Hochschule für Technik und Wirtschaft Dresden

Fakultät Informatik/Mathematik

Studiengang Angewandte Informationstechnologien

Forschungsseminar Sensornetze

Abschlussbericht

Autoren: Falco Baumgärtel 34495, Samuel Bohnet 34820

Eingereicht am: 22.02.2014

Betreuer: Prof. Dr.-Ing. Jörg Vogt

Struktur

Der vorliegende Abschlussbericht untergliedert sich in die folgenden drei Schwerpunkte, wobei der erste Teil im Sommersemester 2013 ausgearbeitet wurde und der zweite und dritte Teil Gegenstand des Wintersemesters 2013/14 darstellten.

I. Integration eines CoAP Client für die Kommunikation zwischen Sensorknoten und FHEM

II. Weiterentwicklung des 802.15.4-Modul für FHEM

III. IPv6-Tunneling

Abkürzungsverzeichnis

CoAP	Constrained Application Protocol
FHEM	Freundliche Hausautomation und Energie Messung
URI	Uniform Resource Identifier
WSN	Wireless Sensor Network
WSNPHD	Wireless Sensor Network Physical Device
IPv6	Internet Protocol Version 6

Abbildungsverzeichnis

Tabellen- und Listingverzeichnis

Integration eines CoAP Client für die Kommunikation zwischen Sensorknoten und FHEM

Einleitung

Die grundlegende Fragestellung, welche uns während des Semesters beschäftigt hat, kann wie folgt formuliert werden:

„Welche CoAP-Implementierungen eignen sich für die Integration in die bestehende Systemlandschaft und wie können Sie verwendet werden, so dass Befehle des FHEM-Servers zum Sensorboard weitergeleitet werden sowie eine Sensoransteuerung ohne FHEM (z.B. über die Konsole, andere Endgeräte) durchführbar ist?"

1 Systemanalyse

Während dieses Semesters wurde ein 802.15.4 genanntes FHEM-Modul entwickelt, welches die Sensoransteuerungsbefehle des Nutzers mittels FHEM verarbeitet.
Zur Übermittelung dieser Befehle an die Sensoren wird eine Möglichkeit benötigt, diese in ein konformes Protokoll zu wandeln und an das jeweilige Sensorboard weiterzugeben. Die Systemkomponenten, welche für die drahtlose Kommunikation zwischen Sensor und Nutzer notwendig sind, können der folgenden Abbildung entnommen werden.

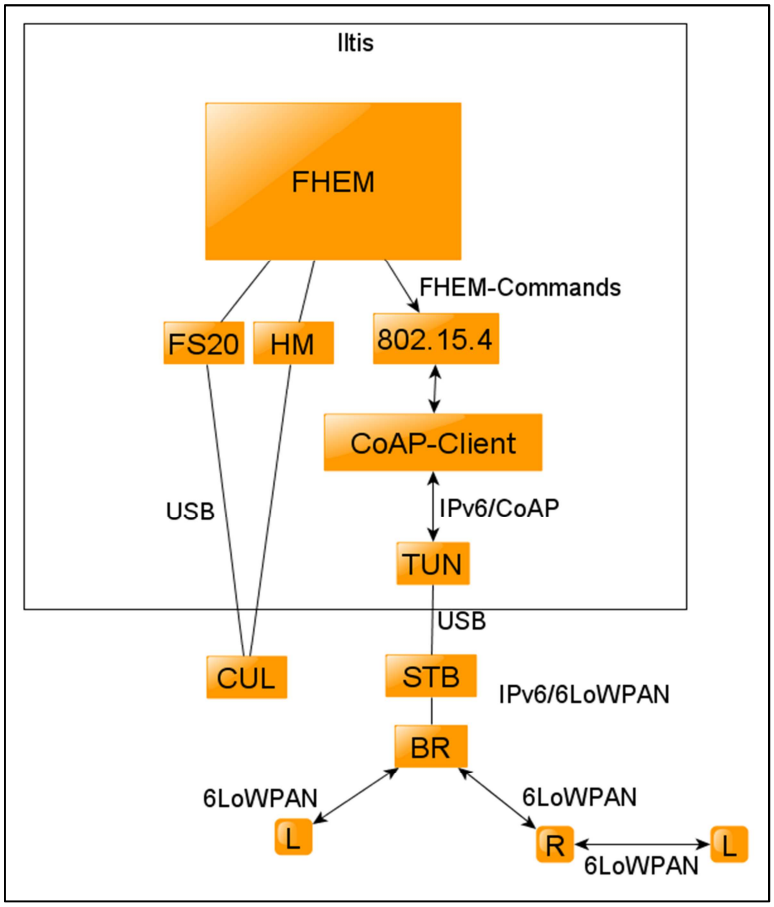

Abbildung 1 Grobentwurf der Systemlandschaft

Folgende Systemkomponenten müssen bei der Erarbeitung des Lösungskonzepts berücksichtigt werden:

Server der HTW-Dresden „Iltis“

- Nur im HTW-Netz erreichbar
- Debian-OS

- Hostet den FHEM-Server

Sensorknoten der HTW-Dresden

- Contiki-OS
- Erbium CoAP Server für die Kommunikation mit Sensoren

2 Ergebnisse aus früheren Semestern

Während des Sommersemesters 2012 und des Wintersemesters 2012/13 wurde von den vorherigen Teilnehmern des Forschungsseminars der Ansatz verfolgt, die Sensoransteuerung über eine selbstentwickelte, zentrale Plattform zu verwirklichen. Hierbei wurde auch eine eigene CoAP-Implementierung in Python entwickelt.

Zum Ende des WS12/13 stellte sich heraus, dass das junge CoAP-Protokoll so schnell wächst, dass eine Entwicklung im Rahmen des Forschungsseminars nicht lohnenswert ist – bei Fertigstellung der prototypischen Implementierung wurden bereits vier neue CoAP-Versionen veröffentlicht.

Im Sommersemester 2013 startet das Forschungsseminar mit einem mit einem neuen Ansatz: Anstelle der Implementierung einer eigenen Plattform zur Sensoransteuerung wird nun die bereits bestehende Implementierung FHEM (Freundliche Hausautomatisierung und Energie-Messung) genutzt und soll um die Möglichkeit, der Sensoransteuerung 6LoWPAN (IPv6 over Low power Wireless Personal Area Network) erweitert werden.

3 Constrained Application Protocol

Das Constrained Application Protocol ist ein Protokoll zur Nachrichten-/Datenübertragung zwischen Applikationen, deren Funktionalitäten eingeschränkt sind. Besonders hervorzuheben sind hierbei Applikationen von Geräten, die wenig Speicher (kleiner ROM sowie RAM) und sehr wenig Strom (Batterie) zur Verfügung haben. Die daraus resultierende Nutzung von Netzwerken, die nicht permanent verfügbar sind - bspw. durch die Verwendung von 6LoWPAN - wird mit CoAP stark vereinfacht.

Das Protokoll wurde für die Machine-to-Machine (M2M) Kommunikation entworfen, wie es z.B. in der Hausautomation oder bei Intelligenter Energie (smart energy) der Fall ist. Es wird seit 2010 als IETF-Standard von Sensinode, SkyFoundry und Pacific Gas & Electric (ab Version 0.4 auch von der Uni Bremen) entwickelt.

CoAP stellt ein Request/Response Interaktionsmodel zwischen zwei Applikationen bereit, unterstützt mittels eines eingebautem Discovery-Modus die selbstständige Auffindung von Services und Ressourcen und beinhaltet wichtige Grundkonzepte des Internets wie URIs und Internet Media Types (MIME-Types).

Dadurch wird eine einfache Kombinierung mit http gewähreistet, um Übertragungen über das World Wide Web zu ermöglichen. CoAP unterstützt Mehrpunktverbindungen (multicast) und zeichnet sich, besonders durch den geringen Overhead und seine Einfachheit, zur Nutzung in eingeschränkten Umgebungen aus. [1]

4 Anforderungsanalyse CoAP Client

Die Hauptaufgabe des CoAP-Clients besteht aus der Bereitstellung einer Schnittstelle zum Sensorknoten über den dort bereitgestellten Erbium-CoAP-Server. Diese soll einerseits eine permanente Verbindung mit dem FHEM-Server ermöglichen, um dessen Sensoranfragen rund um die Uhr bedienen zu können, als auch einfache Abfragen über die Konsole bzw. andere Programme ermöglichen.

Die Sensoren sollen mittels folgender Methoden ansprechbar sein:

- DISCOVER
  Auslesen aller mit dem Sensorboard verbundenen Geräte und deren Funktionalitäten
- GET
  Zur Abfrage von Werten der Sensoren (bspw. Luftfeuchtigkeit)
- SET
  Zum Setzen eines Wertes (bspw. Licht anschalten)
- OBSERVE
  Zyklische Abfrage von Sensorwerten (bspw. Überwachung der Raumtemperatur)

Zur Kommunikation mit dem FHEM Modul wurde gemeinsam mit dessen Entwicklern ein Datenaustauschformat entwickelt, dass unterstützt werden muss:

```
<method>|<uri>(|<payload>)
```

„Method“ steht hierbei für die jeweilige Methode (siehe oben), „uri“ für die Adresse des jeweiligen Sensors und der optionale „payload“ für die Daten, die an den Sensor übergeben werden können.

5 Analyse vorhandener CoAP Implementierungen

Aufgrund des gescheiterten Versuches, in den vorigen Semestern eine eigene CoAP-Implementierung zu entwickeln, haben wir uns in diesem Semester dazu entschlossen, eine bereits vorhandene Implementierung zu nutzen.

Dies soll den nachfolgenden Studenten des Forschungsseminars ermöglichen, sich schnell einarbeiten zu können und Aktualisierungen der Software mit einfachen Mitteln durchführen zu können.

Bei der Suche nach vorhandenen Implementierungen konnten wir einen guten Überblick durch eine häufig aktualisierte Übersicht im CoAP-Artikel der englisch-sprachigen Wikipedia gewinnen (Siehe Abbildung 2 CoAP-Implementierungen Stand 19.07.2013).

Implementations [edit]

Name	Programming Language	Implemented CoAP version	Client/Server	Implemented CoAP features	License	Link
libcoap	C	coap-13	Client + Server	Observe, Blockwise Transfers	BSD/GPL	http://sourceforge.net/projects/libcoap/develop
nCoap	Java	coap-08	Client + Server	Observe (draft 06), Blockwise Transfers (draft 04, partially)	BSD	https://github.com/okleine/nCoAP
jcoap	Java	coap-08?	Client + Server		Apache	http://code.google.com/p/jcoap/
coapy	Python	coap-03	Client + Server		BSD	http://sourceforge.net/projects/coapy/
TinyOS CoapBlip	nesC/C	coap-13	Client + Server	Observe, Blockwise Transfers	BSD	http://docs.tinyos.net/tinywiki/index.php/CoAP
RESTful Contiki	C	coap-03	Server		3-clause BSD	http://www.contiki-os.org/ (rest-example)
Erbium for Contiki	C	coap-13	Client + Server	Observe, Blockwise Transfers	3-clause BSD	http://www.contiki-os.org/ (er-rest-example)
Californium	Java	coap-13	Client + Server	Observe, Blockwise Transfers, DTLS	3-clause BSD	https://github.com/mkovatsc/Californium
Copper	JavaScript (Browser Plugin)	coap-13	Client	Observe, Blockwise Transfers	3-clause BSD	https://github.com/mkovatsc/Copper https://addons.mozilla.org/de/firefox/addon/copper-270430/
CoAP implementation for TinyOS	nesC/C	coap-08	??		??	http://telecom.dei.unipd.it/pages/read/90/
CoAP implementation for Go	Go	coap-10	Client + Server	Core + Draft Subscribe	MIT	https://github.com/dustin/go-coap
Sensinode C Device Library	C	coap-12	Client + Server	Core, Observe, Block, RD	Commercial	http://www.sensinode.com/EN/products/trial-download.html
Sensinode Java Device Library	Java SE	coap-12	Client + Server	Core, Observe, Block, RD	Commercial	http://www.sensinode.com/EN/products/trial-download.html
Sensinode NanoService Platform	Java SE	coap-12	Cloud Server	Core, Observe, Block, RD	Commercial	http://www.sensinode.com/EN/products/trial-download.html

Abbildung 2 CoAP-Implementierungen Stand 19.07.2013

Neben den kommerziellen Lösungen (die auf Grund des open-source-Anspruchs des Forschungsprojektes direkt herausfallen) in C und Java der Firma Sensinode, die maßgeblich an der Entwicklung von CoAP beteiligt ist, fallen vor allem weitere C und Java-Entwicklungen auf, aber auch die Unterstützung der Betriebssysteme TinyOS und Contiki.

Da in unserem Forschungsprojekt die Sensorboards unter dem Betriebssystem Contiki betrieben werden, fiel unser Augenmerk recht schnell auf die Implementierungen „RESTful Contiki“, und „Erbium for Contiki“.

Die Nutzung einer weiterentwickelten CoAP-Version ließ uns dann erste Tests mit Erbium durchführen (siehe Testumgebung). Die funktionale Abhängigkeit der Erbium CoAP Engine von dem Betriebssystem Contiki ließ uns dazu veranlassen, andere CoAP Implementierungen in Betracht zu ziehen.

Aufgrund der Aktualität (derzeit version 0.13) und der Programmiersprache Java haben wir uns für die Nutzung des Californium Frameworks (Cf-Framework) entschieden. Die Entwicklung der Implementierung wird von der ETH Zürich vorangetrieben, die ebenfalls für die Entwicklung der Erbium CoAP-Implementierung in Contiki verantwortlich ist.

6 Entwurf

Das WSN-Modul des FHEM Servers kommuniziert mit einem festgelegten Datenaustauschformat über eine Socket-Verbindung mit dem CoAP-Client. So sind CoAP-Client und WSN-Modul unabhängig voneinander. Der CoAP-Client wurde um die Implementierung der Schnittstelle (Format: „<method>|<uri>(|<payload>)“) erweitert. Abbildung 3 veranschaulicht die wesentlichen Softwarekomponenten, welche Bestandteil der Kommunikation WSN-Modul - CoAP-Client - CoAP-Server sind.

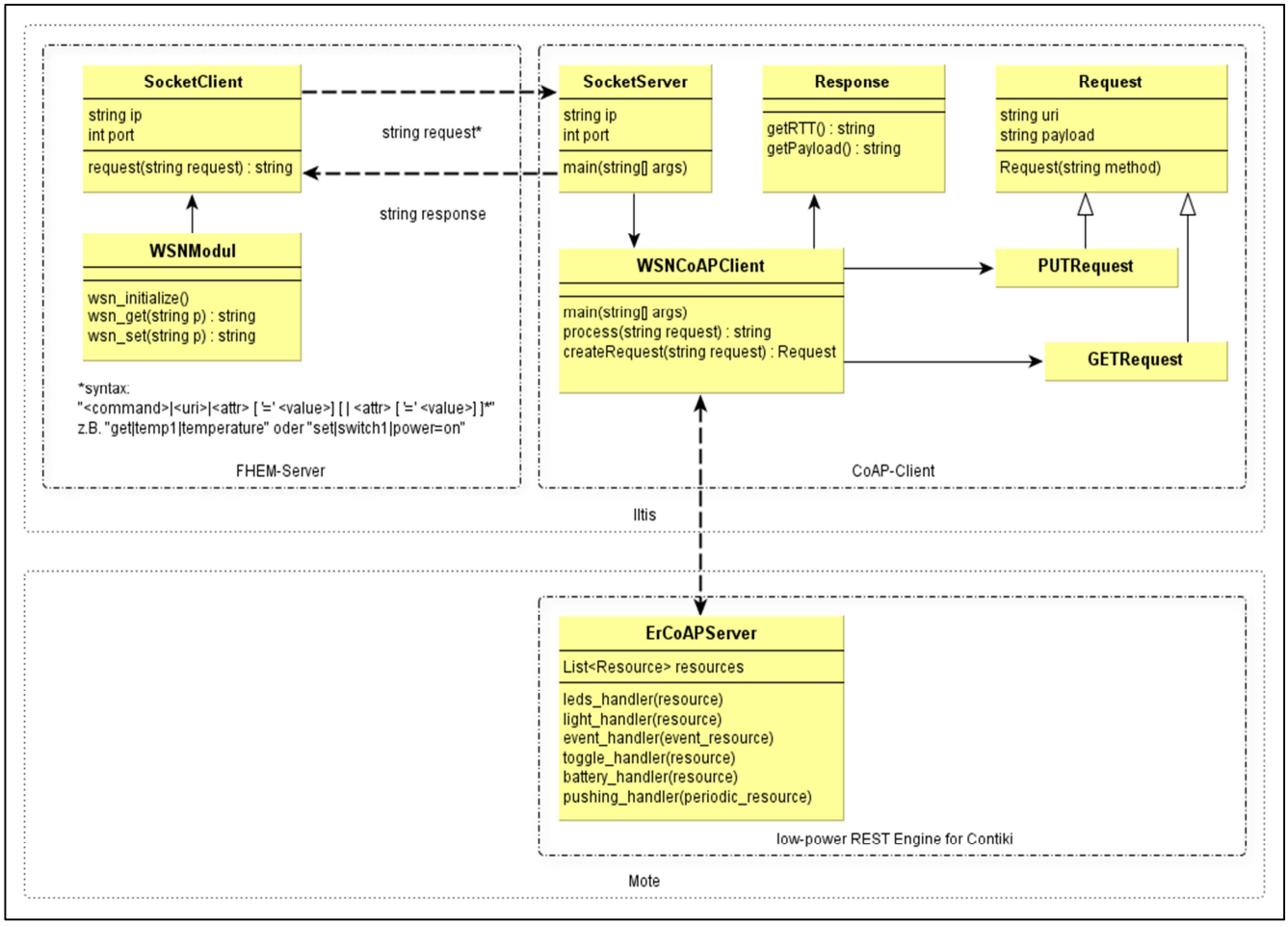

Abbildung 3 Entwurf der Softwarekomponenten des FHEM-CoAP-Server

Die Softwarekomponenten *Response*, *PUTRequest* und *GETRequest* stellen Klassen des Californium-Frameworks dar. Des Weiteren stellen die Klassen *SocketServer* und *WSNCoAPClient* jeweils einen Programmeinstiegspunkt bereit, so dass eine direkte Abfrage des Californium CoAP-Client ermöglicht werden kann. Der CoAP-Client sendet und empfängt CoAP-Pakete vom Sensorknoten, welcher mit den entsprechenden Akteuren kommuniziert.

7 Implementierung

Der Aufbau des Californium Framework ist sehr modular und erweiterbar. Außerdem stellt es eine Implementierung des *CoapClient* bereit, welcher den Anforderungen entsprechend angepasst wurde. Wenn der Client eine neue Nachricht erhält, wird ein *CoapResponseEvent* Ereignis ausgelöst, welches die Übermittlung der CoAP-Nachricht an den Absender des Requests nach sich zieht. Für die Zuordnung von Anfrage und Antwort kann die Methode *getRTT()* der *Response*-Klasse verwendet werden, welches das entsprechende *Request*-Objekt zurückgibt. Der Aufbau des modifizierten *CoapClient*, sowie der Schnittstellen für die ereignisgesteuerte Prozessierung können dem Klassendiagramm in Abbildung 4 entnommen werden. Das Klassendiagramm stellt den derzeitigen Stand der Implementierung dar.

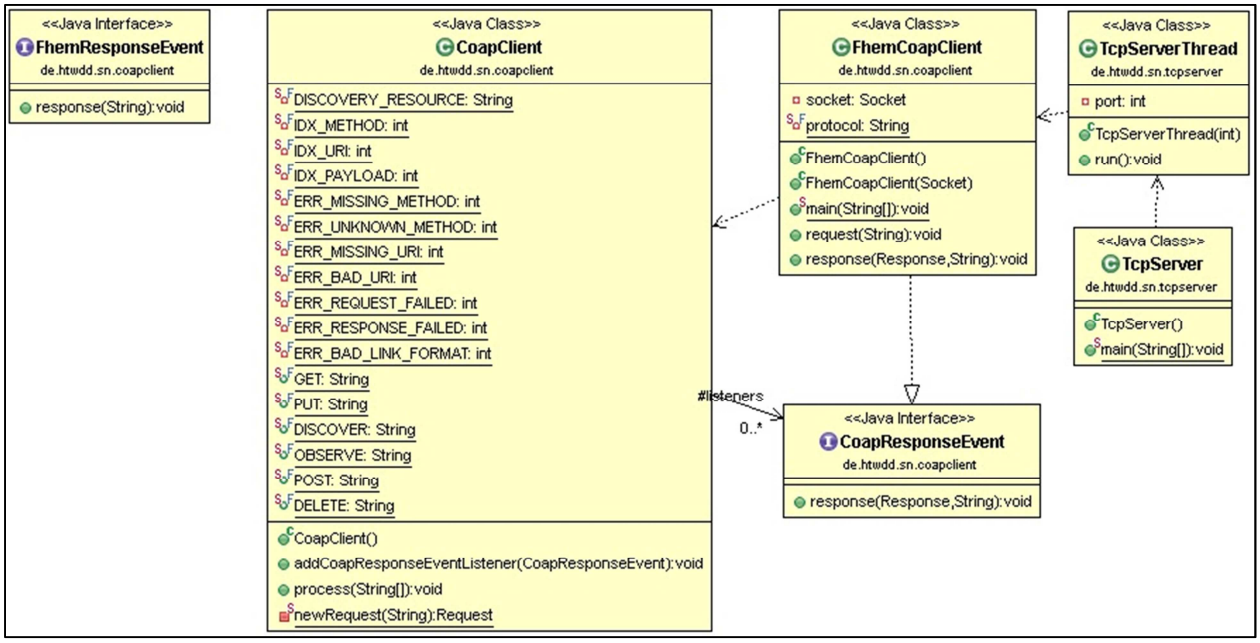

Abbildung 4 Klassendiagramm des WSN CoAP Client

8 Testumgebung

Das in Contiki-OS integrierte "Cooja" simuliert ein drahtloses Netzwerk von Sensorknoten (motes). So konnte während der Entwicklung mit einem simulierten Erbium Server und aktueller CoAP Implementierung getestet werden.

9 Zusammenfassung und Ausblick

Es wurde eine geeignete CoAP-Implementierung verwendet und um Funktionalitäten erweitert, sodass eine dauerhafte Verbindung mit dem FHEM-Server besteht und die entsprechenden CoAP-Methoden für FHEM und andere Endgeräte der Heimautomatisierung nutzbar sind.

Da die Entwickler des 802.15.4-Moduls das Forschungsseminar zum Wintersemester 2013/14 verlassen, wird die Zusammenführung der Entwicklung des FHEM-Moduls und des CoAP-Clients mit einer bisher noch nicht voll umfassende Fehlerbehandlung inklusive sämtlicher Fehlernachrichten für das FHEM-Modul beginnen.

Mit weiteren Tests und Implementierungen wird die Funktionalität der Discover, Get, Set und Observe-Befehle verifiziert und umgesetzt.

Die Ergebnisse in diesem Semester konzentrierten sich vor allem darauf, Anfragen vom FHEM-Modul zu akzeptieren und weiterzureichen. Zukünftig soll auch außerhalb der Iltis-Umgebung über den CoAP-Client auf die Sensoren zugegriffen werden können, beispielsweise mit dem CoAP-Plugin „Copper“ für Firefox, das von der ETH Zürich entwickelt wird.

Weiterentwicklung des 802.15.4-Modul für FHEM

Problematik

Damit ein Nutzer der freien Hausautomatisierungssoftware (FHEM) 802.15.4-Sensornetze integrieren kann, ist der entwickelte Prototyp auf fehlende Funktionalitäten zu überprüfen. Die bereits bestehende Funktionalität gilt es zu testen sowie weiter zu entwickeln. Im Hinblick auf eine Veröffentlichung des 802.15.4 Modul für FHEM stellt die Ausarbeitung einer stabilen Lösung einen weiteren Gegenstand der Problematik dar.

1 Analyse des 802.15.4-Modul für FHEM

Damit der Prototyp um Funktionalitäten erweitert werden kann, ist zunächst dessen Funktionsweise und Architektur zu erfassen.
Das 802.15.4-Modul ist ein Programmmodul, welches beim Start durch die FHEM Hauptschleife initialisiert wird. Es dient dem Anlegen, Abfragen und der Manipulation von Sensoren bzw. Aktoren eines 802.15.4-Sensornetzes. Zur Kommunikation mit den Sensorknoten wird auf Anwendungsschicht CoAP verwendet, dessen Implementierung nach umfangreicher Evaluierung in eine externe Anwendung ausgelagert wurde. Aus diesem Grund wurde das 802.15.4-Modul für die folgenden zwei Aufgaben untergliedert:

- Die Kommunikation mit einem TCP-Server wird von dem Modul WSNPHD (WSN Physical Device) übernommen.
- Die logische Verwaltung von Resourcen eines Sensorknoten wird von dem Modul WSN (Wireless Sensor Network) realisiert.

Zunächst muss eine Instanz des WSNPHD-Modul mithilfe der FHEM Kommandozeile definiert werden, damit eine Kommunikation über das spezifizierte Datenaustauschformat mit einem externem Programm (CoAP Client) erfolgen kann.

```
define <name> WSNPHD <ip>:<port>
```

Nachdem das neu definierte Gerät initialisiert wurde und eine Verbindung zum TCP-Server erfolgreich hergestellt werden konnte, können die eigentlichen Resourcen definiert, abgefragt und manipuliert werden.

```
define <sensor_name> WSN <uri>
```

Bei der Abfrage eines spezifischen Sensors wird der Befehl vom WSN-Modul in das entsprechende Datenaustauschformat umgewandelt und an das zugeordnete WSNPHD-Modul übermittelt. Eingehende Nachrichten werden wiederum vom WSNPHD-Modul empfangen und vom WSN-Modul interpretiert. Die Trennung in zwei separate Module ermöglicht die

bidirektionale Kommunikation von FHEM und CoAP Client, welche in Abbildung 1 anhand einer Temperaturabfrage schematisch dargestellt ist.

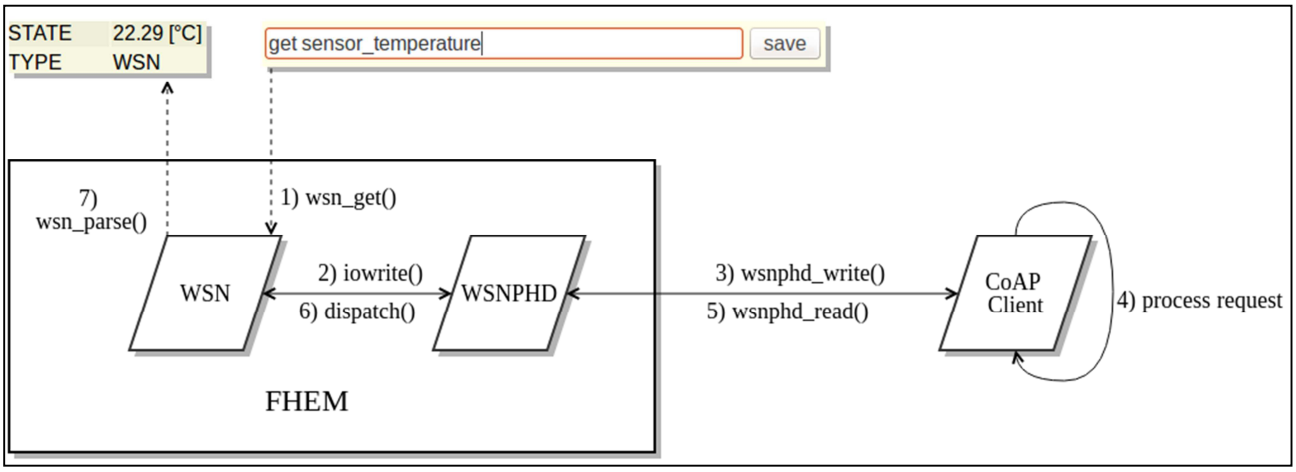

Abbildung 5 Kommunikation zwischen FHEM und CoAP Client

Des Weiteren kann man Abbildung 1 entnehmen, in welcher Reihenfolge die Prozeduren der 802.15.4 Module bei einer Sensorabfrage aufgerufen werden. Da die Prozeduren wsn_get und wsn_parse die Verarbeitungslogik der zu verschickenden bzw. eingehenden Nachrichten implementieren, sind sie für die Weiterentwicklung von besonderer Bedeutung.

2 Stand der Funktionalität

Neben den für die Interaktion mit FHEM notwendigen Funktionalitäten sind im Hinblick auf die Realisierung von CoAP folgende Methoden vorbereitet worden:

- GET
- DISCOVER
- PUT

Das Abfragen von Sensoren mittels GET ist funktionsfähig implementiert. Der DISCOVER Vorgang kann zunächst für eine Sitzung durchgeführt werden. Es wird dabei für jede entdeckte Resource ein eindeutiger Name generiert und als WSN-Gerät definiert. Dabei werden die Geräte aber nicht in der FHEM Konfiguration gespeichert, weshalb der DISCOVER Vorgang bei einem Neustart von FHEM wiederholt ausgeführt werden muss. Das Setzen von Aktoren mittels PUT ist programmtechnisch vorbereitet, ist jedoch nicht funktionsfähig umgesetzt. In diesem Sinne wurden bisher keine ausreichenden Maßnahmen zur Fehlerbehandlung sowie sinnvolle Fehlerbeschreibungen für den Nutzer realisiert. Für nähere Informationen wurde die o.g. Funktionalität in der Projektdokumentation „Integration des 802.15.4-Sensornetzes in FHEM“ ausführlich beschrieben.

3 Weiterentwicklungen

Wie man den in 2. beschriebenen Stand der Funktionalität entnehmen kann, ist es notwendig die grundlegenden Operationen für die Verwendung von CoAP (Entdecken, Schreiben und Abonnieren von Ressourcen) zu implementieren bzw. auszubauen. Für den sinnvollen Einsatz außerhalb des seminarinternen 802.15.4-Sensornetzes stellt die Behandlung von Ausnahmen einen weiteren Schwerpunkt der Weiterentwicklungen dar.

3.1 Entdecken von Ressourcen

Damit unbekannte Ressourcen eines Sensorknoten automatisch von FHEM erkannt werden, muss ein DISCOVER Vorgang gestartet werden.

```
get <wsnphd_device_name> discover <uri>
```

Da der Befehl keinen Bezug zu einer spezifischen WSN Ressource hat und direkt an den CoAP Client geschickt wird, kann er nur von einem zuvor definierten WSNPHD-Modul verarbeitet werden. Zusätzlich zur Uri ist es sinnvoll dem FHEM Nutzer Metadaten der neu entdeckten Ressource zur Verfügung zu stellen. In Listing 1 ist eine CoAP DISCOVER-Antwort in Textform dargestellt.

```
</.well-known/core>,
</hello>;title="Hello world: ?len=0..";rt="Text";ct="text/plain",
</actuators/leds>;title="LEDs: ?color=r|g|b, POST/PUT mode=on|off",
</actuators/toggle>;title="Red LED",
</sensors/sht21_temperature>;title="Temperature in [°C]";rt="Temperature-
C";obs;ct="text/plain",
</sensors/sht21_humidity>;title="Relative Humidity in [%RH]";rt="Humidity-
rel";obs;ct="text/plain"
```

Listing 1 Syntax einer CoAP DISCOVER Antwort

Das Link-Format der entdeckten Ressourcen wurde von der IETF unter dem RFC 6690 („Simple Semantics for Machines“) standardisiert und definiert u.a. folgende Semantik:

- resource type (rt=)
  Bezeichnung oder Angaben über Modell, Typ der Messwerte etc.
- content type (ct=)
  Format des Payload der Ressource
- interface description (if=)
  Definiert die Möglichkeiten des Zugriffes auf die Ressource (z.B. GET)
- observe (obs)
  Gibt an, ob die Ressource periodisches Abfragen unterstützt
- ...

Die o.g. Attribute müssen in FHEM konforme Attribute überführt werden. Dabei muss beachtet werden, sofern es sinnvoll ist, dass die bereits in FHEM standardisierten Attribute wieder verwendet werden, um Redundanzen zu vermeiden. Daher wurde ein Mapping der Attribute vorgenommen:

- Das Attribut „rt“ dient der Bestimmung des Typs der Messwerte eines Sensors und ist in Tabelle 1 beispielhaft erläutert.

Tabelle 1 Vereinbarungen für die Verwendung des Resource Type Attribut

rt=	**Bedeutung**
Temperature-C	Temperatur in °C
Pressure-hPa	Luftdruck in hPa
Humidity-rel	relative Luftfeuchtigkeit in %

- Eine Beschreibung der Ressource liefert das Attribut „title“
- Das Attribut „obs“ kennzeichnet, dass die Ressource für periodische Benachrichtigungen abonnierbar ist

Das Mapping in FHEM konforme Attribute wird in Listing 2 dargestellt. Im Hinblick auf die Definition von weiteren Attributen ist lediglich die Prozedur DefineAttribute mit der entsprechenden Konvertierung aufzurufen.

```
# resource type as unit (e.g. Temperature-C)
DefineAttribute($1, $devname, "rt", "unit");
# title as comment
DefineAttribute($1, $devname, "title", "comment");
```

Listing 2 Aufruf der DefineAttribute Prozedur des WSN Modul

Nach der Definition der entdeckten Ressourcen und ihren Attributen werden die Änderungen in der FHEM Konfigurationsdatei gespeichert.

3.2 Setzen von Aktoren

Das Setzen von Aktoren bzw. Schaltern wurde bereits dahingehend vorbereitet, dass eine WSN Ressource ein „set“ Befehl nach folgender Syntax verarbeiten kann.

```
set <wsn_name> <payload>
```

Im Hinblick auf den CoAP Standard wurde die Methode PUT verwendet, welche für das Überschreiben von Ressourcen geeignet ist. Da das Protokoll im Sinne des REST Paradigma konzipiert wurde, sind bei der Definition des Befehl zusätzlich Query Parameter (?key=value) zu berücksichtigen. Diese können optional beim Setzen von Aktoren angegeben werden.

```
set <wsn_name> [query_parameter] <payload>
set 0:0:0:221_actuators_leds color=r mode=on
```

An dieser Stelle ist jedoch zu erwähnen, dass der Lösungsansatz, Query Parameter als zusätzlichen Parameter einzuführen, nicht optimal ist, da dies der allgemeinen FHEM Notation widerspricht. Sinnvoller wäre eine Bereitstellung der Ressource mit eindeutiger URI, sodass nach o.g. Beispiel für die verschiedenen Farben in FHEM jeweils eine WSN Ressource verwaltet wird.

3.3 Beobachten von Ressourcen

Unter dem OBSERVE Modus versteht man die Registrierung des Client beim Server für periodische Benachrichtigungen. Die Registrierung wird durch ein CoAP GET Paket realisiert, welchem das Flag „obs“ als Option angefügt wird. Da mehrere Sensoren parallel beobachtetet werden können, müssen die empfangenen Antworten dem ursprünglichen Absender zugeordnet werden. Hierfür wird von der Californium Bibliothek die Schnittstelle *ResponseHandler* bereit gestellt.

Die Registrierung und Abmeldung einer Ressource für OBSERVE zieht in FHEM Anpassungen des GET Kommandos nach sich, welches wie folgt definiert ist.

```
get <wsn_name> observe=1
get 0:0:0:221_sensors_sht21_temperature observe=1
```

Mit der Angabe des Flag observe=0 kann eine Ressource entsprechend de-registiert werden. Ein Ansatz für Weiterentwicklungen wäre wie die Verarbeitung von query parametern, mit welchen der OBSERVE Modus genauer spezifiziert werden kann. Im Kontext der IETF spricht man von einem „Conditional Observe“, welcher durch z.B. die Angabe von Extremwerten oder einem Zeitintervall die periodischen Benachrichtigungen des CoAP Server steuert.

3.4 Behandlung von Ausnahmen

Im Hinblick auf die Veröffentlichung der Module für die Verwaltung von 802.15.4 Sensornetzen in FHEM stellt die Behandlung von Ausnahmen einen weiteren Schwerpunkt der Weiterentwicklungen dar. Dabei wurden hauptsächlich die Status Codes der CoAP

Antworten überprüft und FHEM zur Benachrichtigung des Nutzer bereitgestellt. Die folgende Auflistung beschreibt die Behandlung der wichtigsten Ausnahmen, die bei der Kommunikation mit den Ressourcen auftreten können:

- falsches Format der angegebenen URI
- keine Antwort vom CoAP Server (Timeout)
- angeforderte Operation (GET, PUT, OBSERVE) wird von Ressource nicht unterstützt
- Sensor oder Aktor ist nicht erreichbar
- Abbruch der TCP-Verbindung zum CoAP Client
- Falscheingaben der FHEM Kommandozeile

Wenn eine Anfrage des CoAP Client in ein Timeout läuft, stellt die Californium Bibliothek Einstellungen bereit, mit denen u.a. die maximale Anzahl der erneuten Verbindungsversuche konfiguriert werden kann. Unerwartete Ausnahmen werden vom CoAP Client und FHEM im entsprechenden Verzeichnis protokolliert.

4 Hinweise zur Installation

FHEM:

1. WSN- und WSNPHD Modul in Anwendungsordner von FHEM verschieben.
2. WSNPHD Gerät anlegen (z.B.: define tcpclient WSNPHD 127.0.0.1:50009)

CoAP Client

1. Java installieren (mindestens Version 7)
2. CoAPClientTcpServer in screen session starten (java -jar CoAPClientTcpServer.jar 50009)
3. Unter FHEM Status des WSNPHD überprüfen (Status: Opened)
4. CoAP Client ist bereit für die Verarbeitung der FHEM Befehle

Wahlweise kann auch die Anwendung FHEMCoAPClient mit dem entsprechenden Datenaustauschformat für FHEM gestartet werden. (java –jar FHEMCoapClient.jar “get|[aaaa::121:2eff:ff00:1966]:5683/sensors/sht21_temperature”)

5 Ausblick

Abschließend ist zu sagen, dass der im Vorsemester entwickelte Prototyp zu einer stabilen Lösung weiterentwickelt wurde und die wichtigsten Funktionalitäten für das Entdecken, Lesen, Setzen und Abonnieren von Ressourcen ausgebaut bzw. realisiert wurden. Die in Kapitel 3.2 und 3.3 genannten Ansatzpunkte für Weiterentwicklungen bzw. Anpassungen können als Ausgangspunkt für weiterführende Arbeiten verwendet werden. Zusätzlich ist zu beachten, dass die Fragestellung der Sicherheit des Systems bisher außen vor gelassen wurde. Da sich das CoAP Protokoll offiziell noch im Entwicklungsstadium befindet, bleibt es weiterhin erforderlich, Anpassungen zu analysieren und entsprechend Client und Server zu aktualisieren.

IPv6-Tunneling

Einleitung

Die grundlegende Fragestellung, welche mich während des Semesters beschäftigt hat, kann wie folgt formuliert werden:

„*Wie kann ein Sensor mittels IPv6 über das World Wide Web abgefragt werden, wenn dieser in das lokale IPv6 Netzwerk eines Servers integriert ist, der nur über das IPv4 Protokoll mit dem internen Netzwerk der HTW verbunden ist?*”

1 Wiederholung IPv6-Adressaufbau

Da im Folgenden stark auf den IPv6-Adressen eingegangen wird, soll an dieser Stelle eine kurze Erläuterung zum Aufbau dieser Adressen stattfinden.

Eine IPv6 Adresse besteht aus 32 Hexadezimalstellen und ist damit 128 Bit lang.
Die ersten 64 Bit beinhalten die Netzwerkadresse, über die das Netzwerk einzelner Hosts identifiziert wird (siehe Abbildung 6). Die letzten 64 Bit beinhalten die Adresse des einzelnen, sich im Netzwerk befindlichen Gerätes.

Die Netzwerkadresse kann wiederum nach Adresstyp, Netz und Subnetz unterschieden werden. Der Adress-Typ kennzeichnet die Art der IPv6-Adresse, beispielsweise ob es sich um eine lokale Adresse handelt oder um eine global gültige. Das Netz kennzeichnet das übergeordnete Netzwerk, beispielsweise die Adresse des genutzten Internet-Service-Providers. Zusätzlich kann ein Subnetz aufgespannt werden, so dass sich verschiedene Netzwerke an einem Internetanschluss einrichten lassen können.

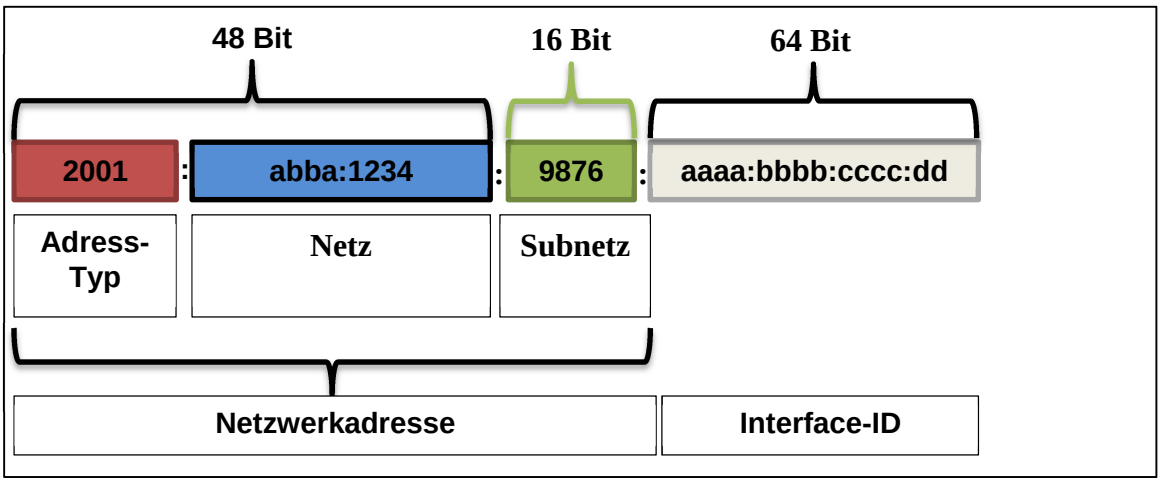

Abbildung 6 IPv6 Adressaufbau

2 Systemanalyse

Wie in der einleitenden Frage schon angesprochen, sind die Sensoren dem lokalen IPv6-Netzwerk des forschungsseminareigenen Servers Iltis zugeordnet. Innerhalb dieses Netzwerkes wird das Präfix *aaaa::* genutzt, das bedeutet alle Sensoren sind vom Iltis aus mit einer IPv6 Adresse erreichbar, die das Format *aaaa::1111:2222:3333:4444* besitzt. Die Sensoren sind ausschließlich über den Server Iltis zu erreichen (siehe Abbildung 7).

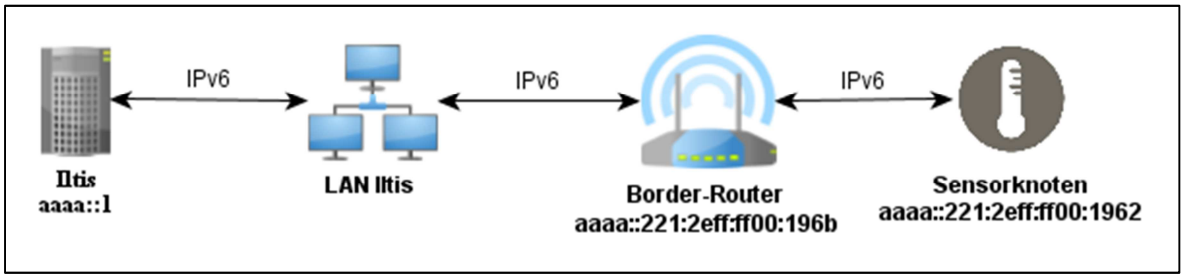

Abbildung 7 IPv6-Verbindung Iltis zu Sensor

Der Iltis Server verfügt über einen Anschluss an das IPv4-Netzwerk der HTW Dresden. Er kann nur von anderen Netzwerkteilnehmern aus erreicht werden, die ebenfalls im mit dem HTW Netzwerk verbunden sind. Ein direkter Zugriff aus dem World Wide Web ist also derzeit nur möglich, wenn eine VPN-Verbindung zur HTW besteht. Diese wiederum ist nur mittels IPv4 zugänglich (siehe Abbildung 8).

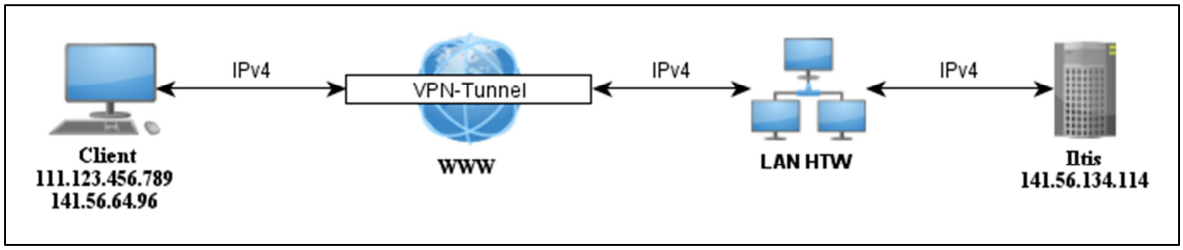

Abbildung 8 VPN-Verbindung zum Iltis

Zusätzlich existiert ein zweiter Server der Fakultät Informatik/Mathematik, der Ilux150 genannt wird. Dieser hat, da er sich im HTW Netz befindet, einen direkten Zugang zum Iltis. Des Weiteren ist Ilux150 auch mit dem World Wide Web verbunden, also von jedem Internetanschluss aus erreichbar. Beide Möglichkeiten werden in Abbildung 9 dargestellt.

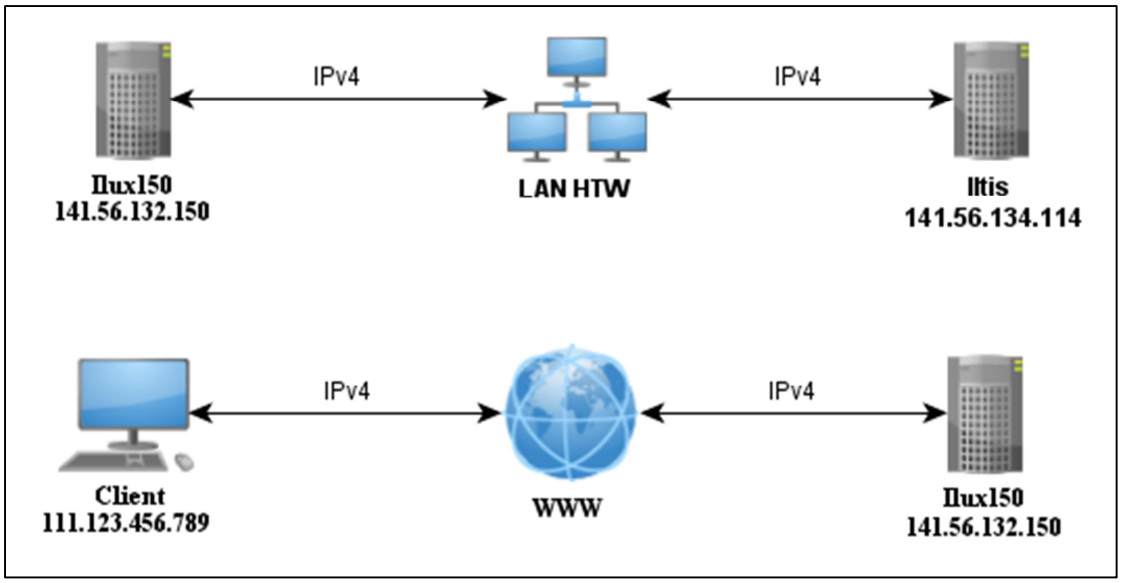

Abbildung 9 Verbindungsmöglichkeiten des Ilux150

Eine Verbindung zu einem IPv6 Netzwerk besteht vom Ilux150 aus gesehen nicht. Er kann also ebenfalls nicht auf die Sensoren zugreifen. Auch vom World Wide Web aus ist er nur mittels IPv4 erreichbar, nicht aber mittels IPv6.

Praktisch gesehen kann eine Verbindung von außerhalb des HTW-Netzes zum Iltis hergestellt werden, indem ein SSH-Tunnel genutzt wird (siehe Abbildung 10). In diesem Fall muss der Client den Tunnel konfigurieren – wir möchten allerdings eine Möglichkeit finden, Sensoren auch bspw. von einem Internetcafé heraus abzufragen.

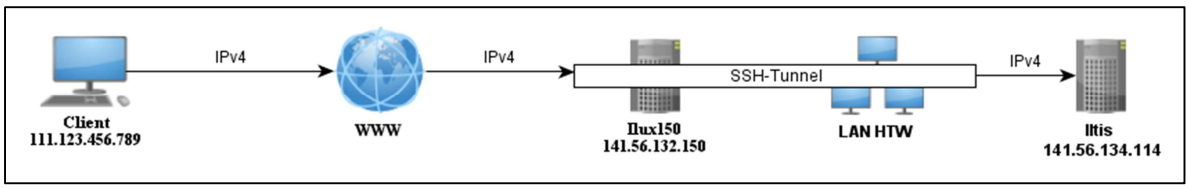

Abbildung 10 SSH Tunnel zum Iltis

Beide Server nutzen ein Linux-Betriebssystem, auf dem ilux150 ist openSuse 12.1 installiert, auf dem Iltis läuft Debian 6.0.8. Da unklar ist, ob wir für unsere Zwecke Netzwerkeinstellungen am Ilux150 vornehmen können (dürfen), konzentriert sich dieser Forschungsbericht auf das Betriebssystem Debian.

3 IPv6-Übergangsmechanismen

Bereits bei der Entwicklung von IPv6 wurde daran gedacht, dass es lange dauern würde, einen Umstieg von IPv4 zu IPv6 zu realisieren.

Aus diesem Grund wurde beschlossen, Mechanismen zu entwickeln, die eine Einführung von IPv6-Netzwerken ermöglichen, ohne den Zugang zum den bereits vorhandenen IPv4-Netzwerken zu verlieren. Diese Mechanismen sind als IPv6-Übergangsmechanismen bekannt.

Es gibt eine Reihe verschiedener Lösungsansätze, ein Blick auf Wikipedia (siehe Abbildung 11) zeigt in einer Liste 12 verschiedene Mechanismen an.

Ziel dieses Forschungsseminars ist es, herauszufinden, ob - und wenn ja, welche - Tunnelmechanismen wir verwenden können, um einen Zugriff auf die Sensoren von außerhalb zu gewährleisten.

Nachfolgend werden einige dieser Übergangsmechanismen vorgestellt.

IPv6-Übergangsmechanismen	
4in6	Tunneling von IPv4 in IPv6
6in4	Tunneling von IPv6 in IPv4
6over4	Transport von IPv6-Datenpaketen zwischen Dual-Stack Knoten über ein IPv4-Netzwerk
6to4	Transport von IPv6-Datenpaketen über ein IPv4-Netzwerk
AYIYA	Anything In Anything
Dual-Stack	Netzknoten mit IPv4 und IPv6 im Parallelbetrieb
Dual-Stack Lite	Wie Dual-Stack, jedoch mit globaler IPv6 und Carrier-NAT IPv4
6rd	IPv6 rapid deployment
ISATAP	Intra-Site Automatic Tunnel Addressing Protocol
NAT64	Übersetzung von IPv4-Adressen in IPv6-Adressen
Teredo	Kapselung von IPv6-Datenpaketen in IPv4-UDP-Datenpaketen
SIIT	Stateless IP/ICMP Translation

Abbildung 11 IPv6-Überangsmechanismen

3.1 Tunnel-Mechanismus

Viele der Übergangsmechanismen bauen darauf auf, dass IPv6-Pakete in IPv4-Pakete gepackt und dann über das IPv4 Netzwerk verschickt werden. Im Prinzip das IPv6-Paket einen zusätzlichen IPv4 Header zugewiesen. Indem die Protokoll-Version des IPv4-Headers auf 41 gesetzt wird, weiß der Empfänger, dass als Payload ein IPv6-Paket geliefert wird. Abbildung 12 verdeutlicht diesen Vorgang.

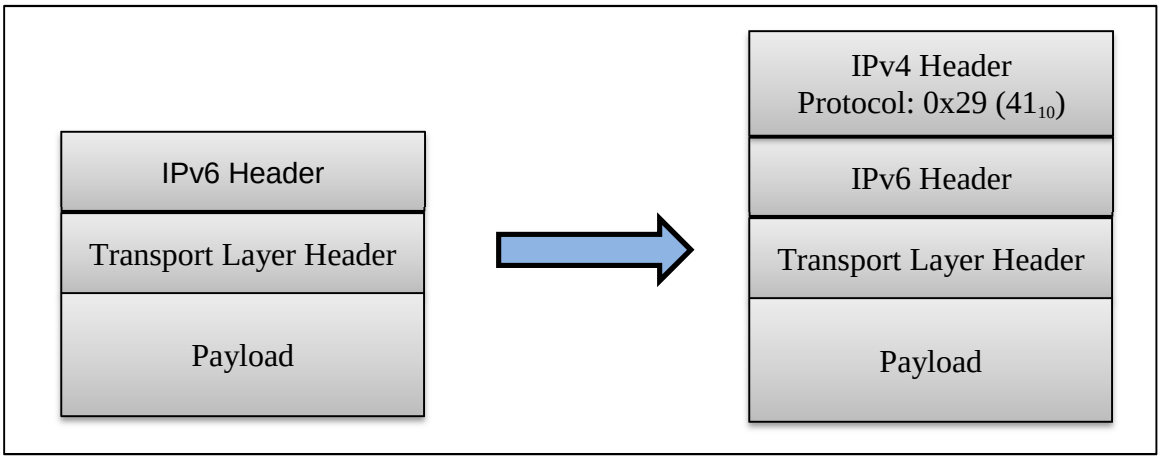

Abbildung 12 Einkapseln des IPv6 Paketes in IPv4

Diese Darstellung stellt den vereinfachten Fall dar, bei dem das IPv6 Paket die Maximalgröße des IPv4-Payloads nicht übersteigt. Ansonsten müssen weitere Regeln bezüglich der Fragmentierung beachtet werden – nähere Informationen sind im RFC 4213 ab Sektion 3.2 zu finden.

3.2 6in4

Die 6in4-Technologie wurde erstmals 1996 im RFC 1933 veröffentlicht, die letzte Spezifierung erfolgte 2005 im RFC 4213. 6in4 wird genutzt, um zwei Systeme über IPv6 miteinander zu verbinden, die untereinander nur eine IPv4-Verbindung besitzen. 6in4 Tunnel (auch "manual tunnels", "static tunnels", "protocol 41 tunnels", "configured tunnels" genannt) sind Point-to-Point Tunnel, die manuell konfiguriert werden.

Hierbei wird bspw. Rechner A am Tunnelanfang eine beliebige (lokale) IPv6 Adresse zugewiesen, bei der Konfiguration aber auch gleichzeitig die IPv4-Adresse des Tunnelendpunktes Rechner B mitgeteilt.

Rechner B bekommt wiederum eine lokale IPv6-Adresse im Adressbereich der IPv6-Adresse von Rechner A zugewiesen. Auch hier wird nun die IPv4-Adresse des Gegenübers angegeben. Dadurch wissen beide Tunnelendpunkte, wie Sie ihr gegenüber erreichen können. Die Konfiguration für dieses Projekt ist in 4.1 Verbindung vom Ilux150 zum Iltis beschrieben.

Da hierbei manuell IPv6-Adressen auf die jeweiligen IPv4-Adressen des anderen Systems geleitet werden, ist die Implementation aufwändiger als bei automatischen Systemen.

Für Systeme, die bei jedem Neustart eine andere IPv4-Adresse von einem DHCP-Server zugewiesen bekommen sind die statischen 6in4 Tunnel eher ungeeignet, da bei jeder Anpassung der IPv4-Adresse der Gegenseite der Tunnel angepasst und neu aufgebaut werden muss.

Abhilfe könnte hierbei Tunnelbroker mit statischer IPv4-Adresse, wie bspw. der Hurricane Electric Free IPv6 Tunnel Broker, schaffen. Nutzt man solch einen Tunnel Broker und übermittelt bei jedem Reconnect automatisch die aktuelle IPv4 Adresse per DynDNS an den Broker, kann dieser seinen Tunnelendpunkt automatisch anpassen und die Funktionsfähigkeit wiederherstellen. Für unsere Zwecke ist dies allerdings nicht notwendig, da statische IPv4 Adressen vorhanden sind.

3.3 6to4

6to4 ist ebenfalls ein Tunnelmechanismus, der IPv6-Pakete über IPv4-Netzwerke überträgt. Im Gegensatz zu 6in4 liegt das Hauptaugenmerk darauf, einen einzelnen Rechner zu konfigurieren.

Er wurde im Februar 2001 im RFC 3056 veröffentlicht und im Juni 2001 zur erleichterten Konfiguration im RFC 3068 um eine Anycast-Adresse für 6to4 Relay Router erweitert.

Mittels 6to4 wird es ermöglicht, ein Gerät mit IPv4-Zugang zum World Wide Web eine globale IPv6 Adresse zuzuweisen und somit eine IPv6-Verbindung zum WWW herzustellen.

Hierbei wird die IPv4-Adresse des Clients in eine 32 Bit lange Hexadezimale Adresse umgewandelt und an die Stelle der Netzadresse zugewiesen. Als Adresstyp wird der für den 6to4-Mechanismus vorbehaltene Typ *2002* genutzt. Standardmäßig ist kein Subnetz gesetzt und die Interface-ID mit *0:0:0:1* angegeben.

In Abbildung 13 wird verdeutlicht, wie die IPv4-Adresse *171.186.18.52*. mittels 6to4 in eine IPv6 Adresse umgewandelt wird.

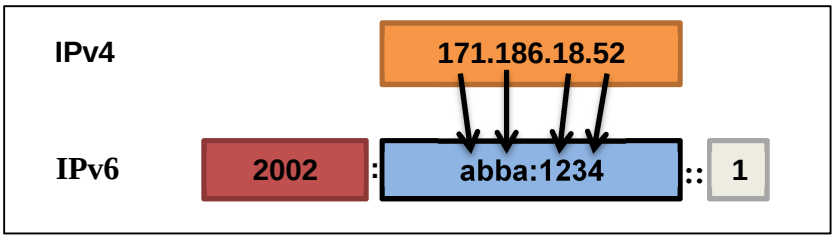

Abbildung 13 Umwandlung IPv4 zu 6to4 IPv6

Damit die Zuordnung eines Netzwerkteilnehmers gewährleistet werden kann, muss dieser also über eine global gültige IPv4 Adresse verfügen. Es reicht nicht, die IPv4 Adresse eines Netzwerkrouters o.ä. anzugeben.

Will der Client nun ein IPv6-Paket über seine IPv4 Verbindung versenden, wird dieses mittels IPv4 Tunneling in ein IPv4-Paket gepackt und anschließend über die vorher konfigurierte Anycast-Adresse *192.88.99.1* versendet.

Unter dieser Adresse sind öffentliche 6to4 Relay Router eingerichtet, die das IPv4 Paket auswerten und als natives IPv6-Paket an den Empfänger weiterleiten.

Es wird also das jeweils nächste öffentliche 6to4 Relay angesprochen und über dieses das eigentliche Ziel erreicht.

Das Antwortpaket wird nun an die 2002er IPv6 Adresse des Clients geschickt. Hierbei wird der Adresstyp erkannt und das IPv6-Paket nun wiederum an das nächste öffentliche 6to4 Relay (des Zieles) geleitet. Dort wird die IPv4-Adresse des Clients berechnet und mittels IPv4-Tunneling das IPv6 Paket wiederum in ein IPv4 Paket gepackt. Dieses erreicht nun den mittels 6to4 konfigurierten Rechner.

Eine anschauliche Grafik sowie die 6to4 Konfiguration innerhalb dieses Projektes sind in Kapitel 4.2 zu finden.

4 Lösungsansatz

Um auf die Sensoren im lokalen IPv6-Netzwerk des Iltis-Servers zugreifen zu können, ohne selbst im Netzwerk der HTW angemeldet zu sein, ist es unerlässlich, einen Server als Weiterleitung zu nutzen, der eine Verbindung ins World Wide Web besitzt. Als Beispiel für solch einen Server soll für diese Lösung wieder der Ilux150 dienen.

4.1 Verbindung vom Ilux150 zum Iltis

Es gilt also zunächst, den Zugriff auf die Sensoren vom Ilux150 aus herzustellen. Hierzu wird sich der in Kapitel 3.2 vorgestellten Technologie 6in4 bedient, um einen Punkt-zu-Punkt Tunnel zwischen Ilux150 und Iltis herzustellen. Beide Server haben eine statische IPv4-Adresse, die sich nicht ändern soll, daher muss die Konfiguration nur einmal geschehen und sollte dann dauerhaft funktionieren. Als IP-Adressen für die beiden Endpunkte wurde das bisher verwendete Präfix *aaaa::/64* des bereits genutzten Sensornetzes als Orientierung genutzt.

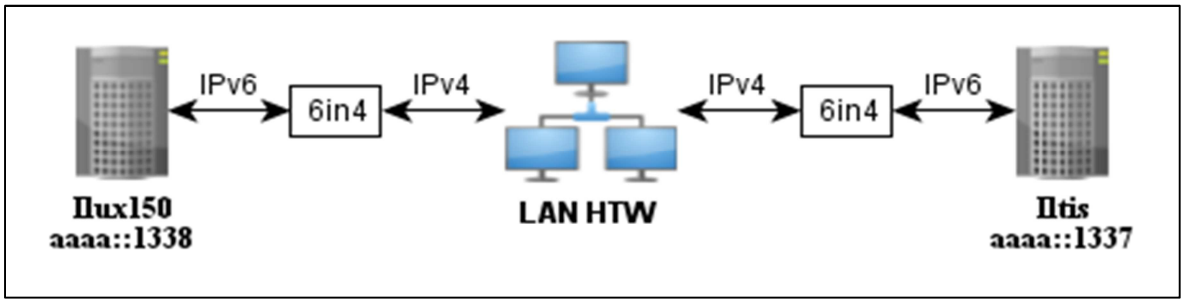

Abbildung 14 - 6in4 Verbindung vom Ilux150 zum Iltis

Die Einträge in die jeweilige /etc/network/interfaces sind wie folgt zu setzen:

Tabelle 2 6in4 Konfiguration der Server

Ilux150	Iltis
auto 6in4tun iface 6in4tun inet6 v4tunnel address aaaa::1338 netmask 64 endpoint 141.56.134.114	auto 6in4tun iface 6in4tun inet6 v4tunnel address aaaa::1337 netmask 64 endpoint 141.56.132.150 gateway aaaa::1338

Wird nun die Verbindung bspw. vom Ilux150 aus mittels ping6 aaaa::1337 getestet, kommen die erwarteten Antwortpakete innerhalb kürzester Zeit zurück – die Verbindung funktioniert also. Als nun versucht wurde einen Sensor zu erreichen, indem man bspw. ping6 aaaa::221:2eff:ff00:1962 ausführt, wurden sämtliche Pakete verworfen.

Nach einiger Fehlersuche stellte sich heraus, dass die Routingfunktionalität für IPv6-Pakete am Iltis nicht aktiviert war. Daher wurden die eingehenden Pakete schlichtweg verworfen. Dies ließ sich mit folgendem Befehl beheben:

```
sudo echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Da nun sämtliche IPv6 Pakete (gemäß der Routing-Tabelle) im Iltis weitergeleitet werden, ist ein Zugriff auf die Sensoren möglich (siehe Abbildung 15). Da der Sensorknoten bereits als Router fungiert, mussten hier keine weiteren Einstellungen vorgenommen werden.

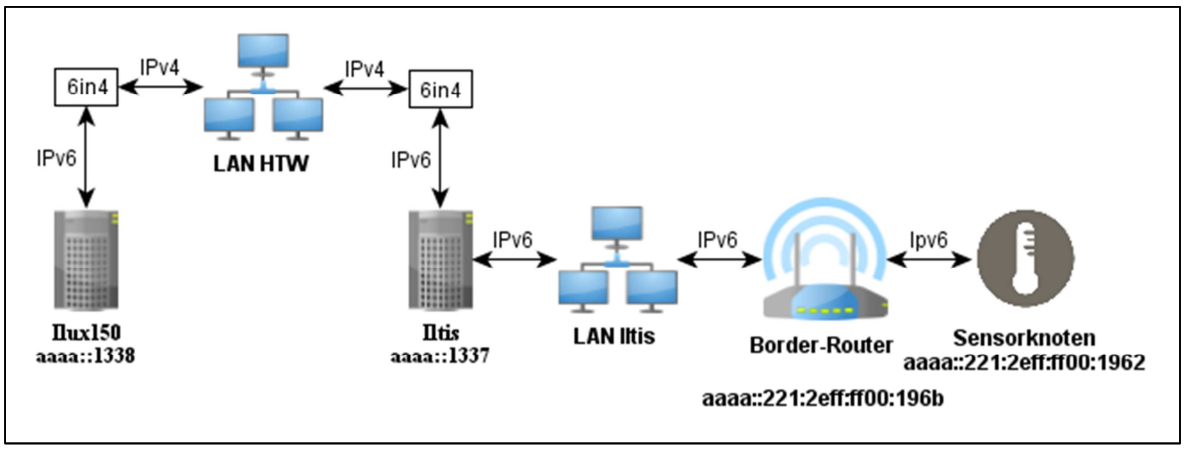

Abbildung 15 - Verbindung vom Ilux150 bis zu den Sensoren

Da im 6in4-Tunnel auf Seiten des Iltis der Ilux150 als Gateway angelegt wurde, wird auch die Antwort des Sensors über den Ilux150 geleitet und gelangt von dort aus zum Client.

4.2 Verbinden des Ilux150 mit dem IPv6-WWW

Die Sensoren sind nun vom Ilux150 aus zu erreichen, da eine IPv6-Verbindung zum Iltis hergestellt ist. Als nächstes ist es erforderlich, den Ilux150-Server über das Internet unter Verwendung von IPv6 zugänglich zu machen.

Hierfür bietet sich die in Kapitel 3.3 vorgestellte Konfiguration einer 6to4 Verbindung an.

Wird die IPv4-Adresse *141.56.132.150* des Ilux150 in eine 6to4-IPv6 Adresse umgewandelt, ergibt sich die IPv6-Adresse *2002:8d38:8496::1*.

Nun kann der 6to4-Tunnel in der /etc/network/interfaces eingerichtet werden:

```
auto tun6to4
iface tun6to4 inet6 v4tunnel
        address 2002:8d38:8496::1
    netmask 16
        gateway ::192.88.99.1
        endpoint any
        local 141.56.132.150
```

Listing 3 6to4-Adapter Konfiguration

Damit ist der Ilux150 nun in der Lage, sämtliche verfügbaren IPv6-Adressen des World Wide Web zu erreichen bzw. von dort aus erreicht zu werden (siehe Abbildung 16).

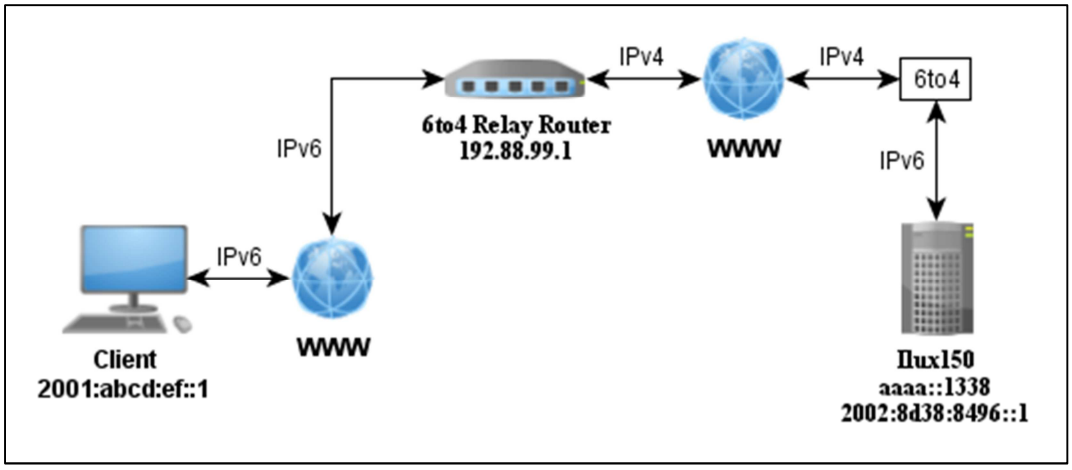

Abbildung 16 - 6to4 Verbindung des Ilux150

Anmerkung: Auch hierbei wird die IPv4-Protokollnummer 41 verwendet, diese darf nicht von Firewalls etc. blockiert sein.

4.3 Anpassungen des lokalen IPv6 Netzwerkes vom Iltis

Mit obig vorgestellten Konfigurationen ist es nun möglich, den Server Ilux150 mittels IPv6 aus dem World Wide Web zu erreichen. Das Ziel soll es sein, die Sensoren des IPv6 Netzwerkes des Iltis Servers abzufragen.

Würde der Client nun versuchen, den Sensor mit der Adresse *aaaa::221:2eff:ff00:1966* zu erreichen, könnte keine Verbindung zu diesem hergestellt werden. Der Client kann nicht wissen, dass Sensoren mit dem Präfix *aaaa::/64* am Iltis-Server angeschlossen sind, also kann auch keine Weiterleitung dorthin erfolgen. Es wäre nun möglich, eine Route einzurichten, die dies berücksichtigt – das Ziel ist aber eine Möglichkeit zu finden, ohne weitere Konfiguration eines Clients auf die Sensoren zugreifen zu können.

Bei der Betrachtung der IPv6-Adresses des Ilux150 (*2002:8d38:8496::1*) fällt auf, dass nach dem globalen Präfix *2002:8d38:8496* direkt eine *1* als Interface-ID kommt. Ein Subnetz wurde hier nicht konfiguriert.

Da der Zugriff auf den Iltis über den Ilux150 erfolgen soll, ergab sich die Schlussfolgerung ein Subnetz einzurichten, dessen globales Präfix die 6to4-Adresse des Ilux150 sein soll.
In Analogie zu dem bisher verwendeten Präfix des internen Sensornetzes wurde als Subnetz-Adresse *aaaa* festgelegt.

Das vollständige Präfix der IPv6 Adressen im Ipv6 Netzwerk des Iltis Servers lautet damit *2002:8d38:8496:aaaa::/64*. Um dem Ilux150 dieses Subnetz bekanntzumachen, muss auch der 6in4-Tunnel mit dem neuen Präfix angepasst werden.

Nach erfolgten Änderungen lautet die Adresse des Sensors nun *2002:8d38:8496:aaaa:221:2eff:ff00:1966*. Fragt ein Client diese an, wird erkannt, dass es sich hierbei um eine 6to4 Adresse handelt, und das Paket von einem 6to4 Relay Router zum Ilux150 weitergeleitet (siehe Abbildung 17).

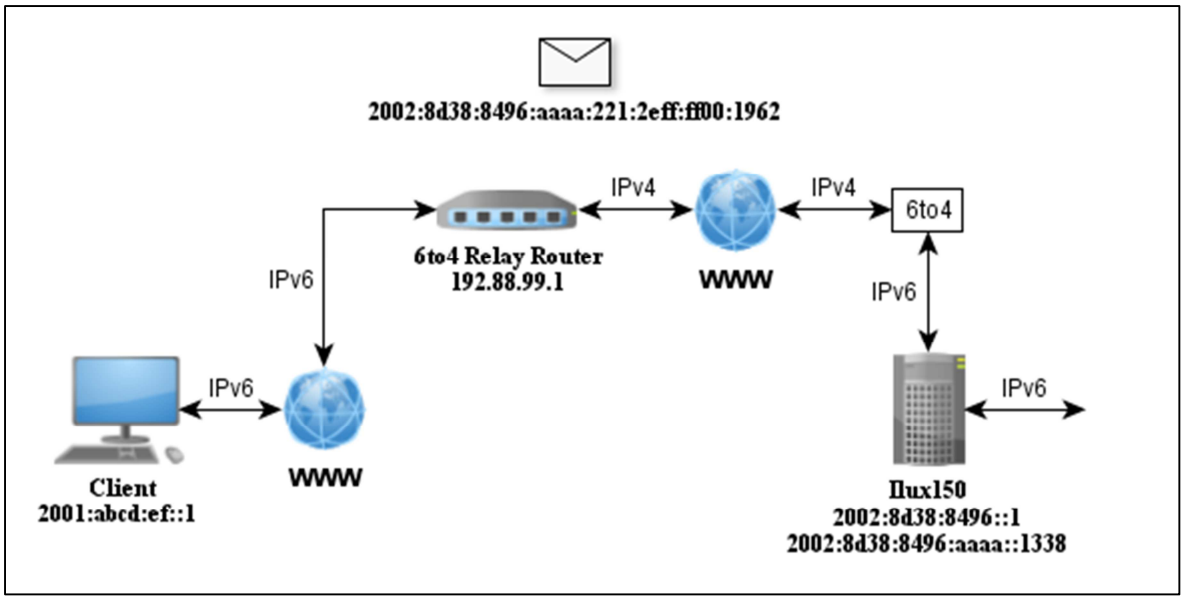

Abbildung 17 - Abfrage des Sensors Teil 1

Auf dem Ilux150 wird erkannt, dass die Nachricht an einen Teilnehmer des Subnetzes *aaaa* gehört, es erfolgt eine Weiterleitung durch den 6in4-Tunnel zum Iltis. Dieser wiederum erkennt, dass die IPv6-Adresse zu seinem Subnetz gehört und leitet das Paket zum Router weiter. Der Router wiederum kennt den Sensor direkt, sendet das Paket dort hin und mit diesem Schritt ist die Anfrage am Ziel angelangt (siehe Abbildung 18).

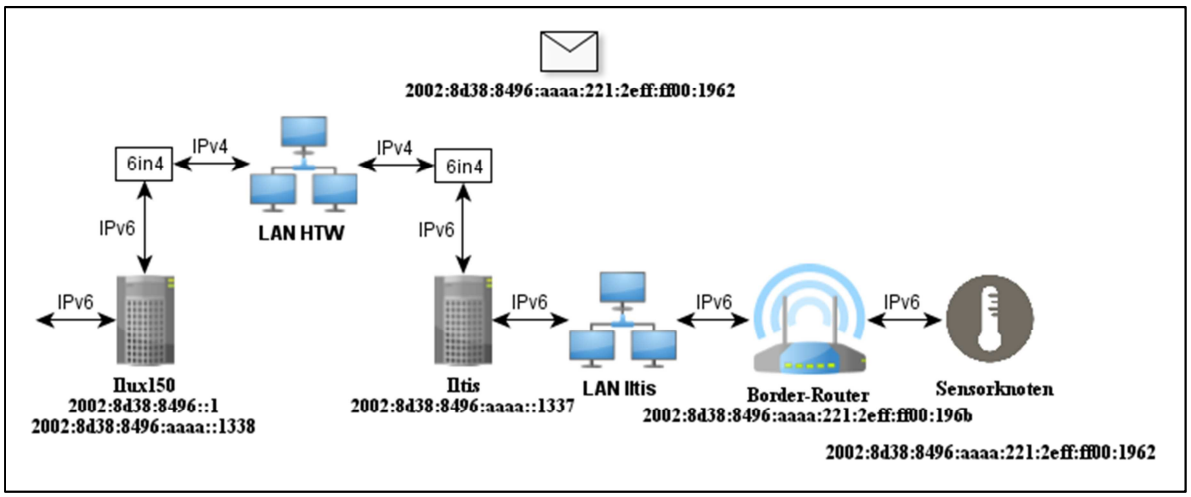

Abbildung 18 - Abfrage des Sensors Teil 2

Die Antwort des Sensors wird nun über den im 6in4-Tunnel auf Seiten des Iltis definierten Gateway, also den Ilux150 geleitet und gelangt von dort aus über den nächsten 6to4 Relay Router zum Client.

5 Fazit und Ausblick

Es konnte eine Lösung gefunden werden, die Sensoren über das World Wide Web erreichbar zu machen. Beim Test der Verbindung konnten keine Probleme der Verbindung festgestellt werden, es funktionierte reibungslos.

Durch die Nutzung von 6in4-Tunneln, muss jede Änderung an den Netzwerkadressen des Ilux150 bzw. des Iltis in der Gegenseite angepasst werden.

Zudem ist der Iltis-Server mit seinen Komponenten nun über den Ilux150 an das IPv6-WWW angebunden, so dass sämtliche verfügbaren Seiten aufgerufen werden können.

Es ist also unbedingt die Sicherheit dieser Konfiguration zu überprüfen und auszuwerten. Für die Sicherheit bei der Verwendung von 6in4 existiert ein eigenes RFC mit der Nummer 3964. Im Februar 2014 wurde zudem das RFC 7123 veröffentlicht, welches sich ebenfalls mit Sicherheitsaspekten von IPv6 in IPv4-Netzwerken befasst.

Da die Anbindung an das IPv6 Internet über 6to4 ein Übergangsmechanismus ist und mit der Verfügbarkeit und Funktionalität der Relay-Router steht und fällt, sollte zukünftig darüber nachgedacht werden, eine native IPv6-Verbindung einzurichten.

Literaturverzeichnis

- Projektdokumentation „Integration des 802.15.4-Sensornetzes in FHEM“
  Marcus Kupke, Markus Fischer
- http://tools.ietf.org/html/rfc6690, [Online] 20 01, 2014
  Constrained RESTful Environments (CoRE) Link Format [Online] 20 01, 2014
- http://tools.ietf.org/html/draft-ietf-core-coap-13
  Constrained Application Protocol (CoAP) draft-ietf-core-coap-13. [Online] 18 01, 2014
- http://tools.ietf.org/html/draft-li-core-conditional-observe-02,[Online] 03 02, 2014
- Constrained Application Protocol. [Online] 07 11, 2013. [Cited: 07 19, 2013.] http://en.wikipedia.org/wiki/CoAP#Implementations.
- IPv6 - Was ist neu? *www.bs-roth.de* . [Online] [Zitat vom: 26. 01 2014.] http://www.bs-roth.de/neu/schueler/projekte2012/ipv6/index.php?Was_ist_neu%3F.
- Heise Netze - IPv6: Das Mega-Netz. *Heise Netze.* [Online] [Zitat vom: 26. 01 2014.] http://www.heise.de/netze/artikel/IPv6-Das-Mega-Netz-221708.html.
- Wikipedia. Liste von IPv6-Tunnelbrokern. [Online] 26. 01 2014. [Zitat vom: 26. 01 2014.] http://de.wikipedia.org/wiki/Liste_von_IPv6-Tunnelbrokern.
- Network Working Group. RFC 4213 - Basic Transition Mechanisms for IPv6 Hosts and Routers. [Online] 10 2005. [Zitat vom: 26. 01 2014.] http://tools.ietf.org/html/rfc4213.
- Inc., Wikimedia Foundation. 6to4 - Wikipedia. [Online] Wikimedia Foundation Inc. [Zitat vom: 17. 02 2014.] http://de.wikipedia.org/wiki/6to4.
- Wikipedia. Constrained Application Protocol. [Online] 11. 07 2013. [Zitat vom: 19. 07 2013.] http://en.wikipedia.org/wiki/CoAP#Implementations.