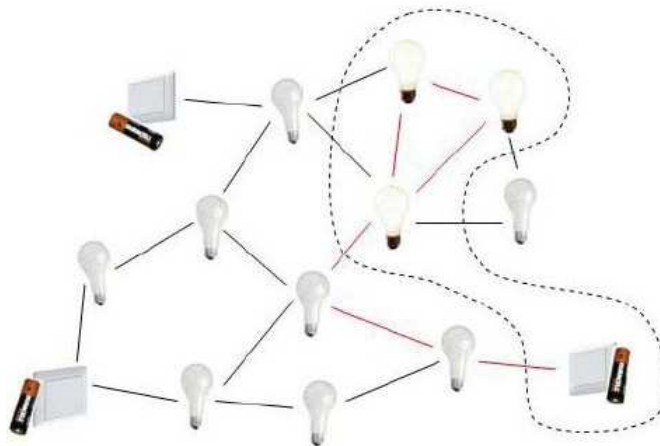


Philippe Bourquin (D-INFK)

## "LightNet"



---

Semesterarbeit  
Sommersemester 2005

Betreuung: Pascal von Rickenbach,  
Distributed Computing Group (DCG)

---



# Zusammenfassung

”LightNet” ist eine mögliche, realistische Sensornetz-Anwendung, bei der Taster und Leuchteinheiten drahtlos miteinander verbunden sind. Dabei sollen von den Tastern ausgelöste Ereignisse über mehrere Leuchten hinweg transportiert werden können. In dieser Arbeit werden mögliche Architekturen und Semantiken von ”LightNet” vorgestellt und auf ihre Probleme hin analysiert. Das Ziel ist es, eine bestimmte Architektur herauszuarbeiten und genauer zu untersuchen. Aufbau, Semantik und Routing in “LightNet” werden die zentralen Aspekte dieser Arbeit sein.



# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.1.1	Sensornetze: Eine Einführung . . . . .	1
1.1.2	Sensornetze zur Lichtsteuerung? . . . . .	2
<b>2</b>	<b>Komponenten und Aufbau von “LightNet”</b>	<b>3</b>
2.1	Grundelemente . . . . .	3
2.1.1	Taster . . . . .	4
2.1.2	Leuchten . . . . .	4
2.2	Netzwerk . . . . .	5
<b>3</b>	<b>Semantik</b>	<b>7</b>
3.1	Gruppen . . . . .	7
3.1.1	Motivation . . . . .	7
3.1.2	Szenarien . . . . .	7
<b>4</b>	<b>Routing</b>	<b>11</b>
4.1	Flooding . . . . .	11
4.2	Routing entlang eines Spannbaums . . . . .	11
4.3	Spannbaumkonstruktion . . . . .	12
4.3.1	Routing Algorithmus . . . . .	12
<b>5</b>	<b>Zuordnung</b>	<b>17</b>
5.1	Zuordnung ohne zusätzliche Infrastruktur . . . . .	17
5.1.1	Taster-zu-Gruppen Zuordnung . . . . .	17
5.1.2	Leuchten-zu-Gruppen Zuordnung . . . . .	17
5.2	Zuordnung mithilfe eines Servers . . . . .	18
5.2.1	Synchronisation . . . . .	18
<b>6</b>	<b>Fazit und Schlussbemerkungen</b>	<b>19</b>
6.1	Ausstehende Probleme . . . . .	19
6.2	Persönlicher Kommentar . . . . .	19
6.2.1	Alternativer Taster mit zwei Funktionen . . . . .	19
6.3	Schlussfolgerungen . . . . .	20
	<b>Literaturverzeichnis</b>	<b>21</b>



# Kapitel 1

## Einführung

### 1.1 Motivation

#### 1.1.1 Sensornetze: Eine Einführung

##### Definition

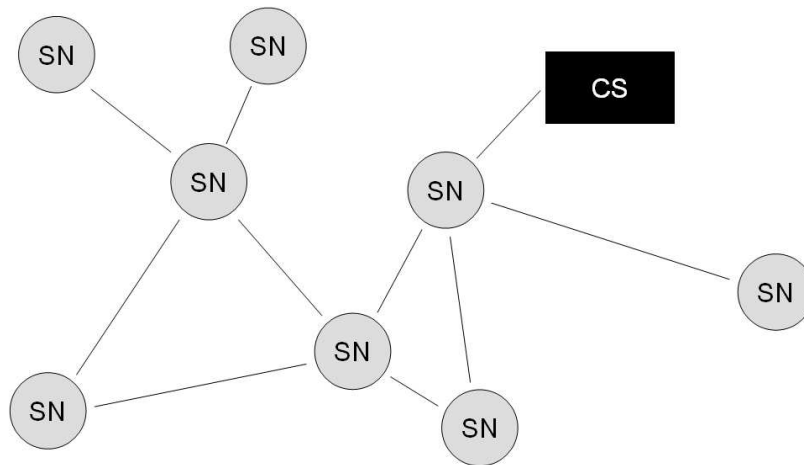
Ein Sensornetz (engl. sensor network) ist ein Rechnernetz aus Kleinst-Computern, sog. Sensorknoten oder auch nur Knoten, die mit Sensoren ausgestattet sind und durch Zusammenarbeit eine gemeinsame Aufgabe bewältigen. [5]

Das Anwendungsgebiet von Sensornetzen ist denkbar gross. Einige in der Forschung oft erwähnte Beispiele sind:

- Tierbeobachtung mit Hilfe von Bewegungssensoren (engl. Habitat Monitoring)
- Gebäude- oder Geländesicherung durch Detektion von Geräuschen
- Einsatz im militärischen Bereich (z.B Information aus feindlichem Gebiet)
- Überwachung der Körperfunktionen eines Sportlers
- Früherkennung von Katastrophen wie Waldbränden und Erdbeben

Die Sensoren können von beliebiger Art sein. Bewegungs-, Audio-, Licht-, Druck-, Feuchtigkeits- und Beschleunigungs-Sensoren und viele mehr sind denkbar. Wenn von Sensornetzen gesprochen wird, sind meistens drahtlose Sensornetze (engl. wireless sensor networks) gemeint. Ihre Energie beziehen die Knoten aus einer Batterie und die Kommunikation erfolgt meistens über Funk oder Infrarot, wobei letzteres eine direkte Sichtlinie zwischen zwei Sensorknoten erfordert. Auch die Komponenten von "LightNet" sollen drahtlos miteinander kommunizieren. Ein typischer Aufbau von einem Sensornetz ist in Abbildung 1.1 schematisch dargestellt.

Ein wichtiges Ziel beim Design von Sensorknoten, Netzwerk und Kommunikationsprotokoll besteht darin, die meist stark limitierten Ressourcen wie Speicher, Rechenleistung, Bandbreite und Energiereserven so effizient wie möglich einzusetzen. Insbesondere die Kommunikation sollte auf ein Minimum reduziert werden, da diese am meisten Energie beansprucht und somit signifikanten Einfluss die Lebensdauer eines Knoten beziehungsweise dessen Batterie hat. Sensornetze müssen als ad-hoc Netzwerk ihre Topologie selbständig und möglichst umfassend selbst ausfindig machen, da nicht davon ausgegangen werden kann, dass eine fixe Infrastruktur zur Verfügung steht. Bekannte Verfahren müssen hierzu in Einklang mit sparsamer Kommunikation gebracht werden.



**Abbildung 1.1:** Schematischer Aufbau eines Sensornetzwerks. Die einzelnen Sensor Knoten (SN) messen Signale aus der Umgebung und leiten diese an einen zentralen Server (CS) weiter.

### 1.1.2 Sensornetze zur Lichtsteuerung?

Wie kann nun ein Sensornetzwerk mit einer Lichtsteuerung sinnvoll kombiniert werden und welche Vorteile gegenüber herkömmlichen Lichtsteuerungsanlagen ergeben sich daraus?

Ein Beleuchtungssystem besteht üblicherweise aus Taster (Lichtschalter) und Leuchten (Lampen), die auf verschiedene Arten miteinander verkabelt sein können um die gewünschten “Taster→Leuchten”-Zuordnungen zu erreichen. Die Taster sowie auch die Leuchten haben ihre fixen Plätze an der Wand, beziehungsweise an der Decke. Bei “LightNet” sollen sowohl Taster wie auch Leuchten beliebig platziert werden können. Weiter sollten auch Zuordnung der Taster zu den einzelnen Leuchten ohne grossen Aufwand geändert werden können. Die eigentlichen Sensoren in so einem Netzwerk stellen die Taster dar, welche mit einem einfachen Drucksensor ausgestattet werden können. Wird dieser Sensor aktiviert, d.h der Taster wird betätigt, so muss der Taster reagieren und eine Aktion ausführen, welche früher oder später zum gewünschten Ergebnis, nämlich das Licht ein- oder ausschalten, führt.

Die Leuchten selbst besitzen keine Sensoren im eigentlichen Sinn, sondern besitzen ein Modul, welches für die Steuerung (An/Aus), Kommunikation und Programmierung (Zuordnung) verantwortlich ist. Da Leuchten im Allgemeinen am gebäudeeigenen Stromnetz angeschlossen sind um ihre Energie zu beziehen, muss hier weniger stark auf eine Minimierung des Energieverbrauchs geachtet werden. Die Taster allerdings werden von einer Batterie gespeist und sollten daher möglichst wenig Energie verbrauchen, so dass die Batterien höchstens alle paar Jahre gewechselt werden müssen.

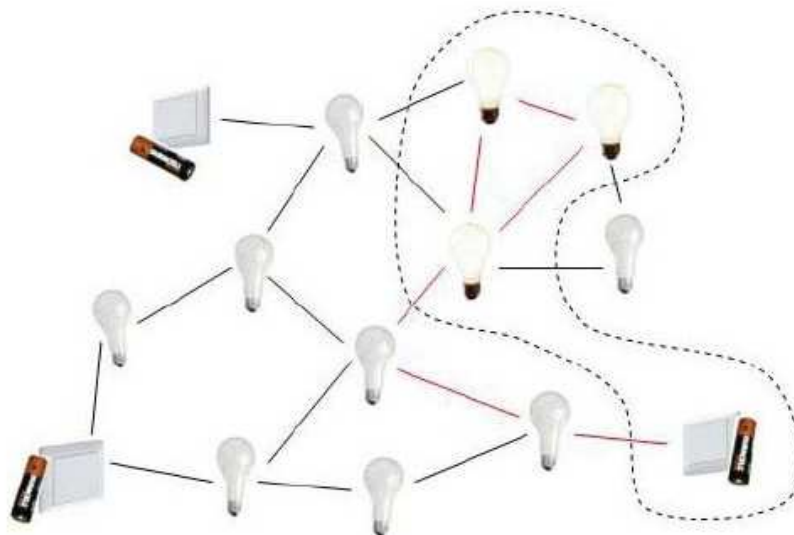
Da nur die Taster mit eigentlichen Sensoren ausgestattet sind, ist Bezeichnung Sensornetz für “LightNet” nicht ganz perfekt, kommt aber der verwendeten Art von Netzwerk am nächsten. Zudem hat auch das Routing, das in Kapitel 4 genauer diskutiert wird, grosse Ähnlichkeit mit demjenigen in Sensornetzwerken. Im nächsten Kapitel wird zunächst mal auf die einzelnen Komponenten und deren Aufgaben eingegangen.

## Kapitel 2

# Komponenten und Aufbau von “LightNet”

### 2.1 Grundelemente

“LightNet” besteht aus zwei Hauptkomponenten, nämlich die Taster und die Leuchten. Je nach Anwendungsgebiet und Anforderungen lassen sich natürlich noch mehr Elemente hinzufügen, wie zum Beispiel ein zentraler Server, an welchem dann die Programmierung der Leuchten vorgenommen würde (siehe dazu Kapitel 5), oder zusätzliche Relay-Knoten, die ausschliesslich zur Weitergabe von Paketen dienen. Ein einfacher Aufbau von “LightNet” ohne zusätzliche Komponenten ist in Abbildung 2.1 illustriert.



**Abbildung 2.1:** Mögliches Setup eines “LightNet” Systems. Im Unterschied zu einem klassischen Sensornetzwerk werden hier die Daten nicht zu einem zentralen Server weitergeleitet, sondern von einzelnen Taster zu mehreren Leuchten innerhalb des Netzwerks.

### 2.1.1 Taster

#### Aufgabe

Die Taster haben die Aufgabe auf Druck hin ein Ereignis auszulösen. Dieses sollte möglichst genau zu dem vom Benutzer gewünschten Resultat führen. Jeder Taster hat also die Aufgabe, die ihm zugeordneten Leuchten entweder alle ein- oder auszuschalten; genau so wie bei herkömmlichen Lichtschalter. Das ausgelöste Ereignis wird danach über Funk an die Umgebung, beziehungsweise an die Leuchten, übermittelt. Der Energieverbrauch soll dabei so gering wie möglich gehalten werden, damit die Lebensdauer der Batterien maximiert wird. Eine Möglichkeit den Energieverbrauch drastisch zu senken besteht darin, die Schalter nur zum Senden von Paketen, aber nicht für dessen Empfang, auszustatten. Denn ständiges horchen auf ankommende Pakete kostet wertvolle Energie. Ist diese Massnahme zu drastisch, so kann wenigstens versucht werden das Zeitintervall, in dem die Taster auf ankommende Pakete horchen müssen, einzuschränken. Mit einem schwächeren Funksignal lässt sich ebenfalls Energie einsparen. Die Funksignalstärke der Taster sollte gerade so gross sein, dass die nächsten paar Leuchten im Empfangsbereich liegen. Auf keinen Fall soll die Anforderung gestellt werden, dass das Signal jede Leuchte im System zu erreichen hat. Stattdessen sollen die Leuchten unter sich Ereignisse weiterleiten, das heisst die Leuchten müssen Routingfähig sein (siehe Kapitel 4).

#### Layout

Es sind verschiedene Tasterformen denkbar. Welche man wählt, hängt davon ab wieviele verschiedene Ereignisse man mit einem Taster auszulösen gedenkt. Ein Taster mit zwei Knöpfen (Subtaster) ist sicherlich imstande mindestens zwei verschiedene Ereignisse auszulösen. Erlaubt man gleichzeitiges oder verschieden langes Drücken der Knöpfe, könnten mit 2 Knöpfen pro Schalter sogar mehr als zwei Ereignisse unterschieden werden. Allerdings nimmt die Benutzungsfreundlichkeit mit zunehmender Komplexität der Taster ab. Um die Bedienung einfach und intuitiv zu halten, sollte das Erscheinungsbild der Schalter von "LightNet" möglichst demjenigen von herkömmlichen Lichtschaltern entsprechen. Deshalb beschränken wir uns im Folgenden auf Schalter mit nur einem Knopf.

### 2.1.2 Leuchten

Die primäre Aufgabe der Leuchten ist, auf Wunsch des Benutzers ein- oder auszuschalten. Bei "LightNet" kommen allerdings noch zusätzliche Aufgaben dazu. Die Leuchten sind wie die Schalter mit einer Kommunikationseinheit ausgestattet, welche per Funk mit anderen Leuchten (oder Schalter) kommunizieren kann. Da die Leuchten ihre Energie zum Betrieb der eigentlichen Leuchteinheit (Leuchtstoffröhre, Halogenscheinwerfer, Glühlampe, ...) sowieso aus dem gebäudeeigenen Stromnetz beziehen, ist Energie keine limitierte Ressource wie bei den Tastern und die einzelnen Komponenten der Leuchte dürfen deshalb auch leistungsfähiger sein. Eine Leuchte, die "LightNet"-tauglich ist, besteht mindestens aus folgenden 3 Komponenten:

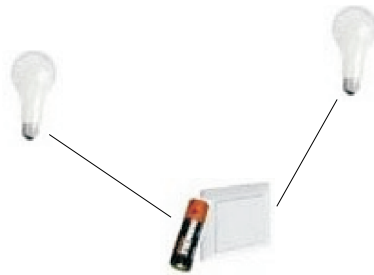
1. Leuchteinheit (Neonröhre, Halogenscheinwerfer, Glühlampe, ...)
2. Controller (CPU, Speicher, ...)
3. Drahtlose Kommunikationseinheit

Der Controller steuert und verwaltet die Gruppenzugehörigkeit einer Leuchte (mehr zu Gruppen in Kapitel 3) und ist für das Routing für die vom Taster losgeschickten Pakete zuständig. Die Kommunikationseinheit empfängt und versendet Pakete

mit einem geeigneten Kommunikationsprotokoll. Damit die Leuchten auch Pakete von den Schaltern empfangen können, muss das verwendete Kommunikationsprotokoll bei beiden identisch sein. Um mit den limitierten Ressourcen der Schalter den Eigenschaften von “LightNet” gerecht zu werden, sollte mit einem proprietären Protokoll mittels “Low-Power”-Funk kommuniziert werden, das heisst die Funksignale der Taster sollten nur gerade soweit wie notwendig reichen, also bis zu den nächstgelegenen Leuchten.

## 2.2 Netzwerk

Das Netzwerk selbst ist ein drahtloses ad-hoc Netzwerk, in welchem Pakete fehlerhaft übertragen werden oder sogar verloren gehen können. Deshalb muss bei der Zugriffskontrolle auf das gemeinsame Kommunikationsmedium zusätzlich darauf geachtet werden, dass Übertragungsfehler detektiert oder korrigiert werden können, wie zum Beispiel mittels eines Cyclic Redundancy Check (CRC). Weiter wird die Annahme getroffen, dass das Netzwerk einen zusammenhängenden, endlichen Graphen bildet, wenn man die Leuchten als Knoten und ihre Kommunikationsverbindungen untereinander als Kanten betrachtet. Die Taster sind hier absichtlich nicht berücksichtigt worden, da der Graph auch ohne Taster zusammenhängend sein sollte. Abbildung 2.2 zeigt ein Gegenbeispiel.



**Abbildung 2.2:** Ungültige Anordnung: Die Leuchten ohne Taster müssen zusammenhängend sein, das heisst die beiden Leuchten müssten untereinander ebenfalls kommunizieren können.



# Kapitel 3

## Semantik

### 3.1 Gruppen

#### 3.1.1 Motivation

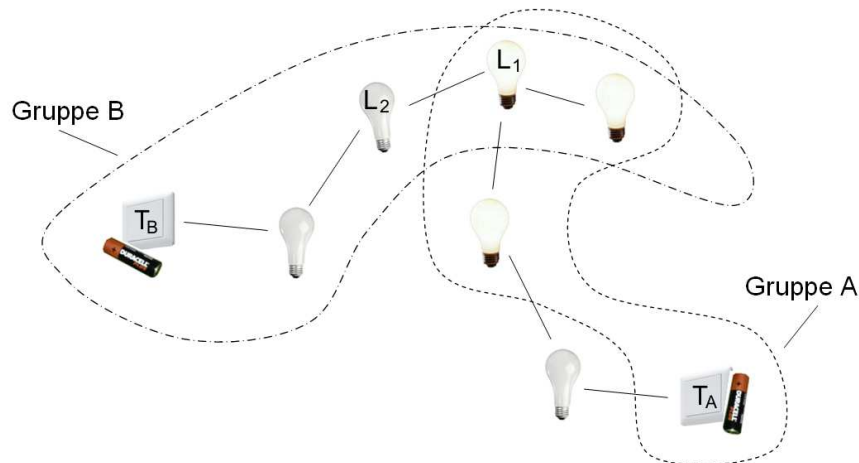
Würde man für jede einzelne Leuchte genau einen dazugehörigen Taster benötigen, erwiese sich dies bei grösseren Räumen mit mehreren einzelnen Leuchten schnell als problematisch; gleich mehrere Taster müssten gedrückt werden um zum Beispiel in einer Vorlesungsaal das Licht einzuschalten. Deshalb werden normalerweise mehrere Leuchten zusammengeschaltet und mit dem gleichen Taster verbunden. Eine solche “1-Taster→mehrere-Leuchten”-Zuweisung muss “LightNet” unbedingt unterstützen. Aber auch das Umgekehrte sollte möglich sein. Angenommen man hat nur eine Leuchte im Raum und möchte diese von zwei Seiten des Raums steuern können, so braucht es ebenfalls eine “mehrere Taster→1-Leuchte”-Unterstützung. Die Anforderung kann nun schnell auf eine “ $x$ -Taster→ $y$ -Leuchten”-Beziehung erweitert werden, wobei  $x \geq 1$  und  $y \geq 1$  ist. Solche Beziehungen lassen sich am einfachsten durch Gruppen realisieren. Einzelne Leuchten und Taster werden zu Gruppen mit eindeutigen Gruppen-Identifikationsnummern (gID) zusammengefasst. Eine Betätigung des Tasters mit Gruppen-Identifikation  $k$  soll nur Auswirkungen auf Leuchten haben, die Gruppe  $k$  angehören und den Zustand (Licht an/aus) der anderen Leuchten nicht beeinflussen.

In Folgenden soll durch zunehmend komplexere Szenarien die verschiedenen Anforderungen an die Semantik von “LightNet” Stück für Stück herausgearbeitet werden.

#### 3.1.2 Szenarien

##### Einfachstes Szenario

Das wohl einfachst denkbare Szenario besteht aus nur einer Gruppe mit einer Leuchte und einem Schalter. Der Schalter löst bei Druck ein Ereignis aus und sendet es der Leuchte. Angenommen die Zuordnungen der Leuchten zu den einzelnen Gruppen sei bereits geschehen, so muss die Leuchte bloss noch überprüfen, ob das empfangene Ereignis für die eigene Gruppe bestimmt ist. Falls ja, schaltet sich die Leuchte ein, beziehungsweise aus, je nachdem in welchem Zustand sie sich befindet. In diesem einfachen Szenario braucht es abgesehen von der Gruppeninformation keine zusätzlichen Informationen im gesendeten Packet. Die von der Leuchte auszuführenden Aktionen sind eindeutig. Dies gilt auch für Anordnungen mit mehreren Schaltern, die verschiedenen, sich nicht überlappenden Gruppen zugeordnet sind.



**Abbildung 3.1:** Gruppe A und Gruppe B überlappen sich, was zu inkonsistenten Zuständen innerhalb der einzelnen Gruppe führen kann. Wird nun der Taster der Gruppe B gedrückt, muss darauf geachtet werden, dass nachher entweder alle Leuchten der Gruppe B an- oder ausgeschaltet sind.

### Überlappende Gruppen

Überlappen sich zwei Gruppen (siehe Abb. 3.1), ergibt sich mit dem oben beschriebenen Verfahren folgendes Problem: Bei einer nicht vollständigen Überlappung der beiden Gruppen, können Zustände erreicht werden, die nicht konsistent sind, d.h. innerhalb einer Gruppe können einige Leuchten an- und andere ausgeschaltet sein. Wünschenswert wäre jedoch, wenn eine Gruppe nach Drücken eines ihr zugehörigen Tasters einen konsistenten Zustand annehmen würde. Folgendes Beispiel illustriert, wie es zu inkonsistenten Zuständen innerhalb einer Gruppe kommen kann:

Angenommen Taster  $T_A$  wurde betätigt und nur die Leuchte  $L_1$  ist an, so würde mit dem simplen Verfahren, bei dem die Leuchten einfach ihren aktuellen Zustand wechseln, beim Drücken von Taster  $T_B$  die Leuchte  $L_2$  angeschaltet und die Leuchte  $L_1$  ausgeschaltet werden. Da  $L_2$  ausgeschaltet und näher am Taster  $T_B$  ist, kann angenommen werden, dass der Benutzer die Gruppe  $B$  einschalten möchte. Der Leuchte  $L_1$  muss also mitgeteilt werden, welche Aktion (hier einschalten bzw. eingeschaltet lassen) diese auszuführen hat. Da die Schalter nur ein Typ von Ereignis kennen, muss diese Aufgabe von der Leuchte  $L_2$  übernommen werden.  $L_2$  fügt beim weiterversenden des Ereignisses ein weiteres Feld zum Packet hinzu, in welchem steht, dass sich  $L_2$  eingeschaltet hat.  $L_1$  überprüft dieses Feld und bleibt eingeschaltet.

Soweit stimmt noch alles. Wird nun jedoch Taster  $T_a$  betätigt, löscht  $L_1$  ab. Der Gruppenzustand von Gruppe  $B$  ist jetzt inkonsistent. Eine Möglichkeit diese Inkonsistenz aufzulösen bestünde darin, dass  $L_1$  sofort ein Aus-Ereignis an alle Gruppenmitglieder aller anderen Gruppen in welchen sich  $L_1$  befindet schickt. Dies würde dazu führen dass anderen Benutzer des Systems einfach das Licht ausgeschaltet würde. "LightNet" sollte aber nicht erlauben, dass anderen Benutzern das Licht abgeschaltet werden kann.

Diese Forderung kann erfüllt werden, indem jede Leuchte für jede Gruppe die Ein- und Aus-Ereignisse zählt. Die Leuchte reagiert erst dann auf ein Aus-Ereignis, wenn für jede Gruppe, in welcher sich die Leuchte befindet, gilt:

$$\#An - \text{Ereignisse} = \#Aus - \text{Ereignisse}$$

Die Zähler werden bei jedem Einschalten der Leuchte auf Null gesetzt. Dies ist nötig, damit vorhergehende Aus-Ereignisse nicht berücksichtigt werden. In Algorithmus 1, welcher das Verhalten der Leuchten beim Empfang von Ein/Aus-Ereignissen beschreibt, wird pro Gruppe nur ein Zähler verwendet, wobei dieser die Differenz zwischen erhaltenen An- und Aus-Ereignissen speichert.

### Algorithmus 1

Das Packet, welches bei einem An/Aus-Ereignis per Broadcast ausgesendet wird sieht folgendermassen aus:

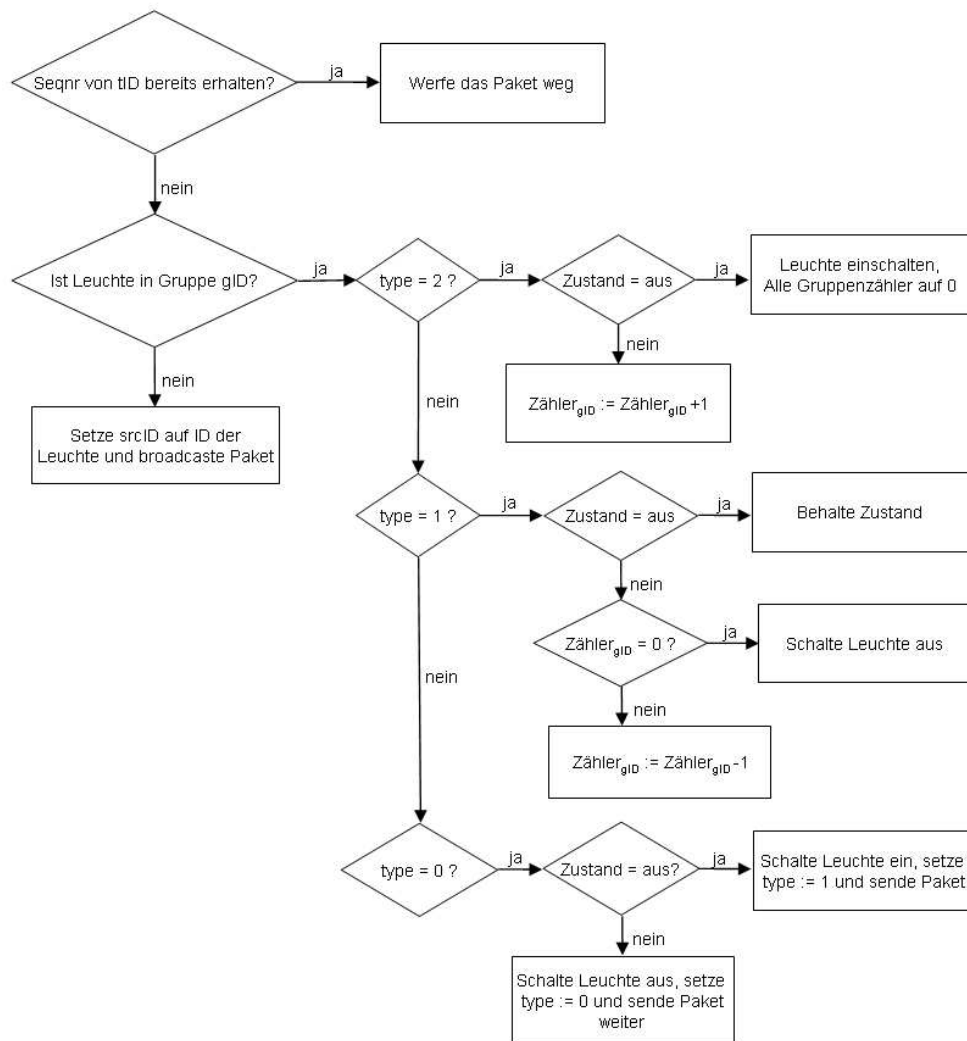
tID	srcID	gID	seqnr	type	ack
-----	-------	-----	-------	------	-----

- **tID**: Taster Identifikationsnummer. Spezifiziert den Taster, der das Packet ursprünglich losgeschickt hat.
- **srcID**: Letzter Absender des Packets. Jeder Knoten, der das Packet weiterleitet überschreibt das Feld mit seiner Identifikationsnummer.
- **gID**: Gruppen Identifikationsnummer. Spezifiziert Gruppe, für die das Packet bestimmt ist.
- **seqnr**: Sequenznummer des Packets. Jeder Taster hat einen eigenen Zähler, dessen Wert beim Absenden eines Packets in das **seqnr**-Feld eingetragen wird und danach inkrementiert wird. Zusammen mit der **tID** identifiziert die Sequenznummer ein Packet eindeutig.
- **type**: Das Typ-Feld spezifiziert die auszuführende Aktion. 1 heisst ausschalten, 2 bedeutet einschalten und 0 steht für nicht definiert.
- **ack**: Legt fest, ob es sich um ein Acknowledgement (ACK) handelt.

Bei Schalterdruck werden lediglich die Felder **tID**, **gID** und **seqnr** ausgefüllt, die restlichen sind alle auf Null gesetzt. Empfängt nun eine Leuchte ein beliebiges On/Off-Ereignis führt sie den Algorithmus dargestellt in Abbildung 3.2 aus. Der Fall, dass das **type**-Feld den Wert 0 hat, tritt genau dann ein, wenn die Leuchte, die das Packet empfängt, Nachbar des Tasters ist, welcher das Packet gesendet hat.

Wichtig ist, dass man auch den (wahrscheinlichen) Fall bedenkt, dass von einem Taster aus mehrere Leuchten der Gruppe direkt erreicht werden können. Haben nun nicht alle denselben Zustand, d.h einige sind an und einige sind aus, würden mit obigen Verfahren die Leuchten einer Gruppe nach Tasterdruck ungleiche Zustände einnehmen und Pakete mit unterschiedlichen Werten im **type**-Feld propagieren. Um dies zu vermeiden, sollte nur eine Leuchte auf Tasterdruck hin reagieren. Dazu müssen sich entweder die Leuchten unter sich entscheiden (Konsensus) oder der Taster wählt eine aus. Die zweite Variante lässt sich leichter implementieren und gibt weniger Kommunikations-Overhead. Dafür müssen die Taster kurzfristig die Möglichkeit haben Signale zu empfangen. Der Ablauf bei Tasterdruck ist dann folgendermassen:

1. Bei Tasterdruck wird ein On/Off-Ereignis mit leerem **type**-Feld gesendet.
2. Der Taster aktiviert den Empfang für kurze Zeit.
3. Eine Leuchte, die ein On/Off-Packet mit **type=0** empfängt, sendet sofort ein ACK-Packet per Broadcast aus.



**Abbildung 3.2:** Algorithmus 1. Wird von einer Leuchte beim Empfang eines On/Off-Packets ausgeführt.

4. Falls das erste ACK-Packet empfangen wird, beendet der Taster den Empfangsmodus sofort und schickt ein weiteres ACK-Packet per Unicast an den Absender des zuvor empfangenen ACK-Packets zurück.
5. Die Leuchte, die das zweite ACK-Packet vom Taster erhalten hat, übernimmt ab jetzt das Routing.

Der Taster weiss mit dem Empfang des ACK-Packets, dass das Netzwerk (zumindest eine Leuchte) erreichbar ist.

# Kapitel 4

## Routing

In diesem Kapitel wird angenommen, dass der Graph, welchen die Kommunikationsverbindungen zwischen den Leuchten untereinander bilden, zusammenhängend und ungerichtet ist, das heisst, die Kommunikation ist immer in beide Richtungen möglich.

### 4.1 Flooding

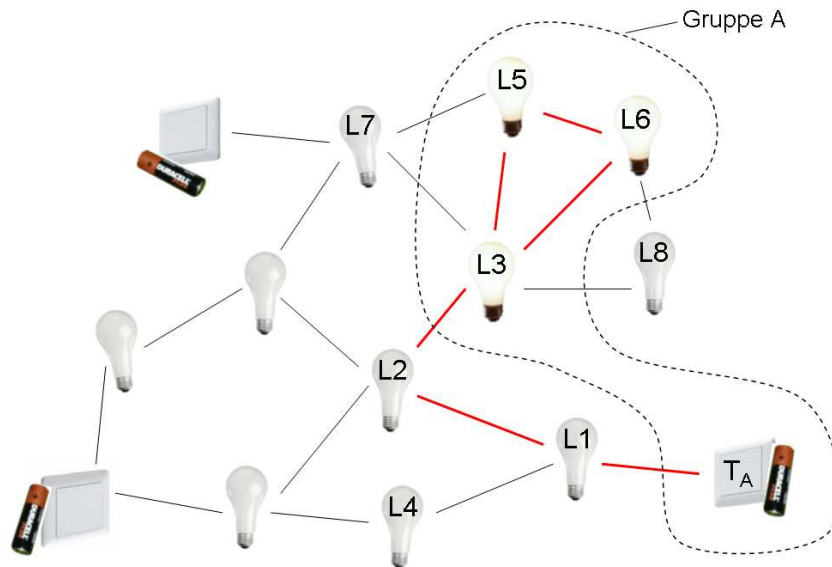
Die einfachste Variante ein Ereignis an alle Gruppenmitglieder weiterzuleiten ist das Flooding. Jede Leuchte sendet ein neues Ereignis an alle Nachbarn weiter. Dabei merkt sie sich die Sequenznummer des Pakets, so dass jedes Paket genau einmal von jeder Leuchte per Broadcast weitergesendet wird.

Flooding birgt den grossen Nachteil, dass bei jedem An- oder Ausschalten irgendeiner Leuchte immer das ganze Netz geflutet wird. Für grössere Installationen kann dies zu gravierenden Performance-Einbussen führen [3].

### 4.2 Routing entlang eines Spannbaums

Um einen effizienteren Routing-Algorithmus auszuführen, brauchen die einzelnen Knoten Informationen bezüglich der Verteilung der Gruppenmitglieder im gesamten Netzwerk. Für jede Gruppe kann ein Spannbaum konstruiert werden, der sich je nach Verteilung der Gruppenmitglieder durch das ganze Netzwerk erstrecken kann. In Abbildung 4.1 ist ein solcher Spannbaum und die zugehörige Gruppe A markiert. Ein On/Off-Ereignis kann nun beginnend beim Taster entlang dieses Spannbaums geroutet werden. Dazu muss jede Leuchte für jede Gruppe wissen, ob sie zu deren Spannbaum gehört oder nicht. Nachdem  $L1$  ein Ereignis von Taster  $T_A$  erhalten hat, genügt es, dass  $L1$  dieses nur an  $L2$  weitersendet, also entlang des Spannbaums.  $L2$  wiederum braucht das Ereignis nur an  $L3$  weiterzusenden, usw. Man beachte, dass sonst immer noch gemäss Algorithmus 1 vorgegangen wird und somit Pakete, die zum zweiten mal ankommen, ignoriert werden (z.B könnte  $L6$  zuerst von  $L3$  und dann von  $L5$  dasselbe Paket erhalten).

Da die Leuchten drahtlos miteinander kommunizieren, können jeweils alle benachbarten Leuchten ein gesendetes Paket sehen. Mit dieser Tatsache kann das oben beschriebene Verfahren vereinfacht werden. Eine Leuchte muss nun nicht mehr wissen, welche ihrer Nachbarn sich ebenfalls im Spannbaum befinden, sondern lediglich, ob sie selbst im Spannbaum ist. Ist dies der Fall, wird das Paket (an alle Nachbarn) weitergesendet, falls nicht, wird es weggeworfen. Nachdem also Leuchte  $L1$  das Paket weitergesendet hat, wird es von  $L2$  und  $L4$  empfangen. Knoten  $L4$  wirft das Paket weg, da er nicht Teil des Spannbaums der Gruppe A ist.  $L2$  hingegen ist



**Abbildung 4.1:** Mögliche Anordnung von Tasten und Leuchten. Eine Kante zwischen zwei Elementen bedeutet, dass diese miteinander kommunizieren können. Die roten Linien stellen den Spannbau für Gruppe A dar.

Teil des Spannsbaums und sendet das Paket erneut per Broadcast weiter. So wird erreicht, dass nicht das gesamte Netzwerk geflutet werden muss, wenn bloss eine kleine Gruppe von Leuchten eingeschaltet werden soll.

### 4.3 Spannbaukonstruktion

Bisher wurde angenommen, dass für jede Gruppe in einem "LightNet" System ein Spannbau existiert, beziehungsweise dass jeder Knoten eine Liste der Gruppen hat, für welche er auf dessen Spannbau liegt. Da die Taster nicht vom Stromnetz abhängig sind, muss angenommen werden, dass diese mobil sind. Das bedeutet, dass ein aufgebauter Spannbau nur solange gültig ist, wie sich die Positionen von Leuchten und Taster nicht ändern (siehe Abb. 4.2).

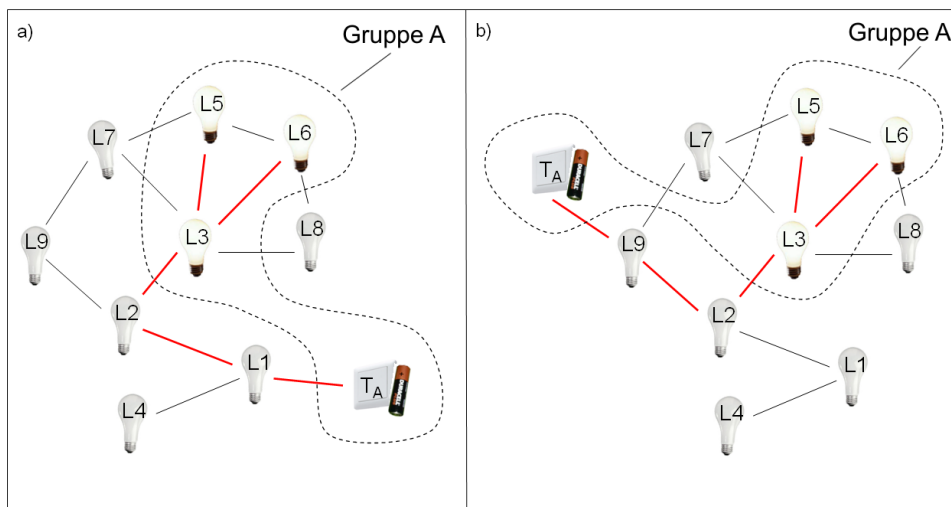
Da das System dynamisch ist, brauchen wir lernfähige Knoten und einen dazu passenden Routing-Algorithmus. Hier nochmals die wichtigsten Anforderungen an die Knoten (Leuchten):

- Jeder Knoten muss für jede Gruppe wissen, ob er **On/Off**-Ereignisse weiter broadcasten soll. Für bestimmte Knoten ist dies jedoch von der Position der Taster abhängig (siehe Abbildung 4.2).
- Die Knoten sollen selbst lernen, ob sie auf dem Weg zu weiteren Gruppenmitgliedern liegen und somit **On/Off**-Ereignisse weiter broadcasten müssen

Damit die Knoten einen Spannbau aufbauen können, muss entweder ein neues Protokoll speziell zur Spannbaukonstruktion entworfen werden, oder der bestehende Algorithmus wird erweitert. In dieser Arbeit wird der zweiten Variante gefolgt.

#### 4.3.1 Routing Algorithmus

Es gibt 3 Typen von Paketen:



**Abbildung 4.2:** Wird ein Taster der Gruppe A verschoben, ändert sich auch der Spannbaum von Gruppe A. Knoten L9 gehört in Szenario a) nicht zum Spannbaum, aber nach dem Verschieben von Taster  $T_A$  (Szenario b)) schon. Mit Knoten L1 verhält es sich gerade umgekehrt.

- **On/Off-Paket:** Steuert das Ein- und Ausschalten der Leuchten
- **ACKREQ-Paket:** Wird generiert, wenn ein Knoten wissen will, ob er im Spannbaum ist
- **ACK-Paket:** Wird von einem Knoten generiert der ein ACKREQ-Paket empfängt und entweder in der Gruppe oder im Spannbaum entsprechender Gruppe ist.

Der Routing Algorithmus ist in Abbildung 4.3 als Flussdiagramm dargestellt. Beim ersten Betätigen eines Tasters aus einer neuen Gruppe, wird ein komplett neuer Spannbaum aufgebaut. In diesem Spannbaum befinden sich alle Gruppenmitglieder, Knoten zwischen Taster und erstem Gruppenmitglied, sowie Nicht-Gruppenmitglieder, die zwei Gruppenmitglieder miteinander verbinden. Nur Knoten, die im Spannbaum der Gruppe  $k$  sind, senden die On/Off-Pakete für Gruppe  $k$  weiter. Für jede Gruppe existiert also ein separater Spannbaum. Der Algorithmus ist für den Fall optimiert, dass Gruppenmitglieder der gleichen Gruppe benachbart sind und "Cluster" bilden. Aber auch wenn dem nicht so ist und z.B Mitglieder der Gruppe A und B abwechslungsweise auftreten funktioniert der Algorithmus.

### Analyse

Ein Problem bei der Analyse des Algorithmus' tritt dann auf, wenn ein Knoten, der entweder ein ACKREQ-Paket generiert oder ein ACKREQ-Paket weitergeleitet hat, keine Antwort, d.h kein ACK-Paket erhält. Die Frage die sich stellt ist, wie lange der Knoten auf eine Antwort warten soll, bevor er sich entscheidet, dass er nicht zum Spannbaum der Gruppe gehört. Eine plausible Möglichkeit wäre, dass man jeden Knoten mit einem Timer ausstattet, welcher nach einer bestimmten Zeit ein Timeout auslöst. Doch wie soll sich der Knoten in der Zeit verhalten, in der er "wartet"? Ereignisse, die für andere Gruppen bestimmt sind kann der Knoten unabhängig bearbeiten. Solche, die für dieselbe Gruppe bestimmt sind, werden solange in eine Warteschlange gestellt bis der Knoten weiss, ob er nun zum Spannbaum dieser Gruppe gehört oder nicht. Denn ohne diese Information, würden ihm 2 Möglichkeiten offenbleiben, welche beide nicht sinnvoll sind:

1. Paket ignorieren, d.h wegwerfen. Als Konsequenz könnten nicht alle Leuchten der Gruppe auf einen Tasterdruck reagieren.
2. Paket broadcasten. Dies kann dazu führen, dass das Netz mit weiteren unnötigen ACKREQ-Pakete geflutet würde.

Angenommen, es existiert noch kein Spannbaum, dann wird in der Lernphase das ganze Netz geflutet. Die erste ACKREQ-Nachricht wird vom ersten Knoten, der nicht in der Gruppe ist, erzeugt und per Broadcast an alle Nachbarn verschickt. Bis jetzt wurde also genau 1 Nachricht gesendet. Die Nachbarn wiederholen den Broadcast, usw. Angenommen, im gesamten Netzwerk existieren keine Gruppenmitglieder, müssen bei einer Netzwerkgrösse von  $n$  Knoten,  $n$  Nachrichten versendet werden. Existiert jedoch ein Cluster aus  $k$  Gruppenmitgliedern unter diesen  $n$  Knoten, so werden  $n - k$  ACKREQ-Nachrichten,  $k$  On/Off-Nachrichten und mindestens 1 ACK-Nachricht versendet. Insgesamt also  $n + 1$  Nachrichten. Bei  $r$  Clustern sind es mindestens  $n + r$  Nachrichten. Im ungünstigen Fall, erreicht eine ACKREQ-Nachricht ein Cluster von mehreren Seiten, d.h mehrere Knoten innerhalb eines Clusters antworten mit einer ACK-Nachricht. Doch auch hier bleibt die Nachrichtenkomplexität in  $O(n)$ . Die Zeitkomplexität hängt von den gewählten Timeouts ab, diese wiederum hängen vom längsten Pfad zwischen zwei Clusters derselben Gruppe ab, das heisst sie müssen bei grösseren Netzwerken tendenziell höher gewählt werden als bei kleinen.

Wird eine Tasterposition geändert, d.h ein Taster wird innerhalb des gesamten Netzwerks verschoben, muss nicht der ganze Spannbaum neu gelernt werden sondern nur der Weg vom Taster zum ersten Spannbaummitglied. Führt mehr als ein Weg zum existierenden Spannbaum, so werden gemäss dem Routing-Algorithmus diese alle benutzt. Dies erhöht die Ausfallsicherheit und verkürzt die durchschnittliche Reaktionszeit einer Leuchte in der Gruppe.

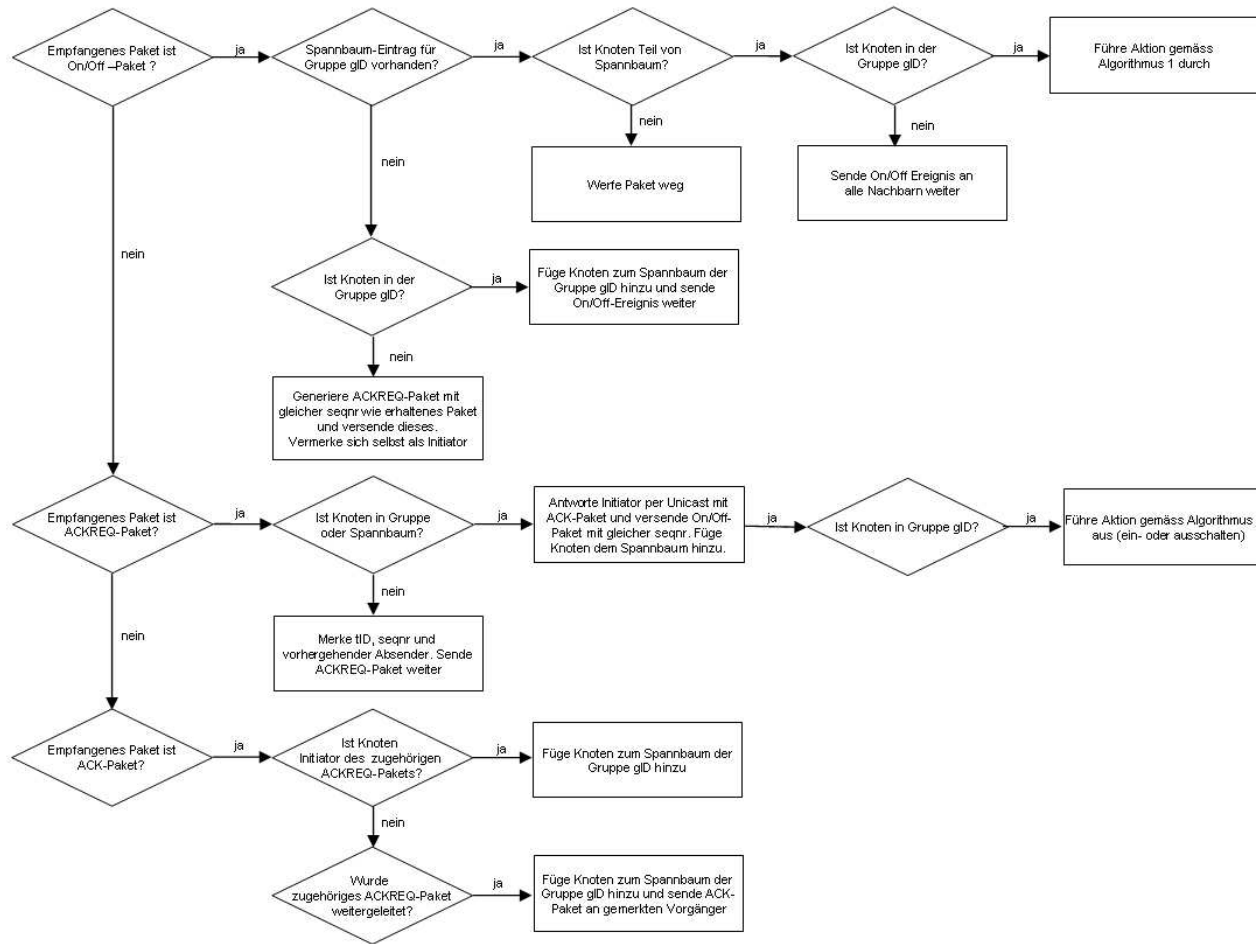


Abbildung 4.3: Routing Algorithmus



# Kapitel 5

## Zuordnung

### 5.1 Zuordnung ohne zusätzliche Infrastruktur

Für kleinere Einsatzgebiete von “LightNet” wie z.B im Home-Bereich wäre es praktisch die Konfiguration ohne zusätzliche Hilfsmittel vornehmen zu können. Unter Konfiguration wird in erster Linie das Gruppenmanagement verstanden. Wie bringt man die einzelnen Leuchten dazu, auf bestimmte Taster zu reagieren und auf andere nicht?

#### 5.1.1 Taster-zu-Gruppen Zuordnung

Zunächst müssen die Taster Gruppen zugeordnet werden. In einem einfachen System könnte die Gruppenidentifikation durch Jumper oder ähnliches direkt am Taster eingestellt werden. So würden sich zum Beispiel mit 8 Jumpern  $2^8 = 256$  verschiedene Gruppen einstellen lassen.

#### 5.1.2 Leuchten-zu-Gruppen Zuordnung

Die Leuchten werden mithilfe der bereits zugeordneten Taster ”programmiert”. Dazu muss ein Taster, der zu der Gruppe gehört zu der die Leuchte hinzugefügt werden soll, in unmittelbare Nähe dieser gebracht werden und per Knopfdruck das Zuweisungssignal ausgelöst werden. Die Übertragungsstärke dieses Zuweisungssignals sollte so eingestellt sein, dass es eine maximale Reichweite von etwa einem Meter hat. So kann vermieden werden, dass unbeabsichtigt weitere Leuchten der Gruppe hinzugefügt werden. Zur Bestätigung könnte entsprechende Leuchte kurz aufleuchten. Neben der “Zuweisungsfunktion” braucht ein Taster noch eine weitere Funktion, die “Resetfunktion”. Mit dieser soll ermöglicht werden, die Leuchte aus entsprechender Gruppe (diejenige die der Taster hat) zu löschen. Bei langem Drücken des ”Reset-Knopfs” (z.B mehr als 3 Sekunden) wird die Leuchte aus allen Gruppen entfernt.

Diese Zuweisungsmethode ist in kleineren Netzwerken einfach zu handhaben und es wird keine zusätzliche Hardware benötigt. Für grössere Anlagen wie z.B Flughäfen, Bahnhöfe, Firmengelände, Freizeitparks, usw., ist diese Variante ungeeignet, da für eine neue Zuweisung physikalischer Zugriff auf jede einzelne Leuchte notwendig wäre.

## 5.2 Zuordnung mithilfe eines Servers

In grösseren Gebäuden oder Aussenanlagen ist die Beleuchtung heutzutage meist computergesteuert. Bei einer Umrüstung auf “LightNet” könnte sich diese Infrastruktur zunutze gemacht werden oder die nötige Funktionalität könnte leicht eingebettet werden. Schliesst man einen zentralen Server an ein “LightNet” System an, so können die Zuweisungen der Leuchten zu den Gruppen von diesem aus erfolgen.

### 5.2.1 Synchronisation

Der Server, welcher ebenfalls drahtlos mit dem Netzwerk kommuniziert, sollte idealerweise möglichst viel Information über das Netzwerk besitzen. Als erstes muss er alle am System angeschlossenen Leuchten kennen. Dazu werden die IDs der angeschlossenen Leuchten abgefragt. Für diesen Zweck würde sich ein ECHO-Algorithmus eignen [1]. Beim ECHO-Algorithmus ist es aber nötig, dass jeder Knoten seine Nachbarn kennt. Dies lässt sich jedoch einfach realisieren indem jeder Knoten in bestimmten (nicht zu kurzen) Abständen eine bestimmte Nachricht - eine sogenannte WHOIS-Nachricht - per Broadcast an alle seine Nachbarn sendet, worauf diese dem Absenderknoten mit ihrer Identifikationsnummer antworten.

Die von den Blätter zum Server rücklaufenden Pakete enthalten die Knoten-IDs und die Struktur des Spannbaums, so dass der Server nach Terminierung des ECHO-Algorithmus’ ein genaues Bild vom Netzwerklayout und dem gewählten Spannbaum hat. So weiss er, welchen Pfad zu einer bestimmten Leuchte genommen werden muss und kann die Pakete per Unicast (inklusive gespeichertem Pfad) an den Zielknoten senden. So kann jeder Knoten einer odereren mehreren Gruppen zugewiesen werden, ohne dass das gesamte Netzwerk geflutet werden muss. Ein möglicher Aufbau von einem Zuweisungs-Paket kann folgendermassen aussehen:

ServerID	Dest	ListLength	Group1	...	GroupN	PathLength	Path
----------	------	------------	--------	-----	--------	------------	------

- **ServerID:** Eindeutige Identifikationsnummer des Servers. Ist rein sicherheitstechnisch dabei, so dass die Leuchten nach dem ersten Kontakt mit dem Server nur noch Zuweisungen von diesem Server annehmen.
- **Dest:** Identifikationsnummer der Leuchte, deren Zuweisung geändert werden soll.
- **ListLength:** Anzahl der Gruppen, zu welchen die Leuchte gehören soll. Hat dieses Feld den Wert 0, bedeutet dies, dass die Leuchte aus allen Gruppen entfernt werden soll.
- **Group 1..N:** Auflistung der einzelnen Gruppen, zu denen die Leuchte zugewiesen werden soll.
- **PathLength:** Länge des Pfads vom Server zu der Leuchte.
- **Path:** Liste der Knoten auf dem Pfad zur Leuchte.

# Kapitel 6

## Fazit und Schlussbemerkungen

### 6.1 Ausstehende Probleme

Folgende Punkte müssten bei einer konkreten Implementierung noch genauer untersucht werden:

- Design eines geeigneten Protokolls für den Zugriff auf das gemeinsames Kommunikationsmedium (MAC-Protokoll)
- Sicherheit (Verschlüsselung, Authentisierung, ...)
- System mit mehreren Servern (bei der Server-basierten Zuweisung)
- Verhalten bei Ausfall von einem oder mehreren Knoten (Fehlertoleranz)

### 6.2 Persönlicher Kommentar

Die Möglichkeiten ein solches “LightNet” System zu realisieren sind zahlreich und ohne Testumgebung ist es schwierig herauszufinden, welche davon am besten geeignet ist. Es gibt auch hier keine eindeutig beste Lösung für alle Anwendungen von “LightNet”. Die zu wählende Implementation hängt vom Einsatzgebiet und den Anforderungen an das System ab. Die in dieser Arbeit herausgearbeitete Realisierung ist Schritt für Schritt über viele andere, zuerst betrachtete Lösungsansätze entstanden.

#### 6.2.1 Alternativer Taster mit zwei Funktionen

Es war auch ein Ansatz dabei, bei dem die Taster trotz dem 1-Knopf-Design zwei Funktionen hatten, welche durch verschieden langes Drücken ausgelöst werden konnten. Dabei wurde durch das kurze Drücken dieselbe Aktion ausgelöst wie in Algorithmus 1 beschrieben, und durch das lange Drücken wurden alle Leuchten der Gruppe ausgeschaltet, was sich als nützlich erweisen kann, wenn nicht alle Leuchte innerhalb der Gruppe denselben Zustand haben. Jedoch lässt sich eine Gruppe auch ohne diese zweite Funktionalität gesamthaft ausschalten, nämlich durch maximal zweimal kurzes Drücken. Ist die vom Taster ausgewählte Leuchte an, so werden alle Leuchten ausgeschaltet und es genügt ein einziger Tasterdruck, ist diese Leuchte jedoch eingeschaltet, so muss der Taster zweimal betätigt werden, da sich nach dem ersten Druck die gesamte Gruppe zuerst einschaltet.

### **6.3 Schlussfolgerungen**

Vom theoretischen Standpunkt aus gesehen würde sich “LightNet” mit den beschriebenen Algorithmen realisieren lassen. Energiesparsamkeit und Reichweite der Signale müssten allerdings in einer Beispielanwendung oder realistischen Simulation überprüft und angepasst werden. Weiter stellt sich die Frage, wie Leistungsstark die einzelnen Knoten und mit wieviel Speicher diese ausgestattet werden müssen, um alle ankommenden Pakete schnell genug verarbeiten zu können.

# Literaturverzeichnis

- [1] E. J.-H. Chang. Echo algorithms: Depth parallel operations on general graphs. *IEEE Trans. Softw. Eng. SE*, 1982.
- [2] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, 2002.
- [3] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Department of Computer Science and Information Engineering, National Central University, Chung-Li, 32054, Taiwan*.
- [4] Andrew S. Tanenbaum. Computer networks. 2003.
- [5] Wikipedia. The free encyclopedia. Available at <http://www.wikipedia.org>.