devices. WiFi Direct is a layer that auto configures one of the devices as a soft application and it brings important security features, ease of setup, and higher performance that is not currently available in ad hoc mode. Table 2.1 presents an overview of the most used tethering applications with both their advantages and disadvantages.

All in all, a framework that enables tethering across different hardware platforms

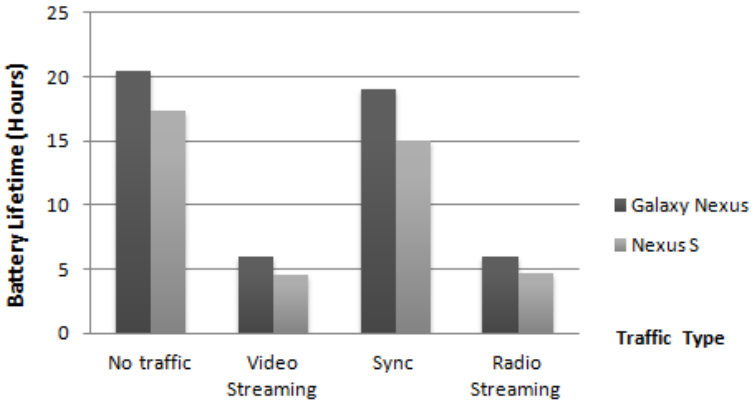



Figure 2.2: Battery lifetime of two different devices under test, WiFi Tethering various types of traffic.

and operating systems, with the ability to leverage different radios depending on availability is needed. It should manage the device discovery, connectivity and security requirements for authenticated and encrypted communications per service. Work in this direction has been started by projects like AllJoyn [80], an open source peer-to-peer software development framework that enables ad-hoc, proximity-based and device-to-device communication.

Even though applications that enable tethering do exist, they only allow a dyadic relationship rather than a network of cooperating users. The existing applications assume tethering is used for simple purposes and do not take into consideration the complex interaction processes among people. The technical part of current research is focusing on enhancing capabilities of devices and enabling cooperation, i.e., data connection sharing, among users, by discussing the technical challenges.

## 2.3. Technical Challenges

In this section, we explore technological challenges of mobile tethering regarding energy consumption, handling of bandwidth requirements, and protecting privacy of providers and users. We test how severe these challenges are through testing a self-developed personal tethering application based on existing open source code and on the android stack. The application enables a cooperative network of users to tether data connection with own devices, but also with friends, family, co-workers and total strangers. The application was tested on two Samsung devices: Samsung Nexus S and Samsung Galaxy Nexus.

### 2.3.1. Energy Consumption

Energy consumption is one of the challenges to prolong battery lifetime. Although the battery lifetime varies depending on the usage patterns, smartphones drain more power than the legacy cell phones. To reduce energy consumption, users can close the Internet connection when the device is idle. However, tethering keeps the smartphone connection always-on and both the WiFi and mobile broadband interfaces consume energy at the same time.

To observe the effect of tethering on the energy consumption of the smartphones, we ran various scenarios with a connection provider and two consuming devices (Nexus S, Galaxy Nexus). First, the depletion time of the phones has been investigated to give a rough estimate of energy consumption, see Figure 2.2. We tested different types of traffic, i.e. being idly connected to the tethering device with no traffic; video streaming; email synchronization; and radio streaming.

Next, voltage probes were connected to the battery. A mobile measurement setup has been used based upon the National Instruments USB-6009 data acquisition card (DAQ) [1]. The setup employing the DAQ and a very low-ohmic high-side shunt has been applied to a smartphone as is with no modification regarding its power supply or possible mode of operation. Hence, the setup allows for deducing general statements on power dissipation and energy consumption at an accuracy of what could possibly be achieved with other published measurement approaches for smartphones or built-in measurement capabilities of a smartphone. Using the setup and previous scenario of two consumers (i.e., client) and provider (i.e., server that shares its cellular Internet connection over WiFi) devices, we have analyzed the following test cases:

- Case 1: Provider not tethering
- Case 2: Provider tethering but no other device connected
- Case 3: Consumer 1 connected but no traffic generated
- Case 4: Consumer 1 connected and radio streaming
- Case 5: Consumer 2 connected but no traffic generated
- Case 6: Both consumers connected and streaming

Figure 2.3 shows the increase in the power consumption. From case 1 to 6 there is a 46% increase in power consumption that is considerably high. The reason for this high energy consumption is that WiFi hotspots are designed to be connected to an electric supply. There are power saving modes in the 802.11 standard [98]

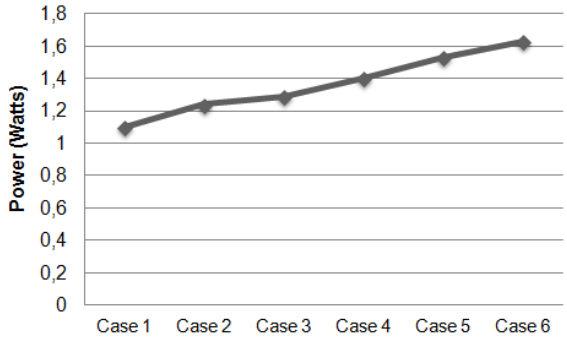



Figure 2.3: Average power per test case.

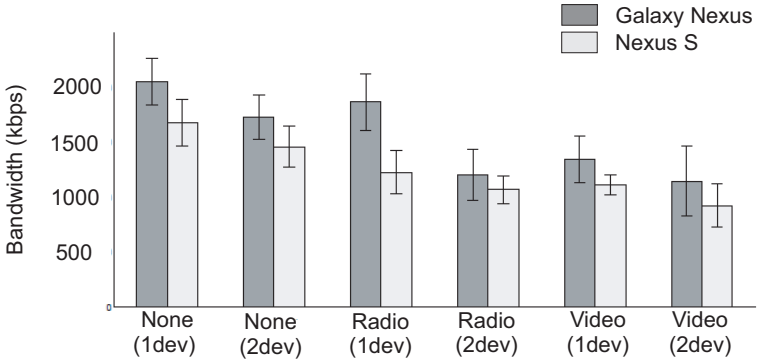



Figure 2.4: Bandwidth (download) comparison of the two devices under test while tethering. Comparison of different types of traffic (no traffic, radio streaming and video streaming) taking in consideration multiple devices connect to the provider (one or two devices connected). Values represent the average of the measured download bandwidth (20 samples) with 95% confidence interval.

for the client devices. Unfortunately, the hotspot is supposed to be always on. In the ad hoc mode of 802.11 standard, the beacon generation task is divided among the devices to save energy. WiFi tethering depends heavily on the tethering device, which cannot go into sleep mode. There is some promising work [41] on sleep interval adaptation algorithms for WiFi tethering. The sleep intervals can be adapted to the ongoing traffic patterns of various applications without changing the 802.11 protocol. However, at this point we can only state that the energy impact is major only in the case of continuous usage for a longer period of time and with certain type and amount of traffic.

### 2.3.2. Bandwidth

It is important to analyze whether sharing data connection is indeed feasible contrary to the general belief that simultaneous usage restricts the bandwidth and degrades the performance of the device. Available bandwidth on both the provider (tethering device) and consumer devices was analysed. Both provider and consumer bandwidth performance were closely inspected with different types of background traffic. Figure 2.4 shows the maximum available bandwidth of provider devices with different number of consumers and types of background traffic originating from consumers. The tests confirm that the number of connected consumers and the amount of background traffic has a significant effect on the available bandwidth. However, perceived performance and bandwidth still satisfy the requirements of many applications.

To improve the provider's performance, further traffic shaping software can be employed in order to limit the consumer's traffic. Figure 2.5 illustrates the performance of the consumer devices. In general, the devices perform on a similar level when connected to the same network operator.

The android technical sheets state that depending on the device type, up to eight devices might be connected. However, our tests show that connecting more than four or five devices has major implications on the data connection. Connectivity is still possible, but insurmountable limitations and delays are to be expected.

A device can supply its Internet connection to clients, depending on the tethering device model and some other parameters, such as network type, coverage or congestion. The performance varies with different scenarios (e.g., urban, indoor or outdoor). Using standard models related to the WiFi signal strength is required for an in-depth analysis. The signal strength and the data rate decline when moving further away from the hotspot. Important challenges related to network performance and acceptable WiFi coverage might impose limitations on the quality and distance of mobile tethering. Nevertheless, based on performed tests, it can be stated that tethering is a feasible service.

### 2.3.3. Security and Privacy

Both the consumers and the providers are concerned about security and privacy if strangers are involved. If certain security mechanism are not enforced, there is a risk that unauthorized third parties may "borrow" the bandwidth, using the wireless connection to access the Internet. Packets might get intercepted or, even worse, someone might gain unauthorized access to the device, which can get involved in illegal actions. Problems may arise for rooted mobile devices, which might not have

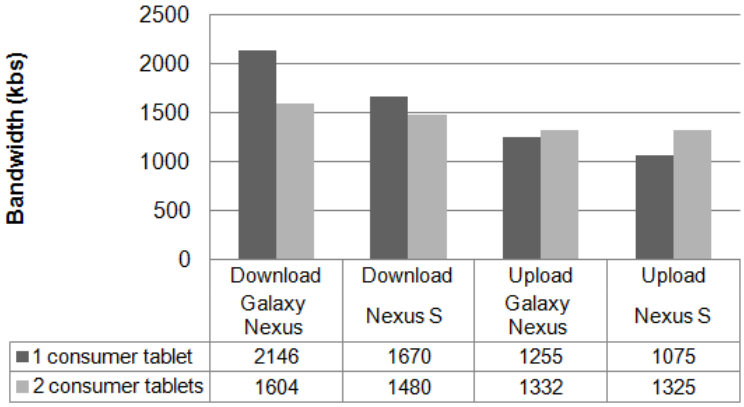



Figure 2.5: Bandwidth statistics (download and upload) of the consumer devices (Samsung Galaxy tablets 10.1). The tablets are sharing the data connection of a provider that uses the built in feature to tether. The tethering devices (Nexus S, Galaxy Nexus) provide data access to multiple devices (one or two). The tablets are just connected without producing any relevant traffic.

the same levels of encryption as regular devices. Some available technologies can meet security concerns as shown in Figure 2.6. WiFi protected access (WPA) and WPA2-Personal ensure confidentiality between providers and consumers. Adversaries cannot eavesdrop communication, attack with replay messages or achieve man-in- the- middle attacks. However, the provider and the consumer must establish a security association before communication that requires exchanging a shared key. Even if the key distribution is handled, a provider can be an adversary. Providers should not be able to eavesdrop the traffic of consumers. In order to ensure privacy of the consumer against a malicious provider, end to end security mechanisms are required. Transport layer security or IPsec prevents the malicious provider to tap into the traffic of the consumer. Although a provider cannot tap into the traffic with SSL, the provider can still identify the end point of the communication. IPsec offers better security however it degrades the performance and the consumer has to find or establish an IPsec end point prior to the communication.

The above solutions can protect the consumer. However, the provider may also be concerned about possible illegal activities of the consumer. The provider does not have control over the traffic of the consumer. If required, the provider should point out the real source of the illegal traffic.

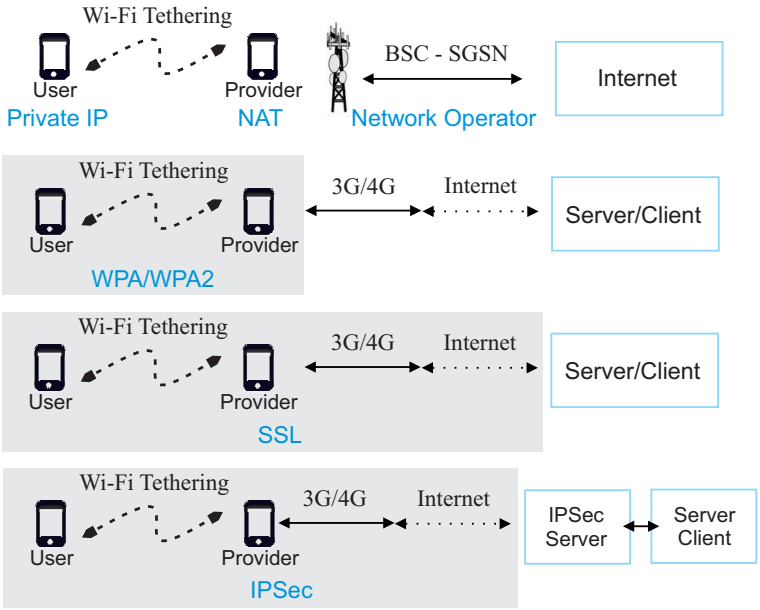



Figure 2.6: Security protocols that ensure the confidentiality of the consumer.

## 2.4. Social Perspective

To better understand the user's perspective on tethering and motivations for sharing, a conjoint analysis was executed, focusing on the rules of and conditions for cooperation (provider or consumer) mentioned in Chapter 1.

### 2.4.1. Conjoint Method

Conjoint analysis is a method to determine, measure, and predict consumers' preferences with regard to different features that define a product or a service [38, 68, 87]. Conjoint analysis identifies trade-offs consumers make for selecting different features of products or services [12, 37], by estimating the importance and utility values that consumer assign to features of a service or product. Conjoint analysis assumes that these features significantly influence the decision process simultaneously. In order to obtain and calculate the importance and utility values ordinary least squares regression or logit analysis are normally used. In conjoint analysis, unlike conventional survey approach in which respondents are asked to estimate how much value they assign to each attribute, the objective is to capture the preferences in a series of choices or ratings. Respondents' choices or ratings make it possible to compute the relative importance of each attribute under investigation. In other words, instead

of "stated importance", conjoint analysis uses "derived importance" values for each attribute or feature [34].



### Design of the Conjoint Instrument: Attributes and levels

There are several basic steps in designing a conjoint study. First, the data collection approach (online survey or pen-and-paper questionnaire) needs to be identified. Second, the most important attributes (features) and the levels of attributes (level can be defined as the set of values the attribute can take) should be identified. In conjoint analysis the levels of attributes describing a service or product are combined together to form a description of hypothetical attribute bundles [57]. In the current study, several attributes and levels with regard to the mobile tethering from both consumer and provider perspectives (see Table 2.2 and 2.3).

Table 2.2: Conjoint attributes and levels (Consumer)

| **Attributes** | Levels | |
|---|---|---|
| Costs | Higher | No connection |
| Quality of Service | Lower | Normal |
| Battery Lifetime | Longer | Shorter |
| Person to Share with | Familiar (Family, Friend, Co-worker) | Not Familiar (Unknown Person, Public) |
| Subscription Type | Limited | Unlimited |

The third step is to select an appropriate conjoint analysis approach [54, 67, 75, 84, 94]. Full-profile conjoint analysis or full-concept approach was selected. In a full-profile conjoint analysis, each conjoint or card shows a complete product or service consisting of a different combination of levels of all attributes. The advantage of this approach over the other methods is threefold. First, in full profile conjoint, all attributes are assumed to be independent. Second, it enables researchers to obtain information on users' preferences and what they value most with regard to a product or a service (each attribute level and the corresponding utilities). And third, full profile conjoint is applicable when the number of attributes is not very large (usually up to 8 attributes). In full profile approach, respondents are requested to rate, rank, or score a set of profiles (cards discussing the bundled attributes) presented to them. In the current study, respondents were asked to rate their preferences. In this study

Table 2.3: Conjoint attributes and levels (Provider)

| **Attributes** | Levels | |
|---|---|---|
| Costs | Normal | Higher |
| Quality of Service | Lower | Normal |
| Battery Lifetime | Longer | Shorter |
| Person to Share with | Familiar (Family, Friend, Co-worker) | Not Familiar (Unknown Person, Public) |
| Subscription Type | Limited | Unlimited |

we will make use of an orthogonal design to reduce the number of the cards (also labeled as conjoint). We design two sets of conjoint attributes: one for the consumer (Table 2.2) and one for the provider perspective (Table 2.3).

The *Cost* attribute implies whether actual subscription costs will increase due to the use of tethering. The attributes related to costs are (1) higher costs than normal and (2) normal costs or no connection.

Next, *Quality of Service* due to bandwidth reduction can be minimized while tethering (sharing bandwidth), and tethering can be perceived as a degradation of the device's data performance and quality of services. Bandwidth is perceived by users as the average rate of successful data transfer through the communication channel (transmission speed). This attribute can have two levels, (1) "normal", i.e. there will be no differences in quality of service or (2) "lower", i.e. lower bandwidth than normally.

Mobile tethering and sharing the Internet connection with others might have a significant impact on energy consumption as discussed in Section 2.3.1 (i.e., battery lifetime). When the connection is shared, there will be continuous impact on the tethering device battery lifetime, depending on different traffic classes. In the current study two attribute levels are discussed, i.e. "longer" which means there is a minor impact on the battery and its lifetime, battery will not deplete soon, or "shorter" which means there is a high impact on the battery, depletion with a high rate, shorter lifetime and battery will deplete soon.

One of the major concerns with tethering is the *Person to Share with*. Today, consumers typically provide mobile tethering to own devices or devices that belong to close friends. However, an important question is why a user should allow total

strangers to exploit their private connection. Are people willing to share their connection for incentives (money or virtual currency) or are they just expecting to receive reciprocal treatment in future interaction? The familiarity and level of acquaintance with different people might be important for sharing from both perspectives user and the person with the tethering device [94]. In the current study, two groups of people are identified. The first level (Family, Friend, and Co-worker) refers to people who are known and sharing can be less problematic. The trust issue may play less significant role in this scenario. The second level (Unknown Person, Public) concerns people who are not familiar and in this scenario sharing data access might be a risk and the level of trust is of utmost important.

*Subscription type* with the data provider in this study is considered to be either unlimited or limited. If the subscription is unlimited then there will be no impact on the data usage and a person who is sharing his/her data access with someone else does not have to worry about data usage. In contrast, if the subscription has a limited data usage (per month/ Per Mb) then sharing the connection may potentially lead to problems.

When full profile conjoint approach is used all combinations of the attributes and levels are considered. In the current study the combination of all the attributes and levels creates 32 ($2^5$) possible service profiles/conjoints for each perspective (consumer or provider). Johnson et al. [51] and Pignone et al. [77], argued that it would be a difficult task for respondents to answer all the questions when the number of profiles is too high and therefore the number of profiles should be reduced. An orthogonal design takes only the main effect of each attribute level into account. When orthogonal design is used, interaction effects between attributes will not be analysed. Statistical Package for the Social Sciences (SPSS) software version 18 was used in the current study to generate the orthogonal design. The result of orthogonal design created eight unique conjoints out of the 32 possible attribute bundles. This number of conjoints is small enough to be included in a survey and large enough to compute the relative importance of each attributes and their levels. The utility scores for each attribute level are called a part-worth. The computed utility scores for each level of attribute provide a quantitative measure of the preference for separate parts of the product (assigned to the multiple attributes). The larger values indicate greater preference.

### Design of the Conjoint Instrument: Dependent variables

As dependent variable, we measure the likelihood that a person would share the data connection or utilize a shared connection in exchange for different types of incentives. As we discussed in Chapter 1, a series of rules for the evolution of cooperation in

nature have been presented in [70]. For instance, individuals cooperate if they are genetic relatives to each other (kin selection). Cooperation may emerge among unrelated individuals if future interactions are probable (direct reciprocity), when altruistic behaviour may be required in reverse direction. Furthermore, individuals help other peers if they have enough reputation (indirect reciprocity).

Whether skills, resources or goods are concerned, people do cooperate [16]. Assets and resources can be shared for free or in exchange for services or money. Some popular examples of collaborative consumption include car sharing or couch surfing. These services, where people interact with total strangers involve credit exchange or reputation. With regard to mobile tethering, this means that the user can either pay money for the connection or the provider can earn reputation or virtual currency that can be used to get connection when needed. People might be willing to share their connection for an incentive (not necessarily real money) or they are just expecting to receive the same treatment (service) in another situation, at another time (e.g. roaming). Specifically, we contrast the intention to share or receive a tethered connection in exchange for financial compensation, virtual currency or no compensation.

Next to the questions regarding compensation, we explore the effect of privacy concerns by asking the users to rate the extent to which they would be concerned over their privacy in a given situation.

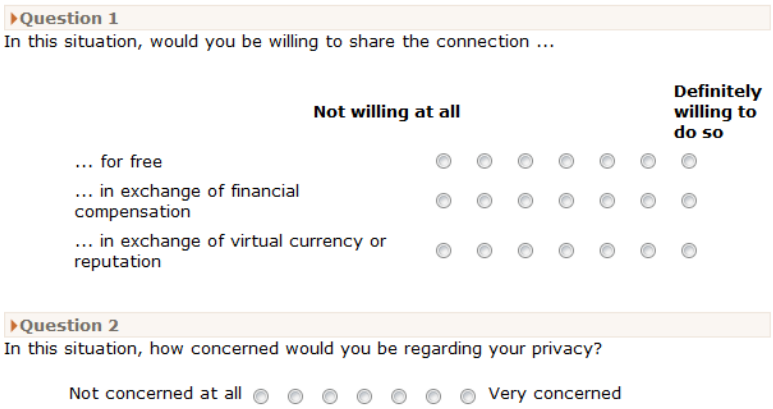
### Questionnaire

The questionnaire was pre-tested by a number of experts and smart-phone users who were well acquainted with the conjoint analysis as well as mobile communication services to verify the accuracy of the questionnaire and to check for ambiguous expressions. An adjusted questionnaire was distributed among the 83 respondents.

An example case and its question (i.e., snapshot of the questionnaire tool) for a provider are as follows:

You are in a situation in which you are willing to share your mobile data connection with somebody else, so you are the connection provider:
**Case 1**

- Normal network costs: sharing does not incur extra costs
- Sharing your connection reduces the performance of your device and data connection
- Minor impact on the battery and its lifetime, battery will not deplete soon
- Share the connection with familiar persons (family, friends or colleagues)
- You offer unlimited amount of Mbs to the person(s) using the connection




### Sampling

Data were collected making use of a web questionnaire (online survey) distributed from July $7^{th}$ until July $20^{th}$, 2012. The respondents have become aware of the objective of the study by a short description explaining the mobile tethering distributed within a university setting. As Compeau et al. [21], argue this is not problematic seen the fact that in conjoint analysis, like in experimental research the interest is not in generalization to a specific population but in understanding the way certain attributes relate to decision making of consumers. The interest is more in understanding the role of relevant concepts (internal validity), than the external validity of the results. In total, 74 complete questionnaires were obtained (see Table 2.4 for more information on respondents).

Table 2.4: Respondents' background information

| | | | | | | |
|---|---|---|---|---|---|---|
| **Operator** | Vodafone:28 | Orange:9 | Lebara:7 | T-Mobile:4 | KPN:2 | Others:24 |
| **Phone Brand** | HTC:9 | Nokia:23 | Samsung:21 | iPhone:6 | LG:3 | Others:13 |
| **Occupation** | | Student:45 | | Non-Student (Academics):29 | | |
| **Gender** | | Female:24 | | Male:50 | | |
| **Average Age** | | | 26.81 | | | |

### 2.4.2. Results

Conjoint analysis creates utility functions that indicate the perceived value of a feature and how sensitive consumer perceptions and preferences are to changes in that feature. We also present the importance of each attribute in percentage. The validity of the conjoint models is indicated by Pearson's $r$ and Kendall's $\tau$ [85], which are the indicators of correlation between the estimated and observed values. They should exceed the recommended benchmark values, respectively .80 for Pearson's $r$ and .70 for Kendall's $\tau$. The conjoint models' values in the current study were above the recommended values for both perspectives (consumer and provider). This indicates that there is a strong relationship between the rating and the utilities. However Kendall's $\tau$ for the rating of financial compensation and for concern of privacy (Question 2 and 4) seen from a tethering consumer perspective were slightly lower than the recommended threshold value. These utility functions indicate the perceived value of the feature and how sensitive consumer perceptions and preferences are to changes in product features. To evaluate the utility of the attributes (i.e., sensitivity of consumer/provider perceptions on an attribute), simple dummy variable regression analysis was used. From the tethering consumer perspective, when asked for the intention to share for free, the familiarity of the other person is most important with 44% (See Table 2.5). The utility value is .338 for sharing with known people. It is noteworthy that battery lifetime has the lowest importance rate (1%) and presumably this attribute is the least important criterion for the respondents while using tethering technology. A similar pattern can be found for other dependent variable (questions 2-4). Again the person to share with has received the highest importance rate, between 40% and 45%. Subscription costs to the operator have a moderate level of importance. It seems that for the majority of the respondents, tethering with people that are familiar is of utmost importance and bears the lowest privacy risks. Moreover, the results show that technological

issues such as battery lifetime, subscription type and impact on quality of service while sharing the connection are less important.

The results for the provider of tethering view mirror the results for tethering consumer (See Table 2.6). In other words, sharing with other people (known vs. unknown) receives the highest importance values. Strikingly, with regard to the question "how concerned would you be regarding your privacy" when using tethering technology, familiarity of the person to share with has received the highest importance rate (89%). Next, the conjoint results show that quality of service is the least important criterion for respondents when taking a tethering provider view. Quality of service has received the lowest importance rate for question 1, 3 and 4 with similar values. Only in question two, "sharing in exchange of financial compensation" the results are different and respondents indicated that subscription type is the least important criterion and the importance rate is 7%.

In conclusion, from both the tethering provider as well as consumer of tethering capabilities perspective, it is utmost important that people cooperate with persons with whom they are familiar (family, friend and co-worker). Mobile tethering becomes more acceptable when the other party is acquainted. Next to that, costs also represent a critical issue, while technological challenges appear to be not that relevant. Tethering capability providers are willing to share with strangers only if there is a form of payment. Moreover, if the provider decides to share the connection for money energy consumption seems to have a higher influence relative to other technical aspects.

Consumers would prefer of course a free connection. However, they are still willing to pay in certain scenarios like roaming. As a consequence of paying for the tethered data connection they require a certain level of Quality of Service (QoS) and guaranteed bandwidth.

All in all, our survey indicates that a cooperative hub of familiar people using the mobile tethering features of their devices can be achieved. Using proposed schemes that involve financial or reputation compensation can extend present usage.

Table 2.5: The conjoint results for the dependent variable questions (Consumers' Perspective). Utility values and importance indicate the given value to an attribute by the respondents. We highlight the overall most significant attribute in yellow while the highest valued in blue and the lowest in red for each question. The cases where statistical significance lacks are highlighted with green.

| Attributes | Levels | Q1: Sharing for free | | Q2: In exchange of financial compensation | | Q3: In exchange of virtual currency or reputation | | Q4: How concerned would you be regarding your privacy? | |
|---|---|---|---|---|---|---|---|---|---|
| | | Utility | Importance | Utility | Importance | Utility | Importance | Utility | Importance |
| Costs | No connection | 0.276 | 36% | 0.046 | 12% | 0.108 | 19% | 0.186 | 13% |
| | Higher | -0.276 | | -0.046 | | -0.108 | | -0.186 | |
| Quality of Service | Normal | 0.022 | 3% | -0.038 | 10% | -0.004 | **1%** | 0.203 | 14% |
| | Lower | -0.022 | | 0.038 | | 0.004 | | -0.203 | |
| Battery Lifetime | Longer | -0.004 | **1%** | 0.029 | **8%** | 0.050 | 9% | 0.225 | 16% |
| | Shorter | 0.004 | | -0.029 | | -0.050 | | -0.225 | |
| Persons involved | Familiar (Family, Friend, Co-worker) | 0.338 | **44%** | 0.154 | **40%** | 0.246 | **44%** | -0.647 | **45%** |
| | Not-Familiar (Unknown Person, Public) | -0.338 | | -0.154 | | -0.246 | | 0.647 | |
| Subscription Type | Unlimited | 0.123 | 16% | 0.117 | 30% | 0.154 | 27% | -0.175 | **12%** |
| | Limited | -0.123 | | -0.117 | | -0.154 | | 0.175 | |
| Pearson's $r$ | | .904$p < .001$ | | .916$p < .001$ | | .952$p < .000$ | | .927$p < .000$ | |
| Kendall's $\tau$ | | .764$p < .004$ | | .643$p < .015$ | | .857$p < .001$ | | .593$p < .022$ | |

Table 2.6: The conjoint results for the dependent variable questions (Providers' Perspective). Utility values and importance indicate the given value to an attribute by the respondents. We highlight the overall most significant attribute in yellow while the highest valued in blue and the lowest in red for each question.

| Attributes | Levels | Q1: Sharing for free | | Q2: In exchange of financial compensation | | Q3: In exchange of virtual currency or reputation | | Q4: How concerned would you be regarding your privacy? | |
|---|---|---|---|---|---|---|---|---|---|
| | | Utility | Importance | Utility | Importance | Utility | Importance | Utility | Importance |
| Costs | Higher | -0.608 | 37% | -0.043 | 9% | -0.416 | 36% | 0.019 | 3% |
| | Normal | 0.608 | | 0.043 | | 0.416 | | -0.019 | |
| Quality of Service | Normal | 0.040 | **2%** | 0.059 | 12% | 0.029 | **2%** | -0.019 | **2%** |
| | Lower | -0.040 | | -0.059 | | -0.029 | | 0.019 | |
| Battery Lifetime | Longer | 0.176 | 11% | 0.158 | 33% | 0.213 | 19% | 0.019 | 3% |
| | Shorter | -0.176 | | -0.158 | | -0.213 | | -0.019 | |
| Persons involved | Familiar (Family, Friend, Co-worker) | 0.763 | **47%** | 0.186 | **39%** | 0.416 | **37%** | -0.625 | **89%** |
| | Not-Familiar (Unknown Person, Public) | -0.763 | | -0.186 | | -0.416 | | 0.625 | |
| Subscription Type | Limited | -0.047 | 3% | -0.035 | 7% | -0.072 | 6% | -0.019 | 3% |
| | Unlimited | 0.047 | | 0.035 | | 0.072 | | 0.019 | |
| Pearson's $r$ | | $.981 p < .000$ | | $.974 p < .000$ | | $.966 p < .000$ | | $.999 p < .000$ | |
| Kendall's $\tau$ | | $1.00 p < .000$ | | $.929 p < .001$ | | $.786 p < .003$ | | $.964 p < .001$ | |

## 2.5. Conclusions



Mobile tethering can be a new market opportunity, not only for customers, but also for network operators. This is why we analyzed not only the technical performances of tethering, but also the users' motivation. Regardless of the opportunities, many mobile tethering ideas only exist as a concept, and a number of technological improvements need to be made, for instance in WiFi energy consumption. In the future, mobile interactions will be driven by network technologies that enable sharing and pooling of resources on a scale never possible before, maximizing the potential of presently available applications and exploiting wireless resources on mobile devices.

The energy consumption and bandwidth measurements in this chapter show promising results for the future of tethering. Energy consumption is still high, but ongoing research tries to address the problem. Also, the availability of bandwidth is affected by tethering specifically if the user moves away form the core hot-spot. Dealing with security and privacy is a problem. Although security might be solved via network protocols, privacy is directly related to access of resources on the devices used during tethering. In order to solve the privacy problems operators can be involved as a trusted third party and apply network security protocols.

Currently, tethering is mainly employed to connect devices that are owned by the same person. Cooperative services, i.e. cooperative web browsing for mobile devices and mobile cloud computing can take advantage of further applicability. The conjoint analysis in this chapter shows that familiarity of the persons that a connection is shared with is a key issue. Respondents only wish to share a connection with somebody who is known or familiar. Similarly, respondents are only willing to use a shared connection from a familiar person. Costs are the second important decision criterion. Credit exchange or the use of a virtual currency can support tethering with unknown persons. All in all, social and costs related issues are much more important to users in their decision to adopt mobile tethering than technological issues like battery lifetime and bandwidth limitations.

A limitation of our study is that the technological tests were based on an application that only works on Android, which was only used on Samsung devices. Other operating systems and other devices may perform differently. Furthermore we are aware that tethering is available and developing rapidly, making research in this domain like shooting on a moving target. Nevertheless we hold the opinion that doing techno-economic research in combination with user research in a service design process is valuable. Insights based on research with a small number of concepts helps to develop a more generic model that moves away from existing acceptance and user research.

With regard to tethering, we may conclude that tethering represents a viable technology. Tethering can bring added value not only to the users, but also to the mobile telecommunications industry. Even though tethering is dependent on the performances of the telecommunication infrastructure and on the network operators per ensemble, the perceived quality of tethering supports an increased usage, will support further development.

With regard to cooperation, we may conclude that though costs are still a concern, social relations are crucial in sharing resources. In Chapter 4, we will create a decentralized social-device network to automate the familiarity based cooperation. Before that, we will investigate a prior step in cooperation in the next chapter, which is the discovery of resources. Our discovery protocol includes semantic representations of resources as well as their owner information.

# 3

# Service Knowledge Discovery in Smart Machine Networks

In Chapter 1, we described our vision of "devices that know their owners" in accordance with the outcome of Chapter 2 that social relations are crucial in cooperation. Before further elaboration of our vision and explaining its core design, in this chapter we investigate the problem of discovery of surrounding devices with their capabilities. We consider discovery of resources and capabilities of devices as one of the initial steps towards achieving a cooperative platform of smart devices.

Today's operational service-discovery protocols carry simple text-based uniform resource identifiers that are not expressive enough. Machines cannot comprehend the meaning of a new service that is not in their knowledge base. In addition to being more expressive, service-discovery protocols must compensate for the diversity to improve cooperation between the devices that use different application protocols and operate on different communication interfaces.

In this chapter, we propose the Smart Discovery Protocol (SDP) that outperforms the traditional service-discovery protocols with three main features: *(i)* a more expressive semantic representation of services, *(ii)* operating at the network layer to deal with diversity, and *(iii)* unifying existing service-discovery protocols. SDP represents services with ontologies as some recently proposed semantic service-discovery protocols. It further enhances the success of semantic representations by creating a unified platform that can carry legacy discovery services. The underlying transport mechanism of SDP is designed as an add-on to the Neighbor Discovery Protocol

(NDP) of the IPv6 standard. The metadata is carried in the payload of ICMPv6 packets. Simple text-based representations of other service discovery protocols are embedded in type-length-value options of NDP. Authenticity of the devices is ensured by the IPv6 Secure Neighbor Discovery protocol. Unlike previous semantic approaches on service discovery, we have implemented our protocol on real hardware. The results demonstrate the feasibility of carrying semantic representations of the services and integration of other service discovery protocols.



## 3.1. Introduction

The computational power of networked devices enhances every day and these devices can accomplish very complex tasks. For instance, a smartphone can observe our daily travel patterns and suggest improvements. However, these smart devices are not *swiss army knives*: they cannot encompass all kinds of capabilities and they have limitations that cannot be surmounted by simply boosting their processing power. Take a portable smartphone as an example, it cannot have a 50 inch display or wash our clothes. Often smart devices have to utilize the resources and capabilities of other devices and complement each other for accommodating the needs of their owners. Moreover, as we mentioned in Chapter 1, cooperation must be accomplished without human intervention because the configuration requirements may be unmanageable as the number of devices increases rapidly.

One of the significant challenges in establishing a cooperative network of smart devices that complement each other is the lack of an expressive and autonomous service knowledge discovery system. In the scenario illustrated in Figure 3.1, a device searches for a piece of missing context information. Alice's camera does not have any capabilities for discovering its geographic position. To determine its location, it queries the devices in the vicinity as to whether or not they are capable of providing the location information. Instead of asking for specific services that give GPS, 3G triangulation or cell id information, simply any service that can extract location information is queried. Nearby devices that know they have hardware for location information advertise their specifications with the properties of the equipment such as the precision of the location information. After identifying the devices that have the required capability, the camera requests the location information from one of them. This scenario clearly depicts that devices have to be smart and infer on collected pieces of information. We refer to this process of making machines understand the semantics of service descriptions as service knowledge discovery.

Semantic Web researchers achieve a similar goal of making machines capable of understanding the content in a web page without human intervention. Machines

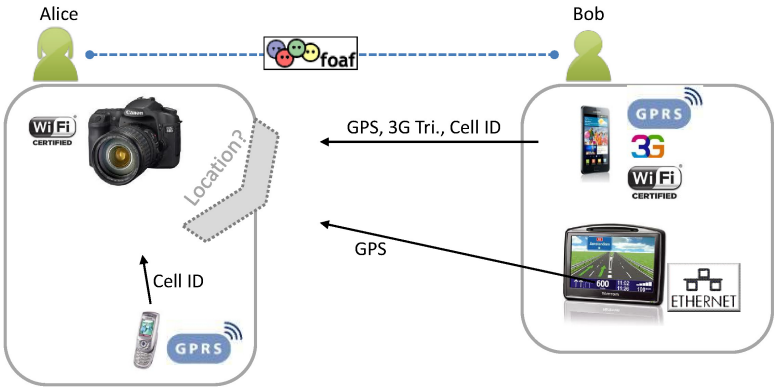



Figure 3.1: Missing location information is queried from neighbor nodes. The device owners and network interfaces are also shown.

collect the information scattered in several web pages through the logical relationships among them. The power of machine-understandable web pages originates from the representation of knowledge in ontologies. In an autonomous network of smart devices, we need a service knowledge discovery protocol that can evolve and adapt to future requirements just as in the Semantic Web. The ontologies comprising the information about the services should be distributed among the devices. With ontologies, the discovery system builds a knowledge base that evolves and comprises new systems and services. Fortunately, the tools for handling ontologies are available, and the scientific challenge for us is to propose a distribution system for the information. Some of the recent proposals for service discovery (e.g., mRDP [95]) also promote the use of ontologies as knowledge representation. However, they are not able to deal with heterogeneity of both the networks and applications. Compatibility with legacy systems in these architectures is also missing.

There are many operational and even recently proposed service discovery protocols that are being used today in many systems. However, there is still no widely deployed autonomous service discovery protocol that exists in every device. The reasons are as follows: firstly, although some of them improve expressiveness by using ontologies, still many of them prefers simple representations that do not evolve for future requirements of the services. Secondly, there are various application protocols for services and existing service discovery protocols add new application protocols to the list instead of decreasing the heterogeneity. While deploying a service, a separate application layer protocol for discovery has to be installed. Thirdly, companies push vendor locking. Each service discovery protocol has its own island of connected devices where there is no interaction with an outsider.

In this chapter, we propose the Smart Discovery Protocol (SDP) as an add-on to the Neighbor Discovery Protocol (NDP, IETF RFC 4861) of IPv6, which is positioned in the kernel and comes as pre-installed in the operating systems. SDP is independent of the application layer and can exist in heterogeneous networks by the convergence to IP. It operates on ICMPv6 packets that carry semi-structured (ontological) service representations and queries. *Not only the knowledge about the resources and capabilities that form a service, but also the details about the service owner and the context are included in the representations.* In the type-length-value (TLV) options of NDP, the ICMPv6 packets carry the Uniform Resource Identifiers (URI) that define legacy services. Semantic data is carried in the payload of these packets. Multicast advertisement and solicitation messages are employed to maintain scalability. Secure Neighbor Discovery (SEND, RFC-3971) is used for authenticating the collaborating devices. For confidentiality, multicast group security proposals can be employed. The contributions of SDP are the following:

- SDP is a semantic service knowledge discovery protocol that decreases human intervention in service discovery for devices complementing each other and enables cooperation among the machines.
- SDP can operate in all IPv6-based networks independent of the lower and upper layers.
  - Devices that operate on the network layer like routers can collect more information about the structure of the network by looking at ICMPv6 packets and improve the QoS.
- Legacy systems are unified in one message type.
- Low-power devices are also involved in discovery by using URI identifiers of existing service discovery architectures.

In the next section, we propose and describe SDP. First, the challenges and requirements of a service discovery protocol are given, then the design of SDP is matched with the requirements. We present the performance of SDP in Section 3.4. A thorough comparison to existing service discovery architectures are given in Section 3.5. Finally, we conclude and discuss future work.

## 3.2. Smart Discovery Requirements

In this section, we present the Smart Discovery Protocol (SDP) that paves the way for an autonomous cooperative network of smart devices. We present the chal-

lenges of service knowledge discovery in smart machine networks using the scenario illustrated in Figure 3.1.

In the light of these challenges, we will explain service representation format and ontologies for readers who are not familiar to semantic technologies in Section 3.2.2. Then the SDP protocol will be described starting with Section 3.3. We will elaborate the packet format, message types, legacy support features, protocol operation, and security.



### 3.2.1. Challenges and Requirements of Knowledge Discovery

The challenges of service discovery have been identified in [64] and they are still valid with an increasing importance. We summarize and elaborate some of those challenges that are still open research questions and introduce new challenges (last two) to indicate the need for a new knowledge discovery protocol.

Nodes in a network need to employ a service knowledge discovery system to obtain extensive information about the other collaborating devices and establish their own knowledge base. The combination of resources and capabilities can be referred to as services as defined in the Information Technology Infrastructure Library[1]. Resources are the hardware components and the capability is the software running on the hardware that makes it usable. The knowledge discovery involves not only the resources and the capabilities of devices, but also the context and the user-specific information including the social network profiles of the users. When a new service appears in a network, devices should be able to infer its functionalities and start using it autonomously. The information and inferred knowledge of the services should also involve the context of the physical and social environments that can be a distinctive factor in service selection. The discovery protocol should abstract the heterogeneity both in terms of networking and semantics. While satisfying all these requirements, the protocol should accommodate the legacy (operational) service discovery systems.

#### Knowledge Representation

Service identification is a serious problem that cannot be addressed by just standardization of the list of services in the market. Firstly, there is a diverse set of services. It is rather difficult to keep an up-to-date list of the services and distribute it. Even if such a standardization is accomplished by the vendors, each vendor reflects its own categorization. Secondly, users cannot remember all the keywords for an exact naming of the services or user may want to query a service with different

[1] http://www.itil-officialsite.com

categorizations such as purpose. In the scenario presented in Figure 3.1, camera asks for "location" and the peers reply if they have any capabilities for providing location information, such as GPS position, cell id or 3G triangulation. Instead of the exact name match, the purpose of the resources are matched. Lastly, services can alter their inputs, outputs or procedures, knowledge representation should be capable of evolving to adapt such future demands of the services.



#### Personalization for Authorization

Services do not live in a closed world environment where they only interact with trusted parties. There are untrusted peers that access management should eliminate by requiring a trust relationship between the services. Since resources like smartphones have one-to-one relation with their owner, trust between the resources is indeed the trust relation between their owners. In the scenario presented in Figure 3.1, camera and cell phone belongs to Alice while smart phone and navigation device belongs to Bob. The trust between the devices indeed the trust between the owners of the devices. When camera asks for a service which can provide location information, others check the owner of the camera. If a trust relation does not exist between the owners in the social network then service query is not replied. Authorization can be easily addressed by using trust relationships of the owners' themselves, incorporating the personal information in the service representation.

#### Context Dependency

Context information determines the value of the service. Depending on the context, service may become useless or crucial. In the scenario presented in Figure 3.1 for instance, if all the devices were in an indoor environment, the GPS information would be useless since GPS satellite signals do not penetrate indoors. Most reliable information in that case would be the 3G triangulation. Therefore, navigation device and smart phone should not offer their GPS services or the node which uses the GPS service should infer that the information is not reliable. If the service representation involves a piece of information about the constraints of the GPS, machines take better decisions by combining the information with the context.

Ontologies are more expressive data representations than URI like description of the services that are being used in operational service discovery protocols. The above issues can be addressed by the ontologies. In Section 3.2.2, the ontologies will be explained in detail.

#### Network Intrinsic Approach

The smart machine networks are full of heterogeneity of the devices in terms of hardware and software. Differently from [64], today we can claim that network

diversity is being unified in IP based networks. Several researchers call the IP layer as the narrow waist of the hourglass model which is a bridge between the lower and the upper layer. Even if devices operate in different networks or use different application layer protocols, IP layer is the common interface. Therefore, a discovery protocol in the network layer can be supported by many devices. Machines do not have to implement separate application layer protocols which are not used by the service itself. In the scenario, GPS, 3G triangulation services can be provided by different application layer protocols. Devices like camera may use WiFi while phones may use the cellular network. To overcome the heterogeneity in network architectures and to avoid gateways, the discovery protocol should operate at the IP layer.



#### Unified Service Discovery

Currently many systems deploy various service discovery protocols which depend on simple text based matching of the service identifiers. These protocols are not compatible with each other. As a consequence, there are disjoint islands in which only compatible devices exist. And the islands are closed to foreign standards. To enhance the adoption of new service discovery protocols, they should be designed to be compatible with the legacy discovery protocols. In Section 3.5, existing service discovery protocols are reviewed. Most of the protocols transport URI like identifiers and have distributed architectures that do not require a central broker. Unification can be achieved by carrying the URI like identifiers of each standard along with the semantic descriptions.

#### Responsive Discovery

The delay of discovering a service should not disturb the related applications. We expect that mostly the discovery of services happen in the background. A discovery is initiated when a device joins to a network regardless of being required. Such a service discovery does not have time constraints, whereas still some services may be searched on demand. For instance, due to the lack of a local service such as a location service, discovery may start following a user request. In that case, we demand that the discovery process should finish in the order of seconds.

### 3.2.2. Knowledge Representation

In this section, we represent knowledge representation for readers who are not familiar to Semantic Web. Knowledgeable reader may skip this section and proceed to Section 3.3 for the details of SDP.

Semantic Web was introduced in 2001 [90] aiming at making the web machine-

```
@prefix foaf:           <http://xmlns.com/foaf/0.1/> .
@prefix ns1:            <http://en.wikipedia.org/wiki/> .
@prefix ns2:            <http://live.dbpedia.org/resource/> .
@prefix ns3:            <http://live.dbpedia.org/property/> .
@prefix owl:            <http://www.w3.org/2002/07/owl#> .
@prefix ns6:            <http://live.dbpedia.org/ontology/> .
@prefix rdfs:           <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbpedia:        <http://dbpedia.org/resource/> .
@prefix ns13:           <http://dbpedia.org/datatype/> .
@prefix rdf:        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
ns1:Samsung_Galaxy_S_II        foaf:primaryTopic        ns2:Samsung_Galaxy_S_II .
ns2:Samsung_Galaxy_S_II        rdf:type        owl:Thing .
ns2:Samsung_Galaxy_S_II        rdf:type        ns6:Device .
ns2:Samsung_Galaxy_S_II        rdfs:label "Samsung Galaxy S II"@en.
ns2:Samsung_Galaxy_S_II        ns3:imagesize        "200"^^xsd:int;
        ns3:predecessor        ns2:Samsung_Galaxy_S ;
        ns3:successor        ns2:Samsung_Galaxy_S_III ;
        ns6:weight        130 ,130.41 ;
        ns3:display        "AMOLED with 480\u00D7800 pixels"@en,        "800"^^xsd:int ;
        ns3:cpu        "2"^^xsd:int ;
        ns3:memory        "1"^^xsd:int ;
        ns3:input        "Multi-touch screen, headset controls,
                proximity and ambient light sensors,
                3-axis gyroscope, magnetometer, accelerometer,
                aGPS, and stereo FM-radio"@en ;
        ns3:storage        "16"^^xsd:int .
ns2:Samsung_Galaxy_S_II ns3:connectivity "210.0"^^ns13:second ;
        ns3:gpu        "Adreno 220"@en ,  "PowerVR SGX540"@en ,  "ARM Mali-400 MP"@en ;
        ns3:battery        "120000.0"^^ns13:second ;
        ns3:memoryCard        "microSD"@en ;
        ns3:networks        "802"^^xsd:int ,  "Dual band CDMA2000/EV-DO Rev."@en ,
            "HSPA+: 21/42 Mbit/s,  HSUPA: 5.76 Mbit/s LTE 700/1700 Rogers Only"@en ,
              "WiMAX 2.5 to 2.7 GHz;"@en , "UMTS: 850, 900, 1700 , 1900, and 2100 MHz"@en ;
        ns3:soc        "Samsung Exynos 4 Dual 45nm"@en ;
        ns3:rearCamera        "8"^^xsd:int ;
        ns3:frontCamera        "2"^^xsd:int .

<http://example.com/foaf#me> a foaf:Person ;
    foaf:mbox_sha1sum "50a842005e63853ab00d2d46dab152d2e16e92e3" .
```

Figure 3.2: Sample OWL instance document gathered partially from DBpedia. It advertises the device as a smart phone, gives details about the properties of the device with the owner information.

understandable. At present, humans are the only contributors to the Web, we create the web pages and understand the content. Machines are just dealing with the distribution of the content. However, when they become aware of the content, they may adapt to human behavior and needs.

It is crucial to create a vocabulary which machines can comprehend. As a first attempt, the Resource Description Format (RDF) was proposed to represent the

```
PREFIX prop: <http://live.dbpedia.org/property/>
PREFIX ont:  <http://live.dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

select ?phone ?who ?os ?weight
  where
  {
        ?phone    prop:input ?gps;
                     FILTER regex(?gps, ".*a?GPS.*")
                  prop:battery ?battery ;
                  ont:operatingSystem ?os ;
                  ont:weight ?weight .
        ?who      a   foaf:Person ;
                  foaf:mbox_sha1sum
                     "50a842005e63853ab00d2d46dab152d2e16e92e3".
  }
```

Figure 3.3: Sample service query message written in SPARQL vocabulary. GPS property is filtered with a regular expression.



resources in triples: *subject-predicate-object*. To be able to define domain specific vocabularies, RDF Schema (RDFS) was designed and it became possible to describe classes, sub-classes and properties of RDF resources. However, still there was a requirement for defining complex relationships between the objects modeled with RDFS. Therefore, the Web Ontology Language (OWL) was created. As it is seen the abbreviation OWL is not consistent with Web Ontology Language. The reason is that OWL sounds better and it honors the *One World Language* artificial intelligence project at MIT around mid-70s.

With OWL it is possible to construct new classes by simple set operations such as union and intersection. Existential quantifiers *for all* ($\forall$), *there exists* ($\exists$) and even cardinality constraints (such as max or min) become available to describe inter dependence of the classes. After the success of OWL1, OWL2 was accepted in 2009 as a W3C standard which promoted researchers to implement tools for manipulating it. To query data sets, SPARQL query language is used. SPARQL is similar to SQL that is used to fetch data from an RDF data content. Therefore, it is a perfect fit for resource and capability solicitation. Since OWL and SPARQL are widely accepted by the community, we also favor the use of them as the metadata format. However, SDP does not restrict itself in one format, it can carry any semi-structured data format. It is an evolutionary system which adapts to unforeseen future requirements and services.

### OWL and SPARQL

Semantic Web makes the web machine-understandable. Similarly, SDP in smart machine networks makes the resources and capabilities machine-understandable. Advertisement and solicitation messages employ the Semantic Web standards. For instance, the device definition is presented with OWL ontology language in the advertisement messages and solicitation (query) messages are composed of queries expressed in SPARQL.



An example smart advertisement message from a smart phone is depicted in Figure 3.2. The specification is an OWL instance document serialized with Notation3 [13]. The device details are taken from DBpedia but shortened to fit in one column. The owner identifier (digest of the email address) is also given in Friend of a Friend (FOAF) social network language.

In Figure 3.1, the location capability is requested from the neighboring devices. A solicitation message that contains SPARQL query like the one in Figure 3.3 can be sent to discover a device with GPS service. In the query, GPS service that belongs to a specific person is requested by appending the digest of the email address. Since the GPS can be in different formats, a regular expression is used to increase the possibility of a match.

First three challenges mentioned in Section 3.2.1, knowledge representation, personalization and context dependency can be addressed with the use of the ontologies which are described above. With ontologies instead of exact name match, services can be queried with different categorizations. Ontologies can adapt to changes in the definition of the services since they are more expressive than the URIs. Semantic definitions of the services do not require an acceptance from a standards association which can slow down the adoption of a new service or an update in a service. FOAF ontology clearly shows the ability of personalizing the services. The owner information can be embedded into the service definition. Context and hardware information can also be used in ontologies.

## 3.3. The Service Knowledge Distribution Protocol

In Section 3.2.1, we mentioned that SDP is implemented in the network (IP) layer to abstract the heterogeneity. Furthermore, SDP has to work in ad hoc fashion in opportunistic networks.

There are different candidates for a messaging protocol namely HTTPU, SOAP and DNS. In Section 3.5, protocols that use these messaging protocols will be summarized. Key features of the protocols related to service discovery are compared

Table 3.1: Comparison of different carrier protocols for semi-structured data.

| | HTTPU & HTTP | SOAP | DNS | *ICMPv6* |
|---|---|---|---|---|
| Transport Layer | UDP& TCP | UDP | UDP | --- |
| Reliability | With TCP | --- | --- | --- |
| Serialization | Any text | XML | Structured without payload | Structured with payload |
| Header overhead and parsing complexity | Medium | High | Medium | Low |
| Operating Layer | Application Layer | Transport layer | Application Layer | Network Layer |
| Multicast | With UDP | ✓ | ✓ | ✓ |
| Security (Authentication and Integrity) | With SSL | SSL or WS-Sec | DNSSEC | SEND |

in Table 3.1. All the protocols have multicast support without reliability. Security covers only authentication and integrity. Though SSL is mentioned for some protocols, in fact it is not available with UDP and multicasting. ICMPv6 is a better choice in terms of complexity and network intrinsic feature that is required for heterogeneity. Moreover, serialization is flexible with both structured and payload fields. Structured parts (TLV options) are used for security and representing the legacy services, whereas payload is used to carry semi-structured data.

SDP has two types of packets. The Smart Advertisement (SA) and the Smart Solicitation (SS) messages are similar to the Neighbor Advertisement (NA) and the Neighbor Solicitation (NS) messages of the ICMPv6 Neighbor Discovery Protocol (RFC-4861). The NA and NS messages are used to discover the MAC-IP address association of the neighboring devices in a network. SA and SS messages are also ICMPv6 messages which differ from NA and NS in terms of the functionality. Since SA and SS messages operate at the network layer, they are independent from the application layer protocols. They work on any data link layer protocol.

### 3.3.1. Packet Format

The packet format is presented in Figure 3.4, *type*, *code* and *checksum* are the common ICMPv6 fields. *Total* field stores the number of packets that semi-structured data is divided into and *sequence* is used to re-assemble the payload. *Security options* are defined in Secure Neighbor Discovery (RFC 3971), used for authentication

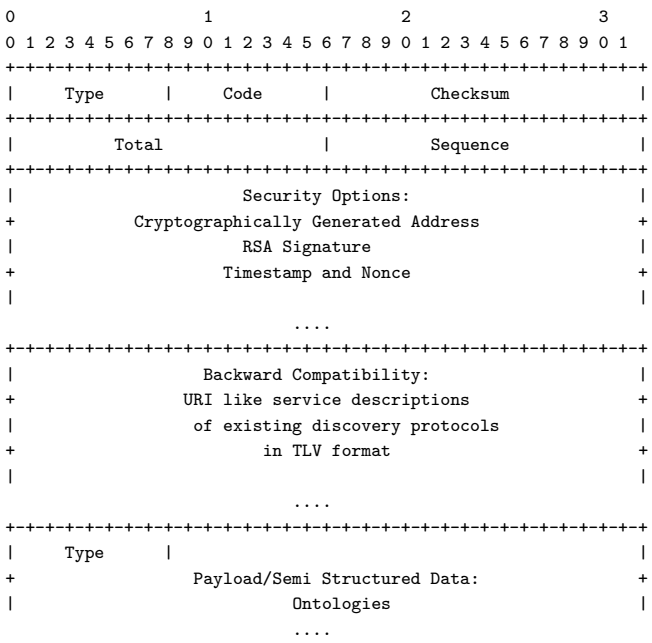



Figure 3.4: Packet format for the Smart Advertisement and the Smart Solicitation messages.

and integrity. The *Backward compatibility* part includes the URI representations of services expressed as in the existing architectures. Lastly, *payload* field involves the ontologies, semantic definition of the services and the device. While parsing the packet to distinguish the payload field from the TLV options, payload starts with a predefined type value but does not have length and value fields.

### 3.3.2. Protocol Operation

The Smart Discovery Protocol operates in ad hoc mode and no central entity is required to distribute the service definitions. Devices send the SA message to declare their existence in the network. The SS message on the other hand aim to query a required service inside the network. Both the SA and SS are multicast messages however, the SA message which is sent as a reply to the SS, is a unicast message.

All the device details, service descriptions and owner profile are placed in a data store, it can be a database or the file system. When a service is added or updated, the data store is also refreshed. A reasoning engine like FaCT++ [92] runs and infers new properties about the services. For instance, a device has GPS hardware and GPS is classified as a location supplier, then the device is considered as it can supply location information. Later on when the device gets a query, it does not have

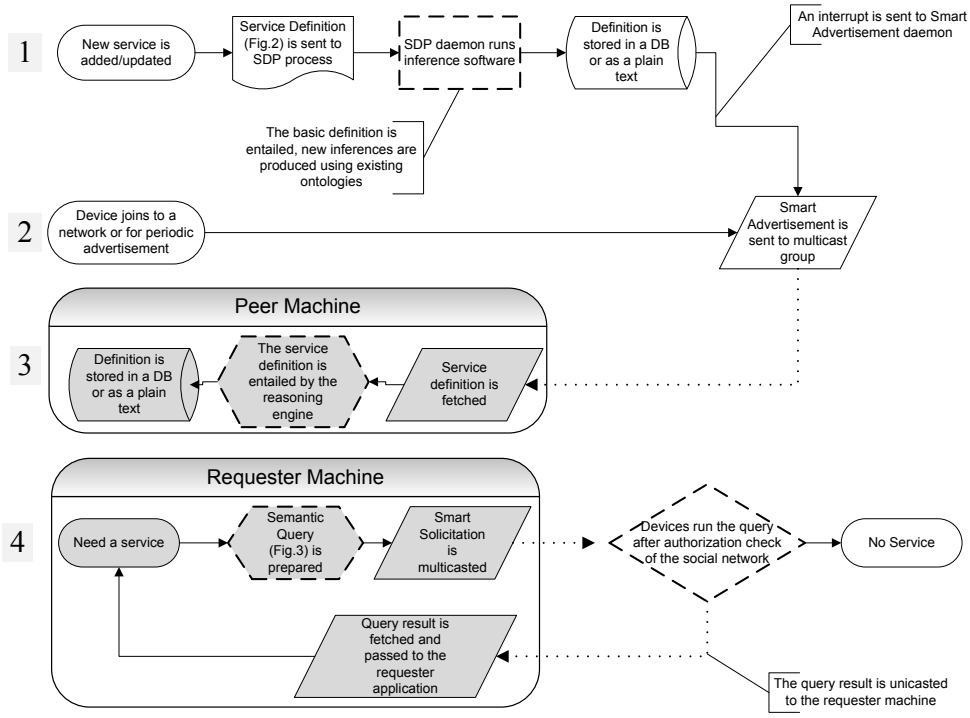



Figure 3.5: The process of the Smart Discovery Protocol

to run the reasoning engine again, therefore it can be concluded that reasoning is an offline procedure. Some devices may lack support or do not have enough computing power to run reasoning software. In such cases the reasoning step can be eliminated. A detailed service description may eliminate the need of inference.

In Figure 3.5, we present the protocol operation. The dashed boxes represent the semantic operations like inferences which do not exist in operational service discovery protocols. An SA message declares the existence of a device and its services in the network. The semantic data describing the resources, capabilities and even the owner details as presented in Figure 3.2 encapsulated in SA are sent when the node joins the network, in case of a change in the information or periodically (presented as the cases 1 and 2 in Figure 3.5). The message is published to a predetermined multicast group address. Peer devices can overhear the SA messages inside the network and create a local database of services for future use (case 3 in Figure 3.5). In need of a service, the local database may be queried first. However, local cache does not guarantee the existence of the service.

To query a functionality, a multicast SS message is published to the group, shown

as the fourth case in Figure 3.5. The payload involves a semantic query in SPARQL vocabulary (such as Figure 3.3). When the devices get the SS message, they run the query in their local semantic data store. Then they reply with unicast SA messages that give the details of the service. The size of the reply message depends on the query, if the details of just one service is required one packet may be enough. It is best practice to prepare a query whose result fits into just one packet.



### 3.3.3. Unified Service Discovery for Legacy and Low Power Devices

In Section 3.2.1, the diversity of the service discovery protocols is mentioned. Devices that use the same service discovery protocol implicitly establish clusters that are disjoint from the devices which adopt another standard. Therefore new service discovery protocols should embrace the legacy standards.

Converting messages of different service discovery protocols to each other is not an easy task. Protocols have diverse set of functionalities which may not have a counterpart in the corresponding protocol. INDISS [17] is an interoperability system for service discovery protocols. In the prototype of the INDISS system, the messages in Simple Service Discovery Protocol (SSDP) and Service Location Protocol (SLP) are converted to each other. The messages are first converted to a intermediary scheme where the messages semantically matched to each other. It is stated that there are still some functionalities that do not match. For instance, SSDP does not have a central entity, whereas SLP employs a directory agent to store all the service definitions inside the network.

Table 3.2: SDP messages and corresponding counterparts of other protocols.

| SDP | SLP | SSDP | DNS-SD |
|---|---|---|---|
| Multicast SA | SAAdvert | ssdp:alive | --- |
| Multicast SS | SrvRqst | ssdp:discover | Query |
| Unicast SA | SrvRply | HTTP/1.1 200 OK | Response |

INDISS divides the service discovery events into three: “Registration Events”, “Discovery Events” and “Advertisement Events”. As seen in Table 3.2, SDP covers only discovery (solicitation) and advertisement events. There are also registration related events such as *SrvReg, SrvDeReg* in SLP which are not covered in SDP. Therefore, SDP cannot be matched exactly with other protocols. However, matched messages can satisfy a distributed service discovery protocol. For instance in SLP

protocol the matched messages are the ones that are used in distributed discovery. Uncovered messages are mainly required for the communication with a central server that is responsible for collecting all the service details and dispatching them from one source.

INDISS like interoperability systems aim to convert the messages to each other without any change in the services. Since semantically matching the standards is not easy and requires excessive effort, in SDP a different approach is taken. As shown in Figure 3.6 simple text representations (sometimes in URI format, eg. SLP, DNS-SD) of the services in different discovery protocols are carried along with ontologies in single SDP message. Devices which prefer using the parsing and service matching APIs of legacy protocols instead of ontology precessing, keep consuming these simple representations. Both sides again supports SDP messages in the their operating system. The initiator embeds the URI representation inside the packet with semantic representation. The receiver omits the semantic representation and only fetches the URI representation that it supports. The URI services are matched by using existing APIs. By combining all the protocols in one message, less packets are transmitted in total. Legacy devices are supported with a small software update which only parses the SDP message and dispatches the messages to corresponding protocols. Moreover, INDISS like systems can incorporate SDP in their design to decrease the message traffic in the network.

One question still remains open is how we will embed the different messages into one SDP message which can be parsed easily. Most of the service discovery protocols summarized in Section 3.5 such as SSDP, SLP and DNS-SD, distinguish the services by URI like texts and mainly text based matching is performed. The URIs are mostly short texts, for instance DNS-SD TXT records are intended to be around 200 bytes or less. In order to embed the legacy service messages, type-length-value (TLV) options of the ICMPv6 are used. New TLV options are created for each text based service discovery protocol. The size of these options are determined by the length field which is an 8-bit unsigned integer in units of 8 octets and results in $2^8 \times 64 = 16Kb = 2KB$. After parsing the TLV options they are passed to their protocol's daemon.

Another motivation for these TLV options is that they are easier to parse for low power devices. Other semantic oriented service discovery protocols exclude such devices from their design. Low power devices like sensor nodes can easily parse TLV options and fetch the URI identifiers of the services expressed in different discovery protocols. In a single message both computationally high and low powerful devices are addressed which enhances the adoption of the protocol, and makes the transition to semantic technologies smoother and easier.

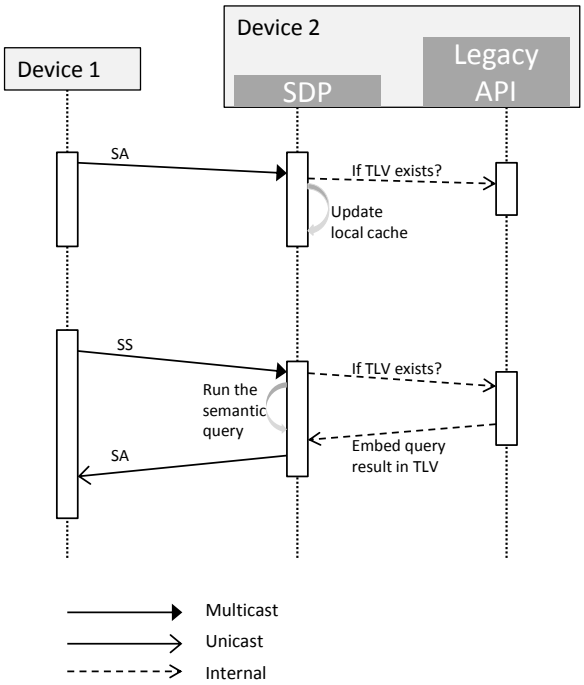



Figure 3.6: SDP messages are given in the sequence diagram. If there is a TLV that carries a legacy service identifier, the legacy service API is called.

### 3.3.4. Reliability

SDP is a distribution protocol for the ontologies and operates at the IP layer where end-to-end reliability is not guaranteed. For scalability multicast messages are preferred in all the discovery protocols. However, reliability in multicast messages requires huge overhead on the transport layer, basically reliability is traded for scalability. There are experimental protocols like the Pragmatic General Multicast reliable transport protocol (RFC 3208), which try to maintain reliability with negative acknowledgement (NACK) packets. NACK packets can reduce the traffic but still the loss of the NACK packet is a problem.

In SDP, like other service discovery protocols, periodic retransmissions are employed to enhance the reception rate but still reception is not guaranteed. Both the SA and the SS messages are retransmitted with increasing intervals in the order of two and after some number of retransmission process ends. The duration of the interval and the number of retransmissions depend on the medium and background traffic. In a medium with low bit error rate small number of retransmissions are enough.

Another issue is the replies to the SS messages. Since the nodes in the network get the SS message at the same time, their replies may collide. Although the data link layer and MAC protocols avoid collisions, it is still preferred to send the replies after a random back-off time.

#### Fragmentation and Traffic Shaping

In Figure 3.4, it is seen that there are "total" and "sequence" fields which are used to reassemble the fragmented message. For fragmentation another option is IPv6 fragmentation header. However, if IPv6 fragmentation were used, every retransmission message would be considered as a different message by the stack. When one of the packets of a message drops, IPv6 fragmentation removes the whole message and partial message is not passed to the SDP. By depending on our own fragmentation method, in retransmissions the peer can reassemble a message by gathering packets from different transmissions.

Additionally, SDP employs traffic shaping to decrease the congestion. Especially in wireless networks like 802.11x, multicast messages can easily incur congestion due to limited bandwidth reserved for them. Therefore, SDP puts time delays between the consequent packets to decrease the congestion. The performance increase employing the traffic shaping is presented in Section 3.4.

### 3.3.5. Authentication and Integrity

In service discovery, adversaries can inject fake services or alter the definitions of the existing services. Therefore, depending on the requirements of the network, authentication and integrity checks should be devised. As summarized in Table 3.1, all the carrier protocols aim to maintain the authentication and integrity to establish trust and prevent the denial of service attacks which can easily be done by advertising non-existing services. SDP also ensures the authentication and integrity with Secure Neighbor Discovery (SEND, RFC 3971). Since SDP is an add-on to NDP and SEND is designed to secure NDP, SDP is also covered with SEND.

SEND introduces four new options to neighbor discovery protocol:

- **Cryptographically Generated Address:** Used to guarantee that address-owner association is valid with the asymmetric key encryption.
- **RSA Signature:** The digest of the packet is signed by the private key of the source. The signed digest authenticates the source and guarantee that no other node can alter the packet, any change on the packet data is detected by the peers.
- **Timestamp:** Timestamp option is employed to prevent the replay attacks.

- **Nonce:** In the solicitation-advertisement message pairs, randomly generated nonce values are used for association.

### 3.3.6. Confidentiality

Though authentication and integrity is offered by many service discovery protocols, confidentiality is not considered in protocols that employ multicast messages. Any node inside a network can analyze and trace each device with its offered services. In a trusted network such as office and home environment, the risk may be low. However, in an open network malicious nodes can overhear the discovery packets and easily build an inventory of network.

In open networks, it is advised to depend on a secure overlay network such as virtual private networks (IPSec). With IPSec all the traffic is secured. However, it is a complex protocol which requires detailed pre-configuration and has bootstrapping problems. Service discovery protocols are designed to be simple and have less overhead, therefore security should also be addressed without heavy protocols. All in all, we also omit confidentiality like other protocols and consider service discovery as a case study for the researchers working on multicast group security.

## 3.4. Performance Evaluation of SDP

The multicast nature of the all the discovery protocols trades off the reliability of the messages. SDP also operates on unreliable multicast ICMPv6 packets, which incur drops due to congestion and noise in the wireless medium. Compared to other service discovery protocols that operate at higher layers, obviously SDP imposes lesser load in terms of packet headers. However, the metadata payload carried over SDP significantly increases the load. We deal with increased load and its consequence, packet drops by the traffic shaping method defined in Section 3.3.4. The web service based systems like DPWS are expected to carry XML metadata which is close to the size of the metadata of SDP. On the other hand protocols that carry just the URI definitions of the services like DNS-SD and SLP, have lesser load.

Despite of the high load incurred by the SDP on the network with respect to simpler service discovery protocols, the traffic is still less than the load of a web page. For instance, the size of the HTML page in the URI *google.com* is around 11 KB whereas the metadata in Figure 3.2 is 1.8 KB.

SDP differs from the other protocols with its design in dealing with heterogeneity and unified discovery. However, still the performance and especially reliability of the protocol should be assessed.

### 3.4.1. Experiments on Real Hardware

In order to assess the performance of the protocol and especially to observe the reliability, the protocol is implemented in Ruby programming language which is served as an open-source project[2]. The ICMPv6 structure described in Section 3.2 is created with *type* value assigned to 200/201 which are reserved for private experimentation in the ICMPv6 standard. Different metadata sizes are used in the experiments. For the experiment environment "*Eduroam*", the largest WiFi network of TUDELFT is used. The network represents a crowded office environment and the experiments are carried out between 13:00 and 17:00 while the network is active. The signal levels of the devices that we experimented vary between -80 dBm and -40 dBm; the bit-rates vary between 18 Mb/s and 54 Mb/s depending on the location. The structure of the experiments is as follows: There are one sender and four receiver laptops. Sender and receivers are being served by different access points. Three of the receivers operate in 802.11a network and the rest is in 802.11g network. The sender laptop either announces its service details or queries for a service and waits for the reply. The latency of the announcement and query-response packets are given as a metric for performance.



We start with experiments on the latency of advertisement and solicitation messages with their responses. Then, we compare SDP against SLP and show that SDP does not lead to extra latency while unifying legacy discovery protocols. Lastly, we demonstrate the performance of the traffic shaping.

#### Results on Message Latencies

First, we experimented the latency of the advertisement messages. The metadata of the services with different sizes are fragmented into SA packets and sent as a batch 100 times to different number of machines. At the receiver side the duration from the first message till the last one is presented in Figure 3.7. As the metadata size increases, more packets are transmitted leading to an increase in the duration. Multicasting the traffic provides scalability as the receivers incur similar latencies.

Second, we experimented the round trip delay of solicitation messages and their reply including the semantic data parsing. The solicitation message in Figure 3.3 is sent to the peers. The peers run the query and the result is sent back with unicast messages. Both the solicitation and its reply fit into one ICMPv6 packet. In Figure 3.8, the time from the start of the solicitation message till the reception of the reply is presented. Even with the semantic data parsing the round trip time is lower than a half second. Unfortunately, there exists variance which is due to the channel conditions.

[2] github.com/yunus/SDP

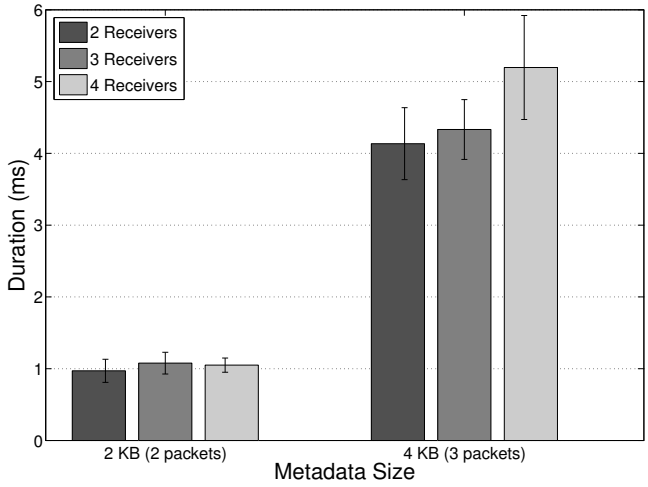



Figure 3.7: The durations between the reception of the first message till the last one with different metadata sizes are presented. The MTU is 1500 bytes.

### Comparison of SDP to SLP

One of the novel features of SDP is that it is capable of carrying legacy service messages. The initiator inserts the URI identifier of the legacy protocol in a TLV field of SDP. As shown in Figure 3.6, the receiver parses the TLV, uses the APIs of an existing library of the legacy protocol to parse the identifier. If the message is a solicitation, legacy API checks whether the request matches the service or not. If the service is matched in the reply whole service definition is embedded in another TLV field of SDP.

In order to validate our design, we compared our implementation with SLP. The jSLP[3] library is used to send SLP service request and service reply messages. In SDP for parsing and matching the service identifier again the jSLP API is used. In both protocols a service is queried with the identifier "service:test" and the reply contains "service:test:myService://my.host.com". Figure 3.9 shows the duration in solicitation messages similar to Figure 3.8. In this test differently from the previous one, SDP makes an additional API call to the jSLP while also performing a semantic query. When Figure 3.9 and Figure 3.8 are compared, we observe that calling an external library does not lead to an increase in the duration. Moreover, both protocols, SLP and SDP perform similarly. The SDP protocol does not impose higher latencies than SLP, which can lead to timing issues on the legacy services.

[3]http://jslp.sourceforge.net/

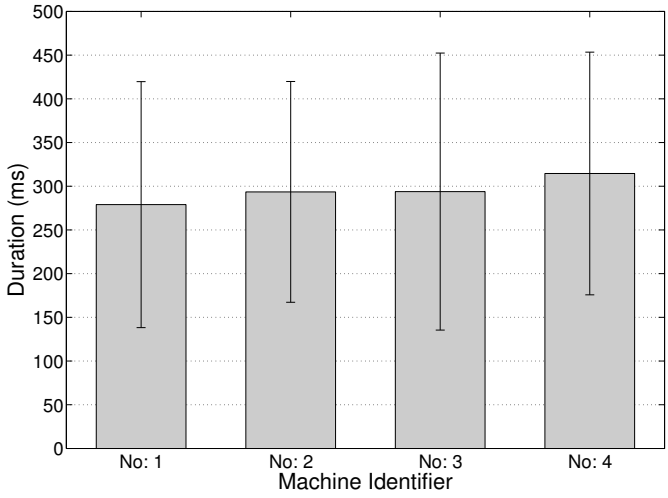



Figure 3.8: Duration from the start of a solicitation message until its reply arriving to the initiator.

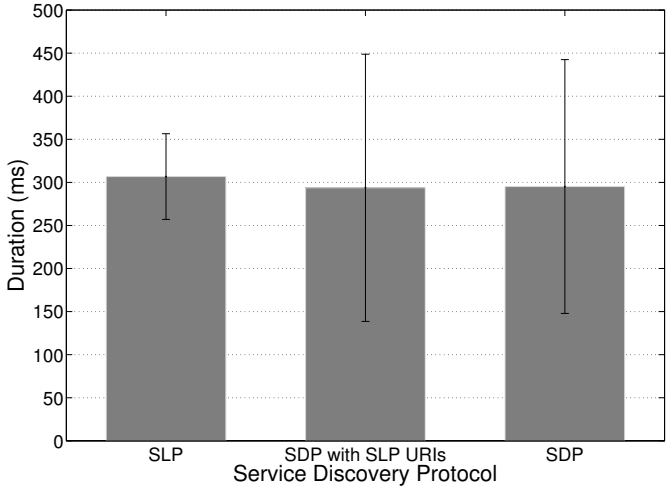



Figure 3.9: Comparison of SLP solicitation messages to SLP URIs carried in SDP.

### Results on the traffic shaping feature

To observe the effect of message size on the packet reception rate, 2000 messages are transmitted and their reception rates are presented. Half of the messages are composed of just one packet and the other half is composed of four packets. As seen in Figure 3.10, the increase in the number of packets decreases the rate of the

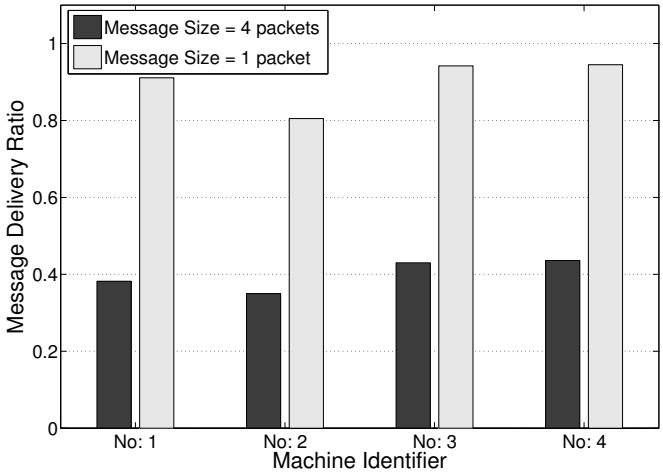



Figure 3.10: Successful arrival ratio of the advertisement messages in different machines with different metadata sizes.

successful message reception. It is also observed that the second machine has higher drop rate with respect to others. This shows that the retransmission threshold should be determined by considering different devices and the noise levels that they encounter.

Until now traffic shaping is not employed in the experiments. As mentioned in Section 3.3.4, traffic shaping is used to avoid congestion. Fragmented parts of a message are transmitted by inserting sleep intervals among them. Figure 3.11 presents the effect of the traffic shaping on the dropped messages. Among the partially arrived messages, packets are grouped according to their sequence ids. Without traffic shaping, as the id increases the reception rate decreases. When traffic shaping is employed the reception rate becomes the same for all. Moreover, the overall successful transmission rate increases from 0.4 to 0.5. Exceptionally, in this example the reception rate for the last packet is larger than the others even if the traffic shaping is enabled. The reason is that last packet is smaller than the others. Other packets use the whole MTU (1500 bytes). However, the last packet carries the rest of the message which is 800 bytes and small packet effected from the noise less than the longer ones.

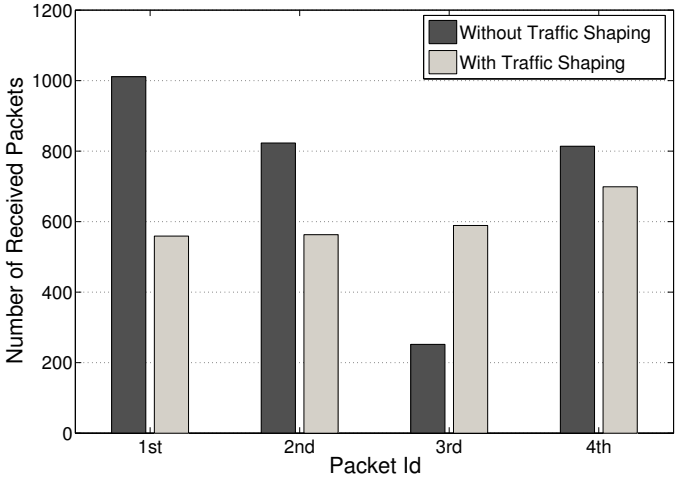



Figure 3.11: A message is composed of four packets. Among the dropped messages, the number of received packets with respect to their ids are given. The last packet has smaller size and hence its success rate is higher.

## 3.5. Existing Architectures

Service discovery is an active research area started in 1990s. There are many proposals up to now. In this section, we present some of the well-known and widely-supported service discovery protocols and compare them with SDP.

### 3.5.1. Service Location Protocol

Service Location Protocol (SLP) is a standard defined in RFC 2608 and RFC 3224. The main communication protocol is UDP multicasting. Unicast TCP messages are also used for long messages to improve the reliability. The messages are in URI formats like "service:printer-detector.1234://example.com:8080" which requires exact matching. Three different roles exist:

- *User Agent (UA)*: looks for the services,
- *Service Agent (SA)*: provides the service and announce it,
- *Directory Agent (DA)*: stores the list of services to solve scalability issues. It is optional.

In the absence of the DA, the system works in a distributed fashion. Otherwise, the service announcements are cached by DA and again the search queries are replied by DA.

### 3.5.2. Universal Plug and Play

Universal Plug and Play[4] (UPnP) is a set of network protocols which aims at seamless discovery and control of devices without human intervention. Leading companies in the electronics industry support the research on UPnP and some end products have already been commercialized. The devices such as computers, network printers, smart phones, televisions discover each other when they are attached to the same network. Then, they can exchange data and configuration parameters. It should be also noted that UPnP is more than just a service discovery protocol, it is an architecture in which pervasive devices control and exchange data among each other in a peer-to-peer way.

Simple Service Discovery Protocol (SSDP) [48] is an outdated IETF Draft but adopted by UPnP community as the service discovery protocol. Similar to SLP, it is based on multicast search messages. However, the protocol used for transportation is HTTPU (HTTP over UDP). UDP is preferred for HTTP transmission to reduce the overhead of TCP signaling and to use multicast instead of unicast. SSDP client multicasts an HTTPU discovery message to a predefined multicast channel. The services which listen to the channel replies with unicast HTTPU messages when the queried service matches. Apart from this request-response scheme, services can also announce their presence when they first join into the network.

Unique Service Names (USN) are URIs which uniquely define the services. USNs are used to handle the change of the point of attachment of the services in the network. An example of request and response message from [48] is given in Figure 3.12. As it is seen, it is an HTTP message whose payload involves some predefined key, value pairs like “Host”.

### 3.5.3. Device Profiles for Web Services

Device Profiles for Web Services (DPWS) [71], proposed by Microsoft, is similar to UPnP, designed as a plug-and-play architecture which involves discovery, control, and eventing of the services. Differently from UPnP, every service is considered as a web service and therefore all the standards depends on web services [71]: WSDL 1.1, XML Schema, SOAP 1.2, WS-Addressing, and further comprises WS-MetadataExchange, WS-Transfer, WS-Policy, WS-Security, WS-Discovery and WS-Eventing.

WS-discovery [72] is the service discovery protocol used in DPWS. SOAP over UDP is chosen as the transport protocol and messages are multicast to enable ad hoc mode of operation. Besides the ad hoc mode of operation, there is a managed

[4] http://www.upnp.org/

```
M-SEARCH * HTTP/1.1
S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6
Host: 239.255.255.250:reservedSSDPport
Man: "ssdp:discover"
ST: ge:fridge
MX: 3

----------------

HTTP/1.1 200 OK
S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6
Ext:
Cache-Control: no-cache="Ext", max-age = 5000
ST: ge:fridge
USN: uuid:abcdefgh-7dec-11d0-a765-00a0c91e6bf6
AL: <blender:ixl><http://foo/bar>
```

Figure 3.12: Example of SSDP request and reply messages.



mode in which a centralized proxy exists to coordinate the traffic. There is also a dynamic mode which combines both ad hoc and managed schemes. Centralized mode is mainly proposed to reduce the multicast traffic load in the network.

### 3.5.4. Zero-Configuration Networking

Zero-Configuration Networking[5] (Zeroconf) uses Multicast DNS/DNS-SD (IETF Draft standard) for service discovery. Multicast DNS enables well-known Domain Name System (DNS) application without the existence of a central server. Devices can query the services using the multicast messages. Similar to the other service discovery protocols, a standardized set of URIs are used to identify the services. Mainly, it is being supported by Apple Inc. in the name of Bonjour[6] with Apache 2.0 License and as another open source implementation for Linux and BSD machines in the name of Avahi[7].

DSN-SD (DNS-Service Discovery) uses different record schemes for identifying and configuring the services. The Service (SRV) (RFC-2782) records are in the form of "*Instance.Service.Domain*" and defines the target host-port pair of the service address. The DNS Text (TXT) records are "*key=value*" pairs and are used to provide additional information about the service such as the queue name of a printing machine. DNS Pointer (PTR) (RFC 1035) records are used in the form of "*Service.Domain*" to discover available instances of a service.

[5] http://www.zeroconf.org/
[6] https://developer.apple.com/opensource/
[7] http://avahi.org/

### 3.5.5. Semantic Service Discovery

All of the above protocols employ text-based matching which can be considered as a problem. As mentioned in Section 3.2.1 and in Section 3.2.2, ontologies are better options for representation. Multicast Resource Discovery Protocol (mRDP) [95] is a semantic service discovery protocol. OWL is used as the message format. Only solicitation messages are allowed, advertisement is not considered. The solicitation messages are over multicast HTTPU packets for scalability and the responses are in unicast HTTP packets to guarantee the delivery. Although semantic data is carried as in SDP, the choice of the transport protocol differs significantly. In SDP, instead of heavy protocols like HTTP, ICMPv6 is used that operates on the network layer and low power devices are addressed with legacy support. SDP offers authentication, integrity via SEND. Advertisement messages are supported which helps caching the services and decrease the response time.

UPnP architecture may also be mixed with the semantic languages [91]. UPnP messages can be converted to ontologies on which other devices may infer. The ontology created in [91] may be used in the SDP since SDP does not restrict itself in one language.

INDISS [17] interoperability system and in general the Amigo[8] project uses OWL-S[9] (Semantic markup for web services). In another example, home device interoperability is improved by using ontologies on top of SOA based service discovery protocols such as UPnP and DPWS [29].

There are some works in pervasive and sensor network environments. The proposed service discovery scheme in [100] tries to convert the natural language queries to machine understandable ones for the sensor like small devices. In [81], service discovery algorithms for pervasive environments are proposed which aim to guide service discovery with the context information or personal preferences of the users.

When we move to the web domain, there are many works that concentrate on semantic web services. Researchers try to develop new algorithms for better matching of the services for the requirements of the users. In [76], an ontology framework which categorizes the services according to their functions is proposed. S-MatchMaker [58] improves service discovery by involving quality of the services in the selection process. Semantic web services research supports our research with the tools that are used in the matching of the web services. Most of these algorithms can be incorporated in our work as a back-end system for service matching.

[8] www.amigo-project.org
[9] http://www.w3.org/Submission/OWL-S/

Table 3.3: Comparison of different service discovery protocols.

| | SLP | SSDP | DNS-SD | WS-Discovery | mRDP | ***SDP*** |
|---|---|---|---|---|---|---|
| Carrier Protocol | UDP & TCP | HTTPU | DNS | SOAP& UDP | HTTPU& HTTP | ICMPv6 |
| Operating in Heterogeneous Networks | --- | --- | --- | --- | --- | ✓ |
| Ad hoc mode | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Service Description | Text Based (URI) | Text Based (USN) | Text Based (DNS-SRV) | Structured Text (XML) | Ontology (OWL) | Ontology (OWL) & Text Based (ALL) |
| Evolves for future requirements | --- | --- | --- | --- | ✓ | ✓ |
| Unified Discovery | --- | --- | --- | --- | --- | ✓ |
| Inference | --- | --- | --- | --- | ✓ | ✓ |
| Advertisement Message | SAAdvert | ssdp:alive | --- | Hello | --- | Multicast SA |
| Resource Information | --- | --- | --- | --- | --- | ✓ |
| Context Information | --- | --- | --- | --- | --- | ✓ |
| Personalized | --- | --- | --- | --- | --- | ✓ |
| Reliability | Long messages in TCP | Retransmission | Retransmission | Retransmission | Reply with HTTP (TCP) | Retransmission |
| Authentication and Integrity | --- | --- | DNSSEC | SSL or WS-Sec | Partial SSL | SEND (IPv6) |
| Confidentiality | --- | --- | --- | --- | --- | --- |

### 3.5.6. Comparison with SDP

Many organizations have proposed state-of-the-art protocols like UPnP, DPWS and Bonjour which target service discovery satisfying specific sets of requirements making those protocols better than others in one way or another. Due to the prominent and distinctive features of these protocols and with the competitive support of the companies behind them, there is still no dominant service discovery mechanism. As a result there are disconnected islands of devices that can only interact with compatible ones belonging to the same vendor. Another main incompetency of the present protocols is their inability to infer beyond the shared pieces of service definitions. The operational service discovery architectures are based on simple text matching of the service descriptions. Generally, a URI ( e.g., *service:printer-detector.1234://example.com:8080*) is published in a network. Other devices that can look up and match the text are able to recognize and consume the service (the printer in this example). When a new type of service is developed, the standardization bodies must come up with a new URI that define the service, and all the machines should upgrade their data stores. While existing services evolve; new services appear everyday. That is why the devices should embrace the change by inferring the meaning of a service by themselves.

Our proposal, SDP, is a service discovery protocol that carry semantic representations of the resources and the capabilities of the devices which *also include the owner information*. We compare SDP with other service discovery platforms in Table 3.3, but we should note that UPnP and DPWS like protocols involve many other features apart from discovery. Therefore, we do not claim that SDP outperforms all other protocols, it can only provide ideas about the incompetencies of other protocols.

As shown in the table, firstly, all the protocols support ad hoc mode operation and use multicast messages for scalability. Ad hoc mode is crucial for the networks without a central authority. Secondly, recent proposals on service discovery have a tendency on employing more expressive representations of the services like XML in WS-Discovery and ontologies in mRDP. Many researchers agree on the expressive capabilities of ontologies and the importance of such intelligent architectures that will minimize human intervention. SDP also motivates the use of semantics. *However, main contribution of SDP is the distribution protocol which operates on the network layer to eliminate heterogeneity.* SDP can crawl in different network architectures provided that they use IP as the network layer. Fortunately, IP is becoming a convergence point for many different network architectures.

Finally, another contribution of SDP is that it combines different service discovery protocols by carrying their service identifiers together. The TLV fields of

ICMPv6 packets can store the identifiers, and when the peer network stack gets the identifier it pushes the identifier to the original handler of the standard. However, in the long run with the involvement of vendors, service representations can converge to ontologies.

## 3.6. Conclusions



Devices are not *swiss army knives* that incorporate all the required functionality in one item. They need to cooperate and share their functionalities with others. As a first step towards the cooperative networks of devices, devices should discover each others' services. Operational service discovery protocols that we use today do not adapt to future requirements. They use simple text-based representations of the services instead of more expressive ontologies that allow inferencing on gathered information. Moreover, existing service discovery protocols create their own islands where only devices from the same vendor can participate in communication.

The Smart Discovery Protocol proposed in this chapter is a semantic service knowledge discovery protocol that operates at the network layer. Being embedded in the operating system, SDP is independent of the application layer protocols and the communication interfaces. Inspired by the Neighbor Discovery Protocol, ICMPv6 messages carry service definitions and service queries. The resources and capabilities are carried together with context and owner information. The underlying protocol does not provide reliability, but we showed that rate limitation can be incorporated improve reliability.

Semantic Web tools and vocabularies like OWL and SPARQL are used to describe the services. The URIs used in existing service discovery protocols like SLP are embedded in TLV options of the ICMPv6 packets. In this way, several discovery messages are unified in one message. Moreover, while other service discovery protocols do not include owner information, SDP can include URIs that points to social profiles of devices with owner information. In this way it is possible to create a cooperative network of social devices that discover each other's services without human intervention.

SDP still requires extensive testing in heterogeneous networks to determine the parameters like traffic shaping and number of retransmissions. SDP works in ad hoc mode. However, many other discovery protocols employ optional centralized servers that store all the service details, to decrease the number of messages sent in the network. Although we do not expect any issues in home networks, in enterprise networks a storage server may be required for scalability. In such a case a central scheme should be developed. Fortunately, ICMPv6 packets are parsed by

the routers. Therefore, routers can act as storage servers.

# 4

# Secure-by-Default IoT via Decentralized Social-Device Networks

In this chapter, we elaborate our vision of "devices that know their owners", which is described in Chapter 1 and create decentralized social-device networks (DSDN). A social-device cooperates with other devices that exist inside the trusted social network of its owner(s). By automating cooperation and access control over social devices, we create cooperative as well as secure-by-default IoT systems. After describing DSDN, we identify its core challenge of decentralized social network search, whose complexity is quadratic for indirect relations. We solve this search problem in both computationally unconstrained and constrained environments. In unconstrained environments, we limit the search space by incorporating proximity information. As an example application we deploy DSDN on an access point by modifying the open-source Hostapd project and EAP-TLS standard where the access point captures WiFi beacons for proximity detection. For constrained environments, we propose a delegation-based architecture, in which a server with more computational power searches social networks instead of a constrained device. We modify the Datagram Transport Layer Security (DTLS) standard, which is designed for constrained devices, to incorporate delegation.

## 4.1. Introduction

In the next decade, humans will be outnumbered by pervasive devices, ranging from highly capable smartphones to constrained sensors and actuators [4]. Due to the abundance of devices and also the hardware constraints (e.g., lack of a display and I/O interfaces), consumers skip complicated deployment procedures. They tend to employ default security and privacy settings, which eventually leads to security and privacy vulnerabilities. Research by HP indicates that 80% of the most common IoT devices fail to require strong passwords and account enumeration reveals 70% of the accounts [45]. Moreover, exploitations have already started as demonstrated by the 2014 incident of the public streaming of many residential webcams by adversaries [35].



The use of insecure or default passwords may be eliminated via educating the users. Nevertheless, scale (i.e., abundance of devices) will remain a significant challenge for the industry as well as for the consumer market. Scalability may be sustained to some extent by creating tools that keep track of passwords or introducing authorization servers like the standardization efforts of the IETF-ACE (Authentication and Authorization for Constrained Environments) group. However, for a complete solution, we should focus on the behavior of users. Consumers share their resources, such as Internet connection, with their real-life social networks (e.g., one's family or friends) while in industry, device access is restricted based on role information.

In this chapter, we address the abundance of devices that lead to management problems by introducing **autonomy** and we eliminate the complicated procedures by creating a **secure-by-default** system, that is a secure system by default settings. We argue that an autonomous and secure-by-default system is possible if devices are social in the sense that they recognize their owners and owners' social networks. Then, devices can take access control decisions on behalf of their owners. For instance, HVAC sensors spread in a house could share their readings with the Nest thermostat in the living room by inferring that both of them are owned by the same family.

We envision that the only step that the owner of a resource server (e.g., NEST thermostat) should do is declaring her ownership–even with the devices that have no user interface. That is, unlike other imprinting techniques where a symmetric key is installed into a resource server either via physical contact or wirelessly in a faraday cage [56], the owner does not interact with the resource server. Instead, the ownership is declared in the Internet. An example scenario for a smooth human-computer interaction is as follows: The resource owner will declare the ownership


Figure 4.1: The resource owner declares the ownership just by scanning a QR code or RFID/NFC tag that contains the security credentials to update the social profiles of the devices. No other action is required by the resource owner for securing the deployment.



just by scanning a **QR code** or **RFID/NFC tag** on the resource server, which contains a link to the social profile of that server with its security credentials. As shown in Figure 4.1, a smartphone will do the scanning. The app will access the server's social profile and embed the ownership relation. *No other action is required by the owner, the rest is* ***automated***. The access point of the residential network will recognize the resource server with its owner and grant network access. Later, the server will authenticate and authorize resource requests based on its owner's social network (i.e., the relation between the resource owner and clients).

Several researches have claimed that all such authentication and authorization can be automated by IoT devices if they had the notion of a social network that involves role information as well [9, 40, 96]. Being centralized, where a server acts as a proxy, leads to drawbacks such as scalability, single point of failure, offline (without Internet connection) operation and especially privacy in these architectures. We argue that decentralized architectures are better in avoiding these drawbacks. Without end-to-end security, proxy-like entities can eavesdrop the communication and violate privacy. Moreover, compared to today's centralized social network platforms like Google+, Facebook and Twitter, a social network of devices will have a similar scale–may be even more since each person can have multiple devices. Therefore, a centralized approach demands huge investments for its infrastructure. On the other hand, decentralization distributes the cost. Finally, centralized architectures inhibit interoperability and lead to vendor lock-in, which we should avoid to improve the utilization of constrained devices by increasing their interactions.

In this chapter, we advance the state-of-the-art by abandoning the centralized solution, we propose **decentralized social-device networks (DSDN)** to auto-

mate device accesses. As shown in Figure 4.2, each device is represented by a URI that points to a social profile of that device. The ownership information is embedded into the social profile. Authentication and authorization are achieved via the WebID-TLS standard [86], which grants access based on social relations between the owners of devices. The drawback of our decentralized approach is its computational overhead, especially while discovering the social relations that are distributed in the Internet. After we identify the problem, we solve it with a context-aware approach in unconstrained environments (i.e., networks of computationally powerful devices such as smartphones, access points, and laptops) and by employing a delegation-based architecture in the constrained case (i.e., networks of computationally weak devices such as embedded systems, see Section 4.6 for more detail).



**DSDN in unconstrained environments.** As a case study for unconstrained devices, we have created a social access point (Social-AP) that grants network access over DSDN. Social-AP discovers the owner of an authenticating device, such as a smartphone, by checking its social profile. Social-AP has its own social profile, and hence, knows its owner as well. If both owners are socially connected and access control rules permit, the smartphone is allowed to join the network. Here the challenge is that, when the smartphone and Social-AP are not related directly, for instance their owners have a common friend, the search complexity becomes quadratic and the whole process lasts minutes. We show that the simple heuristic of limiting the search to friends and devices in physical proximity makes for a scalable solution. After all, it is highly unusual for people to grant access to random acquaintances if not accompanied by a "known" mutual friend. This intuition is corroborated in [5], where it is shown that the likelihood of having a social connection to a person is inversely proportional to the actual distance to that person. We track regular WiFi probe requests to determine which devices are in the vicinity, and sort them according to time of arrival, checking the most recent ones first. This strategy was experimentally verified to greatly reduce the search time; for a social network of neighbor degree 4, the worst case performance of an indirect friend search was decreased from 1 minute to a mere 11 seconds.

**DSDN in constrained environments.** IoT also includes constrained devices that cannot perform the social profile search at all (See Section 4.6). Therefore, we enhance our DSDN proposal to constrained devices and constrained-device networks as well. Constrained devices delegate the search to an authorization server. To realize the architecture, we have modified the Datagram Transport Layer Security (DTLS) standard, which replaces TLS in constrained environments, for DSDN and delegation. Our real-life experiments indicate that the communication overhead of

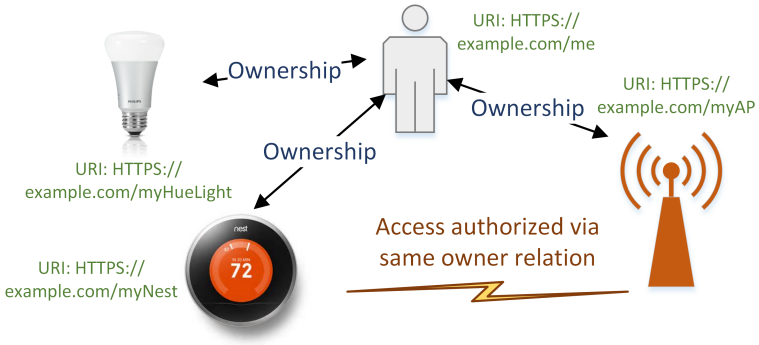



Figure 4.2: All devices have a WebID pointing to their social profiles, which store ownership information. Devices allow access to each other based on the ownership information.



the delegation process is as low as 7%, while the amortized cost of added latency is less than a second.

The rest of the chapter starts with a background information. Section 4.3 describes DSDN and identifies the complexity of social network search. Section 4.4 and 4.6 present our solutions for social network search, which are followed by sections for real-life evaluations in unconstrained and constrained environments, respectively. Section 4.8 analyses the security threads. Section 4.9 investigates the state of the art and positions DSDN. Finally, Section 4.10 concludes the chapter.

## 4.2. Background

Before describing DSDN and our improvements for constrained and unconstrained environments, we would like to explain the Datagram-TLS (DTLS) and WebID-TLS protocols since they are at the core of our proposal. Moreover, we give some information about WiFi probe requests, which we employ for context-aware search optimization.

### 4.2.1. DTLS for End-to-End Secure Channel in Constrained Environments

Datagram Transport Layer Security (DTLS) protocol secures the channel between two end points regardless of the number of intermediate connections. Both peers determine a common secret key to encrypt the application layer traffic by exchanging plain-text messages over an insecure channel. All this message traffic is called a DTLS handshake. We present a DTLS handshake with certificate exchanges in Figure 4.3. Messages are grouped, which are called *flights*. In case of a message

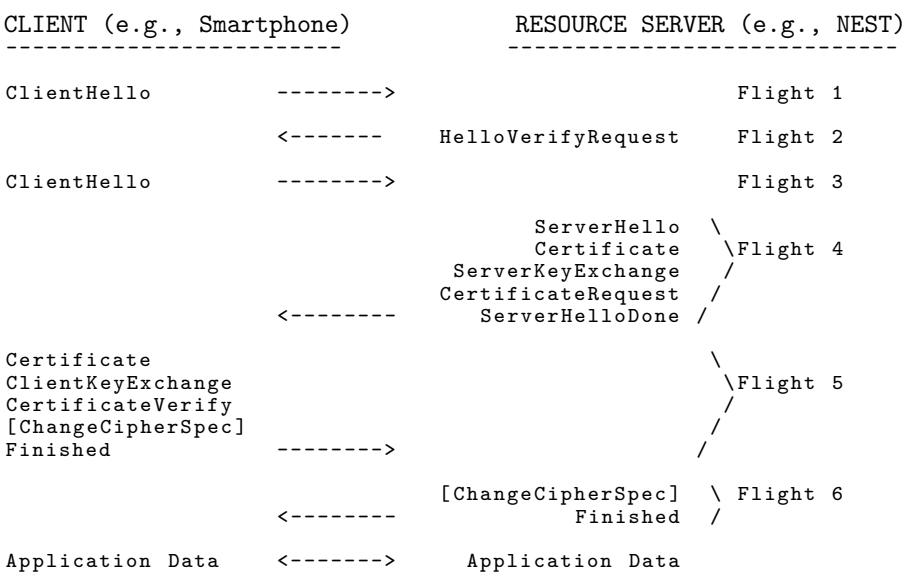



Figure 4.3: A DTLS handshake with certificate exchanges. At the end of flight 6, both parties share a secret symmetric key to encrypt the application layer traffic. In DTLS-PSK (Pre-shared key) mode, certificate related messages are omitted.

lost, the whole flight of that message is retransmitted. The handshake is initiated by the client via the `ClientHello` message. At the end of the Flight 5, both peers possess the same secret key to encrypt the application traffic. `ChangeCipherSpec` packet signals the start of the encrypted traffic.

The given handshake example is certificate based and requires more messages than DTLS-PSK (Pre-shared Key) mode. In addition to extra messages of certificates, cryptographic operations are also heavier such that they take seconds in a constrained node (See Section 4.7). On the other hand, despite of its drawbacks, certificates alleviate the key distribution problem of the PSK mode. In constrained nodes, Elliptic Curve Cryptography (ECC) is preferred over Rivest-Shamir-Adelman (RSA) based cryptography due to shorter key lengths, 160 vs 1024 bits. Fortunately, to further shorten a handshake and avoid expensive certificate operations, *session resumption* feature has been added to the standard. The symmetric key that has been created in a previous handshake is re-used to generate a new one. Hence, peers do not have to repeat the whole handshake and they avoid certificate related operations.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf=
"http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:pingback="http://purl.org/net/pingback/"
         xmlns:cert="http://www.w3.org/ns/auth/cert#">
  <foaf:Person rdf:about= "https://wonderland/alice/card#me">
    <foaf:name>alice </foaf:name>
    <foaf:givenName>alice</foaf:givenName>
    <foaf:img rdf:resource="https://wonderland/alice.png"/>
    <foaf:nick>alice</foaf:nick>
    <foaf:mbox rdf:resource="mailto:alice@wonderland.com"/>
    <foaf:knows rdf:resource="https://wonderland/BOB/card#me"/>
    <foaf:knows rdf:resource="https://wonderland/CHARLIE/card#me"/>
    <cert:key>
      <cert:RSAPublicKey>
        <cert:modulus rdf:datatype=
        "http://www.w3.org/2001/XMLSchema#hexBinary">
          c2e98ceadf831ab308735c8b30e54a76fe76a9d6...
        </cert:modulus>
        <cert:exponent rdf:datatype=
        "http://www.w3.org/2001/XMLSchema#int">
          65537</cert:exponent>
      </cert:RSAPublicKey>
    </cert:key>
  </foaf:Person>
</rdf:RDF>
```

Figure 4.4: Example public profile of Alice. The profile declares the public key information via "cert:key" and the social network via "foaf:knows" keywords.

### 4.2.2. The WebID Protocol for Decentralized Authentication & Authorization

WebID is a *single-sign-on* and distributed authentication standard [86]. It is designed as an authentication and authorization protocol for distributed online social networks (DOSNs). Distributed social profiles written in Friend-of-a-Friend (FOAF) [24] vocabulary establish DOSNs as an alternative to the centralized social networks. FOAF is a vocabulary with which a person can publish his social network in a profile document on the web. An example is shown in Figure 4.4. In this example, Alice declares her friendship to Bob and Charlie using "*foaf:knows*" statements. Note that URIs are used to clearly identify people instead of their plane names. In WebID protocol a person, company, organization or any other thing are defined by a URI and the URI is placed in a X509v3 certificate. With this certificate and a public social profile, a user can login to a server with WebID protocol.

The protocol is embedded in the Transport Layer Security (TLS) protocol. WebID simply replaces the certificate authority (CA) based verification step of TLS

with a social network based one. Instead of a CA, social profiles and trust between the people verify a certificate. It resembles the Web of Trust (WoT), where certificate owners sign each other's certificate to indicate a social relation and trust. The difference is that in WebID, social networks are incorporated to establish trust instead of certificate chains.

When a client requests authentication by handing a certificate to the server, the TLS protocol starts off by verifying whether the client holds the private key that corresponds to the public key in the certificate. Then the WebID protocol takes over control for performing two essential steps: identity verification and trust establishment. In the identity verification step, which is the authentication step, certificate verification is done with social web profiles. A certificate contains the URI address of the personal profile document of the client in the optional Subject Alternative Name (SAN) field. When the authenticating server receives the certificate, it follows the URI link in the SAN field and fetches the profile document such as the example shown in Figure 4.4. SPARQL, the semantic web query language [79], is used to query the profiles. The server checks the equality of the public key provided in the certificate and the personal profile document. If the public keys are the same, it concludes that the client is the owner of both the certificate and the personal profile document. In case of a compromise of a private key (e.g., through theft), the certificate can be revoked by simply removing its public key from the profile. The URI serves as the identity of the authentication requester; the name, surname or email address fields are not taken into account. Although the identity of the client is verified as the URI, the server still cannot grant access without a social tie between the client and the server. An authorization (trust) step is required. In this authorization step, the authenticating server crawls the profiles on the web to discover a social tie between the client and the server. The social network ties are declared by the "*foaf:knows*" statements. As presented in Figure 4.4, Alice declares that she knows Bob by using the "*foaf:knows*" statement. The time consuming part of the WebID protocol is this trust and authorization step where a social tie from the server to the client is discovered. After the identity verification and trust steps, TLS takes control back and proceeds further to encrypt the channel with a symmetric key.

### 4.2.3. WiFi Probe Requests for Proximity Detection

Detecting which devices are present in the vicinity of a device is key to bounding the search for indirect trust relations and WiFi probe requests indicate the presence of a wireless device in an access point scenario. There is no need for developing custom solutions as we can simply leverage the normal WiFi registration process.

Wireless client devices detect the APs by scanning the WiFi channels. There are two modes of scanning: passive and active. Passive scanning sniffs every channel for AP beacons, whereas active scanning sends probe requests to APs to check their presence. Active scanning is preferred over passive since it decreases the AP detection time, hence the duration for AP association. In the 802.11 standard the frequency and burst size of the probe requests is left to the vendor. The standard states that on every WiFi channel the supplicant sends one or more probe request and waits for the replies from APs for some time before moving to the next channel. In Section 4.4.1 we will show that most devices probe the channels at least once every 200 $s$ providing an accurate view on the presence of the individual devices (and their owners).

We should also note that with iOS 8 a MAC address randomization feature was introduced for protecting privacy which unfortunately, disrupts beacon tracking. However, there are many restrictions on randomization such that mostly it is inactive. It is claimed that randomization only works when cellular data and location tracking is off [63], and the phone is in locked state.



## 4.3. Decentralized social-device Networks

In this section we explain our proposal of decentralized social-device networks (DSDN) for a secure-by-default IoT. Moreover, we introduce the challenge of crawling social profiles in our decentralized architecture. In Sections 4.4 and 4.6, we address the high complexity of DSDN itself and social network search for unconstrained and constrained environments.

In the rest of the chapter, we follow IETF terminology given in Table 4.1 about the entities involved. Resource server (RS) is the device that holds a resource such as a light bulb or a smart thermostat and owned by resource owner (RO). Client (C) is the device that accesses an RS such as a smartphone and owned by client owner (CO). Profile servers (PSs) are distributed and they hold social profiles of both devices (i.e., RS, C) and humans (i.e., RO, CO) in a machine readable format (e.g., FOAF). The PS of a resource server (RS) is pre-provisioned while manufacturing, hence the PS of an RS belongs to the manufacturer. Assigning a new PS is only possible after getting control of the RS.

The WebID-TLS standard (See Section 4.2.2) is the basis for DSDN. Not just humans but also devices have a URI and a social profile that identifies them. A client sends a certificate to an RS and the RS employs the WebID protocol for the certificate verification and the authorization of the client. The client is authorized by trust relations embedded in social profiles distributed in the Internet.

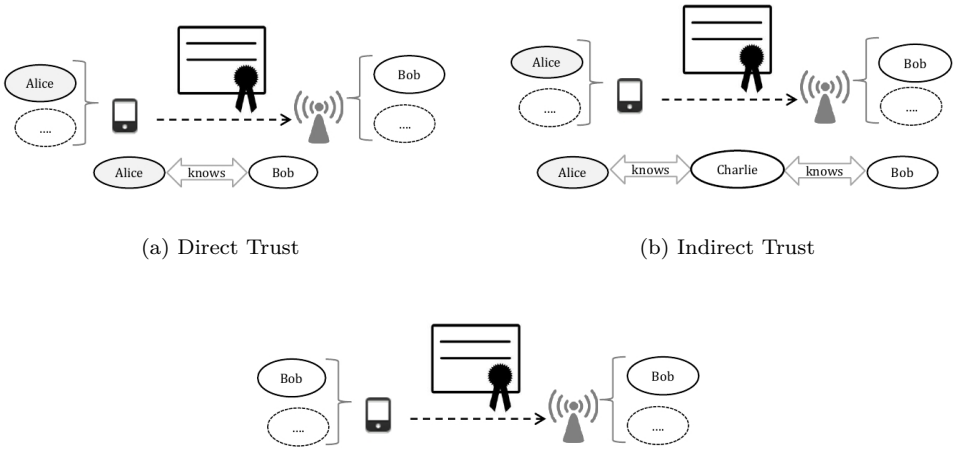
Table 4.1: Role definitions partially adopted from IETF ACE working group

| | |
|---|---|
| *Resource* | A sensory information or an actuator (e.g., heat sensor, door lock). |
| *Resource Server (RS)* | A (constrained) device which hosts a resource only to authorized entities (e.g., thermostat, access point). |
| *Resource Tag* | a QR code or NFC/RFID tag that contains credentials to access PS for ownership imprinting. |
| *Client (C)* | An entity which attempts to access a resource on an RS (e.g., smartphone, laptop). |
| *Resource Owner (RO)* | Owner of the constrained RS (e.g., Bob). |
| *Client Owner (CO)* | Owner of the client (C) device (e.g., Alice). CO and RO are socially connected with relations such as same person, direct friend or friend of a friend. |
| *Profile Server(s) (PS)* | Cloud services that publish the social profiles of all the above entities; also acts as the authorization server for RS with constrained devices (See Section 4.6). PSs are decentralized since each entity can have its own PS. |
| *Smartphone App (SApp)* | Used by RO to scan a tag attached to RS and embeds the ownership relation into PS. |

**Social profiles of devices.** A device like an access point (AP) or a tablet can have multiple owners. If we placed the certificate of a person directly into a device, only one owner can be served. Therefore, we create a new <certificate, public profile> pair for the device. In the public profile, the device can present all its owners. An added benefit is that with individual profiles for devices we can create device-based access control rules.

**Proof of ownership.** To avoid complicated ownership setup, we consider that every RS is tagged with a QR code or RFID/NFC that includes necessary credentials such as a cookie to access the social profile of the RS. The only action required by the owner is reading the tag via a smartphone application. The application reads the tag, accesses to the social profile and embeds the ownership information.

The first person who scans the tag takes over the ownership of RS. There is no message exchange with RS, rather RS's social profile in PS is altered. However, we should note that the tags should be secured to prevent ownership override by




(a) Direct Trust

(b) Indirect Trust

(c) Same Owner

Figure 4.5: Different types of trust. Oval shapes represent the owners of the devices. Smartphone sends a certificate to an access point for authentication and authorization.

adversaries. The ownership transfer can be done in two ways: (i) the tag is scanned by another entity, (ii) the current owner imprints the new owner to RS's social profile while removing her own information. Both of these transfer mechanisms are accomplished autonomously via a smartphone application (SApp). Social-profile can also clear the ownership information based on a timer, if only temporary ownership were allowed.

**Access control process and relation types.** Now we will show how authorization based on profiles works in a WiFi AP scenario. Assume that client is a smartphone owned by Alice and it tries to access the AP of Bob. When the AP gets the certificate of the smartphone, first it verifies the certificate by following the embedded profile URI and checking if the keys "match". After verification, the AP learns the owner of the smartphone, Alice, by checking the social web profile of the smartphone. As the AP knows its owner, Bob, a trust (friendship) relation should be discovered now. There are three possibilities:

- **Direct Friends:** Alice and Bob are direct friends. In their social web profiles, they declare that they know each other by "*foaf:knows*" statements (Figure 4.5(a)),
- **Indirect Friends:** Alice and Bob are not direct friends, but they have a

common friend called Charlie (Figure 4.5(b)). The AP should search the social profiles of Bob and Alice for Charlie or any other mutual friend,

- **Same Owner:** Bob owns both the smartphone and the AP and he tries to access to his own AP (Figure 4.5(c)).

For the sake of simplicity, we assume that having a social connection to a person is enough for trust. However, in real life, we may not want to grant WLAN access to our complete social network. Therefore a chain of access control rules should be involved in the process. The role of the client's owner may become crucial in authorization. We leave access control as future work.



### 4.3.1. Social Network Search and Its Analysis

In DSDN authentication and authorization, messages between the devices (RS-Client), and between resource and social profiles determine the latency. Among two, the latter is the main contributor of the latency. Crawling social profiles is both time consuming and varying part of whole process. Therefore, we present an analysis of the search in this section, and in the upcoming sections we elaborate our solutions for both constrained and unconstrained environments.

The algorithm of the search is given in Algorithm 1. Implicit *HTTPS* calls of the functions to social profiles are highlighted by comments. **k** and **n** are the number of owners and common friends, respectively. After certificate verification, first the *same owner* relation is checked. Since a resource server (RS) may have more than one owner (e.g. a family access point), *same (common) owners* can be multiple as well. For each same owner, the possession of client is checked in line 6 (possession is declared by "*foaf:knows*" statements). At the first verified possession of client, the search completes successfully. The possession check is required to verify the client owner. For instance, in case of a stolen client, if possession is not checked, the "new" owner can access all the APs that the previous owner can. When there is no successful *same owner* relationship, the friends of each RO are fetched and compared to the client owners to find a *direct friend*. In case of a match, the direct friend is checked for possession of client. From the nested loops in Algorithm 1, the direct friend relation seems to have $O(k^2)$ complexity. However, the complexity is actually "$2k \rightarrow O(k)$", since each direct friend is a client owner and we do only 1 possession check per owner by removing it from client owners list (see line 17). In the absence of a trusted *direct friend*, the *indirect friend* relationship is checked. All the friends of client owner(s) are fetched and compared to the friend list of ROs. Then each common friend (direct friend) is verified if it knows the RO. The complexity of indirect search is "$k + 2nk \rightarrow O(nk)$".

**Algorithm 1:** Certificate verification and social tie searching. **k** and **n** are the number of owners and common friends, respectively.

```
   /* All the owners and friends are expressed as URIs */
   /* Client_URI: URI of the client */
   /* RS_URI: URI of the resource server  */
1  if not VerifyCertificate( Client_URI) then                          // HTTPS
2      return FAIL
3  clientOwners = FetchOwnersOf( Client_URI)                           // HTTPS
4  resourceOwners = FetchOwnersOf( RS_URI)                             // HTTPS
   /* SAME OWNER RELATION with complexity:  k */
5  foreach co of Intersect(clientOwners, resourceOwners) do              // +k
6      if co isOwnerOf( Client_URI) then                               // HTTPS
7          return SUCCESS
8      else
9          remove co from clientOwners
   /* DIRECT FRIEND RELATION with complexity:  2k */
10 RO_Friends = []
11 foreach ro of resourceOwners do                                     // +k
12     friends = FetchFriends( ro)                                     // HTTPS
13     foreach df of Intersect( friends, clientOwners) do              // +k
14         if df isOwnerOf( Client_URI) then                           // HTTPS
15             return SUCCESS
16         else
17             remove df from clientOwners
18     push friends into RO_Friends
   /* INDIRECT FRIEND RELATION with complexity:  k+2nk */
19 foreach co of clientOwners do                                       // +k
20     friends = FetchFriends( co)                                     // HTTPS
21     foreach idf of Intersect( RO_Friends, friends) do               // +n
22         if idf isFriendWith( co) then                               // HTTPS
23             if co isOwnerOf( Client_URI) then                       // HTTPS
24                 return SUCCESS
   /* No connection was found.  */
25 return FAIL
```

Each one of the *friend fetches*, *certificate verification* and *possession checks* necessitates a separate *HTTPS* connection. To decrease the number of HTTPS connections, the RS can cache the URIs of its owner(s), and the URIs of the friends of the owner(s) with their public keys. The friends list of an owner also involves the devices he possesses since both the friendship and the ownership are declared with the same "foaf:knows" statements[1]. It is, however, unfeasible to cache all the indi-

[1]The relation between the device and its user is not friendship, it is ownership. This lack of discrimination has a consequence in the effectiveness of the search for trust relations through (in)direct friends; our protocol also iterates over an owner's devices instead of just considering its

Table 4.2: The number of HTTPS connections required for authentication and authorization where $k$ is the number of owners per client and resource server, $n$ is the number of common friends (See Algorithm 1).

| | Number of HTTPS connections | |
|---|---|---|
| | Without Cache | With Cache |
| Same Owner | $3+k$ | 0 |
| Direct Friend | $3+2k$ | $2+k$ |
| Indirect Friend | $3+k+k+2nk$ | $2+k+2nk$ |



rect friends due to their abundance (see Section 4.5.1). The worst case performances are detailed in Algorithm 1 and summarized in Table 4.2. In worst case scenarios all the owners and friends have to be queried for possession and friend relations.

An indirect friendship search has quadratic $O(nk)$ complexity and as a consequence it can easily lead to huge numbers of HTTPS calls. Assume that a tablet and an AP both have $k = 4$ owners (a small family) and each of them has around 200 friends. If the number of *common* friends $n = 100$, according to Table 4.2, $3 + k + k + 2 \times k \times n = 811$ HTTPS connections are required in the worst case. In Section 4.5.1, we show that making such a large number of HTTPS connections requires tens of minutes. In the next sections we solve this search problem.

## 4.4. DSDN in Unconstrained Environments: Social Access Point

In this section, we propose a solution for search space problem defined in the previous section for unconstrained environments. In an unconstrained environment, devices have memories of hundreds of MBs and capable of working on HTTP protocol with structured text parsing (e.g., FOAF in XML). We exploit the presence information of the clients in the vicinity to limit the search space.

As an example scenario, we chose an access point (AP) as RS and a smartphone as client. AP employs DSDN for access control, that is it becomes social (Social-AP). Social-AP allows network access to the social network of its owner(s) without requiring any passwords from clients. Though Social-AP is capable of searching distributed social profiles, as we have shown in the previous section, indirect relations

trusted peers in the social network. As a future work, we plan to propose an ontology to include the concept of ownership, and we consider refining that to the level of principal owner, second tier owner relationships to enhance search efficiency to the maximum possible.

pose a big challenge. In the next section, we address the quadratic complexity of indirect search by incorporating proximity information.

### 4.4.1. Collecting Presence Information

Intuitively, in WiFi authentication, direct friends who bridge the indirect friends to the AP, are probably in close proximity at the time of association. We can bound the number of common direct friends by sensing the ones who are currently in the vicinity of the AP. Moreover, by including the time of arrival, we can sort the common direct friends according to their temporal distance to the indirect friend. The code for bounding and sorting can be added to line 21 of Algorithm 1.



We use WiFi probe requests (see Section 4.2.3) to determine the direct friends in the vicinity of the AP. From previous authentications the AP stores the URIs of direct friends with their devices' MAC addresses in its local cache. When the AP catches a probe request for the first time from a device or the device was absent for a while, the time of arrival of the device is updated. When an indirect friend initiates the authentication, the time of arrival and the presence of direct friends are used to improve search performance. The crucial part in this context-aware system is the detection time of the devices in the vicinity. It is of no use, if detection takes hours. As explained in Section 4.2.3, there is no standard for interval and burst size of probe requests. In [23], it is stated that an expected interval is around 50-60 seconds. To verify that assumption we collected WiFi probe requests in an office environment for two hours. Figure 4.6 shows the inter-arrival time distribution from 35 different devices. Overall 87 devices had been recognized. However to have enough data to present statistical results, only the devices with more than 20 probe request readings are displayed. The firmware of the WiFi card and also the channel noise determine the inter-arrival times. While some devices have almost constant intervals like the first device, some have extremely variable intervals like device #9. The results do not indicate a common inter-arrival range as in [23]. Nevertheless, we can still conclude that inter-arrival times have a median of 56 seconds and 95 percentile of 179 seconds. With the presence detection in minutes, proximity based common-friend sorting and bounding can work effectively.

## 4.5. Evaluation of The Social-AP

To validate our design of decentralized social WiFi AP and to determine the gain of sniffing probe requests, we implemented a prototype version and tested it on real hardware. We altered the Extensible Authentication Protocol-TLS (EAP-

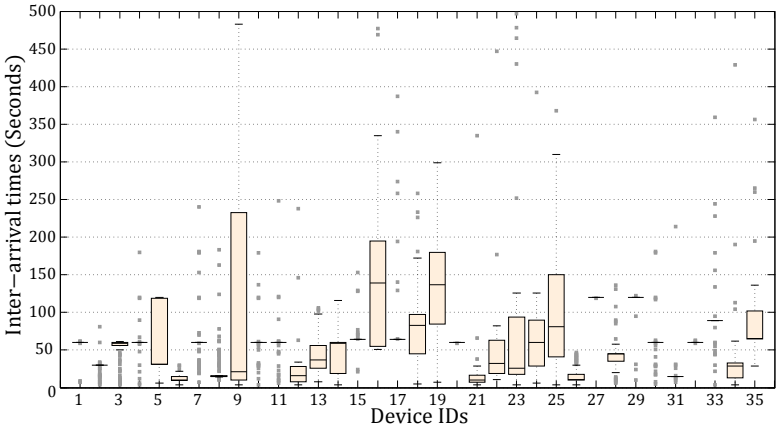



Figure 4.6: Inter-arrival times of the 802.11 probe requests from 35 devices in 2.4 GHz frequency band over two hours.

TLS) standard, and renamed our version as EAP-SocTLS [2]. EAP-TLS is part of Hostapd [3], an opensource access point software implemented purely in C. In the certificate verification part of the Hostapd code, we compare the public key to the one in the social web profile. After verification, we call an external Python library [4] for the authorization part (i.e., searching for trust relations). Although all the extensions can be embedded directly inside Hostapd, Python was selected for its ease-of-use. The lines of code required to complete EAP-SocTLS are about 450 for the Python library and 400 for Hostapd. An SQLite3 database stores the probe requests. Note that our modification is completely transparent to the client side, who follows the normal EAP-TLS process when requesting service from the AP. The only requirement for the client device is to have a certificate with a link to the corresponding social-profile page.

The integration of the WebID protocol into Hostapd was non-trivial due to Hostapd being large and complicated project in terms of lines of code and features. We embedded the WebID process in the OpenSSL *tls_verify_cb* callback function, which is invoked when an error is detected such as (in our case) an unfamiliar certificate failing the built-in CA-based verification. Our heuristic for speeding up the search process for indirect friends relies on presence information that is collected in the background (separate thread); All sniffed probe requests with the MAC addresses and timestamps are pushed to an SQLite3 database. This information is used to filter the common-friends by their temporal distance to the current suppli-

[2] https://github.com/yunus/Hostapd-with-WebID
[3] http://w1.fi/hostapd/
[4] https://github.com/yunus/python-webid

cant being authorized. One challenge in the filtering is that OpenSSL does not pass the MAC address of the supplicant to the *tls_verify_cb* call. Fortunately, we are still able to access the MAC addresses of all the devices that are currently being authorized by Hostapd. Therefore, we use a pool of MAC addresses for the filtering in a round-robin fashion. Once we are done verifying a (in)direct trust relationship we give control back to Hostapd by returning from the *tls_verify_cb* function with the authorization result, and the protocol continues further with the original EAP-TLS sequence.

### 4.5.1. Results

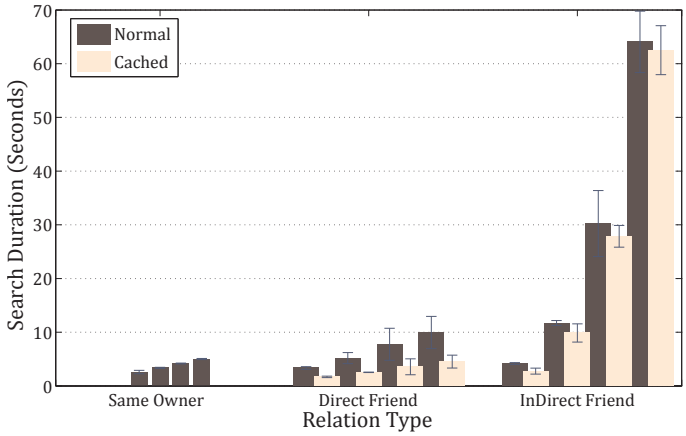


Our experimental setup consists of a Samsung Galaxy S2 smartphone (client) connecting to a Dell Ultrabook with an Intel Centrino Advanced-N 6230 network card (802.11g) serving as the AP. For the tests, we created a synthetic social network with uniform structure at the `https://rww.io` website, which is designed for semantic web applications and supports WebID. Although that web site has SPARQL support, for convenience our EAP-SocTLS implementation fetches complete profiles and processes them locally at the AP.

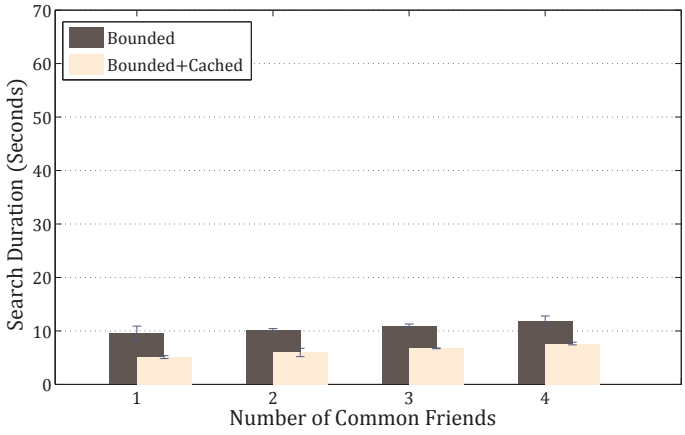
In the following experiments we show the performance of EAP-SocTLS, in particular with respect to the size of the social network (scalability). We study the three different trust types (same owner, direct/indirect friend), and vary the neighbor degree (number of friends/owners) from 1 to 4. In all scenarios we report worst case performance where all possible links (relations) are crawled. The number of common owners increases from 1 to 4 in the **same owner** case. In the **direct friend** case the number of owners on both sides increase with each of the AP owners knowing all client owners and vice-versa. In the **indirect friend** case the device owners and common friends increase together. Note that while the number of owners is increased for all types of trust, the effective number of common friends in the indirect friend case increases quadratically (from 1 to 16).

The duration required for legacy EAP messaging for certificate and symmetric key exchange depends on the quality of a wireless channel (inducing packet loss and retransmits) and was found to typically take less than one second. With respect to the duration of searching for trust relations (Figure 4.7) as implemented in the external Python library, EAP messages do not contribute much to the results. This prompted us to focus on the search part.

The duration of the search with different trust types and with increasing neighbor (friend) degree is given in Figure 4.7(a), Following the analysis in Section 4.3.1, the *normal* case indicates a linear increase in number of (same) owners and direct friends,




(a) Social relation search with increasing neighbor degree from 1 to 4.




(b) Indirect friend search with context Information.

Figure 4.7: Durations for social relation discovery with different relations. Neighbor degree increases from 1 to 4 in Figure 4.7(a) for each relation type. Neighbor degree is 4 in Figure 4.7(b) while the number of common friends increases.

and a quadratic trend for the indirect friend relation. We also present results for the case of an AP with a cache large enough to store the details of AP's owners and their respective friend lists. Usage of the cache for the *same owner* case reduces the search time to zero as a simple lookup suffices; we do not plot these results for clarity. For the other trust types, the performance improves a little, but caching

Table 4.3: Predictions for search duration of higher neighbor degrees. Indirect Friend relation is assumed to be quadratic and the rest as linear.

| | Worst Case Search Duration (Seconds) | |
|---|---|---|
| Neighbor Degree | **10** | **100** |
| Same Owner | 9 | 82 |
| Same Owner Cached | $\sim 0$ | $\sim 0$ |
| Direct Friend | 23 | 224 |
| Direct Friend Cached | 10 | 96 |
| Indirect Friend | 537 | 5279 |
| *Indirect Friend Bounded & Cached | 12 | 88 |
| *: Based on neighbor degree: 4; only the number of common friends changes. | | |



does not change the fundamental linear and quadratic behavior. One might argue that friends-of-friends information can also be cached, which would bring down the search time to near zero, but the amount of storage requirement would then become prohibitive. In [93], for example, it is shown that in Facebook, friends-of-friends grow even faster than quadratic. For a person with 100 friends in Facebook, the number of unique friends-of-friends averages to 27,500.

Observe that, even with caching enabled, the search time for indirect friends grows to over one minute for a neighbor degree of 4. In order to bound the indirect friend search, the AP collects and exploits presence information as explained in Section 4.4.1. To show the effect, we placed one direct friend in the vicinity and then add more direct friends linearly up-to 4. Figure 4.7(b) shows that the use of context information reduces the complexity for an indirect friend search to a linear process.

Creating a large social network is costly. Therefore, we resorted to extrapolating the linear/quadratic search times to study the effects of scalability. In Table 4.3, the estimated time required for searches with a neighbor degree of 10 and 100 are given. The naive Indirect Friend search becomes infeasible; even with degree of just 10, already 537 seconds are required. However, when applying presence information only 12 seconds are needed. In the real life we expect to obtain even better performance

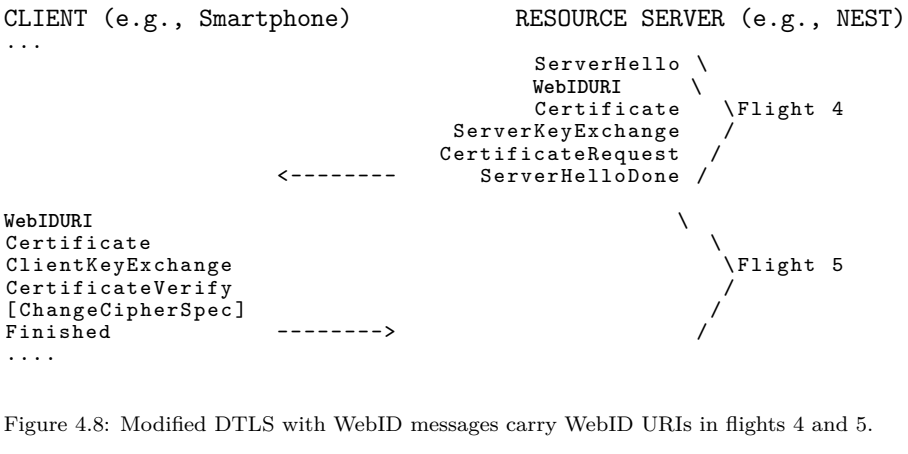



Figure 4.8: Modified DTLS with WebID messages carry WebID URIs in flights 4 and 5.

as we only reported worst case results. Moreover, search performance can be further improved by using better caching algorithms and parallel processing.

## 4.6. DSDN in Constrained Environments: Delegation

In this section, we explain our improvements for DSDN on how to socialize constrained devices as well. We evaluate DSDN in constrained environments and propose a delegation based architecture for handling social profile search. More specifically, we have altered Datagram-TLS (DTLS) protocol (See Section 4.2.1) where devices delegate the social network search to an authorization server. Before that, we want to explain the challenge in constrained environments.

Unlike an unconstrained server (e.g., Social-AP), which can operate on multiple parallel connections to search for a social connection path, a constrained server has limited memory, bandwidth, and computational power. In a duty cycled and multihop network, each TLS connection may require seconds [47] and the whole process may end in minutes, which is not acceptable for user experience. Moreover, assuming that each social connection has hundreds of friends identified by URIs of ∼100 bytes in length, tens of KBs should be fetched and stored in the constrained devices. The capabilities of constrained devices hardly cope with these requirements since according to the IETF classification (RFC-7228), a class 1 device has ∼10 KB RAM, which is five order of magnitude lower than an ordinary server with 1 GB RAM. Therefore, we need a lightweight approach.

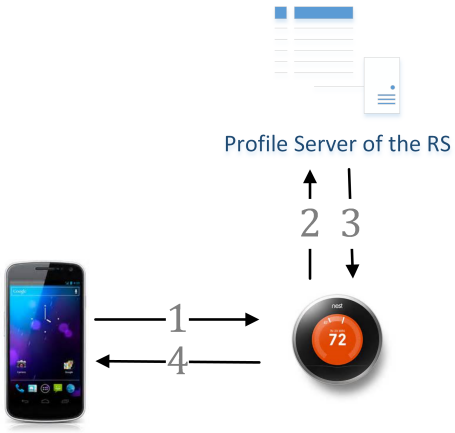



Figure 4.9: The steps of the delegation process. The resource server (Nest) delegates the authentication and authorization of the client to the profile server. All the communication occurs over CoAP-DTLS.



### 4.6.1. Modified DTLS for URI Exchange

In DSDN, all devices and as well as humans are identified by URIs that point to their social profiles. In WebID-TLS protocol (See Section 4.2.2), X509v3 certificates carry these URIs in their subject alternative name fields. Unfortunately, these certificates are too big and difficult to parse for constrained devices. Therefore, IETF recommends the use of Raw public keys (RFC-7250) that only carry public keys, nothing else. Since we need URIs for authentication and authorization over DSDN, we propose new messages for DTLS handshake, called *WebIDURI*, to carry them.

As shown in Figure 4.8, the new messages are added to flights 4 and 5. To keep these messages shorter and thus decrease the overhead, URIs should be kept as short as possible. In our evaluations we show that *WebIDURI* messages with 50 bytes of URIs contribute as low as 7% of the whole DTLS handshake.

### 4.6.2. Delegation of Social Network Search

We follow a **delegation-based** approach to perform DSDN on constrained devices. The social profile server (PS) of a resource server, whose address has been pre-provisioned in the device, also acts as a delegation server. As shown in Figure 4.9, RS asks PS for authentication and authorization of client. In step 1, RS delays the *flight 6* of the DTLS handshake (See Figure 4.3) while consulting PS. In step 3, RS sends the URI and public key of client to PS by using a GET query over Constrained object

Access Protocol (CoAP)[5] protocol that is secured by a second DTLS connection in parallel the first. The format of the query is `x=..&y=..&uri=..` where `x` and `y` represents Elliptic Curve Cryptography (ECC) public key parameters. PS replies with a short message composed of a 1/0 boolean and client's URI to distinguish parallel requests. If the reply from PS is positive, RS continues the DTLS handshake from *flight 6* and shares the resource at step 4. Otherwise, RS sends an access denied DTLS alert to the client.

**Bootstrapping.** The address and public key of profile server, which also acts as an authorization server, are pre-provisioned in RS during manufacturing. Therefore, resource server does not need to discover PS. Moreover, the complete DTLS handshake is only performed in the beginning, later either the connection is kept alive or session resumption is employed. Thus, delegation overhead is minimized after the initial handshake.

**Offline operation.** After the initial delegation process, new requests from the same client can be handled offline without consulting PS. RS stores the public key of client and marks it as authorized, while occasionally refreshing the authorization by a delegation process. On the other hand, as we further discuss in Section 4.9.1, OAuth2 provides temporary access tokens. In our case, DSDN can provide the same functionality via access control lists (ACL), which are passed in JSON format (e.g., JSON web tokens [52]) from PS to RS. However, it is also possible to employ the bearer tokens of OAuth2 since they are well-defined and employed in the industry. The adoption of these options is as simple as switching from a boolean value of authorization decision in step 3 to a bearer token or JSON-based ACL if a constrained device has enough resources to operate on them.

## 4.7. Evaluation of Delegation

We have introduced the delegation process to minimize the communication, memory and computational overheads of the WebID protocol. However, delegation still demands some resources. In this section we describe our implementation and evaluate the protocol on the OpenMote[6] platform. OpenMote is based on CC2538SF53 which is a 32-bit Cortex-M3 microcontroller with 32 KB RAM and 512 KB ROM. Its CC2520 like radio operates in 2.4 GHz and implements 802.15.4 standard.

[5] CoAP is lightweight version of HTTP, designed for constrained node networks with multicast support.

[6] openmote.com

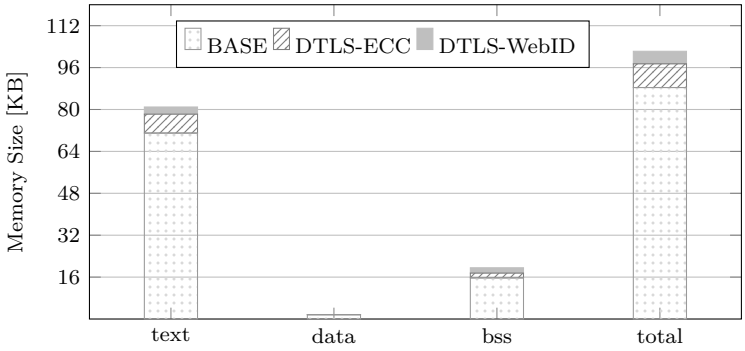



Figure 4.10: Size of memory sections in KBytes for the OpenMote platform. The *BASE* case is composed of Contiki OS+CoAP+DTLS-PSK. Stack is allocated inside the *bss* section.



We have modified Contiki OS 3.x[7], TinyDTLS[8] and Scandium[9] to integrate WebID and delegation support. The constrained RS runs Contiki OS and TinyDTLS while the rest, client and PS, operate on an Ubuntu PC and use Scandium for DTLS. In Contiki OS, we have altered the CoAP implementation for our modified DTLS integration. Scandium and especially TinyDTLS have been heavily modified to allow the new *WebIDURI* handshake messages. Moreover, in TinyDTLS, we have added delegation process as described in Section 4.6.2. Our modified versions of these software are also publicly shared[10].

In the deployment, we have setup a one-hop 6LowPAN network where a Tmote Sky node acts as a slip-radio interface to a border router[11]. We did not employ any radio duty cycling below the CSMA based MAC layer.

### 4.7.1. Memory Overhead

We have analyzed the amount of ROM and RAM requirements by using *arm-none-eabi-size* tool. Though our delegation based system demands more memory, it still fits into available memory. In Figure 4.10, we present the RAM (i.e., *data* and *bss* sections) and ROM (i.e., *text* section) values incrementally. In CC2538 based environments, the stack is also served from the *bss* section, hence these results involve the stack usage as well. The base case, which contributes around 90 KB memory (refer to *total* column), contains Contiki, CoAP and DTLS-PSK. DTLS-ECC feature increases the memory for $\sim$ 9% and finally DTLS-WebID adds $\sim$ 4.7%

[7] github.com/contiki-os/contiki
[8] tinydtls.sourceforge.net/
[9] github.com/eclipse/californium.scandium
[10] github.com/yunus
[11] https://github.com/cetic/6lbr

over DTLS-ECC. The main contributer to the extra memory of DTLS-WebID is the number of concurrent handshakes. To save memory in DTLS-PSK and DTLS-ECC, concurrent connections are not allowed, hence only one client can be in handshake. Whereas DTLS-WebID requires one additional peer for the profile server in parallel to client connection. However, we should note that even if the client connections are increased, still just one profile server connection is enough. In overall, according to RFC-7228 terminology, we can conclude that high end class 1 constrained devices handle delegated WebID protocol with ECC.

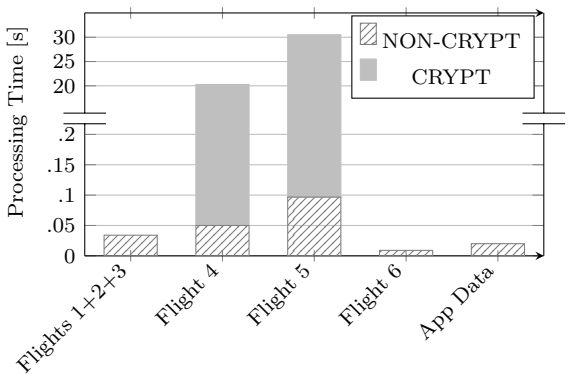
### 4.7.2. Latency



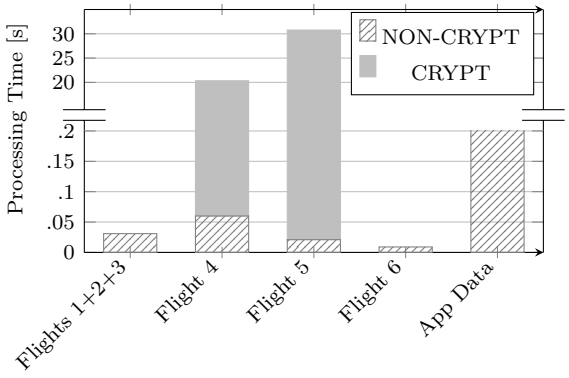
In addition to low memory footprint, delegation operation should have low latency as well for a smooth user experience. In this section, we present the latency of our one-hop 6LowPAN network. We've only focused on the time spent in the resource server since we do not aim to assess the network performance. Moreover, we assume that profile server immediately responds without checking trust connections. We have considered latency of social network crawl in Section 4.5. It is known that public key operations are computationally demanding and takes time. Therefore, new microprocessors like CC2538 have built in hardware acceleration components. Unfortunately, there was no driver for those features while we were experimenting with our setup. We should also note that algorithms used for cryptographic operations perform differently. Here we stick to the algorithms provided by TinyDTLS.

We divide the latency measurements first based on client-RS and RS-PS, then based on handshake flights. In Figure 4.11(a), the latencies of flights for client-RS connection is given. Cryptographic operations dominate the whole process. However, we should note that with software improvements durations for ECC operations can be dropped to less than 5 seconds [47]. Moreover, RSA based hardware accelerators were shown to achieve performances in millisecond scale [46] and we expect to get similar performance for ECC as well. Finally, session resumption feature avoids public key cryptography by employing symmetric keys and hence further requests from the same client would have lower latencies [47].

Between the flights 5 and 6 RS switches its server role and acts as a client for authorization. The latencies of flights for RS-PS communication are given in Figure 4.11(b). Again we observe high latencies. A noticeable difference from RS-client connection is in the application traffic. Application data latency includes the whole round-trip to the profile server and it is higher with respect to client's application data. The reason is that the carried data are more in the case of PS traffic since the data include raw public key and the WebID URI while client data traffic is just a few bytes of sensory information. On the other hand, the main latency is still




(a) Initial handshake with a client. For upcoming requests from the same client, session resumption feature is used to circumvent heavy cryptographic operations.




(b) Resource server acts as a client to fetch authorization information from the profile server. Though the overhead is high, it is a one time operation. Forthcoming clients are authorized by only the application data traffic.

Figure 4.11: Time spent in the resource server. Cryptographic operations are presented separately to indicate the main contributor of the latency. Except the application data traffic, end-to-end latency is not included, which is pretty low with respect to cryptographic operations.

due to public key operations. However, we should note that these time consuming

flights are done once for all the resource requests. In the subsequent requests, there is no need to repeat the whole process to create yet another symmetric key for encrypting the traffic. Only the application traffic is repeated, which is around 0.2 seconds. As a result, the amortized cost of authorization delegation is less than a second.

### 4.7.3. Communication Overhead



In constrained node networks, the communication overhead should be kept to minimum to save energy. Moreover, the retransmissions due to the nature of wireless medium and multihop communication exacerbate the overhead. In our analysis, to focus on the basis of the overhead, we did not account multihop and retransmission delays. In the handshake messages, both RS and client exchange their social profile URIs via the new *WebIDURI* message. In our implementation we set the maximum length of a URI to 50 bytes, however, we advise to keep them as small as possible to avoid message fragmentation. Regardless of the MAC layer, IPv6, UDP, and DTLS headers account for 40, 8 and 27 bytes of overhead, respectively. With a URI of 50 bytes, 125 bytes would be transmitted in addition to MAC overhead. Considering the 127 bytes MTU of 802.15.4, a 50 byte URI is on the edge.

When we consider the overall bytes transferred in our setup, we observe that WebIDURI's contribution is low. To calculate the overhead, we tapped the connection between border router and PS via Wireshark tool. In the messages sent from the RS, 139 out of 2178 bytes of all messages are contributed by *WebIDURI* with maximum URI length. The overall bytes sent from client are fewer since the client software (Scandium) is capable of bundling the messages. From client 77 out of 941 bytes account for the *WebIDURI*. As a consequence, *WebIDURI* messages contribute as low as 7% of all traffic.

Additionally, RS also contacts to PS again over a DTLS connection. RS asks for authorization via the lightweight CoAP protocol. At first, delegation sounds like doubling the communication overhead. However, as we mentioned in the previous section, full handshake happens only in the very first delegation. In the subsequent requests, only application layer data is transmitted. Moreover, compared to other security proposals from ACE working group, which are summarized in Section 4.9.1, all the proposals depend on an external authorization server. Hence, we expect similar overhead in other proposals as well.

## 4.8. Security and Privacy Considerations

In this section, we provide a high level security sketch of DSDN and discuss a number of issues in terms of security and privacy.

### 4.8.1. Security

Notice that DSDN relies on actual standards of DTLS, EAP-TLS and WebID. Therefore, the security of the overall protocol depends on the security of the underlying building blocks. We assume that resource server is populated with profile server's public key in manufacturing, and the implementation of cryptographic primitives are correct. Given these, DSDN is secure as the communication takes place in the encrypted form, end-to-end encryption, using public key cryptosystems. However, with delegation, we introduce a change in the DTLS where WebIDURI is transferred in clear text in flights 4 and 5. This modification creates a risk for an attacker to eavesdrop the communication. The information acquired by the attacker will be the URI of both peers in the communication. This information can be used to track the behaviors of the entities but the attacker cannot access to the content of the communication due to the end to end encryption. The risk of an attacker changing the content of the plain text WebIDURI, as well as other plain text information in the protocol up to the flight 6, is eliminated in the standard by using message digests added to *Finished* messages. To prevent the eavesdropping of WebIDURIs, WebIDURI messages can be placed after flight 6 when the keys are exchanged for secure communication. Being more secure, this approach fundamentally changes the DTLS standard by adding two new flights. Due to the extra flights, the standard gets complicated while latency and energy consumption both increases.

Furthermore, the DTLS standard is designed to guard the nodes to several well-known attacks such as DOS, replay, and man-in-the-middle (Section 5 of [82]). In the following sections, we discuss three scenarios each of which summarizes a case where an involved party is compromised.

**Compromised resource server.** If an adversary takes control of RS and hence acquires the public-private key pair, the information delivered from RS cannot be trusted anymore. In that case clients should omit RS in information collection. The detection of compromise is out of our scope. However, after detecting the misbehaving RS, resource owners remove the social links to RS from their own profiles. Thus, effectively ownership relation is revoked. Clients detect the compromise by searching the profile of RS and inferring that the owner does not have a social connection anymore. Compared to revocation list approach of revoking the key-chain

based certificates and web-of-trust based on PGP certificates, DSDN offers a faster response to compromises.

**Compromised client.** If a client's public-private key pair such as the smartphone in the Social-AP application, were compromised, the attacker can access to Social-AP by using the keys. Fortunately, recovery is as easy as revoking the ownership relation. The owner of the smartphone removes the ownership relation from her social profile. Then, RS does not allow access since WebID protocol does not authorize the smartphone. However, in offline operation the compromised public keys will remain in RS until the next synchronization with PS. This is a typical stale-cache problem. With frequent RS or PS originated updates, the compromised keys can be detected and prevented from collecting the data.



**Compromised social profile server.** The trust between RS and PS is established in manufacturing by imprinting PS's public key and URI into RS. If the private key of PS were compromised, a new public-private key should be installed into RS. To accomplish this, a physical access to RS would be required if there were no redundant trusted public keys installed as backup. Due to the physical access, we consider PS compromise as the more disruptive one. However, since PSs are distributed, the effect of such a compromise remains local with respect to centralized architectures where single point of failure is a concern.

### 4.8.2. Privacy

We have identified two different privacy concerns: *privacy of social profiles* and *privacy of interactions*. The former one is the problem of revealing social relations while the latter is the problem of revealing the actual social interactions with the social network. With respect to the former, unfortunately decentralized social relation search requires the social profiles to be kept in plain text. Therefore, the social relations among people and their devices are publicly available. One solution for better privacy is obfuscating the profile by populating fake device ownerships and friendships. Moreover, complete privacy is achieved by restricting profile access to only close relations such as the same-owner and direct friend because, indirect relations cannot be discovered when profiles are restricted.

With respect to device interactions, in DSDN, an attacker can only detect interactions by eavesdropping the communication as explained in the previous section. If the attacker is not in the same network, DSDN successfully conceals the interactions, because resource servers carry out the social relation search by themselves without revealing the client's identity. However, note that in delegation, profile

server obtains the peer's identity. Consequently, if the profile server belongs to a third party, then the interactions are revealed.

To conclude, DSDN violates the profile privacy–though there are some precautions–while it protects the interaction privacy. We argue that interaction privacy comprises more information than the other. The reason is that, social relations do not necessarily indicate a close bond among peers. As observed in online social networks, people do not interact with all of their friends. On the other hand, interactions of devices are certainly real projections of ongoing and possibly close relationships among the peers. In the next section, we show that only DSDN can protect interaction privacy with respect to the state of the art.



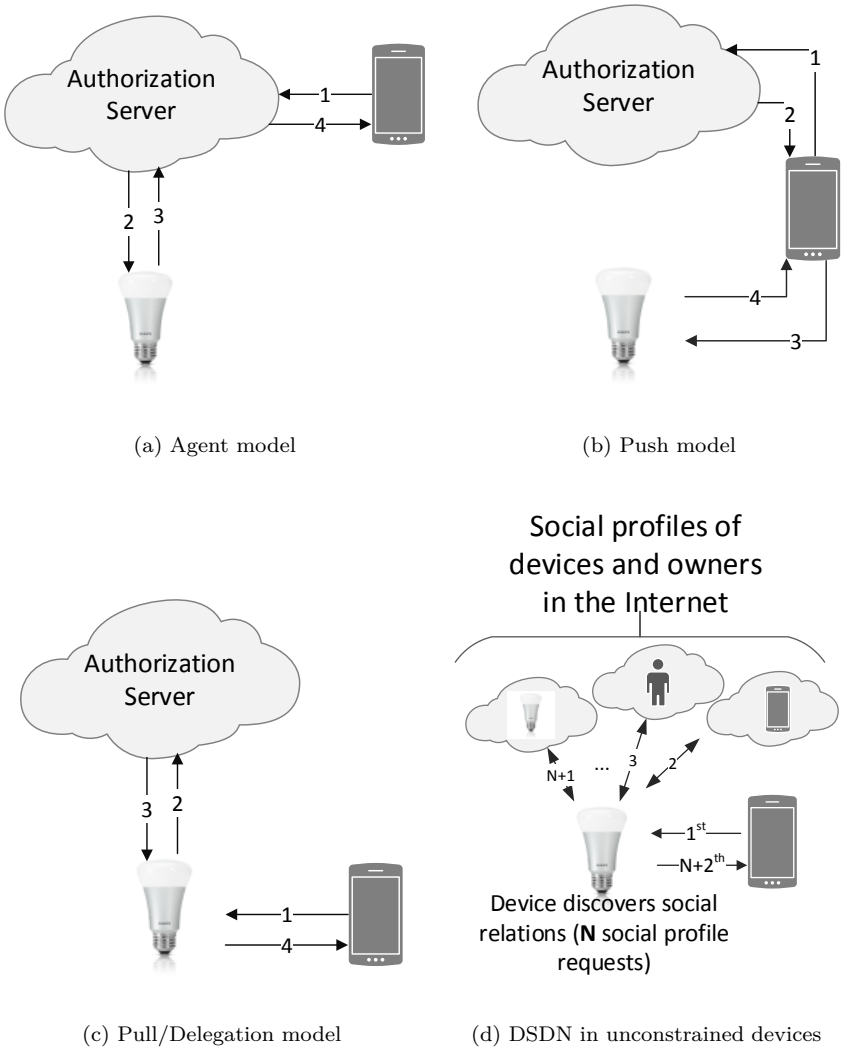
## 4.9. Related work

The significance of social networks in cooperation was mentioned by Mihai et al. [22]. It has been shown that with respect to classical reputation and direct reciprocity (e.g., tit-for-tat), cooperating with relatives/acquaintances is more prominent in cooperation decisions. Atzori et al. [9, 10] and Guinard et al. [40] are leading groups who initiated the idea of integrating social networks in IoT or web of things. The social-IoT (SIoT) architecture, proposed by Atzori et al., mentions not just ownership relation but also co-work, co-location, social-object and parental-object relationships. Among these our DSDN provides primarily social object and ownership relations while using role information potentially offers the rest of the relationships. However, both SIoT and the social access controller (SAC) proposed by Guinard et al. [40] are centralized architectures. SAC, for instance, acts as a proxy between the RS and client, hence it is capable of eavesdropping on the transferred data (i.e., violates end-to-end security).

Motivated by the above studies, we have proposed DSDN as a decentralized alternative. In the next section, we position and compare DSDN to the *currently draft* recommendations of IETF Authentication and Authorization for Constrained Environments (ACE) working group.

### 4.9.1. Comparison to IETF-ACE proposals

There is an ongoing effort in the IETF ACE working group to standardize IoT security. Various scenarios, models and technologies are being discussed. In this section we position and compare DSDN to these *draft* proposals of the working group.




(a) Agent model

(b) Push model

(c) Pull/Delegation model

(d) DSDN in unconstrained devices

Figure 4.12: IETF ACE working group authorization models (a-c) in relation to DSDN (d). DSDN with delegation resembles the pull model. DSDN in unconstrained environments is fully decentralized, all the decisions are made by the resource server.

**Architectures.** Among five, here we mention the three that have minimal communication overhead. That are *pull*, *push* and *agent* models (for a detailed comparison of all models please refer to [39]). All the models with DSDN are depicted in Figure 4.12. The **pull** model is the same as our delegation model, where the RS asks an authorization server (AS), PS in our case, for the access control decision.

Table 4.4: Comparison of authentication and authorization models

| | End-to-end Security | Profile Privacy | Interaction Privacy | Robustness* | Offline Operation† |
|---|---|---|---|---|---|
| Agent | – | – | – | – | – |
| Push | • | – | – | – | ○ |
| Pull/Delegation | • | – | – | – | ○ |
| DSDN | • | – | • | • | ○ |

•, ○, and – represent full, partial, and zero support, respectively.
*: Robust to single point of failure.
†: Without Internet connection.



In the **push** model, first the client obtains a security token (e.g., bearer token for OAuth2) from the AS and passes it to the RS to access the resource. If the RS cannot decipher the token, it may need to consult to AS as an additional step. In the **agent** model, similar to SAC described in the previous section, there is a proxy between the client and the RS.

The comparison of all these models are summarized in Table 4.4. Though the IETF does not mention social network integration, for a fair comparison, we assume that all the architectures can be adapted to social network based access control. Among all the architectures, the agent based model is the weakest since it violates end-to-end security, privacy and it is vulnerable to single point of failure. Pull and push models have similar properties as shown in the table. Both architectures provides end-to-end security since authorization server is out of the traffic route.

**Offline support (lack of Internet connection).** In the push model, clients can re-use old tokens, hence each client should reach the AS at least once. In the pull model a resource server can store the credentials of client as trusted after the first authorization server access. However, in the delegation model, a resource server uses the locally replicated social profile, hence an RS does not have to communicate with the AS all the time. *Moreover, even without any Internet connection if the social profile is replicated over the local area network, then the RS does not need the Internet at all.* This ability comes from the fact that the decision maker can be the RS not just the AS. Note that replicating social profiles locally introduces the stale cache problem and should be taken into consideration in system design.

**Robustness.** All the architectures except DSDN are vulnerable to a single point of failure since in DSDN the decision maker is the resource server. Malfunctioning

of resource servers or profile servers do not disrupt the whole system.

**Privacy.** As explained in the previous section, in DSDN social profiles have to be kept as public. The ultimate solution for profile privacy is storing all the social relations in "only" one authorization server (e.g., only Facebook, the rest Google+, Twitter or others should not exist), then that authorization server discovers the relations inside its protected network without revealing to outside. Another option is trading off interoperability by allowing interactions of devices that belong to the same social network provider. However, such centralized approaches violates interaction privacy, which we believe more important than profile privacy as discussed in previous section. Fortunately, DSDN still protects interaction privacy.



### 4.9.2. Delegation-based Systems

Beyond pushing the authentication and authorization to a separate entity, cryptographic operations can be delegated as well. In gateway like architectures [15, 36], the cryptographic keys of the RS are shared with the gateway. Though effective, such approaches do not provide end-to-end security and raise concerns about the privacy. In a recent work from Hummen et al. [47], the session resumption feature of the DTLS protocol has been exploited to offload the computation. The delegation server handles the public key related computation and shares required session resumption keys with both the RS and the client. With this delegation the communication and computational overheads are avoided in the constrained RS. We do not see any issue in adopting this technique to DSDN or other delegation-based systems. However, we should note that discovery of the delegation server, and the use of IP addresses may limit the adoption. Since the delegation server simply spoofs the IP address of the client to hand over session resumption keys, the network layer should be capable of handling IP-conflicts and adapt its routing table.

## 4.10. Conclusions and Future Work

Our goal has been to create an autonomous and secure-by-default IoT system and our proposal, decentralized social-device networks (DSDN), supports both of these features. Devices and their owners are identified by URIs that point to their social profiles. In these social profiles ownership and friendship relations are declared for cooperation (i.e., authentication and authorization). DSDN is **autonomous** since devices take decisions instead of their owners. In the interaction of devices, authentication and authorization are carried out via a decentralized social network, which is queried by the devices themselves. The only operation required by a human




Figure 4.13: Future of DSDN, where trusted devices assist each other. That is, a constrained device delegates authorization to an unconstrained trusted device of the same owner. Thus, the delegation architecture becomes more robust and preserves interaction privacy. Moreover, in general Internet dependency is fully relieved.

is declaring the ownership, which is just a tag/code scan. Moreover, our approach is **secure-by-default** such that there is no additional security via complicated procedures. From the beginning we employ public key encryption and hence, avoid weak passwords. The system itself takes the responsibility for security; the user is not burdened with complicated setups.

Compared to centralized architectures for social network integration, DSDN is robust to single-point-of-failure, protects interaction privacy and can operate without Internet connection. Unfortunately, decentralization has high computational complexity of searching for social relations such that it increases latency in unconstrained devices (i.e., social access point) while it is not possible to implement in constrained devices at all. For the unconstrained case, we have implemented DSDN on an access point. We assume that for indirect relations, common friends are also in the vicinity. Therefore, our "social" access point uses proximity information to limit the search space via determining the common friend. Proximity detection is achieved by overhearing WiFi beacons. For the constrained case, we have proposed a delegation architecture where social network search is carried out by the profile server. We have modified the Datagram-TLS standard and tested our solution on real hardware. Experiments demonstrate that delegation contributes to as low as 7% of all traffic with added latency of less than a second.

The future of DSDN is trusted device networks where unconstrained devices act as authorization servers for constrained ones. As shown in Figure 4.13, a smart light may contact a trusted access point instead of reaching to the Internet. Thus,

dependency for Internet connectivity is minimized while latency is decreased.

# 5

# An Identity-Oblivious Evolutionary Approach to the Forwarder's Dilemma

In the previous chapter, we proposed decentralized social-device networks (DSDN) for cooperation. We pointed out the complexity of social network search and proposed solutions for both constrained and unconstrained environments. We set the minimal requirements for employing DSDN, which unfortunately exclude highly constrained devices. These highly constrained devices cannot employ cryptographic operations that ensure identity. Therefore, they cannot be involved effectively in DSDN. These devices can either cooperate always and hence, are exploited by free riders or they defect always and hence, are punished as well as isolated by more intelligent devices.

In this chapter, our goal is to provide identity-oblivious meta strategies for highly constrained devices to discover the locally-best strategy in a neighborhood. We choose the forwarder's dilemma as our application scenario since cooperation of devices is fundamental in wireless multihop networks. We modify two meta strategies from evolutionary game theory, Win-Stay Lose-Shift (WSLS) and Stochastic Imitate Best Strategy (SIBS), for wireless ad hoc networks. We redefine fitness by means of packet traffic and propose the *two-hop overhearing* method to measure the fitness. Simulations and real-life experiments show that both WSLS and SIBS are

able to discover the locally-best strategy while they are robust to fake identities. Moreover, we study the effects of local decisions on the evolution of the network. While WSLS promotes cooperation up to half of the network, SIBS achieves full network cooperation. However, unlike WSLS, which is robust against the variation of many network parameters, SIBS is subject to neighbor degree, routing, and mobility. Thus, in the absence of identity information, these meta strategies protect devices against exploitation by free riders and still favor the spread of cooperation.

## 5.1. Introduction

In Chapter 1, we explained *forwarder's dilemma* [30], which is a strategic situation where devices are unwilling to relay packets of other nodes, yet require the help of other nodes to send their own packets. To save their precious energy resources, devices *defect* and drop packets rather than *cooperate* and relay them. As a consequence, multi-hop communication in an ad hoc network becomes impossible.



In the last decade, researchers have aimed at incentivizing or enforcing cooperation with methods that incorporate rewards and punishments, such as *credit exchange*, *direct-* and *indirect-reciprocity*. Nevertheless, these methods share a common Achilles' heel: *they depend on identity information*. Hence, these methods are susceptible to Sybil and slandering attacks, where free riders fabricate and spawn identities at will [26]. Adopting a new identity allows attackers to, for example, whitewash (clear) their reputation as a free rider by pretending to freshly join the network. Unfortunately, there is no complete protection against Sybil attacks without some form of centralized control [26]. Existing proposals use prior trust relationships and even location information [78, 101]. Consequently, a node with enough computational resources can ensure the identities of its neighbors (e. g., by public key cryptography, or pre-shared keys) and employ a cooperation enforcement mechanism. On the other hand, a real-life wireless network may be composed of devices with diverse resources (computation, memory etc. ), which raises the question: how should an identity-oblivious device act in a (potentially) selfish network?

In this chapter, we present a novel approach based on evolutionary game theory (eGT). eGT argues that the imitation of locally fittest strategies (cooperation/defection), called *network reciprocity*, can help cooperation prevail in the network. Taking actions based on the observed "spatial" trends prevents exploitation and promotes cooperation [89]. First, cooperators establish clusters with higher fitnesses than defectors. Then, these clusters expand over time by converting defectors to cooperators with the higher fitness offered by cooperation. Nevertheless, though current results on network reciprocity also seem promising for the wireless setting, we cannot apply them directly. While in eGT the fitness is defined in terms of the number

of neighbors and the identity is assumed to be known, in wireless networks the measure of success is based on packets and identities are easily forged. By re-examining network reciprocity from a wireless perspective, in this chapter we bridge the gap between theory and practice.

### 5.1.1. Contributions

We ask: "how can we devise *meta strategies* suitable for identity-oblivious devices that (*i*) are able to adapt to locally-best pure strategies (cooperation/defection), (*ii*) allow the spread of cooperators, and (*iii*) are resistant to Sybil attacks?" Within this context, we provide three main contributions:

**Contribution 1. Packet-based fitness measure.** Though significant work has been done in a theoretical setting, known meta strategies are not suited for wireless networks. That is due to a simple -yet critical- assumption: the action of cooperation does not matter; only the number of cooperator/defector neighbors determines the fitness of a node. Though this assumption is reasonable for abstract theoretical work, it does not capture the reality of wireless networks, where nodes cooperate or defect by injecting their own packets and forwarding the packets of their neighbors. To this end, our first contribution entails the definition of a fitness measure that captures the behavior of nodes as a function of packet traffic (See Section 5.3).

**Contribution 2. Sybil-resistant meta strategies for wireless networks.** We identified two promising meta strategies for their ability to resist Sybil attacks: the *Stochastic Imitate Best Strategy (SIBS)* and the *Win-Stay Lose-Shift (WSLS)* meta strategies. Both SIBS and WSLS identify the locally fittest strategy by adapting to the observed neighborhood trends. To render WSLS and SIBS Sybil-resistant, we propose that nodes should not compute their fitness directly by asking and hence trusting their neighbors, but indirectly by *overhearing* the neighborhood packet traffic (See Section 5.4). Extensive simulations and experiments on a real wireless testbed demonstrate that, with overhearing, both meta strategies adapt to avoid exploitation (See Section 5.5). Furthermore, our comparison shows that, in the presence of Sybil attacks, SIBS and WSLS outperform the celebrated Tit-For-Tat (TFT) and Generous-TFT (GTFT) meta strategies [88] (See Section 5.6).

**Contribution 3. The evolution of cooperation.** A central question in evolutionary game theory is which strategy, cooperation or defection, spreads and dominates the network. In their seminal paper, Ohtsuki et al. [73], showed that, cooperation does spread globally in the theoretical setting of the game when the benefit to cost ratio is larger than the node degree. The question that we pose is whether

the same holds in wireless networks. Our analysis and experiments draw forth novel insights: While WSLS keeps at most half of the network as cooperator, SIBS can lead to the invasion of cooperators throughout the network. Additionally the performance of SIBS in wireless networks does not only depend on the neighbor degree (as was in previous studies), but also on the routing protocol and the mobility of the nodes (See Sections 5.7-to-5.8).

The chapter concludes with a detailed discussion of related work (See Section 5.9) and a brief comparison of the proposed meta strategies (See Section 5.10).

## 5.2. Motivation and Preliminaries

**Standard model.** We classify the nodes in a forwarder's game into three types: cooperators, defectors (free riders) and smarts. Smart nodes recognizes the strategies of their neighbors and cooperate only with other cooperators. Consider the case of a node $u$. We will use set $\mathcal{V}_u$ to refer to $u$'s neighbors, while set $\mathcal{C}_u, \mathcal{D}_u$ and $\mathcal{S}_u$ represent cooperator, defector and smart neighbors, respectively. Hence, $\mathcal{C}_u \cup \mathcal{D}_u \cup \mathcal{S}_u = \mathcal{V}_u$ and the sets are disjoint. If node $u$ decides to cooperate, then its fitness will be

$$f_u = b\left(|\mathcal{C}_u| + |\mathcal{S}_u|\right) - c\,|\mathcal{V}_u|\,, \tag{5.1}$$

where $b$ is the benefit obtained from each cooperative or smart neighbor, and $c$ is the cost of $u$'s altruistic behavior payed for every neighbor ($b, c \in \mathbb{R}^+$). On the other hand, a defector does not pay any cost and its fitness is simply

$$g_u = b\,|\mathcal{C}_u|\,, \tag{5.2}$$

since only cooperators help a defector. Smart neighbors detect defection and do not help a defector. Notice that, in the standard model when there are no smart neighbors, the fitness of defection is always higher than that of cooperation (i. e., $\mathcal{S}_u = \emptyset \Rightarrow f_u \leq g_u$). This is the reason why, according to classical game theory, defection is the best strategy. Therefore, many researchers proposed mechanisms for building smart nodes.

**Motivation.** To promote cooperation, smart nodes employ mechanisms such as rewards and punishments, which inherently depend on identity. Therefore, these methods are not an option for a constrained node that is identity oblivious. Consequently, defection seems to remain the only viable strategy for the constrained node. On the other hand, defection only pays off if there are unconditional cooperators around. *Among smart neighbors that can detect defection to punish as well as cooperation to reward, the constrained node still benefits more by practicing cooperation.*

To conclude, we define the problem for a constrained node as discovering which strategy is the best in a neighborhood.

**Evolutionary game theory for ad hoc networks.** Evolutionary game theory (eGT) aims to explain the decisions of agents (nodes) where complete or perfect information may not be available. Nodes use *meta strategies* to determine whether they should cooperate or defect based on the actions of their neighbors. Many meta strategies have been studied in the literature, including *(stochastic) imitate best neighbor (SIBN)*, *(stochastic) imitate best strategy (SIBS)*, and *win-stay lose-shift (WSLS)* [43]. Typically, candidate meta strategies assume that the identity of neighboring nodes is known, which makes them susceptible to Sybil attacks. Before explaining how we were able to free SIBS and WSLS (but not SIBN) from using identity information, we describe the original SIBS and WSLS meta strategies for reference.



*Win Stay Lose Shift.* WSLS is a reactive strategy that depends on trial and error. If a node *wins* the strategic game, it keeps playing the same strategy. If the node *loses*, it switches strategy. It is noteworthy that, in WSLS, the network can be neither fully cooperator nor fully defector. In a fully defector network, each node infers that it is not winning against its neighbors and switches to cooperation. When a few nodes switch together, they help each other and consequently their fitnesses become better than full defection. Since a node benefits more by breaking symmetry, the network evolves in a time-varying and stochastic way. In Section 5.3.2, we will discuss how WSLS can be made to operate without any identity information and in Section 5.4 we will explain how a node estimates the cooperation level of its neighborhood by overhearing the packet traffic.

*Stochastic Imitate Best Strategy.* In SIBS, each node $u$ decides whether to cooperate or defect by imitating the aggregate neighborhood behavior with a probability computed locally as

$$P_u^{(card)} = \begin{cases} \frac{F_u^{(card)}}{F_u^{(card)}+G_u^{(card)}} & , F_u^{(card)} > 0 \\ 0 & , F_u^{(card)} \leq 0 \end{cases} , \tag{5.3}$$

where

$$F_u^{(card)} = \sum_{v \in (\mathcal{C}_u \cup \mathcal{S}_u)} f_v \quad \text{and} \quad G_u^{(card)} = \sum_{v \in \mathcal{D}_u} g_v$$

are the aggregate fitnesses of cooperator (involves smart nodes) and defector neighbors, respectively. $f_u$ and $g_u$ are defined in (5.1) and (5.2). The superscript *card*

emphasizes the use of the cardinality of the set of neighbors (which in turn implies knowledge of their identities). By analyzing SIBS in different graph families, such as lattices, small-world networks and random graphs, Ohtsuki et al. showed that the invasion of cooperators is possible when $b/c > |\mathcal{V}|$, that is when the benefit-to-cost ratio exceeds the node degree [73]. We have shown that this result also holds for graph families representative of ad hoc networks, such as random geometric graphs [27]. In Section 5.3.1, we will re-define SIBS-based on packet traffic and explain its robustness to identity changes.

**Assumptions and limitations.** The main assumptions posed in this work are as follows:

**(i)** In this work, we do not prevent all identity related attacks. When there is no trusted identity provider, spoofing identities can be employed for denial-of-service, buffer-overflow, data aggregation, voting, distributed storage or routing attacks [66]. Most of these attacks have malicious intents, which have the aim of collapsing a network or corrupting the collected data. Instead, we target Sybil-attackers who do not forward a packet while still desire to exploit their neighbors. *We classify them as free-riders to highlight that they do not aim collapsing a network.* Their intention is to be regarded as cooperators to raise the chance that genuine nodes are lured into cooperation. Being malicious and disrupting the network would deceive the purpose for the free riders: none of the packets of a free rider are routed. However, as we will mention in Section 5.4.2, if an attacker steals existing identities instead of fabricating new ones, routing protocols can get disrupted. Though the meta-strategies cannot recover the routing protocol, they still avoid exploitation.

**(ii)** We have applied our algorithms in two different routing schemes that are kept as simple as possible to only shed light on the meta strategies. Both of the routing schemes determine the next hop randomly, which is known as *random walk*. The difference between the two lies in the set of candidates. In the simpler one all the neighbors are included in the candidate set, while only cooperators are included in the second. The second routing scheme is called *selective routing*. We have proposed selective routing to represent more intelligent routing schemes that affect the cost of cooperators and consequently the network evolution. meta strategies with selective routing are abbreviated with an 'S-' prefix such as S-SIBS and S-WSLS.

**Methodology.** We will base our conclusions about the proposed meta-strategies on mostly simulations and real-world testbed experiments. First, we identify key factors and give the intuition about their effects based on mathematical analysis

that does not take into account network evolution, then we experiment on these factors and highlight the outcomes.
Unfortunately, standard mathematical tools cannot be used to analyse the evolution of cooperation without relying on oversimplifying assumptions. There are two well known methodologies for studing the evolution of game theoretic strategies over networks. First, one can model the system using stochastic difference equations that evolve over graphs. As is most often the case, even if studied in expectation, the resulting state-space equations are non-linear, and yield few insights. Second, mean field theory can be used in order to understand the state evolution using a simpler model. This in fact was the approach adopted by Ohtsuki et al. to show that (under some simpler/specfic conditions) cooperation can prevail in a network. This technique however assumes that the population is well mixed and does not take into account the graph structure, thus may only be a rough approximation. To conclude, we follow an experimentation based approach to unveil the performance of the above meta-strategies against identity changes.



## 5.3. Sybil-Resistant Meta Strategies

In this section we argue that the original definition (cardinality based) of fitness is *not suitable* for wireless networks. In the wireless setting, it is not the number of cooperator and defector neighbors that matters, but the number of successfully transmitted packets. Therefore, we modify SIBS and WSLS to employ packet traffic as the fitness measure. We also show that identities become less significant as a side effect of using packet traffic.

We begin by noting that in wireless networks the benefits and costs of nodes are a function of packet traffic. Consider a resource constrained node $u$. The benefit of $u$ is proportional to the number of packets it *injects* into the network. On the other hand, $u$'s cost is proportional to the amount of resources consumed for forwarding the packets of other nodes (e. g., the energy consumed for packet transmission). This motivates us to modify the definition of fitness to incorporate the number of packets injected and relayed by each node. In our definition, the fitness of a node $u$ (either defector, $g_u$ or cooperator, $f_u$) is

$$f_u = g_u = b\,|\mathcal{I}_u| - c\,|\mathcal{R}_u|\ , \tag{5.4}$$

where $\mathcal{I}_u$ and $\mathcal{R}_u$ are the sets of packets (successfully) injected and relayed by $u$. It is important to note that the above is a local definition. An alternative, global definition would count the benefit based on the number of packets *delivered* to the destination, as well as on the routing path length. The global definition captures

more precisely the total effort incurred by the network in delivering $u$'s packet, but introduces higher complexity: it necessitates a global view and depends highly on the routing mechanism in use. The number of successfully injected packets –packets received *and* relayed by cooperative neighbors– presents a simpler alternative.

### 5.3.1. Sybil-Resistant SIBS

As discussed in Section 5.2, in SIBS each node periodically identifies and imitates the best strategy in its neighborhood. The only difference between the standard SIBS meta strategy and the one we propose here is the definition of fitness, which instead of cardinality-based becomes packet-based.

The implementation of SIBS is given in Algorithm 2. The algorithm is run periodically by each node in the network. Consider a representative node $u$. For a fixed time period, $u$ overhears the injected and relayed packets in its neighborhood (See Section 5.4). Node $u$ then computes $F_u^{(pkt)}$ and $G_u^{(pkt)}$ by grouping its neighbors into cooperators and defectors based on whether the ratio of injected and relayed packets exceeds an optimism threshold (lines 1 to 7). Finally, $u$ cooperates with a probability $P_u^{(pkt)}$ and defects with probability $1 - P_u^{(pkt)}$ (lines 7 to 11). We use superscript "*pkt*" to highlight the use of packet traffic (as opposed to cardinality) in the formula. The proposed implementation is very lightweight: the algorithm exhibits a complexity of $O(|\mathcal{V}_u|)$, in terms of both space and time.

**Optimism threshold.** How many packets should a node relay to be considered cooperative? The optimism threshold plays an integral part in distinguishing cooperators and defectors (see line 4 in Algorithm 2). Although, we set *optimism* to 2 in the experiments, in general, the value should be set according to the requirements of the nodes and the characteristics of the environment. With a high optimism, everyone will be classified as a cooperator to the benefit of the free riders. Using a low *optimism*, however, can easily lead to misconceptions as corrupted or dropped packets directly influence the sets of injected and relayed packets. Note that the free riders may want to maintain the reputation of a cooperator with minimum effort. They may relay just enough packets as the threshold demands. We consider this as an incentive for the free riders as in [60].

**The effect of grouping.** But if cooperators and defectors are grouped separately based on their identifier and fitness, how can the algorithm be Sybil resistant? The crucial operation against Sybil attacks is the sum operation in lines 5 and 7. Though cooperators and defectors are grouped based on their identity, the aggregated fitnesses matter for strategy selection. As a result, packet traffic becomes the

**Algorithm 2:** Sybil-resistant SIBS.

**Data**: $\mathcal{I}_u$: the number of injected packets in previous round.
$\mathcal{R}_u$: the number of relayed packets in previous round.
$b$: benefit acquired by injecting a packet.
$c$: cost of relaying a packet.
OPTIMISM: user defined threshold.

**Result**: The forwarding strategy of node $u$.

```
/* Compute fitness of cooperation and defection */
```

$F_u^{(pkt)} = G_u^{(pkt)} = 0$

**foreach** $v \in \mathcal{V}_u$ **do**

  $f_v = b\,|\mathcal{I}_v| - c\,|\mathcal{R}_v|$

  **if** $(|\mathcal{R}_v| > 0) \wedge \left(\frac{|\mathcal{I}_v|}{|\mathcal{R}_v|} < \text{OPTIMISM}\right)$ **then**

    $F_u^{(pkt)}$ += $f_v$ `// Cooperator neighbor`

  **else**

    $G_u^{(pkt)}$ += $f_v$ `// Defector neighbor`

```
/* Imitate fittest strategy */
```

**if** $\frac{F_u^{(pkt)}}{F_u^{(pkt)} + G_u^{(pkt)}} >$ `rand()` **then**

  *Cooperate*

**else**

  *Defect*

significant factor rather than the nodes themselves, effectively limiting the effect of changing identity. The effect of grouping is better illustrated by an example: Suppose that node $v$ is a defector which, for some small period, changes its identity to $w$ and cooperates. All the packet traffic of nodes $v$ and $w$ will be added to $G_u^{(pkt)}$ and $F_u^{(pkt)}$, respectively. Since in truth $v$ is a defector who does not intend to stay in cooperative state, any benefit gained by changing its identity is small. For more influence on $P_u^{(pkt)}$, node $v$ may stay in cooperative state longer (as $w$). However, node $v$ is indeed a defector, hence a dilemma.

We have to note that grouping nodes into cooperators and defectors is not the only viable solution. Instead of counting the packet traffic of each node individually, the aggregated traffic can also be used to observe the trend in the neighborhood. For instance, a node may decide to cooperate if the ratio of *overall* injected and relayed packets in its neighborhood is larger than an optimism threshold. *Unfortunately, focusing solely on aggregated traffic brings forth an undesirable consequence; it promotes exploitation.* Consider, for example, a defector neighborhood, in which a single cooperator relays all packets. If a node in this neighborhood employs the aggregated packet count as a decision metric, it can infer that the environment is cooperative since many packets are being relayed. However, by choosing cooperation

the node allows adjacent defectors to exploit its resources.

**Algorithm 3:** Sybil-resistant WSLS.

**Data**: $\mathcal{I}_u$: the number of injected packets in previous round.
$\mathcal{R}_u$: the number of relayed packets in previous round.
$b$: benefit acquired by injecting a packet.
$c$: cost of relaying a packet.
MUTATION: user defined threshold.

**Result**: The forwarding strategy of node $u$.

1 $f_u = b\,|\mathcal{I}_u| - c\,|\mathcal{R}_u|$

/* Initialization */

2 **if** $f'_u == NULL$ **then**

3 *Cooperate*

/* Win Stay, lose shift */

4 **else if** $\left(\texttt{rand()} < \frac{f'_u - f_u}{f'_u}\right) \vee (\texttt{rand()} < \text{MUTATION})$ **then**

5 switchStrategy()

6 $f'_u = f_u$

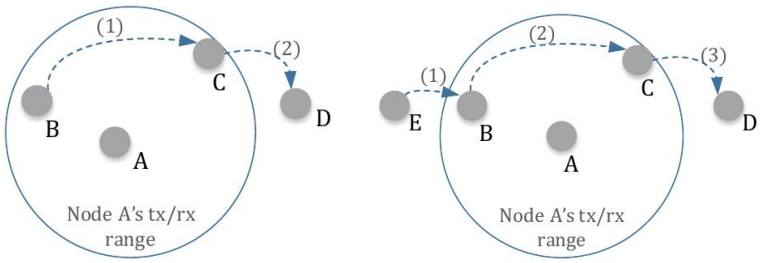
5

## 5.3.2. Sybil-Resistant WSLS

WSLS is a simple meta strategy used in iterated prisoner's dilemma games [69]. In every iteration of standard WSLS, if a node *wins* the game against an opponent, it keeps playing the same strategy. If not, it switches to the other strategy. Because it depends on a direct fitness comparison between nodes, the standard WSLS is sensitive to Sybil attacks. To make WSLS Sybil resistant we employ a different version where each node compares its fitness to its own in a previous time [43].

The logic of the proposed meta strategy is given in Algorithm 3. Initially, nodes do not have any knowledge about their environment, and they can choose their strategy in an arbitrary way. In our implementation, nodes always begin by cooperating (line 3). Every time the algorithm is executed, nodes switch their strategy probabilistically based on the normalized difference between the fitness of their previous and current strategies (line 4). Due to this short term memory, the algorithm exhibits $O(1)$ space and computational complexity.

WSLS is an innovative meta strategy in that it can choose a non-existing strategy in a neighborhood. For instance, unlike SIBS, which favors cooperation when all the neighborhood is cooperative, WSLS may switch to defection. Two phenomena, fluctuations of fitness and mutation probability, are the sources of the innovation. *(i) Fluctuations:* Even if all the neighbors are cooperators, fitness may fluctuate in two consecutive rounds due to varying packet rate and routes. In these fluctuations, if the fitness drops, the node may switch to the other strategy. *(ii) Mutation:* On the




(a) Node B injects a packet, and node C relays it.

(b) Both nodes, B and C relay, but only C is rewarded with a relay.

Figure 5.1: Two examples of the two-hop overhearing method. Arrows indicate the path taken by a packet, while in parenthesis we give the transmission sequence. Node A recognizes whether a packet from B to C was injected or relayed. Packets from E to B cannot be recognized.



other hand, in our simulations, we observed that fitness values stuck at zero without fluctuations when all the neighborhood follow defection. Hence, WSLS could not innovate and switch to cooperation. *Therefore, we introduced a small mutation (line 4) to maintain the innovative behavior and break reciprocal defection.* In the literature, mostly the mutation is not mentioned [43]. However, in one of the initial works, Nowak et al. [69] introduced a probability of 0.001 as *noise*. Although, we also used 0.001 as our mutation value to be consistent with the literature, we should note that especially in adaptation (See Section 5.5), a higher mutation value of 0.05 offered slightly better results. In a real-life deployment, the mutation threshold can be set to a higher value depending on the possibility of a full defection state.

## 5.4. Estimating Fitness by Overhearing

In ad hoc networks, asking neighbors about their strategy and fitness necessitates trust. That presents a problem; by lying about their strategy and identity (as simple as altering their MAC or IP address), free riders can bias fitness computation, and exploit the network. Instead, we propose that each node independently estimates the fitness of its neighbors and itself by overhearing the packet traffic.

**WSLS.** A WSLS node counts only the packets that pass through itself. For relayed traffic, overhearing is not needed. However, to assure the successful injection of a packet, the receiving node should be monitored to check if it forwards the packet to the next hop (success) or drops the packet (failure). Fortunately, *all forwarding can be overheard*, since the next hop of an injected packet is always sent from within the

radio range of the injecting node (assuming symmetric links).

**SIBS.** A SIBS node has to overhear all the packet traffic in its neighborhood to compute the fitnesses. Unlike WSLS, for which one-hop overhearing is enough, SIBS needs two-hop overhearing. The two-hop overhearing method, which we propose, is based on a simple observation: *a node can recognize whether one of its neighbors injected or relayed a packet if it knows the packet's previous location.* After all, packets that are injected cannot exist prior to the time of their injection. Therefore using overhearing we can correctly label packets that are overheard over two hops. Consider for example node A in Figure 5.1(a). Node A recognizes that C relayed a packet since it has overheard the same packet being previously sent by B. On the other hand, when a packet's previous transmission is not overheard, there is no reliable way to recognize if the packet was injected or relayed. In Figure 5.1(b), for instance, node A cannot discern that B relayed E's packet due to the fact that E is not in A's radio vicinity. As a consequence, some portion of the traffic may not be labeled. *Nevertheless, the percentage of labeled traffic is sufficient to estimate the* $P^{(pkt)}$ *with an accuracy of more than* 96% *(See Section 5.4.1).*



The code for labeling injected and relayed packets is given in Algorithm 4. The algorithm is designed for the Contiki OS and is composed of two callback functions. Nodes operate in monitor mode overhearing all packets transmitted in their frequency band and channel. Address fields inside the packets are used to create the neighbor lists. The *input_sniffer* (line 1) function is called for every received packet, while the *output_sniffer* (line 12) is called for every transmitted packet. In the initial reception of a packet, it is only pushed to a set data structure. On the second reception, the relayed and injected counts for neighbors or the node itself are increased. Packets are distinguished by the use of source address, packet id, and –more importantly– the hash of the packet's payload.

### 5.4.1. Two-hop Overhearing Accuracy

In the following we ask: (i) what is the percentage of packet traffic labeled by two-hop overhearing and (ii) how accurately can we estimate fitness based on overheard packets?

**Analysis.** Using two-hop overhearing, a node can only label packets that move for at least two hops within its transmission range. We can characterize the likelihood that a packet is labeled by computing the probability that the overhearing node and the transmitting node share a common neighbor. Assuming a random geometric graph deployed in a torus[1] and that the deployment area is larger than the *tx*

[1]In a random geometric graph, nodes are deployed randomly within a given area. The torus

**Algorithm 4:** Packet overhearing algorithm.

```
Data:
storedPktList: stores the overheard packets including the node's own packets.
sender: the previous hop address of the packet.
source: the address of the packet creator.

/* INPUT SNIFFER: callback function.  Called when a packet is overheard.  */
1  def input_sniffer(sniffedPkt):
2      storedPktCopy = findPkt (sniffedPkt, storedPktList)
3      if storedPktCopy is NULL then
4          push sniffedPkt to storedPktList
5      else
6          n = findNeigh(sniffedPkt.sender)
7          n.relayed++
8          if storedPktCopy.source == storedPktCopy.sender then
               // Previously, packet was heard from source.
9              m = findNeigh(storedPktCopy.source)
10             m.injected++
           // Update stored copy not to count as injected again.
11         storedPktCopy.sender = sniffedPkt.sender

/* OUTPUT SNIFFER: callback function.  Called when a packet is transmitted.  */
12 def output_sniffer(sniffedPkt):
13     if sniffedPkt.source == myAddr then
14         push sniffedPkt to storedPktList
15     else
16         storedPktCopy = findPkt (sniffedPkt, storedPktList)
           /* Below is same as line 6 to 11 */
17         n = findNeigh(sniffedPkt.sender)
18         n.relayed++
19         if storedPktCopy.source == storedPktCopy.sender then
20             m = findNeigh(storedPktCopy.source)
21             m.injected++
22         storedPktCopy.sender = sniffedPkt.sender
```

range, the expected ratio of two neighboring nodes $u, v$ sharing a common neighbor is $E[\frac{Area(u,v)}{Area(u)}] \approx 0.59$, under the unit-disk model [53]. Hence, in expectation $u$ will be able to label 59% of $v$'s packets.

**Experiments on packet labeling.** Next, we study two-hop overhearing in a real-life testbed. In our experiment, nodes inject packets periodically and forward them via *random-walk* routing. The testbed, which is located in our department at TUDELFT, consists of 108 SOWNet G301 nodes with CC1101 radios operating at 868 MHz. The deployment area is almost rectangular with dimensions 14.65 by 81 meters. For more detail please refer to [99]. We experimented with different transmission power levels. As previously shown, at −34 dBm transmission (tx) power

simplifies analysis by allowing us to ignore the effect of the boundary.

Table 5.1: Average percentage of labeled packets for two-hop overhearing in a real-life testbed with different transmission ($tx$) powers, as compared to random geometric graphs (analysis).

| $tx$ power ($dBm$) | $E[\mathcal{V}]$ | $P_{injected}$ | $P_{relayed}$ |
|---|---|---|---|
| $-34$ | $4,43$ | 51% | 45% |
| $-15$ | 26.33 | 59% | 51% |
| $-6$ | 41.01 | 71% | 64% |
| 0 | 42.28 | 76% | 73% |
| analysis | independent | 59% | 59% |

the network becomes partially connected and at all values above it, the network gets connected [99].

Table 5.1 summarizes, for each transmission radius, the percentage of labeled injected and relayed packets, $P_{injected}$ and $P_{relayed}$, respectively. We have two main observations: *First, the percentage of labeled traffic exhibits a dependence on the tx power, but is still similar to the analysis.* As the $tx$ power increases two-hop overhearing becomes more efficient. Nodes close to the boundary are more likely to share neighbors. This boundary effect is amplified by increasing the $tx$ power, in which cases most of the packet traffic occurs in the overhearing range of the nodes. *Second, the percentage of labeled relayed packets is less than the injected for all tx powers.* The explanation for this effect can be seen from Node B's traffic as overheard by A in Figure 5.1. Despite the same route from B onwards, the injected packet sent to C (Figure 5.1(a)) *can* be labeled, while the relayed packet received from E and sent to C (Figure 5.1(b)) *can not*. To prevent free riding the two-hop overhearing method counts relaying only if the packet was heard previously. Therefore, there is a bias towards the injected count.

**Accuracy of $P^{(pkt)}$ estimation.** $P^{(pkt)}$ is a weighted ratio of injected and relayed packets. Since the two-hop overhearing captures a similar fraction (6% difference) of both the injected and relayed packets, the $P^{(pkt)}$ ratio is not affected. Hence, we expect almost zero $P^{(pkt)}$ estimation error. We simulated a network composed of cooperators and defectors where all follow the (S-)SIBS meta strategy. More details on the simulation environment and parameters are in Section 5.5. Figure 5.2 presents the mean absolute error of $P^{(pkt)}$ estimation to ground truth with overhearing windows of different lengths. When the windows are long enough, the error reduces to as low as 0.03, which is enough for discovering the spatial trend. Selective

routing is more restricted as the set of forwarding candidates is smaller, so smaller window sizes suffice to average out the randomness.

### 5.4.2. Resistance to Address Spoofing

With two-hop overhearing, we are certain that a cooperator node has indeed relayed a packet. However, a free rider may still alter the source address and pretend as if the packet is being relayed. We address this by not labeling a packet as *relayed* without having overheard it before. Other attacks by spoofing source address are as follows:

**Spoofed relayed packet.** A free rider can only invite its neighbors to cooperation by relaying some packets itself such that is regarded as a cooperator. Therefore, altering the source address to its own (i. e., hiding cooperative behavior) has no gain for a free rider.

**Spoofed injected packet.** We distinguish three types of attacks, depending on the spoofed source address:

*(i) Changing the source address to that of an existing cooperator.* This is a malicious attack that damages the routing protocol of the network. As mentioned in Section 5.2, a free-rider node does not employ this attack. Otherwise, the network collapses and the attacker itself also cannot be served. Nevertheless, meta-strategies still avoid exploitation. If too many packets are labeled as being injected from a cooperator, then said cooperator will be classified as a defector.

*(ii) Changing the source address to that of an existing defector.* A free rider blames another free rider. Since there is no individual punishment in both SIBS and WSLS, this attack does not affect the strategy decision.

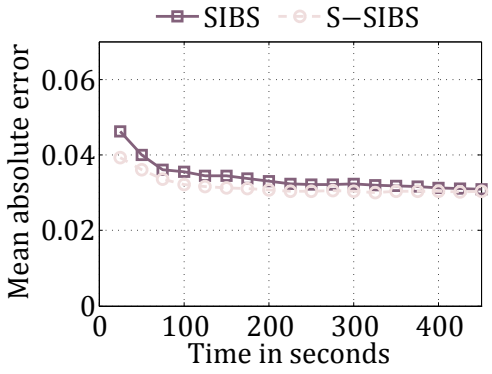



Figure 5.2: Mean absolute error of $P^{(pkt)}$ estimation with increasing overhearing window.

*(iii) Changing the source address to that of a non-existent node (invisibility attack).* A free rider alters the source address of its packets to a non-existent value. Hence, it will be considered as if it does not inject any packet and operates with zero fitness. However, the overall relayed packets by cooperators keep climbing, which lead to lesser fitness of the cooperators. Therefore, SIBS eventually switches to defection.

We should note that the above attacks are threats only to the SIBS meta strategy. In WSLS, nodes are impervious to address spoofing; each node considers solely its own fitness and actions (relaying or injecting a packet).

Table 5.2: Simulation parameters for experiments on meta strategies

| | |
|---|---|
| Operating System | Contiki |
| Setup | 100 nodes in $50 \times 50\ m^2$ |
| Neighbor Degree | $\mu = 9.9$, $\sigma = 3.5$*, $tx$, $rx = 10\ m$ |
| MAC Layer | CSMA |
| Path length | $Random(1-4)$ hops |
| Packet generation rate | $4 + Random(1-3)$ sec |
| Packet Size | 100 bytes |
| * $\mu$ and $\sigma$ denotes mean and standard deviation. | |

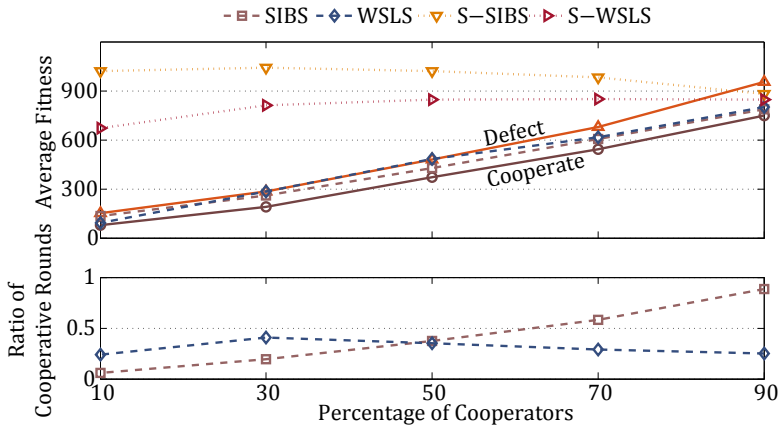


## 5.5. The Local Adaptation of Meta Strategies

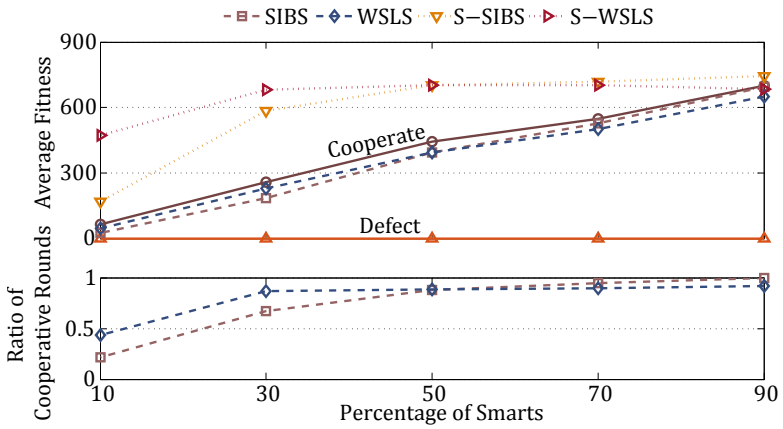
Punishments and rewards are possible only when identity is known. Yet not every device has the computational resources required for validating the identities of its neighbors. This section evaluates how well an identity-oblivious node performs when using the adopted SIBS and WSLS meta strategies. In our experiments, we place a resource constrained node within a heterogeneous network composed of defectors, cooperators, and smarts. We then test the node's ability to discover the locally best strategy. Our results indicate that when there are smart nodes in the network, defectors are punished, whereas nodes with meta strategies discover that cooperation pays off.

### 5.5.1. Experiments on Local Adaptation

In our experiments we traced the behavior of an oblivious node in two scenarios: mixture of defectors first, with unconditional cooperators (scenario $C$) and second,




(a) Scenario $C$, Defection is better.




(b) Scenario $S$, Cooperation is better.

Figure 5.3: Average fitnesses (top) and ratio of cooperative rounds (bottom) of meta strategies in two different scenarios. The legends of the pure strategies, *Defect* and *Cooperate*, are embedded in the figures as they are reference points for comparing the meta strategies. The difference of pure strategies is low in scenario $C$ due to the high $^b/_c$ ratio. *In both scenarios SIBS and WSLS perform close to the fittest pure strategy.* S-SIBS and S-WSLS have selective routing; hence, their fitness values are higher than the rest.

with smart nodes (scenario $S$). While in the former scenario a node always gains more by defecting, in the latter it is often more rational to cooperate (when surrounded by smarts). In each round, after the convergence of $P^{(pkt)}$ estimation (See Section 5.5.2), the traced node reconsiders its strategy according to the overheard

packets in the previous round. We compare the node's fitness under (S-)SIBS and (S-)WSLS to the one obtained by a pure strategy (i. e., always cooperating or defecting). For each scenario, we repeated the experiment 20 times, each lasting for 20 rounds. We set $b/c = 20$ to fulfill the condition of $b/c > |\mathcal{V}|$ (see Section 5.2). The rest of the simulation parameters are given in Table 5.2. All simulations were performed with the Cooja simulator [74].

**Fitness.** The achieved fitness values of the meta strategies with respect to pure strategies are given in the top sub-figures of Figure 5.3. As postulated, in scenario $C$, there is no smart node to punish defectors and hence, defection is always fitter. On the other hand, when smart nodes punish defectors as in scenario $S$, nodes are better of cooperating. *Observe that, in both scenarios, the fitness of WSLS and SIBS is comparable to the best pure strategy.* This supports our claim that both meta strategies are able to adapt locally, even without the knowledge of identity. Moreover, when SIBS and WSLS are compared to each other, we do not observe a significant difference.



**Cooperative rounds**. The bottom sub-figures of Figure 5.3 presents the ratio of rounds that the oblivious node was a cooperator. First consider SIBS. *In both scenarios, SIBS favors cooperation when there are fewer defectors around (i. e., to the right).* The trends are similar in both scenarios, since SIBS does not distinguish cooperators and smarts. However, in scenario $S$ (Figure 5.3(b)), SIBS is more inclined towards cooperation, since defectors have zero fitness. Secondly consider WSLS. In scenario $C$ (Figure 5.3(a)), *WSLS recognizes the absence of punishments, and hence, does not follow cooperation.* On the other hand, in scenario $S$ (Figure 5.3(b)), *WSLS favors cooperation similar to SIBS, since whenever WSLS tries defection it is punished by the smart nodes.* Finally, when we compare the two, especially in scenario $C$, they differ a lot. With lower percentage of cooperators, WSLS favors cooperation more than SIBS. The reason is that, WSLS is an innovative meta strategy; a node with WSLS tries cooperation to consider other options when surrounded with defectors and has zero fitness. Similarly, when surrounded with cooperators, it may switch to defection. Later, we will see similar trends in network evolution (Section 5.7.1 & 5.7.2).

**Selective routing.** We also evaluated the two meta strategies in a selective routing scheme that forwards messages solely to cooperator neighbors (S-SIBS and S-WSLS). *We can see that, by being selective, the traced node achieves a much higher fitness.* Since packets are more likely to be relayed when sent specifically to cooperators (rather than at random), a node employing selective routing achieves higher

fitness. When both meta strategies are compared, especially in scenario $C$, S-SIBS performs better than S-WSLS. We observed that by increasing the mutation value from 0.001 to 0.05, S-WSLS performs almost the same as S-SIBS. With lower mutation values, nodes are stuck in reciprocal defection state longer. However, again in the first scenario, with higher mutation, the performance of WSLS degrades slightly. This shows the significance of the mutation value on the (S-)WSLS performance.

**Are meta strategies exploitative?** To test if meta strategies exploit pure cooperators, we deployed a network composed of only unconditional cooperators, and placed one adaptive node amidst them. We found that SIBS immediately chooses cooperation. However, the WSLS-enabled node chooses cooperation on average 48% of the experiments. Due to its innovative behavior, *WSLS discovers that it is surrounded by pure cooperators and observes that defection has better payoff.*

### 5.5.2. The duration of rounds

Nodes overhear their neighborhood traffic to estimate $P^{(pkt)}$, but for how long? The duration should be as short as possible to conserve energy and to be responsive, yet long enough for accurate $P^{(pkt)}$ estimation. The minimum period should involve at least 1 packet transmission to each neighbor to assess its strategy. Having no prior knowledge of the traffic pattern, and assuming plain random walk routing, the duration is computed as follows: According to the Coupon Collector's problem [49], to be able to send at least one packet to every neighbor, a node needs $|\mathcal{V}| \log |\mathcal{V}| + \gamma|\mathcal{V}| + 0.5$ trials, where $\gamma \approx 0.577$ is the Euler-Mascheroni constant. In our case, the average packet inter-arrival time is 6.5 *sec/pkt*, $|\mathcal{V}| \approx 10$. As a lower bound, the node should keep overhearing at least $6.5 \times [10 \log 10 + (0.577)10 + 0.5] \approx 190$ sec. Nevertheless, as simulations indicate, even shorter durations are enough with SIBS due to overhearing the injected packets from several sources in parallel.

WSLS estimates the probability of switching its strategy by comparing the fitnesses of two consecutive rounds. If the environment remains unchanged, we expect that the overheard traffic also remains the same keeping the probability close to zero. However, the overhearing duration should be long enough to capture similar amount of traffic. We collected the fitnesses of nodes in 10 consecutive rounds in a full cooperator network with $b/c = 20$ (see Table 5.2 for other parameters). Nodes do not change their strategies to guarantee a stable neighborhood, otherwise we cannot observe a convergence to a value. We plot the normalized error between consecutive fitness values ($\frac{f^{t-1}-f^t}{f^{t-1}}$) in Figure 5.4. The elbow point for mean is around 200 seconds which is similar to the above analysis. Longer intervals reduce the variance and consequently improve the decision accuracy. However, the response time of

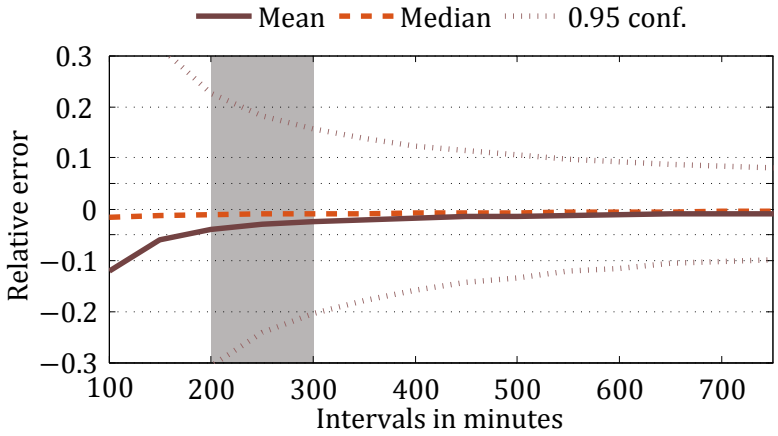



Figure 5.4: Relative error of consecutive fitness values ($\frac{f^{t-1}-f^t}{f^{t-1}}$) with different overhearing intervals. Shaded area represents our operating interval.

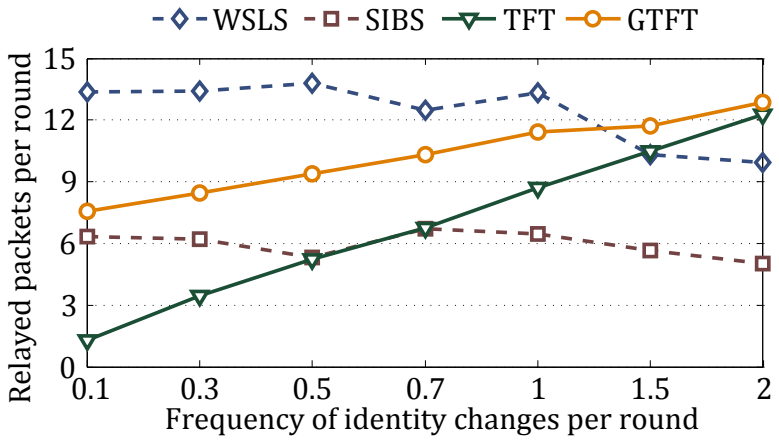


the nodes to variations in the neighborhood also diminishes. In our experiments, we chose the interval of $[200, 300]$, which is enough for overhearing, to maintain asynchrony in strategy updates.

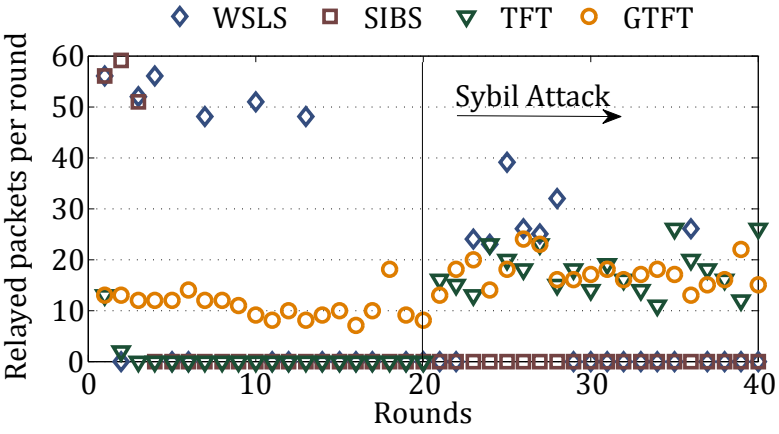
For SIBS, Figure 5.2 shows that the $P^{(pkt)}$ estimation error drops exponentially with longer overhearing periods. The elbow point of the curve is around 75 seconds with the given network parameters. The reason for faster convergence than WSLS is that the node overhears all the neighboring traffic. As a result the packet rate is much higher.

## 5.6. Sybil Attack Resilience

To test the Sybil-resilience of our meta strategies, we placed an identity-oblivious node in a network composed of 80% free riders (i. e., defectors that employ Sybil attack) and 20% cooperators. Since the network is dominated by selfish non-cooperative nodes, the rational choice is to defect. However, by fabricating identities, free riders attempt to hide their behavior, and hence, confuse oblivious nodes into relaying their packets. In this setting, the more Sybil-resilient a meta strategy is, the fewer free-rider packets it relays. We compared SIBS and WSLS to two direct-reciprocity meta strategies known for their good performance (when identity is known): the Tit-For-Tat (TFT) and Generous-Tit-For-Tat (GTFT) [88] strategies. A node using the TFT or GTFT meta strategy starts by relaying the packets of a neighbor and follows the neighbor's strategy as observed in the previous round. The difference between the two is that GTFT becomes sporadically generous and




(a) Average number of *free-rider neighbors'* packets that the traced node relays per round as a function of Sybil attack frequency (i. e., identity change per round).




(b) A representative run: after the $20^{th}$ round, free riders repeatedly fabricate identities with a frequency (i. e., identity change per round) of 2.

Figure 5.5: The effect of Sybil attack on the number of relayed packets from free riders. WSLS and SIBS are resilient to Sybil attack since they relay the same or fewer number of packets with increased frequency of the attack.

gives a new chance to defector nodes.

Figure 5.5(a) presents the average number of relayed packets from free riders

for SIBS, WSLS, TFT, and GTFT, over 20 runs, each lasting for 50 rounds. To illustrate the effect of the attack, we varied the frequency with which free riders fabricate new identities in each round. *Observe that, both SIBS and WSLS curves mostly remain constant and even drop when Sybil attacks become more frequent.* Moreover, SIBS performs better than TFT in higher frequencies, while better than GTFT for all frequencies. *TFT and GTFT were tricked into relaying more packets when free riders fabricated identities more aggressively.* In both meta strategies, the first time that a node encounters a neighbor it always starts by being cooperative. Though this 'grace period' is essential in promoting cooperation (otherwise any nodes using TFT will end up defecting), it becomes a vulnerability when free riders change their identity. Note that in the high frequencies –over 1 identity per round– both SIBS and WSLS start to relay fewer packets. This phenomenon is not the success of SIBS or WSLS; it is a problem of routing. After an identity change, the old identity stays in the routing table for some time leading to an invalid route. Hence, fewer number of packets are addressed to our adaptive node.



Figure 5.5(b) takes a closer look into the meta strategy behavior by depicting the number of relayed packets from free riders for each strategy in a representative simulation run. In this experiment, free riders start to fabricate new identities after the $20^{th}$ round, (highlighted with a vertical line) with a frequency of 2 identities per round. *We can see that SIBS and WSLS are not affected by Sybil attacks.* Both meta strategies quickly realize that defection is the fittest strategy (the network is dominated by defectors). SIBS is more stable and keeps defecting, while WSLS sporadically tries cooperation. GTFT and TFT exhibit a very different behavior. In the first 20 rounds, TFT punishes all the defectors successfully. However, as soon as the defectors start fabricating fake identities, TFT starts relaying again. *Lastly, we should also mention that when SIBS and WSLS are cooperative, they relay more packets than GTFT and TFT.* The reason is that TFT and GTFT still punishes a free rider until it fabricates a new identity while SIBS and WSLS relays blindly. However, under more aggressive attacks, TFT and GTFT will relay as blind as SIBS and WSLS.

## 5.7. Evolution of Networks

In the previous sections, we demonstrated the adaptation to the fittest strategy and the Sybil attack resilience of both meta strategies. *Nodes avoid being exploited by others while not depending on the identities.* Next, we focus on the questions such as what would be the emergent behavior when all the nodes follow the same meta strategy? Will cooperation suppress defection throughout the network? Which

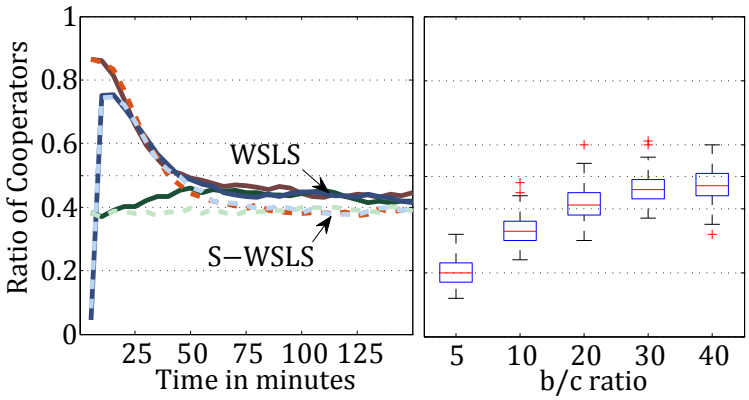



Figure 5.6: Evolution of network under WSLS and S-WSLS. On the Left, the ratio of cooperators over time is presented with $R_c^{init} = \{0.1, 0.5, 0.9\}$ and $b/c = 20$. On the right, ratio of cooperators is presented for only WSLS with different $b/c$ ratios and $R_c^{init} = 0.5$. *(S-)WSLS keep at most half of the network as cooperator.*



factors supervise the evolutionary process?

### 5.7.1. Evolution of Networks under WSLS

How does a network of WSLS nodes evolve over time? For non-zero mutation probability, WSLS possesses no equilibrium[2]. This means that the network never reaches a steady state; nodes continuously change their strategy. Nevertheless, as we show in the following, when considered from a probabilistic standpoint, WSLS does converge, in the sense that the percentage of cooperators in the network stabilizes.

To examine the evolution of (S-)WSLS, we conducted simulations, using the same network setup and simulation parameters as in Section 5.5.1 (see Table 5.2). First, we examined the sensitivity of the evolutionary process to initial conditions. The left part of Figure 5.6 presents the change in the ratio of cooperators over time. We initialized the network by letting a given ratio of nodes to start as cooperators (though their strategy is ultimately controlled by WSLS). *Observe that for both WSLS and S-WSLS the ratio of cooperators in the network soon converged between* 0.38 *and* 0.45. Second, we examined the impact of the $b/c$ ratio presented at the right part of Figure 5.6. Since both WSLS and S-WSLS perform similarly, we only focus on WSLS. *Observe that WSLS does stimulate cooperation, but only to a limited*

[2]We can see this if we express WSLS as a Markov chain. Even though the state space has size exponential to the number of nodes, it can be easily shown that there is always a non-zero probability of changing state. Hence, the chain has no absorbing state.

*extent.* Even with high a $^b/_c$, the ratio of cooperators has an upper limit of 0.5. Other researchers [43] also made the same observation about this upper-limit. However, they missed the effect of the $^b/_c$ ratio on this upper limit. We conclude that although (S-)WSLS does not offer a fully cooperator network, it still keeps a fraction of the network as cooperator. The cooperator fraction reaches at most 0.5 and the $^b/_c$ ratio is the most significant parameter, while the initial fraction of cooperators does not make any difference.

### 5.7.2. Evolution of Networks under SIBS

As discussed in Section 5.2, with SIBS, spatial structures promote the survival and spread of cooperators. While the network evolves, each strategy group, cooperators and defectors, gradually establishes clusters. As evolution proceeds, while defector clusters drop to zero fitness, since none of the cluster members help each other, cooperator clusters increase their fitness. Consequently, defectors also switch to cooperation to improve their fitness. To guarantee the invasion of cooperators, Ohtsuki et al. [73] proved that the benefit-to-cost ratio should be higher than the neighbor degree ($^b/_c > |\mathcal{V}|$), and that the strategy updates should be asynchronous.

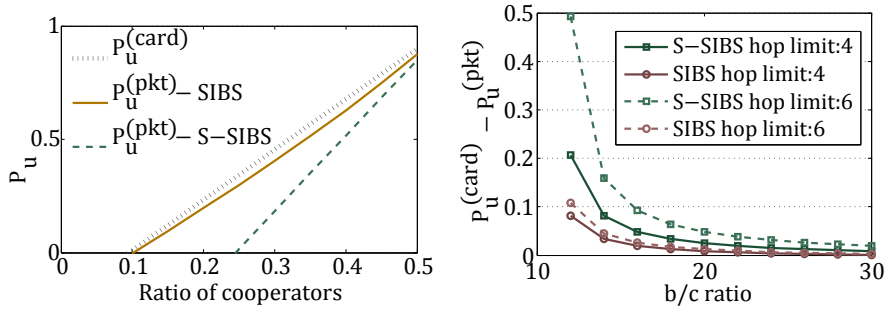
In the following we show that, when we use packet-based fitness, the $b, c$ and $|\mathcal{V}|$ parameters are not the only key factors. Network and traffic parameters such as the path length, routing scheme (See Section 5.7.2), and mobility (See Section 5.8) also influence the evolution process.

#### The effect of the routing

Altering the definition of fitness from cardinality- to packet-based introduces a dependency between the meta strategy's behavior and the routing protocol. Below, we examine this dependency for random walk and selective random walk routing. Our analysis suggests that, with a packet-based fitness, the cost of cooperation becomes a function of the path length and routing type. This result has an important consequence: to ensure that evolution leads to the invasion of cooperators, we need to take into account the characteristics of the routing protocol used.

**SIBS.** Consider the random walk routing scheme, where all the neighbors of a node $u$ are candidates as a next hop without any restriction and suppose that each node generates packets with a fixed rate $\lambda$. While a defector pays no cost (as before), cooperators are burdened with relaying not only the packets injected by their neighbors, but also packets relayed over multiple hops. According to Proposition 1 this phenomenon increases the cost of cooperation.

**Proposition 1.** ***(Random Walk)*** *Consider a network in which nodes generate*




(a) $P_u^{(pkt)}$ analysis with $b/c = 10$, $\eta = 4$

(b) Difference of $P_u^{(card)}$ and $P_u^{(pkt)}$ at $R_c = 0.5$ and with different hop limits (path length) and $b/c$ ratios.

Figure 5.7: $P_u^{(pkt)}$ analysis is visualized and compared to $P_u^{(card)}$. *Cost of being cooperator increases with hop limit and selective routing.*

5

*packets with rate $\lambda$ and forward them (using random walk routing) for $\eta$ hops. Furthermore, let $r$ be the expected ratio of cooperator neighbors of any node $u$. Assuming that there is no packet loss due to congestion or collisions, the expected fitness $f$ and $g$ of a cooperator and defector node, respectively, are*

$$\max\{\lambda\big(br - c(\eta-1)\big), 0\} \leq f \leq \max\{\lambda\big(br - c\frac{1-r^{\eta-2}}{1-r}\big), 0\} \tag{5.5}$$

$$g = \lambda br. \tag{5.6}$$

*proof.* Consider a node $u$. To compute bounds on $u$'s fitness, we will count the packets that $u$ relays when it is a cooperator. To begin with, $u$ relays the packets of its neighbors. If each of its neighbors generates and forwards randomly $\lambda$ packets per round, on average $u$ receives $\lambda|\mathcal{V}_u|/|\mathcal{V}_u| = \lambda$ packets to relay. In addition to the packets injected by its neighbors, $u$ also receives the packets relayed by its cooperator neighbors. Let $R_u = |\mathcal{C}_u|/|\mathcal{V}_u|$ a random variable. describing the ratio between $u$'s cooperators and defector neighbors (here $|\mathcal{C}_u|$ and $|\mathcal{V}_u|$ are also random variables). The expected number of relayed packets of each cooperator node resembles a geometric series:

$$E[\mathcal{R}_u] = E[\lambda(1 + R_u + R_u^2 + \ldots + R_u^{\eta-2})] = \lambda \sum_{k=0}^{\eta-2} E[R_u^k]$$

Since $R_u > 0$ and $h(x) = x^k$ is a convex function for $x \geq 0$, Jensen's inequality gives

$$E[R_u]^k = h(E[R_u]) \leq E[h(R_u)] = E[R_u^k] \,, \tag{5.7}$$

and therefore

$$E[\mathcal{R}_u] \geq \lambda \sum_{k=0}^{\eta-2} E[R_u]^k = \lambda \frac{1 - r^{\eta-2}}{1 - r} \,, \tag{5.8}$$

where $r = E[R_u] < 1$. We can also obtain a trivial upper bound by assuming that all the $\eta$ hops traffic passes over a cooperator:

$$E[\mathcal{R}_u] \leq \lambda(\eta - 1) \,. \tag{5.9}$$

Substituting the above bounds in (5.4) we obtain the desired result.

□

5

We can see that both the path length $\eta$ and the packet generation rate $\lambda$ can increase the overall packet traffic. However, only the path length incurs a higher cooperation cost: whereas $\lambda$ affects both $f$ and $g$ linearly, $\eta$ only affects cooperators.

**S-SIBS:** In addition to path length, the next hop selection procedure also affects the cost of cooperation.

**Proposition 2.** ***(Selective Routing)*** *Consider a network in which nodes generate packets with rate $\lambda$ and forward them (using **selective** random walk routing) for $\eta$ hops. Furthermore, let $r$ be the expected ratio of cooperator neighbors of any node $u$. Assuming that there is no packet loss due to congestion or collisions, the expected fitness $f$ and $g$ of a cooperator and defector node, respectively, is*

$$0 \leq f \leq \begin{cases} 0 & , \; r = 0 \\ \max\{\lambda b - c(\eta-1)\lambda/r, 0\} & , \; otherwise \end{cases} \tag{5.10}$$

$$g = \begin{cases} 0 & , \; r = 0 \\ \lambda b & , \; otherwise. \end{cases} \tag{5.11}$$

*Proof.* Follows in a similar manner as the proof of Proposition 1, with the difference that, because the $\lambda|\mathcal{V}_u|$ packets generated in a neighborhood are sent uniformly only to cooperator nodes, a cooperator node receives $\lambda|\mathcal{V}_u|/|\mathcal{C}_u| = \lambda/r$ packets (instead of $\lambda$). □

*Similar to path length, selective routing also rises the cost of being a cooperator and even with a higher rate.*

We visualize the analysis in Figure 5.7 by providing figures of $P^{(pkt)}$ and $P^{(card)}$ with different path lengths and routing schemes. First, notice that all packet-based fitness plots ($P_u^{(pkt)}$) indicate a lower probability of choosing cooperation than that of cardinality-based ($P_u^{(card)}$) (See Figure 5.7(a)). Second, observe the difference of selective and plain random walk routing. Selective routing significantly increases the cost of being a cooperator. To see the effect of $b/c$ ratio and the tendency towards defection, we depict $P_u^{(card)} - P_u^{(pkt)}$ with $R_c^{init} = 0.5$ in Figure 5.7(b). Fortunately, high $b/c$ ratios decrease the tendency to defect.

Though the analysis does not consider many wireless network challenges, such as packet drops due to collision and congestion, it reveals a main obstacle to the spread of cooperators. Packet-traffic based fitness calculation leads to higher cooperation costs. Cooperators pay more cost when the routing favors cooperators and the increase in path length augments the relayed traffic. We conclude that in addition to the neighbor degree constraint given in the $b/c > |\mathcal{V}|$ requirement, the routing and the path length are also crucial.



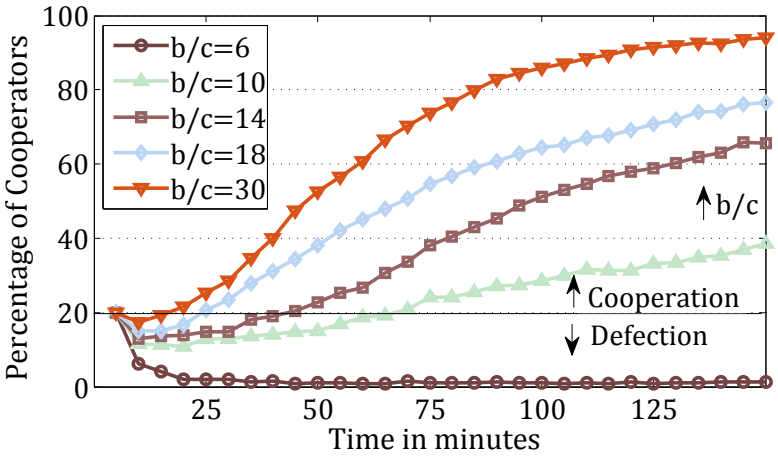
### Experimental evaluation

In this section we demonstrate that while $b/c > |\mathcal{V}|$ holds, the increased burden on the cooperators due to packet traffic slows down the spread of cooperators. Selective routing demands higher $b/c$ ratios for the spread, whereas an unexpected (but fortunate) effect is observed in the case of path length. *Though the cost of cooperation does increase with the path length, it does not necessarily suppress the spread of cooperators.*

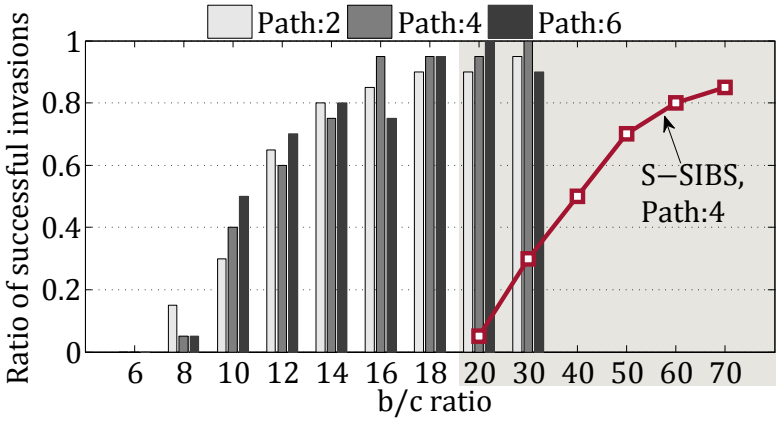
We simulated the SIBS and S-SIBS meta strategies to validate the $b/c > |\mathcal{V}|$ rule, and to observe the effects of different $b/c$ ratios and path lengths. We use the same setup in Section 5.5.1. Each simulation is composed of 20 repetitions and lasts 150 *minutes*. To resemble real life, the rounds are randomized in [200, 300] *seconds* intervals, which also provides asynchrony.

**The $b/c > |\mathcal{V}|$ rule.** Our first objective is to validate the $b/c > |\mathcal{V}|$ rule. Figure 5.8(a) presents the change of the network over time with a fixed path length $\eta = 4$. A initial ratio of cooperators $R_c^{init} = 20\%$. A final ratio of cooperators ($R_c^{end}$) that is below the $R_c^{init}$ indicates the spread of defectors. *The first significant outcome is that $b/c > |\mathcal{V}|$ holds even with packet-traffic based fitness.* The $b/c$ ratio of 6, which is lower than $E[\mathcal{V}] \approx 10$, leads to a defector network while higher ratios encourage cooperation. The second outcome is that the convergence to a fully cooperative network is faster with higher $b/c$ ratios.

**The path length.** But what is the effect of the path length? Analysis has shown




(a) Evolution of SIBS network over time with path length $\eta = 4$. $R_c^{init}$ is indicated with a solid line, over which cooperators are assumed to invading the network.




(b) Ratio of (S-)SIBS-based simulations (out of 20) that are categorized as successful spread of cooperators by $R_c^{end} > R_c^{init}$ metric. The shaded area indicates the larger scale of the X-axis. Unless stated, SIBS is the default meta strategy.

Figure 5.8: Evolution of cooperators under (S-)SIBS strategy with different path lengths and $b/c$ ratios.

that the path length increases the costs of cooperators. Thus, it is possible to observe

a suppression of cooperation spread. To indicate the effect of the path length, we plot Figure 5.8(b). First consider only SIBS (bar graphs). *Fortunately, we do not observe a suppression with longer path lengths in the experiments.* With $b/c = 10$, the path length even promotes the spread. For other $b/c$ ratios (12, 14 etc. ), it is hard to claim a similar effect.

The reason for this counter-intuitive effect (i. e., path length increases the costs of cooperators, but does not limit the spread) is the asynchronous update mechanism. Suppose that a neighbor alters its strategy from cooperation to defection. When the neighbor was cooperator, even for a short duration, it might relay quite a few packets. Consequently, it is still recognized as a cooperator after switching to defection. This misconception is exacerbated by longer path lengths, since there exist more packets to relay than to inject. *As a result, we do not observe any suppression effect from the path length.*

**Selective routing.** To show the effect of routing, we depict the evolution of the network under the S-SIBS meta strategy as a separate line graph in Figure 5.8(b). Notice that the $b/c$ ratios in the x-axis has a different scale and the evolution starts at almost twice of $|\mathcal{V}|$. The reason for higher $b/c$ ratios is that the cooperators are punished more with selective routing. *Therefore, only an increase in the $b/c$ ratio compensates the increased cost.*

**The optimism threshold.** The last parameter that has significant effect is the optimism value. We consider the optimism value as a parameter to control the tendency towards cooperation. We have repeated the simulations with a more pessimistic value of $|\mathcal{I}_u|/|\mathcal{R}_u| < 1$ (default was 2). We observed a decrease in the spread of cooperators. For instance, the ratio of successful invasions that reached 0.4 / 0.8 with $b/c = 10$ / 16 respectively (Figure 5.8(b)), stayed at 0 / 0.4 for the pessimistic threshold. Therefore, the optimism threshold should be high for having a cooperator network. However, there is a trade off; when nodes are too optimistic, free riders can easily be recognized as cooperators if they relay just a few packets.

**A testbed experiment.** To validate our simulations in a real topology with realistic wireless channel, we deployed all the meta strategies to our real-life testbed with $tx = -20dbm$ (see Section 5.4.1). All the experiments start with a 20 percent cooperator ratio and continue for 2 hours. Figure 5.9 presents the average $R_c^{end}$ of the cooperators. Similar to our simulations, both SIBS and S-SIBS promote cooperation when $b/c$ ratio is close to and higher than the neighbor degree. WSLS and S-WSLS, on the other hand, are not influenced by the $b/c$ ratio and keep the network with $\approx 50\%$ cooperators. A notable difference from simulations is the existence of

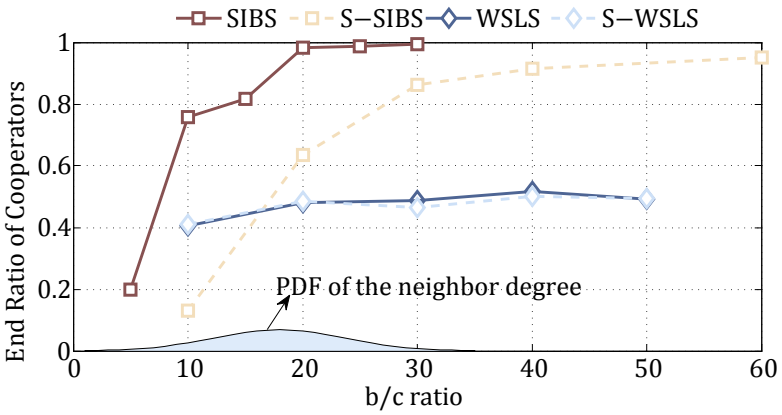



Figure 5.9: Real testbed experiment on the evolution of meta strategies. We present the end ratio of cooperators after 2 hours of experiment with different $b/c$ ratios. Results are the average of five repetitions. $R_c^{init} = 20$, $\eta = 4$.



cooperators even with $b/c$ ratios lower than $E[\mathcal{V}] = 17.9$. The reason is the high variance of the neighbor degree due to the rectangular shape of the testbed (see the probability density function in Figure 5.9).

## 5.8. Mobility

In this section we investigate the effect of mobility on both the adaptation and evolution of meta strategies. To observe the effect of mobility, we added the Random Waypoint mobility model from BonnMotion [8] to our simulations with node speeds 0.5 to 5 $m/s$ (radio range is 50 $m$). Since in this new setup the average link duration drops exponentially from 133 to 20 seconds, we decreased the overhearing period to $30 - 60$ seconds (from $200 - 300$ seconds used previously) to ensure a timely response. With $30 - 60$ seconds, the relative error of fitness in WSLS gets as high as 0.2, while the estimation error of SIBS is still lower than 0.04 (see Figure 5.2).

**Adaptation.** Similar to Section 5.5.1, we evaluate the adaptation of SIBS and WSLS by deploying and tracing the fitness of an oblivious node in an environment with diverse defector ratios. By comparing the ratio of cooperative rounds for the static and mobile cases, *we can see that both SIBS and WSLS still detect the fittest strategy* (Figure 5.10). Nevertheless, while the ratio of cooperative rounds is close to the stationary case under WSLS, nodes tend to follow defection in SIBS at higher speeds.

**Comparison of TFT, GTFT, SIBS and WSLS under mobility**. Mobile

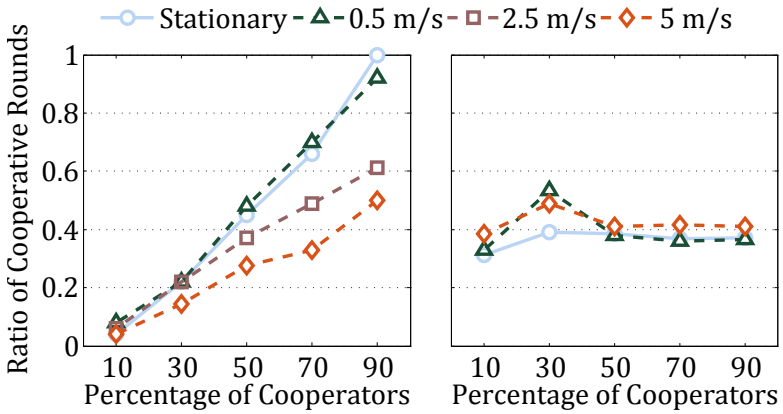



Figure 5.10: Ratio of cooperative rounds of an adaptive node with different ratios of defectors. The median values are presented for SIBS (LEFT) due to variance, while averages for WSLS (RIGHT).

Table 5.3: Comparison of TFT, GTFT, WSLS and SIBS in a defector mobile network. The metric is average number of relayed packets –of only free riders– per round.

| **speed** ($^m/_s$) | **WSLS** | **SIBS** | **TFT** | **GTFT** |
|---|---|---|---|---|
| 0.5 | 2.06 | 0.68 | 1.81 | 3.74 |
| 2.5 | 0.59 | 0.23 | 1.26 | 1.49 |
| 5 | 0.37 | 0.09 | 0.94 | 1.11 |

environments are similar to Sybil attacks such that nodes frequently encounter new nodes. For instance, if a TFT-based node's neighbor list cannot store all the past neighbors, a free rider can whitewash itself by disappearing for some time. With different speeds, we deployed a network of 80% defectors and 20% cooperators with one oblivious node (where there are 100 nodes, but neighbor lists can hold up to 50). The average number of relayed packets of defectors per round is presented in Table 5.3. Firstly, note that the number of relayed packets decreases with higher speeds, since neighbor lists store more invalid neighbors which destroys the routes. Second, observe that in all scenarios, SIBS relay fewer packets of defectors than the rest. Since higher speeds resemble Sybil attacks, (G)TFT performs worse than SIBS and even WSLS over 2.5 $^m/_s$. We conclude that, *though SIBS does not promote cooperation in a mobile environment, still its performance is better than direct-reciprocity methods, TFT and GTFT.*

**Evolution.** When we consider network evolution, our simulation experiments showed that *WSLS maintains the network with a* $40 - 45\%$ *cooperator ratio as*

*in the stationary case.* On the other hand, apart from the slowest speed of $0.5\,^{m}/_{s}$, *SIBS leads to a fully defector network under mobility.* The first factor promoting defection is the tendency to defect as described in the previous paragraph. The second and the most significant effect is the destruction of the cooperator clusters. Without the clusters, cooperation cannot be sustained. Defectors penetrate into a cooperator cluster and show their neighbors that defection offers better fitness.

To conclude; we should admit that in mobile scenarios SIBS cannot promote cooperation. However, both SIBS and WSLS still detect the fittest strategy in their neighborhood and even perform better than TFT and GTFT. Moreover, WSLS sustains cooperation in the network similar to the stationary scenarios.

## 5.9. Related Work



Though classical game theory identifies defection as the equilibrium under the forwarder's dilemma, there are several mechanisms, with which we observe the survival of cooperation. As explained in Chapter 1, these mechanisms are: *Kin Selection*, *Direct Reciprocity*, *Indirect Reciprocity*, *Network Reciprocity* and *Group Selection.* Among these, *Direct Reciprocity* and *Indirect Reciprocity* are famous in the wireless networks research community. However, they mainly depend on identity information.

**Direct reciprocity** depends on the repeated interaction of nodes and follows the *"You scratch my back and I'll scratch yours"* principle. However, in a public wireless network, random encounters may not suggest a future interaction [30]. Additionally, identity information is strictly required. In order to punish identity changes, as a bootstrapping phase, every node should "pay their dues" before injecting packets. However, a node may need immediate operation or it may employ fake identities to protect its privacy. On the other hand, *our meta strategies tolerate fake identities and do not require a bootstrapping phase.*

**Indirect reciprocity** employs *reputations* to reward cooperation. The reputation gained by cooperative behavior is later spent for getting help from any node. Contrary to direct reciprocity, indirect reciprocity can prevail even in the case of random encounters. However, the distribution of reputation in a decentralized system is quite challenging. There is an abundance of different proposals like CORE [62], SORI [42] and others [50, 59]. Indirect reciprocity requires true identity information. Therefore, slandering attacks like good/bad mouthing and whitewashing are hard to avoid. Free-rider nodes retaliate against other nodes that report the node as defector by announcing those nodes as defectors too. In our work, we do not depend on reports from neighbors and hence, our meta strategies are not susceptible

to slandering attacks. Moreover, accumulating reputation and distributing it takes time and leads to energy consumption in a bootstrapping phase [32], whereas *our scheme does not need extra message exchange for any kind of information.*

Yu et al. [101] present a thorough survey on counter measures against Sybil attacks in different platforms. Adding visual contact among the peers [20] for identity is one of the many proposals. A rather interesting work [78], which is totally decentralized, uses mobility to discover attackers. In sparse networks, by overhearing the messages and using machine learning, a node discovers groups of identities that always move together. Those identities are considered as one attacker spawning several identities. Though promising, this only works in sparse networks and the attacker should use several identities at the same time. If the attacker never uses the same identity again, it cannot be detected. In our work, similar techniques can be used to improve the cooperator/defector classification, which currently only depends on the optimism threshold.



## 5.10. Conclusions

Cooperation enforcement for the forwarder's dilemma depends on identity information, which is easy to obfuscate in ad hoc wireless networks. Computationally powerful *smart* nodes can guarantee the identity of their neighbors. However, nodes with low computational power have to stay *identity oblivious*. Since identity-oblivious nodes cannot choose which node to retaliate, they follow pure defection to avoid exploitation. Unfortunately, pure defection may bring isolation since it is recognized by the smart nodes. Therefore, an identity-oblivious node surrounded by smart nodes should discover that cooperation is more rewarding in that case.

In this chapter, we have modified and adapted two meta strategies, SIBS and WSLS, from multi-agent systems so that identity-oblivious nodes still discover the best strategy in their neighborhood. In SIBS, collective fitnesses of cooperator and defector neighbors are compared for the strategy decision. On the other hand, in WSLS, a node switches its current strategy if its own fitness degrades with respect to the previous one. For a better fit to wireless networks, we have changed the definition of fitness measure from neighbor cardinality to packet traffic. For counting the packet traffic, we proposed a novel two-hop overhearing method, which has an accuracy of 96% in $P^{(pkt)}$ estimation.

**SIBS or WSLS?** Our experiments have demonstrated that SIBS and WSLS are able to adopt the locally-fittest strategy. They are both resilient to Sybil attacks. Moreover, SIBS relays fewer packets of free riders than TFT and GTFT under Sybil attacks and mobility. With respect to SIBS, WSLS is rather easy to implement and

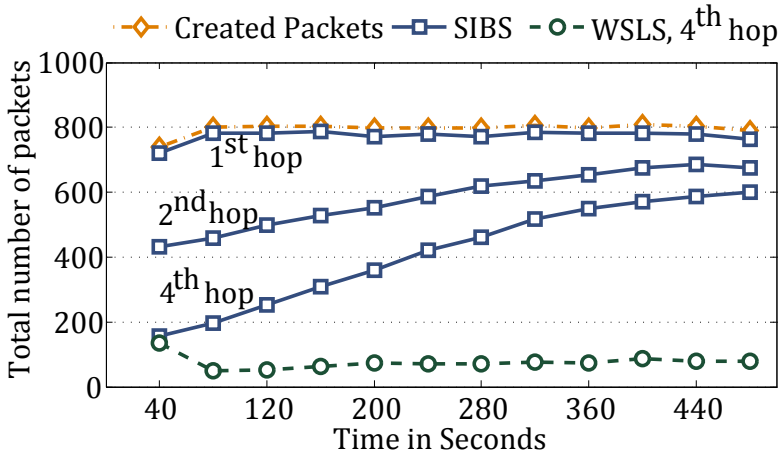



Figure 5.11: The number of packets that reach to a certain hop are presented. $b/c = 50$ and the network starts with 50% defector, 50% cooperator ratio. Packets reach further away with SIBS since *(i)* it promotes cooperation, *(ii)* it has better strategy stability.

Table 5.4: Comparison of WSLS and SIBS

| | **WSLS** | **SIBS** |
|---|---|---|
| $b/c$ ratio | Higher is better | Higher is a "must" |
| Other constraints | None | $\|\mathcal{V}\|$, optimism, routing, mobility |
| Overhearing cost | Only its traffic | All traffic |
| Innovative | ⊠ | □ |
| Evolution | At most half | Full network |
| Space/Time Complexity | $O(1)$ | $O(\|\mathcal{V}_u\|)$ |
| Strategy stability | ≈ 5 consecutive rounds | ≈ 19 consecutive rounds |

more responsive to attacks. Moreover WSLS is innovative, such that it can switch to a non-existent strategy in a neighborhood. For instance, even if all neighbors are cooperative, a WSLS node can switch to defection, whereas SIBS is stuck with the cooperation. Thus, a network with WSLS nodes converge to neither a full defection nor full cooperation state. As a result, while WSLS can transform a network to at most 50% cooperation, SIBS can transform the whole network to full cooperation. Moreover, the strategy stability of nodes, which indicates the persistence of a node following the same strategy consecutively, is higher for SIBS (See Table 5.4). Strategy stability of nodes is crucial for packet networks since

switching strategies frequently destroys the packet routes. With $b/c$ ratios of 20 and 30, the average number of rounds that a node stays in cooperation state is 16.17 and 19.14 for SIBS whereas 5.01 and 5.66 for WSLS, respectively. When we consider the actual path lengths that a packet can travel, SIBS is more promising. As shown in Figure 5.11, packets get more help with SIBS. Unfortunately, mobility has a detrimental effect on the traffic as it inhibits the spread of cooperation, but still SIBS can discover the fittest strategy.

To conclude, SIBS offers a fully cooperative network and hence, improves the network performance more than WSLS. However, SIBS is exposed to more constraints and spends more energy in packet overhearing. Nevertheless, both meta strategies protect the identity-oblivious nodes against exploitation by discovering the locally-best strategy.

# 6

# Conclusions

*Bir elin nesi var, iki elin sesi var.*
*One finger cannot lift a pebble.*

When devices cooperate, they can form large systems for the benefit of their users, which they cannot accomplish by themselves. Unfortunately, if altruistic acts increase costs or deplete resources such as a scarce one like energy, devices tend to defect. To enhance cooperation among devices, first, we investigated the incentives of device owners for cooperation. Then, we proposed autonomous systems where devices cooperate without human intervention. Briefly, the three main outcomes of the thesis are as follows:

- Social relations are the most significant criterion for cooperation,
- Decentralized social-device networks, which we proposed, can automate access control and provide secure-by-default IoT systems,
- In the forwarder's dilemma, when identity cannot be secured, constrained devices can avoid exploitation and promote cooperation by observing the traffic in their locality.

In more detail, the contributions of each chapter are given in the next subsections.

**Chapter 2.** Although cooperation mechanisms had already been identified, preferences of people were not investigated. With a questionnaire based on a mobile tethering application, we revealed that consumers mostly care about social relations while they are concerned with security in cooperation. That is, consumers want to share their cellular connection with familiar people (e.g., family, friends, and co-workers) and be sure that they are not vulnerable to any security attacks. Moreover, our technical analysis on mobile tethering in different platforms indicated the high power consumption. Due to the lack of power-saving features in the connection provider, energy costs of sharing cellular connection is high. For the wide adoption of mobile tethering, energy consumption of connection providers should be decreased.

**Chapter 3.** Today's operational service discovery protocols carry simple text-based uniform resource identifiers that are not expressive enough. Machines cannot comprehend the meaning of a new service that is not in their knowledge base or *cannot request services based on its owner*. In this chapter, we proposed a new service discovery protocol that is **(i)** more expressive via the use of ontologies, **(ii)** application layer independent since designed as an extension to the neighbor discovery protocol and **(iii)** capable of carrying existing discovery protocols. We demonstrated that interoperability with legacy protocols is easy to achieve such that existing libraries can be employed without any modification. We also indicated the importance of limiting the rate of discovery messages because the packets of the carrier protocol, ICMPv6 multicast, are dropped by the routers when the rate is high.

**Chapters 4.** The main contribution of this chapter is the creation of a decentralized social-device network (DSDN) for automated cooperation. Social networks were involved in access control, for instance, an access point learns the owner of a client device and allows access to network if their owners trust each other. Moreover, complicated security setups are replaced with a simple ownership declaration without passwords, which is scanning a QR code or NFC/RFID tag. Additionally, we addressed the social network search problem, whose worst case time complexity is quadratic for indirect relations. For an unconstrained device such as an access point, we demonstrated that WiFi beacons can be incorporated to discover proximity and limit the search space. Moreover, for constrained devices, we proposed an architecture and modified security standards to delegate the social network search. Real-life experiments showed that delegation works with as low as 7% communication overhead and a typical latency of less than a second.

**Chapter 5.** Without guaranteed identity information neither DSDN nor (in)direct

reciprocity mechanisms work since punishments and rewards depend on identities. Unfortunately, highly constrained devices do not have enough resources for cryptographic operations in securing identities. A constrained device can either cooperate always and consequently may be exploited by free riders or defect always and may be punished by unconstrained devices. In this chapter, we analyzed and experimented with two local-observation based cooperation meta-strategies that can be employed by constrained devices, which are Stochastic Imitate Best Strategy (SIBS) and Win Stay Lose Shift (WSLS). These two meta-strategies discover the best local strategy (cooperation or defection) by observing neighborhood traffic without any help from neighboring nodes. We proved that even in the absence of identity information these two methods protect a constrained device from exploitations and punishments while they still promote cooperation. Real-life experiments on over hundred devices and simulations show that WSLS operates with less information than SIBS as well as consuming less energy. Nevertheless, with SIBS cooperation prevails throughout the network while with WSLS only half of the network becomes cooperative. As a result, with SIBS, packets can travel over more hops than WSLS.



## 6.1. Discussion and Future Work

To position our access control architecture, which is based on a decentralized social-device network (DSDN), and highlight the differences from current practices. Before discussing the pros and cons of each architecture, we should indicate a common trend, which is addressing devices via URIs. Instead of assigning local addresses, for global access every device is given a URI. Secondly, we assume that each of these architectures has the capability of social network integration. A brief evaluation of each architecture is as follows:

**The current architecture** is fully centralized where a cloud service acts as a proxy or man-in-the-middle. A centralized architecture is convenient since all the information is collected at and served from one place. However, it raises concerns about the security, there is neither end-to-end confidentiality nor privacy. Central services can view all sensory information as well as social network interactions.

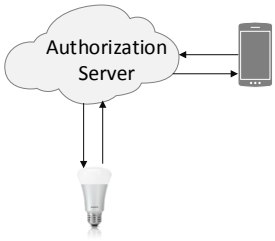



Figure 6.1: Current

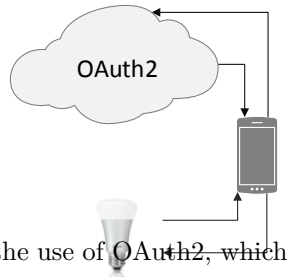



Figure 6.2: Expected

**The expected architecture** is derived from ongoing standardization efforts of the IETF Authentication and Authorization for Constrained Environments (ACE) working group. When this thesis was compiled, the standard had not been ready yet and there were many other candidates. Here, we chose the most probable one due to the use of OAuth2, which is currently the most practiced single-sign-on standard. *This architecture allows end-to-end confidentiality, whereas it does not scale well, and violates privacy by observing real-life social network interactions.* The client device (smartphone) obtains a token from an OAuth2 provider and passes it to the resource server (light bulb) for access. Thus, the OAuth2 provider cannot eavesdrop IoT traffic, it only authenticates and authorizes the client. However, in this architecture, the OAuth2 provider obtains the information of device-to-device and hence, human-to-human interactions since the provider grants the access instead of the bulb. With such information, physical contacts of people are revealed, which is much more informative than current online social networks (OSN). The reason is that OSNs do not necessarily reflect real-life interactions. Finally, Internet connectivity is a must in OAuth2-based architectures; when disconnected a resource server cannot make a decision. (We discuss scalability problems in the next item).



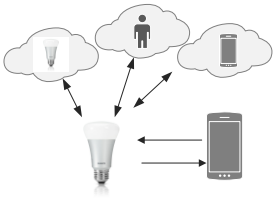


Figure 6.3: Our approach: DSDN

**In our approach of DSDN-based access control**, all the control is pulled down to the devices. Thus, in addition to providing end-to-end confidentiality, all the device interactions are kept hidden. The resource server (light bulb) receives the URI of the client (smartphone) and crawls the social profiles for authentication and authorization. Social profile services on the Internet do not know who interacts with whom. Moreover, offline operation (disconnected to Internet) is also possible since a resource server can memorize its owner's social network beforehand.

With respect to scalability, unlike the OAuth2 architecture where there are only a few providers, our architecture does not have any restriction on social profile servers. Any web server can act as a social profile by simply providing social network information in a machine-readable format such as JSON and XML. In OAuth2, trust for a provider is a manual task, that is a consumer has to explicitly declare its trust for every new OAuth2 provider. Therefore, only a few OAuth2 providers like Google+, Facebook and Twitter exist.

Finally, the drawback of our approach is the demand for more resources; not all con-

strained devices are capable of crawling social networks. In Chapter 4, we demonstrated that social network search can be delegated to the cloud. However, it increases the dependency on the Internet connection. As a future work, we argue that trusted devices should be able to exchange social network information in a local area network.

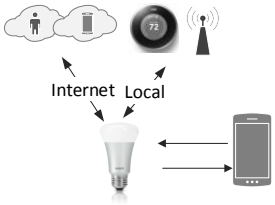


Figure 6.4: Future of DSDN

**The future of our approach** is even more decentralized in that social network information will be searched without Internet connection. Trusted devices, such as an access point and a light bulb of the same person in the same local network, will assist each other in authentication and authorization of clients. Thus, authentication and authorization will be faster, and more robust to intermittent Internet connection.

In addition to enhancing DSDN by distributing social profiles among trusted neighboring devices, there is still room for improvement in several aspects. Firstly, the complexity of decentralized social network crawling should be decreased. In Chapter 4, we proposed the use of WiFi beacons for proximity detection and limiting the search space. For different deployments, we need similar approaches for limiting the search space. Moreover, as a new feature, replicating social networks in constrained devices should be added. We should note that simply copying a whole social network of a person is not a solution since constrained devices cannot hold all the relations.



Secondly, we consider that anomaly detection for revealing compromised resources also requires attention. We mentioned several times that IoT carries sensitive information and hence, confidentiality is of paramount importance. Our proposals avoid many attacks and easily recover from compromised devices by simply removing the ownership relation. However, the detection of a compromised device is still an open question. We need anomaly detection systems for revealing attacks and fix breaches.

Finally, serving social networks in clear text may be considered as a privacy issue for some consumers. In DSDN social profiles are published in clear text for crawlers such that everyone can see each other's social network. The solution is keeping social networks partially in clear text and protecting access via access-control-lists (ACL). When two direct friend's devices interact, both parties may exchange tokens for giving a peer access permissions to their own social profiles. Unfortunately, this restriction does not work for indirect relations since none of the peers can generate a token for the third person's social profile. Nevertheless, we should note that currently online social networks also keep friendship information public. Moreover, regardless of architecture when two devices from different social networks interact, discovering a relation is only possible if these social networks cooperate–they reveal their member's network. Either all IoT devices will be served from the same central service or social networks will (have to) be partially public.

To conclude, in this thesis we start from analyzing cooperation to comprehend

the motivations of people and continue with applying our findings to automate cooperation in IoT. Our proposal, DSDN, promotes cooperation over social relations as well as decreases the burden of consumers in securing their devices. Moreover, we are not silent about highly constrained devices that cannot employ DSDN or any other (in)direct reciprocity mechanisms due to the lack of identity information. We proposed two meta-strategies that constrained devices can employ and discover the best strategy (cooperation or defection) in their locality.

# 7

# Acknowledgments

*Well, I've been down so Goddamn long*
*That it looks like up to me*
*Well, I've been down so very damn long*
*That it looks like up to me*
*Yeah, why don't one you people*
*C'mon and set me free*

Jim Morrison

Doing a PhD takes a long time, terribly long. The newest and trendy technologies at the beginning becomes ordinary and old-fashioned by the end. When I started, Android programming was a must skill, now Arduino programming and data science are a must. Back then IoT was at its infancy, named as sensor networks. Today IoT is at the top of the Gartner's Hype index; as you may guess, "IoT" in the title was not planned at the beginning.

Through the PhD process, my personal and work life have changed a lot as well. I moved to another country–luckily english friendly–, lost 9 kg as well as my hair and got married. At work, unexpected events occured including the shutdown of my initial department. However, that crisis became an opportunity; my new group turned out to be full of brilliant and inspiring people. Apart from the university, I also met many expats–"gurbetçi" in Turkish–, who are all highly educated, funny,

and fascinating. Though it is hard to make friends after a certain age, I was able to make a few thanks to the Dutch beer.

To conclude, I would like to thank to all my family, friends, and colleagues. They are so brilliant that by 2030, they will probably be ruling the earth, so get along with them. I assure you that the world will be a better place.

Fethiye, TURKEY
11/Aug/2015

# Bibliography


[1] NI USB-6009, multifunction DAQ. Last accessed 14.2.2013. [Online]. Available: http://sine.ni.com/nips/cds/print/p/lang/en/nid/201987

[2] (2012) Android wi-fi tether. Last accessed 14.2.2013. [Online]. Available: http://code.google.com/p/android-wifi-tether/

[3] (2012) Open garden. Last accessed 14.2.2013. [Online]. Available: http://opengarden.com/

[4] (2013) More than 30 billion devices will wirelessly connect to the internet of everything in 2020. ABI Research. London, UK. [Online]. Available: https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne

[5] L. Adamic and E. Adar, "How to search a social network," *Social Networks*, vol. 27, no. 3, pp. 187 – 203, 2005.

[6] O. Aksoy, "Essays on social preferences and beliefs in non-embedded social dilemmas," Ph.D. dissertation, Utrech University, 2013.

[7] T. Anantvalee and J. Wu, "Reputation-based system for encouraging the cooperation of nodes in mobile ad hoc networks," in *IEEE International Conference on Communications, 2007. ICC '07.*, june 2007, pp. 3383 –3388.

[8] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "Bonnmotion: A mobility scenario generation and analysis tool," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '10. Brussels: ICST, 2010, pp. 51:1–51:10.

[9] L. Atzori, A. Iera, and G. Morabito, "From 'smart objects' to 'social objects': The next evolutionary step of the internet of things," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 97–105, January 2014.

[10] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (siot), when social networks meet the internet of things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594 – 3608, 2012.

[11] S. Bansal and M. Baker, "Observation-based cooperation enforcement in ad hoc networks," *ArXiv*, 2003.

[12] C. E. Batt and J. E. Katz, "A conjoint model of enhanced voice mail services.

implications for new service development and forecasting," *Telecommunications Policy*, vol. 21, no. 8, pp. 743 – 760, 1997.

[13] T. Berners-Lee and D. Connolly. (2011, March) Notation3 (n3): A readable rdf syntax. World Wide Web Consortium. [Online]. Available: www.w3.org/TeamSubmission/n3/

[14] K. Binmore, "Social norms or social preferences?" *Mind & Society*, vol. 9, no. 2, pp. 139–157, 2010.

[15] R. Bonetto, N. Bui, V. Lakkundi, A. Olivereau, A. Serbanati, and M. Rossi, "Secure communication for smart IoT objects: Protocol stacks, use cases and practical examples," in *2012 IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2012, pp. 1–7.

[16] R. Botsman. (2012) Collaborative consumption hub. Last accessed 14.2.2013. [Online]. Available: http://collaborativeconsumption.com/

[17] Y.-D. Bromberg and V. Issarny, "Indiss: interoperable discovery system for networked services," in *Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware*, ser. Middleware '05. New York, NY, USA: Springer-Verlag New York, Inc., 2005, pp. 164–183.

[18] M. N. Burton-Chellew and S. A. West, "Prosocial preferences do not explain human cooperation in public-goods games," *Proceedings of the National Academy of Sciences*, vol. 110, no. 1, pp. 216–221, 2013.

[19] L. Buttyan and J.-P. Hubaux, *Security and Cooperation in Wireless Networks*. Cambridge University Press, 2007, no. ISBN 9780521873710.

[20] S. Capkun, J.-P. Hubaux, and L. Buttyan, "Mobility helps peer-to-peer security," *IEEE Transactions on Mobile Computing*, vol. 5, no. 1, pp. 43–51, Jan 2006.

[21] D. Compeau, B. Marcolin, H. Kelley, and C. Higgins, "Research commentary-generalizability of information systems research using student subjects-a reflection on our practices and recommendations for future research," *Information Systems Research*, vol. 23, no. 4, pp. 1093–1109, 2012.

[22] M. Constantinescu, E. Onur, Y. Durmus, S. Nikou, M. de Reuver, H. Bouwman, M. Djurica, and P. Maria Glatz, "Mobile tethering: overview, perspectives and challenges," *info*, vol. 16, no. 3, pp. 40–53, 2014.

[23] M. Cunche, M.-A. Kaafar, and R. Boreli, "I know who you will meet this evening! linking wireless devices using wi-fi probe requests," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, 2012, pp. 1–9.

[24] L. M. Dan Brickley, *FOAF Vocabulary Specification 0.98*, Namespace

Document, Rev. Marco Polo Edition, August 2010. [Online]. Available: http://xmlns.com/foaf/spec/

[25] S. Dihal, H. Bouwman, M. de Reuver, M. Warnier, and C. Carlsson, "Mobile cloud computing: state of the art and outlook," *info*, vol. 15, no. 1, pp. 4–16, 2013.

[26] J. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Springer Berlin Heidelberg, 2002, vol. 2429, pp. 251–260.

[27] Y. Durmus and E. Onur, "Imitation as the simplest strategy for cooperation," in *2012 IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Sept 2012, pp. 863–869.

[28] (2007, Sept) Playing games with the planet. The Economist. Retrieved at March 2015. [Online]. Available: http://www.economist.com/node/9867020

[29] C. El Kaed, Y. Denneulin, F.-G. Ottogalli, and L. Mora, "Combining ontology alignment with model driven engineering techniques for home devices interoperability," in *Software Technologies for Embedded and Ubiquitous Systems*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, vol. 6399, pp. 71–82.

[30] M. Felegyhazi, J.-P. Hubaux, and L. Buttyan, "Nash equilibria of packet forwarding strategies in wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 5, pp. 463–476, May 2006.

[31] T. Forde and L. Doyle, "Cellular clouds," *Telecommunications Policy*, vol. 37, no. 2–3, pp. 194 – 207, 2013, cognitive Radio Dynamic Spectrum Assignment.

[32] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 3, pp. 15:1–15:37, Jun. 2008.

[33] (2014, August) Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business. Gartner Inc. Retrieved at April 2015. [Online]. Available: http://www.gartner.com/newsroom/id/2819918

[34] M. S. Garver, Z. Williams, and S. A. LeMay, "Measuring the importance of attributes in logistics research," *The International Journal of Logistics Management*, vol. 21, no. 1, pp. 22–44, 2010.

[35] D. Gilbert. (2014, November) Russian website streaming live footage from 584 uk webcams and baby monitors. [Online]. Available: www.ibtimes.co.uk/russian-website-streaming-live-footage-584-uk-webcams-baby-monitors-1475706

[36] J. Granjal, E. Monteiro, and J. Sa Silva, "End-to-end transport-layer security for internet-integrated sensing applications with mutual and delegated ecc public-key authentication," in *IFIP Networking Conference, 2013*, May 2013,

pp. 1–9.

[37] P. E. Green, A. M. Krieger, and Y. Wind, "Thirty years of conjoint analysis: Reflections and prospects," *Interfaces*, vol. 31, no. 3_supplement, pp. S56–S73, 2001.

[38] P. E. Green and V. Srinivasan, "Conjoint analysis in consumer research: issues and outlook," *Journal of Consumer Research*, pp. 103–123, 1978.

[39] B. Greevenbosch, D. He, and R. Sun, *Comparison of different proposals for ACE*, Internet-Draft, ACE Working Group Informational, December 2014. [Online]. Available: https://tools.ietf.org/id/draft-greevenbosch-ace-comparison-01.txt

[40] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable web of things," in *Proc. PERCOM Workshops*, april 2010, pp. 702 –707.

[41] H. Han, Y. Liu, G. Shen, Y. Zhang, and Q. Li, "Dozyap: power-efficient wi-fi tethering," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 421–434.

[42] Q. He, D. Wu, and P. Khosla, "SORI: a secure and objective reputation-based incentive scheme for ad-hoc networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 2, 2004, pp. 825–830.

[43] L.-M. Hofmann, N. Chakraborty, and K. Sycara, "The evolution of cooperation in self-interested agent societies: A critical study," in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, ser. AAMAS '11, Richland, SC, 2011, pp. 685–692.

[44] M. Holland, *Social bonding & nurture kinship: compatibility between cultural and biological approaches*. Createspace Independent Publishing, 2012.

[45] HP. (2014, July) Internet of things research study. [Online]. Available: http://h30499.www3.hp.com/hpeb/attachments/hpeb/application-security-fortify-on-demand/189/1/HP_IoT_Research_Study.pdf

[46] W. Hu, H. Tan, P. Corke, W. C. Shih, and S. Jha, "Toward trusted wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 7, no. 1, pp. 5:1–5:25, Aug. 2010.

[47] R. Hummen, H. Shafagh, S. Raza, T. Voigt, and K. Wehrle, "Delegation-based authentication and authorization for the IP-based internet of things," in *IEEE SECON*, 2014.

[48] (2000) Simple service discovery protocol (ssdp). IETF Draft. [Online]. Available: https://tools.ietf.org/html/draft-cai-ssdp-v1-03

[49] R. Isaac, *The pleasures of probability*. Springer, 1995.

[50] J. J. Jaramillo and R. Srikant, "A game theory based reputation mechanism to incentivize cooperation in wireless ad hoc networks," *Ad Hoc Networks*, vol. 8, no. 4, pp. 416 – 429, 2010.

[51] R. M. Johnson and B. K. Orme, "How many questions should you ask in choice-based conjoint studies," in *ART Forum Proceedings*, 1996.

[52] M. Jones, J. Bradley, and N. Sakimura, *JSON Web Token*, OAuth Working Group Standards Track, Rev. 32, Dec 2014. [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-32

[53] I. Khalil, S. Bagchi, and N. B. Shroff, "Liteworp: Detection and isolation of the wormhole attack in static multihop wireless networks," *Computer networks*, vol. 51, no. 13, pp. 3750–3772, 2007.

[54] F. Kohne, C. Totz, and K. Wehmeyer, "Consumer preferences for location-based service attributes: a conjoint analysis," *International Journal of Management and Decision Making*, vol. 6, no. 1, pp. 16–32, 2005.

[55] M. Kovatsch, "CoAP for the web of things: From tiny resource-constrained devices to the web browser," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, ser. UbiComp '13 Adjunct. New York, NY, USA: ACM, 2013, pp. 1495–1504.

[56] C. Kuo, M. Luk, R. Negi, and A. Perrig, "Message-in-a-bottle: User-friendly and secure key deployment for sensor nodes," in *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, ser. SenSys '07. New York, NY, USA: ACM, 2007, pp. 233–246.

[57] J. Lee, Y. Cho, J.-D. Lee, and C.-Y. Lee, "Forecasting future demand for large-screen television sets using conjoint analysis with diffusion model," *Technological Forecasting and Social Change*, vol. 73, no. 4, pp. 362 – 376, 2006.

[58] F. Lemos, A. Gater, D. Grigori, and M. Bouzeghoub, "A framework for service discovery based on structural similarity and quality satisfaction," in *Web Engineering*, ser. Lecture Notes in Computer Science, M. Brambilla, T. Tokuda, and R. Tolksdorf, Eds. Springer Berlin Heidelberg, 2012, vol. 7387, pp. 481–485.

[59] Y. Liu, K. Li, Y. Jin, Y. Zhang, and W. Qu, "A novel reputation computation model based on subjective logic for mobile ad hoc networks," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 547 – 554, 2011.

[60] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 255–265.

[61] Selfish. Merriam-Webster. Retrieved at April 2015. [Online]. Available: http://www.merriam-webster.com/dictionary/selfish?show=0&t=1408827003

[62] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proceedings of the IFIP TC6/TC11*, Deventer, The Netherlands, 2002, pp. 107–121.

[63] B. Misra. (2014, September) iOS8 MAC randomization – analyzed! Accessed at April 2015. [Online]. Available: http://blog.airtightnetworks.com/ios8-mac-randomgate/

[64] M. Mutka and L. Ni, "Service Discovery in Pervasive Computing Environments," *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 81–90, Oct. 2005.

[65] J. Nash, "Non-cooperative games," *The Annals of Mathematics*, vol. 54, no. 2, pp. pp. 286–295, 1951.

[66] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis & defenses," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, ser. IPSN '04. New York, NY, USA: ACM, 2004, pp. 259–268.

[67] S. Nikou, H. Bouwman, and M. de Reuver, "Mobile converged rich communication services: A conjoint analysis," in *System Science (HICSS), 2012 45th Hawaii International Conference on*, Jan 2012, pp. 1353–1362.

[68] ——, "The potential of converged mobile telecommunication services: a conjoint analysis," *info*, vol. 14, no. 5, pp. 21–35, 2012.

[69] M. Nowak, K. Sigmund *et al.*, "A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game," *Nature*, vol. 364, no. 6432, pp. 56–58, 1993.

[70] M. A. Nowak, "Five rules for the evolution of cooperation," *Science*, vol. 314, no. 5805, pp. 1560–1563, 2006.

[71] (2009, July) Oasis devices profile for web services (dpws) version 1.1. OASIS. [Online]. Available: http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01

[72] (2009, July) Web services dynamic discovery (ws-discovery) version 1.1. OASIS. [Online]. Available: http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html

[73] H. Ohtsuki, C. Hauert, E. Lieberman, and M. a. Nowak, "A simple rule for the evolution of cooperation on graphs and social networks." *Nature*, vol. 441, no. 7092, pp. 502–5, May 2006.

[74] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *IEEE Conference on Local Computer Networks*, Nov 2006, pp. 641–648.

[75] M. Pagani, "Determinants of adoption of third generation mobile multimedia

services," *Journal of interactive marketing*, vol. 18, no. 3, pp. 46–59, 2004.

[76] A. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," *Services Computing, IEEE Transactions on*, vol. 5, no. 2, pp. 260 –275, april-june 2012.

[77] M. P. Pignone, A. T. Brenner, S. Hawley, S. L. Sheridan, C. L. Lewis, D. E. Jonas, and K. Howard, "Conjoint analysis versus rating and ranking for values elicitation and clarification in colorectal cancer screening," *Journal of general internal medicine*, vol. 27, no. 1, pp. 45–50, 2012.

[78] C. Piro, C. Shields, and B. Levine, "Detecting the sybil attack in mobile ad hoc networks," in *Securecomm and Workshops, 2006*, Aug 2006, pp. 1–11.

[79] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf. W3C. [Online]. Available: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

[80] (2013, July) Alljoyn, the internet of everything. Qualcomm. [Online]. Available: https://www.alljoyn.org/about

[81] K. Rasch, F. Li, S. Sehic, R. Ayani, and S. Dustdar, "Context-driven personalized service discovery in pervasive environments," *World Wide Web*, vol. 14, pp. 295–319, 2011.

[82] E. Rescorla and N. Modadugu, *Datagram Transport Layer Security Version 1.2*, Standards Track, IETF Std., Jan 2012. [Online]. Available: https://tools.ietf.org/html/rfc6347

[83] S. Schulz, A.-R. Sadeghi, M. Zhdanova, H. Mustafa, W. Xu, and V. Varadharajan, "Tetherway: a framework for tethering camouflage," in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, ser. WISEC '12. New York, NY, USA: ACM, 2012, pp. 149–160.

[84] H. K. Shin, A. Kim, and C. W. Lee, "Relationship between consumer's preference and service attributes in mobile telecommunication service," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3522 – 3527, 2011.

[85] D. Sorenson and J. Bogue, "A conjoint based approach to concept optimisation probiotic beverages," *British Food Journal*, vol. 107, no. 11, pp. 870–883, 2005. [Online]. Available: http://dx.doi.org/10.1108/00070700510629805

[86] M. Sporny, T. Inkster, H. Story, B. Harbulot, and R. Bachmann-Gmur, *WebID 1.0, Web Identification and Discovery*, W3C Std., Rev. Editor's Draft, Dec 2011. [Online]. Available: http://www.w3.org/2005/Incubator/webid/spec/

[87] V. Srinivasan, "A model and estimation procedure for multi-stage decision processes," Working Paper, Graduate School of Business, Stanford University, Tech. Rep., 1978.

[88] V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao, "Cooperation in

wireless ad hoc networks," in *Proc. of the INFOCOM 2003.*, vol. 2, march-april 2003, pp. 808 – 817.

[89] G. Szabó and G. Fáth, "Evolutionary games on graphs," *Physics Reports*, vol. 446, no. 4–6, pp. 97 – 216, 2007.

[90] O. L. Tim Berners-Lee, James Hendler, "The semantic web," *Scientific American Magazine*, May 2001.

[91] K. Togias, C. Goumopoulos, and A. Kameas, "Ontology-based representation of upnp devices and services for dynamic context-aware ubiquitous computing applications," in *Communication Theory, Reliability, and Quality of Service (CTRQ), 2010 Third International Conference on*, june 2010, pp. 220 –225.

[92] D. Tsarkov and I. Horrocks. (2012) Fact++ description logic reasoner. [Online]. Available: http://code.google.com/p/factplusplus/

[93] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," *CoRR*, vol. abs/1111.4503, 2011.

[94] L. van de Wijngaert and H. Bouwman, "Would you share? predicting the potential use of a new technology," *Telematics and Informatics*, vol. 26, no. 1, pp. 85 – 102, 2009.

[95] J. I. Vazquez and D. L. de Ipiña, "mrdp: An http-based lightweight semantic discovery protocol," *Computer Networks*, vol. 51, no. 16, pp. 4529 – 4542, 2007.

[96] J. Vazquez and D. Lopez-de Ipina, "Social devices: Autonomous artifacts that communicate on the internet," in *Proc. The Internet of Things*, ser. LNCS, C. Floerkemeier, M. Langheinrich, E. Fleisch, F. Mattern, and S. Sarma, Eds., 2008, vol. 4952, pp. 308–324.

[97] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[98] *IEEE Std 802.11*, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Std., 2007.

[99] M. Woehrle, M. Bor, and K. Langendoen, "868 MHz: a noiseless environment, but no free lunch for protocol design," in *9th int. conf. on Networked Sensing Systems*, ser. INSS, jun 2012, pp. 1–8.

[100] S. Yang, Y. Xu, and Q. He, "Ontology Based Service Discovery Method for Internet of Things," *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pp. 43–47, Oct. 2011.

[101] Y. Yu, K. Li, W. Zhou, and P. Li, "Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 867 – 880, 2012.