Eine formale Analyse des Verbindungsaufbaus bei Bluetooth

Fabian Franzelin

14. Juli 2009

Hauptseminar: Kryptographische Protokolle
Chair for Foundations of Software Reliability and Theoretical Computer Science, Technische Universität München

1 Einführung

Im 10. Jahrhundert lebte ein dänischer Wikingerkönig namens Harald Blauzahn. Er war bekannt für seine außergewöhnliche Kommunikationsfähigkeit. Er diente den Entwicklern von Bluetooth als Namensgeber für eine Kommunikationsschnittstelle, die in den letzten Jahren im Bereich der Welt des kabellosen Datenaustausches stark an Bedeutung gewonnen hat. Vor allem bei Batterie betriebenen Endgeräten wie Handys und PDAs ist Bluetooth der gängige Standard für einen unkomplizierten und schnellen Datenaustausch über kurze Distanzen. Es ist deshalb wichtig zu wissen, wie effizient Bluetooth funktioniert und wie groß der Energieverbrauch im schlechtesten anzunehmenden Fall ist. In diesem Paper wird gezeigt, wie formale automatische Verifikationstechniken zur Analyse der Performance des Bluetooth Protokolls angewendet werden können. Dafür schauen wir uns das Bluetooth Protokoll erstmal genauer an.

2 Verbindungsaufbau

Um zu kommunizieren organisieren sich Bluetooth Geräte in sogenannten *Piconets*, die aus einem *Master* und bis zu sieben *Slaves* bestehen. Der Aufbau eines Piconets unterteilt sich in zwei Schritten, den für uns interessanten *Inquiring Prozess* und den *Page Prozess*. Beim Inquiring Prozess versucht das Master Gerät sich in der Nähe befindende Bluetooth Geräte zu finden. Der Page Prozess findet im Anschluss statt und baut eine feste Verbindung zwischen den Kommunikationspartnern auf. Nachfolgend wird der Inquiring Prozess beschrieben, da er den energieaufwendigsten Teil des Bluetooth Protokolls darstellt und damit wesentlich für eine Performance-Analyse ist.

2.1 Inquiring Modus

Ein potentielles Master Gerät sendet solange auf 32 der 79 lizenzfreien Frequenzbändern des ISM-Band (Industrial, Scientific and Medical Band) zwischen 2,402 GHz und 2,480 GHz bis entweder eine Obergrenze an akzeptierten Antworten

eingegangen sind oder eine Zeitschwelle überschritten wurde. Jedes Bluetooth Gerät verfügt über eine 28-Bit breite Uhr, die alle 128μs tickt. In zwei aufeinander folgenden Zeitintervallen sendet das Gerät auf zwei verschiedenen Frequenzen und hört im Anschluss auf denselben Frequenzen ob ein Gerät antwortet. Danach wird mit zwei anderen Frequenzen weiter gemacht. Eine schematische Darstellung sehen Sie in Abbildung 1.

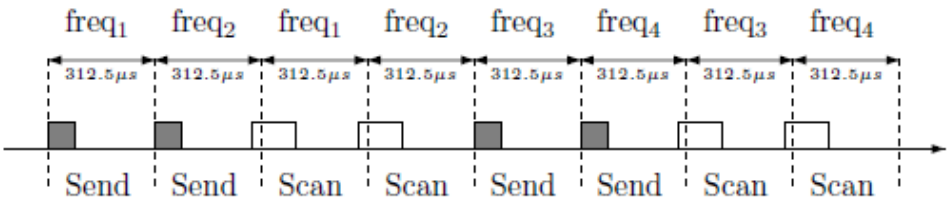

Abbildung 1: Inquiring Prozess

Die 32 Frequenzen werden in zwei verschiedenen Frequenzketten $A = [1, \ldots, 16]$ und $B = [17, \ldots 32]$ aufgeteilt, die jeweils 16 Frequenzen enthalten. Die aktuelle Frequenz lässt sich nach der folgenden Formel berechnen:

$$freq = [CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}\ mod\ 16]\ mod\ 32$$

wobei CLK_{i-j} den Bits $i, \ldots, j$ der Uhr entspricht und k ein Offset ist um entweder die Frequenzkette A oder B zu wählen. 16 aufeinander folgende Frequenzen bilden einen Frequenzzug, der 256 Mal wiederholt wird, ehe ein neuer Zug gebildet wird usw.

2.2 Scanning Modus

Bluetooth Geräte, welche gefunden werden wollen, scannen auf denselben 32 Frequenzen, auf denen das Master Gerät sendet. Um zu gewährleisten, dass sich die verwendeten Frequenzen von Master und Slave irgendwann überschneiden und damit ein erstes Hallo stattfinden kann, ist die Hopping Rate (Rate, wie schnell zwischen verschiedenen Frequenzen gewechselt wird) im Scanning Modus kleiner als im Inquiring Modus. Um zu vermeiden, dass zwei Geräte zur selben Zeit eine Antwort an den Master schicken wurde nach jeder Antwort eine zufällige Wartezeit eingebaut, ehe der Empfänger wieder damit beginnen kann auf einer Frequenz zu lauschen. Die aktuelle Lauschfrequenz wird auch als *Phase* bezeichnet. Der genaue Scanning Vorgang ist in Abbildung 2 veranschaulicht.

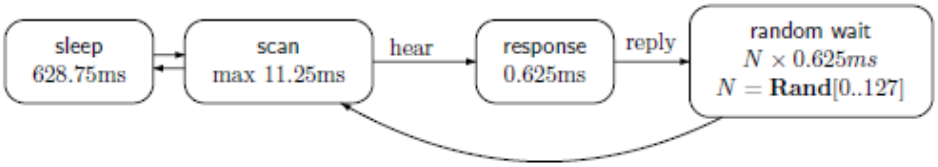

Abbildung 2: Scanning Prozess

Diese schematische Darstellung des Inquiring Prozesses reicht aber nicht aus um Abfragen der Art *"Wie groß ist die Wahrscheinlichkeit, dass sich zwei Ge-*

sprächspartner nach maximal zwei Sekunden mindestens einmal miteinander gesprochen haben?". Dafür braucht man zunächst eine mathematische Beschreibung des Protokolls.

3 Umsetzung

Es erstmal nicht klar ersichtlich, wie sich der Inquiring Prozess in ein mathematisches Modell gießen lässt. Was man aber erkennen kann ist, dass sich beim Scannen von Nachrichten ein probabilistisches Verhalten einschleicht, was durch Markov-Ketten beschreibbar wäre. Eine Markov-Kette ist informell gesprochen ein gerichteter endlicher Graph, dessen Kanten mit Wahrscheinlichkeiten gewichtet sind sodass die Summe der Gewichte der ausgehenden Kanten eines Knotens gleich eins sein muss. Damit ließe sich z.B. der Graph in Abbildung 2 fast als eine zeit-diskrete endliche Markov-Kette interpretieren, denn zudem können wir feststellen, dass die Menge der Zustände unseres Protokolls endlich sind und dass Transitionen nur zeit-diskret schalten können, da die Bluetooth Uhr den Takt des Protokolls vorgibt. Zudem ist die Markov Eigenschaft erfüllt, dass ein zukünftiger Zustand lediglich vom aktuellen Zustand abhängt und nicht von den vorangegangenen. Ein Problem besteht aber darin, dass in diesem Graph einige Kanten nicht mit Wahrscheinlichkeiten gewichtet sind, sondern mit Ereignissen wie *"Hören einer Nachricht"*. Dieses Problem würde den Rahmen dieser Arbeit sprengen, deshalb sei auf die Arbeit [Kwiatkowska et al.(2007)Kwiatkowska, Norman, and Parker] verwiesen welche eine genaue Abhandlung davon bietet. Eine einfache Veranschaulichung bietet auch die Definition der Formel *hear* des PRISM Beispiels im nächsten Abschnitt, das einen Ausschnitt des Empfänger Quellcodes zeigt. Wir können jetzt auf jeden Fall festhalten, dass sich der Inquiring Prozess als eine *dicrete-time Markov chain (DTMC)* beschreiben lässt. Nachdem wir jetzt eine mathematische Beschreibung unseres Modells haben, wollen wir festlegen, was genau wir nun wissen wollen:

1. Wie lange dauert ein Inquiring Prozess für unterschiedlichste Startkonfigurationen der beteiligten Geräte, ehe der Master mindestens eine oder zwei Antwort von einem Slave bekommen hat?

2. Ist das Ereignis *"Habe zweite Antwort erhalten"* unabhängig vom Ereignis *"Habe erste Antwort erhalten"* und damit bloß die Aneinaderkettung von zwei mal dem Modell für den Erhalt von bloß einer Nachricht? Oder hat die erste erfolgreiche Antwort einen positiven Einfluss auf die Empfangszeit für die zweite Antwort?

3. Hat sich die Wartezeit des Senders von Protokollversion 1.1 auf 1.2 signifikant verbessert?

Damit stehen einige nicht triviale Fragen im Raum, die nicht ohne Weiteres gelöst werden können. Um sinnvolle Ergebnisse zu bekommen müssen alle möglichen Pfade durch das System in Betracht gezogen werden. Allein alle möglichen Anfangsszenarien, welche sich bei der Abhängigkeit der Startfrequenz, auf der gesendet oder empfangen werden soll, von der aktuellen geräteeigenen Uhr ergeben sind schon zu viel für eine vollständige Analyse. Deshalb nimmt man an, dass der Empfänger erst dann beginnt zu hören, wenn der Sender bereits erste

Pakete abgeschickt hat. Dies ist ein durchaus nachvollziehbares und logisches Szenario und vereinfacht das Modell erheblich. Es bleiben zwar trotzdem noch 17,179,869,184 mögliche Startkonfigurationen für den Fall, dass der Sender bloß auf eine Antwortet wartet übrig, ist aber eher praktikabel. Vor allem deshalb, da das im nächsten Abschnitt vorgestellte Analysetool raffinierte Tricks bietet um die Anzahl an Zuständen des Modells zu minimieren.

4 PRISM

PRISM ist ein automatisches formales Verifikationstool für die Analyse von quantitativen Eigenschaften eines Systems mit probabilistischem Verhalten. Ein wesentliche Vorteil von PRISM ist, dass es erlaubt Markov-Ketten und insbesondere DTMCs auf eine einfache und übersichtliche Art zu definieren und zudem über BDDs in der Lage ist deren Struktur zu minimieren und z.B. dass nicht erreichbare Zustände automatisch gelöscht oder redundante Zustände zusammengeführt werden. Die high-end Modellierungssprache für die Modelldefinition, besitzt folgenden Aufbau: Ein Modell besteht immer aus einem *Typ*, verschiedenen *Modulen* und *Konstanten*. In PRISM werden drei Typen von probabilistischen Modellen unterstützt: *discrete-time Markov chains (DTMCs), Markov desicion processes (MDPs) und continuous-time Markov chains (CTMCs).*

Als Beispielmodul sei ein Teil des Empfänger Moduls angeführt:

```
// Wenn der Empfänger etwas hört
// freq ... Frequenz vom Sender
// freq1 ... Frequenz vom Empfänger
// train ... Frequenzoffset des Senders (0 = Zug A, 1 = Zug B)
// train1 ... Frequenzoffset des Empfängers (0 = Zug A, 1 = Zug B)
formula hear = (freq1 = freq & train1 = train & send = 1);

module receiver1

       y1 : [0..2 * maxr + 1];
       // Uhr des Empfängers, maxr ... maximum random delay = 127
       receiver : [0..3];
       // 0 - Nächster Scanning Zustand
       // 1 Lauschen
       // 2 Antworten und eine zufällige Wartezeit erzeugen
       // 3 Wartezeit abwarten ;)
       ...
       // Zeit verstreichen lassen
       [time] receiver = 0 & y1 = 1 -> (y1' = y1 - 1);
       ...
       // Nichts zu hören
       [time] receiver = 1 & !hear -> (y1' = y1);
       // Etwas zu hören
       [] receiver = 1 & hear ->
         (receiver' = 2) & (y1' = 2) & (freq1' = 0) & (train1' = 0);
       ...
       [reply]  receiver = 2 & y1 = 0 ->
           // antworten und zufällige Wartezeit initiieren
```

```
            1/(maxr + 1) : (receiver' = 3) & (y1' = 0)
          + 1/(maxr + 1) : (receiver' = 3) & (y1' = 2 * 1)
          ...
          + 1/(maxr + 1) : (receiver' = 3) & (y1' = 2 * 127);
        ...
endmodule
```

Wie im obigen Beispiel zu sehen ist, besteht ein Modul aus Variablen und sog. *Wachen*, welche das Verhalten des Modells beschreiben. Solche Wachen sind folgendermaßen definiert:

$$[a]\, g \rightarrow \lambda_1 : u_1 + \cdots + \lambda_n : u_n;$$

1. **a** $\rightarrow$ **Aktion:** Ein globales Ereignis, das die Überprüfung aller Prädikate, die an dieses Ereignis gebunden sind auslöst. Dies passiert Modulübergreifend.

2. **g** $\rightarrow$ **Prädikat:** Eine Folge von logischen Operationen, welche festlegen ob die angebundene Update Anweisung ausgeführt wird oder nicht. Prädikate können als Zustände des Moduls angesehen werden.

3. $\lambda_k \rightarrow$ **Wahrscheinlichkeit:** Eine Wahrscheinlichkeit, mit der das folgende Update ausgeführt wird. Die Bedingung $\sum_{i=0}^{n} \lambda_i = 1$ innerhalb einer Wache muss dabei erfüllt sein.

4. $u_k \rightarrow$ **Update:** Eine Folge von Anweisungen, welche sich auf das System auswirken.

Für die Analyse des Inquiring Prozesses interessieren wir uns primär um das zeitliche Verhalten des Protokolls und müssen deshalb jedes Mal wenn die Uhr tickt und einige Transitionen schalten Kosten vergeben können. Dies ist in PRISM über bestimmte Kostenkonstrukte möglich, die Gewichte an Transitionen und an Zuständen vergeben werden. Zur formalen Verifikation von probabilistischen und quantitativen Eigenschaften des Modells unterstützt PRISM zudem die *Probabilistic Computation Tree Logic (PCTL)*, welche es ermöglicht Abfragen der Art *"Was ist die worst-case Wahrscheinlichkeit über allen möglichen Anfangskonfigurationen des Modells, dass ein Fehler bis zum Zeitpunkt T auftritt?"* an das Modell zu stellen was folgendermaßen in PCTL übersetzt werden kann: *P = ? [F ≤ T error {init}{max}]*. Für einen genauerern Einblick sei auf [A. Hinton and Parker(2006)] verwiesen. Dies ist genau die Art von Abfragen, welche im vorigen Abschnitt vorgestellt wurden und welche nun effizient und einfach an unser Modell gestellt werden können.

5 Ergebnisse

In linken Teilbild der Abbildung 3 sehen Sie die Verteilung der Startkonfigurationen über ihre Terminierungszeit. In dieser Abbildung kann man gut erkennen, dass sich die verschiedenen Konfigurationen um insgesamt fünf Punkten bündeln. Dies ist deshalb der Fall, weil die großen zeitlichen Unterschiede zwischen den Zuständen durch das Schlafen legen des Empfängers entsteht. Die Prozedur dazwischen ist zeitlich gesehen zum Schlaf-zustand relativ kurz.

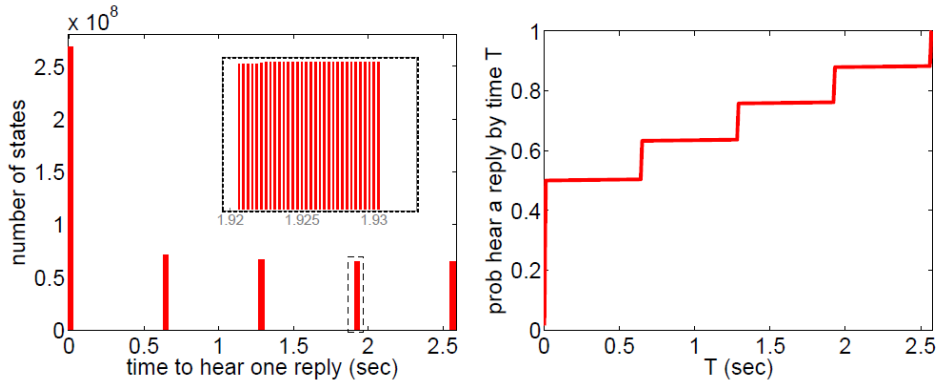

Abbildung 3: Ergebnisse der Simulation bei einer Antwort

Beim Warten auf zwei Antworten verhält sich die absolute Verteilung im linken Teilbild der Abbildung 4 ähnlich wie in Abbildung 3, mit dem Unterschied, dass nicht fünf Peaks erkennbar sind, sondern acht, wobei die letzten vier relativ zu den ersten fünf selten auftreten.

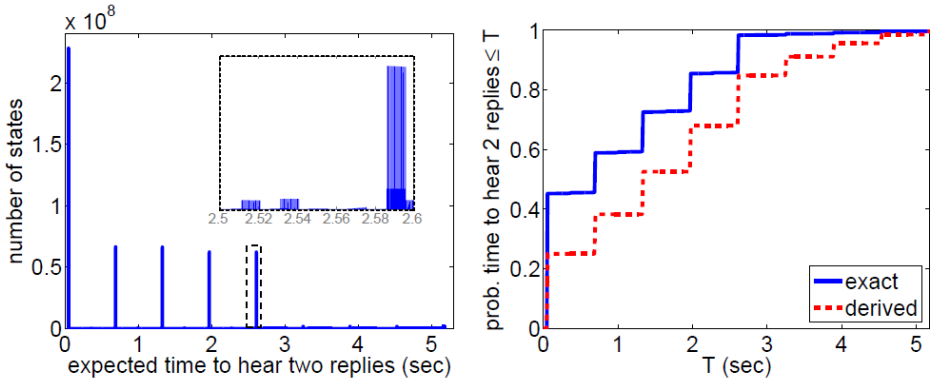

Abbildung 4: Ergebnisse der Simulation bei zwei Antwort

In Tabelle 1 sind die Verteilungsfunktionen von verschiedenen Protokollversionen (1.1 und 1.2) eingetragen, welche mit PRISM bestimmt wurden. In der Spalte K befindet sich die Anzahl, wie oft sich ein Empfänger schlafen gelegt hat, ehe er eine Antwort an den Sender geschickt hat. Die Variable n entspricht der Anzahl, auf wie viele Antworten der Sender gewartet hat, ehe der Prozess beendet werden konnte. Die Tabelle 1 beinhaltet die Wahrscheinlichkeitsverteilungen der verschiedenen Szenarios mit Angabe der verschiedenen Bluetooth Versionen unter der Voraussetzung, dass die Menge der Startkonfigurationen gleich-verteilt ist.

Die Spalteneinträge entsprechen akkumulierten Wahrscheinlichkeiten aus den rechten Teilbildern der Abbildungen 3 und 4. Die letzte Spalte entspricht den aus Spalte zwei berechneten Werten der Verteilungsfunktion, falls das Ereignis *"Schicke erste Antwort"* und das Ereignis *"Schicke zweite Antwort"* unabhän-

K	Version 1.1 $n = 1$	Version 1.2 $n = 1$	Version 1.2 $n = 2$	Version 1.2 (abgeleitet) $n = 2$
0	0.461240	0.500305	0.455379	0.250305
1	0.596265	0.633575	0.590829	0.383657
2	0.731585	0.759062	0.728684	0.526981
3	0.857913	0.879674	0.855329	0.681114
4	0.984295	1	0.984218	0.849408
5	0.988269	1	0.988294	0.911750
6	0.992398	1	0.992514	0.956496
7	0.996294	1	0.996519	0.985521
8	1	1	1	1

Tabelle 1: Ergebnisse

gig voneinander sind. In Spalte drei befinden sich die tatsächliche Verteilung der Startkonfigurationen, wenn der Sender auf zwei Antworten des Empfängers wartet, die mit PRISM berechnet wurden. Da Spalte drei und vier nicht übereinstimmen, kann man behaupten, dass die beiden Ereignisse entgegen der Annahme abhängig sind.

Literatur

[A. Hinton and Parker(2006)] G. Norman A. Hinton, M. Kwiatkowska and D. Parker. Prism: A tool for automatic verification of probabilistic systems. *In H. Hermanns and J. Palsberg (editor) Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, Lecture Notes in Computer Science(3920):441–444, 2006.

[Duflot et al.(2006)Duflot, Kwiatkowska, Norman, , and Parker] M. Duflot, M. Kwiatkowska, G. Norman, , and D. Parker. A formal analysis of Bluetooth device discovery. *Int. Journal on Software Tools for Technology Transfer*, 8(6):621–632, 2006.

[Homepage(2009)] PRISM Homepage. Bluetooth device discovery, 2009. URL `http://www.prismmodelchecker.org/casestudies/bluetooth.php`.

[Kwiatkowska et al.(2007)Kwiatkowska, Norman, and Parker] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270. Springer, 2007.

[Rajeev Alur(1999)] Thomas A. Henzinger Rajeev Alur. Reactive modules. *Formal Methods in System Design*, Formal Methods in System Design(15):7–48, 1999.