pimoto – Ein System zum verteilten passiven Monitoring von Sensornetzen

Rodrigo Nebel, Abdalkarim Awad, Reinhard German, Falko Dressler

Rechnernetze und Kommunikationssysteme, Universität Erlangen-Nürnberg

Kurzfassung. Im vorliegenden Beitrag stellen wir ein Monitoringkonzept für drahtlose Sensornetze und eine Implementierung speziell für eine Architektur bestehend aus Sensorknoten des Typs *BTnode* vor. Die hierarchisch aufgebaute Architektur ermöglicht das verteilte passive Monitoren des anfallenden Datenverkehrs innerhalb eines oder mehrerer drahtloser Sensornetzen. Die Analyse des anfallenden Datenverkehrs findet über ein eigens entwickeltes Plugin für das Netzwerkanalyse-Tool *Wireshark* statt. Um diese Funktionalität unseres Werkzeugs zu veranschaulichen, haben wir abschließend einen Versuch durchgeführt, dabei die Vorgehensweise beschrieben, die Ergebnisse erörtert und zugleich auch die Bedeutung des Tools für die Lehre aufgezeigt.

1 Einleitung

Die Popularität von vernetzten eingebetteten Systemen ist in den letzten Jahren stark gestiegen. Ein hervorstechendes Beispiel sind drahtlose Sensornetze. Im Allgemeinen versteht man unter Sensornetzen den losen selbstorganisierten Verbund kleiner eingebetteter Systeme, welche, mit Sensoren und Radioschnittstellen bestück, ein Ad hoc Netzwerk aufbauen, um Messdaten z.B. zu einer Senke (Analysestation) zu transportieren.

Die im Bereich drahtloser Sensornetze eingesetzten Kommunikationsmethoden sind sehr komplex. Daher ist das Verhalten des Gesamtnetzes oft schwer vorherzusagen. Gerade im Bereich der Entwicklung neuer Methoden werden daher Möglichkeiten der Analyse und des Debuggings von Kommunikationsmethoden benötigt. Da für das Debugging eingebetteter Systeme im Allgemeinen ein direkter Zugang zu den einzelnen Systemen benötigt wird, ist dies in größeren Netzen schwer oder gar nicht mehr realisierbar (gerade da auch mehrere Knoten synchron beobachtet werden müssen). Weiterhin besteht auch im Betrieb eines Sensornetzes oft der Anspruch, Kommunikationsverbindungen zu beobachten, um Fehlerzustände zu erkennen oder Systemparameter zu optimieren. In dieser Arbeit wird ein System zum passiven Monitoring von drahtlosen Sensornetzen vorgestellt. Dieses System, *pimoto*, erlaubt es, Radioübertragungen passiv zu belauschen und die empfangenen Daten zu speichern bzw. für eine weitere Verarbeitung an einen zentralen Server zu übermitteln. Um den Betrieb in verteilten Sensornetzen zu ermöglichen, wurde eine hierarchische Architektur entwickelt. Als Analysewerkzeug setzen wir das im Kommunikationsbereich verbreitete Werkzeug *Wireshark* [2] ein, welches eine graphische Analyse vereinfacht.

Um das natürliche Verhalten der zu überwachenden Sensornetze nicht zu beeinflussen, legten wir besonderen Wert auf ein *passives* Monitoring-Tool. Es soll weder durch weitere Softwarekomponenten auf den Knoten, noch durch das Einspeisen zusätzlicher Nachrichten in das Sensornetz das Netzverhalten unnatürlich beeinflusst werden. Im Gegensatz dazu steht das *aktive* Monitoring, bei dem bewusst in die Architektur eingegriffen wird um an die gewünschten Informationen zu gelangen.

Nach diesem Prinzip funktioniert *Nucleus* [3]. Es stellt eine Menge von TinyOS Komponenten zur Verfügung, die in eigene Anwendungen auf den Sensorknoten integriert werden mit dem Ziel einen Informationsaustausch zwischen diesen Komponenten und einer Auswertungsinstanz zu ermöglichen. Ein ähnliches Monitoring-System ist *Sympathy* [4]. Es werden ebenfalls zusätzliche Softwaremodule auf den Sensorknoten installiert und in periodischen Abständen Informationen an die sog. *Sympathie*-Senke geschickt. Diese übernimmt dann die Auswertung der Daten. Indessen werden bei *ScatterWeb* [5] erst nach Aufforderung durch den Anwender die relevanten Informationen an die Auswertungsinstanz *ScatterViewer* gesendet.

Passive Monitoring-Systeme dagegen sind *TWIST* [6] und *Wit* [7]. Erstgenanntes verbindet alle Monitoring-Knoten über ein USB-Kabel an ein Netzwerk und leitet die Informationen an eine Auswertungsstelle weiter. *Wit* [7] hingegen wurde mit nur genau einer speziellen Zielsetzung entwickelt: Es soll anhand verloren gegangener Datenpakete die Effizient der MAC-Schicht (802.11) in drahtlosen Sensornetzen überprüfen.

Unser Tool jedoch stellt die gesamte Kommunikation aus dem Sensornetz für eine Analyse zur Verfügung. Dabei wird – wie oben bereits genannt – auf ein natürliches Verhalten des zu überwachenden Sensornetzes Wert gelegt und aus diesem Grund die Kommunikation passiv mitgeschnitten. Die komplett kabellose Architektur sowie die Integration des professionellen Netzwerkanalyse-Tools *Wireshark* zur Auswertung des Datenverkehrs versprechen ein sehr gutes Handling. Da sich außerdem die komplette Funktionalität *Wiresharks* auf den erlauschten Datenverkehr anwenden lässt, sind die Möglichkeiten der Datenanalyse im Gegensatz zu allen oben kurz vorgestellten Systemen immens.

2 pimoto – Passive Island Monitoring Tool

Die folgenden Anforderungen ergeben sich für die Entwicklung und den Einsatz von *pimoto*. Erstens soll das Tool in einer Umgebung eingesetzt werden können, die evtl. über größere geographische Bereiche verteilt ist. Zweitens soll das Werkzeug die Kommunikation in Sensornetzen rein passiv belauschen, um keinen Einfluss auf die Kommunikation auszuüben. Drittens soll die Analyse zeitnah erfolgen, d.h. die Datenverarbeitung in den Monitorsystemen muss Echtzeitanforderungen genügen und viertens soll das Werkzeug direkt in der Lehre einsetzbar sein, muss also u.a. über eine einfache Bedienungsoberfläche verfügen.

Wir entwickelten eine hierarchische Architektur für den Einsatz von *pimoto*, d.h. mehrere Monitoringknoten können im Sensornetz verteilt angebracht werden. Die komplette Architektur ist in Abbildung 1 gezeigt. Spezielle *pimoto* Sensorknoten monitoren den Verkehr passiv und transportieren die erlauschten Datenpakete über

eine zweite Radioschnittstelle (Bluetooth) weiter an einen PC. Dieser PC verwaltet eine Reihe von Monitorknoten, sammelt die empfangenen Daten und schickt diese über eine weitere Drahtlosverbindung (WLAN) weiter an ein Serversystem. Letzteres analysiert die empfangenen Nachrichten mittels eines *Wireshark*-Plugins, um die komplette Sensorknotenkommunikation zu dekodieren und zu visualisieren.

Die wesentlichen Aspekte für den Betrieb des Systems sind optimierte Übertragungsprotokolle für die Weitergabe und Analyse von Monitordaten sowie die Möglichkeit der lokalen Zwischenspeicherung. Ersteres wird durch den konsequenten Einsatz der push-Strategie erreicht. Da im Sensornetz nur geringe Datenraten möglich sind und die Bandbreite über Bluetooth zu WLAN stetig wächst, sind hier keine Einschränkungen zu befürchten. Außerdem ist nur so eine Echtzeitanalyse der empfangenen Daten durch maximal reduzierte Latenzzeiten möglich. Da Bluetooth keine zuverlässigen Transportprotokolle zur Verfügung stellt, ist auch der zweite Aspekt, die Zwischenspeicherung und evtl. Neuübermittlung bei Übertragungsfehlern, für eine zuverlässige Datenauswertung relevant.

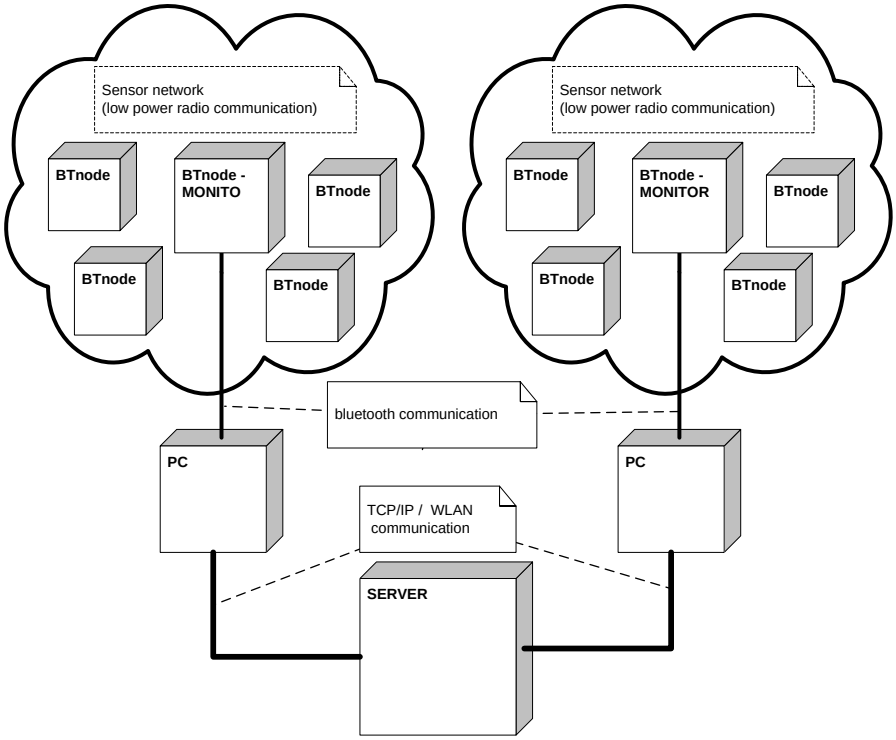

Abb. 1 Hierarchische Struktur: Architektur mit zwei „Monitor Inseln“

3 Implementierung

Implementiert wurde *pimoto* auf Sensorknoten vom Typ *BTnode* [1], welche neben der Radioschnittstelle zum Sensornetz über ein Bluetooth Interface verfügen. Das auf den Knoten verwendete Betriebssystem ist die *BTnut System Software* in der Version 1.6. Im Wesentlichen besteht sie aus Nut/OS, einem einfach gehaltenem Betriebsystem für kleine eingebettete Systeme mit dem ATmega128 Mikrocontroller von ATMEL, erweitert um die spezifischen Treiber für die Hardwaremodule des Sensorknotens *BTnode*.

3.1 Systemkomponenten

Ein wesentlicher Aspekt bei der Implementierung der Monitorsoftware war die Tatsache, dass die Monitorknoten im sog. promiscuous mode agieren, d.h. es werden alle Datenpakete mitgeschnitten, ungeachtet ihrer Zieladresse. Diese Fähigkeit wird momentan von der verwendeten B-MAC Umsetzung nicht unterstützt. Aus diesem Grund erweiterten wir die *BTnut*-Treiber entsprechend.

Für die Kommunikation über Bluetooth zwischen dem Monitorkonoten und einem PC wurde das *rfcomm*-Protokoll gewählt. Dieses emuliert eine serielle Kabelverbindung und garantiert – nach erfolgreichem Verbindungsaufbau – einen kompletten und unverfälschten Datenaustausch. Die vom PC erhaltenen Datenpakete werden im Anschluss daran an einen Server zur Auswertung über TCP/IP geleitet.

Alle weiteren Komponenten wurden unter Linux unter Verwendung von Standardprotokollen realisiert.

3.2 Synchronisation von verteilt aufgezeichneten Ereignissen

Während der Entwicklung von *pimoto* wurde als herausragende Schwierigkeit die genaue Synchronisation von verteilt aufgezeichneten Ereignissen (Datenpaketen) identifiziert. Für die Analyse von Kommunikationsprotokollen muss die exakte Reihenfolge von Datenpaketen rekonstruiert werden können. Da die Monitoringknoten nicht über synchronisierte Uhren verfügen und dies technisch auch nicht sinnvoll realisierbar ist, wurde ein Trick angewandt, der die relativen Laufzeiten der Knoten an den PCs korreliert.

Dazu wird jedem durch den Monitorknoten aufgezeichnetem Radiopaket ein weiteres 4 Byte großes Datenfeld hinzugefügt. In diesem neuen Feld werden die Millisekunden von Reboot des Monitorknotens an gespeichert. Da jeder *BTnode* über einen internen Zähler verfügt, der die Millisekunden seit Reboot zur Verfügung stellt, ist dieser zusätzliche Eintrag ohne großen Aufwand möglich. Anhand dieses Wertes kann später die genaue Empfangsuhrzeit des Pakets errechnet werden.

Dazu subtrahiert der Monitorknoten unmittelbar vor der Übertragung eines Pakets an den PC diesen Wert von dem seines aktuellen Zählers und überschreibt mit diesem Ergebnis den alten Wert. Der PC empfängt das Paket und subtrahiert von seiner aktuellen Uhrzeit den erhaltenen Wert. Das Resultat ist eine auf Millisekunden genaue Empfangsuhrzeit des Radiopakets durch den Monitorknoten. Sind mehrere sog. Monitorinseln im Einsatz (vgl. Abb. 1) ist eine synchrone Uhrzeit auf den PCs Vorrausset-

zung für eine einwandfreie Ermittlung der Paketempfangszeiten, andernfalls sind die Empfangszeiten der Radiopakete beider Monitorinseln verfälscht, da die Referenzzeiten auf den PCs unterschiedlich sind.

3.3 Analyse mit Wireshark

Für die Analyse mit *Wireshark* wurde das Plugin *BTnode Radio Protocol* für das im Sensornetz eingesetzte B-MAC Protokoll entwickelt. Die wesentliche Aufgabe des Plugins ist es, die Decodierung und Interpretation des erlauschten Datenverkehrs vorzunehmen.

Um an den Datenverkehr zu gelangen, schneidet *Wireshark* die Kommunikation zwischen PC und Server mit. Diese findet über TCP auf einem selbst festgelegtem Port statt. *Wireshark* schneidet nun die komplette Kommunikation mit und das entwickelte Plugin *BTnode Radio Protocol* analysiert die im Nutzdatenfeld von TCP übertragenen Radiopakete. Die Besonderheit liegt nun darin, dass sich in den Nutzdaten eines TCP Pakets genau ein von einem Monitorknoten mitgeschnittenes Radiopaket befindet. Unser entwickeltes Plugin extrahiert und interpretiert nun jedes einzelne Radiopaket aus den TCP-Nutzdaten und visualisiert die Details auf der graphischen Benutzeroberfläche von *Wireshark*. In Tabelle 1 sind die einzelnen Werte eines dekodierten Pakets zusammengefasst.

Tabelle 1. Felder eines Pakets nach der Dekodierung durch Wireshark

Feld	Bedeutung
Monitor MAC (6 Byte)	MAC Adresse des BTnode Monitorknotens
Source node (2 Byte)	B-MAC Quelladresse des Pakets
Destination node (2 Byte)	B-MAC Zieladresse des Pakets
Length of Data (2 Byte)	Länge der Daten
Type (1 Byte)	Typ der Anwendung
Seconds (4 Byte)	Sekunden zur Berechnung der Empfangszeit
Milliseconds (2 Byte)	Millisekunden bei der Empfangszeit
Time (0 Byte - errechnet)	Empfangszeit des Pakets durch den Monitorknoten, berechnet aus „Seconds“ und „Milliseconds“
Date (Length of Data)	Daten der Länge „Length of Data“

Auf den ersten Blick erscheint diese Variante der Paketübertragung umständlich und mit viel Overhead verbunden. Der große Vorteil ist bei der Analyse zu erkennen, denn es steht die gesamte Funktionalität von *Wireshark* zur Verfügung. So können bspw. spezielle Filter auf alle oder nur auf einen Teil der Datenpakete angewandt werden. Mit diesen werden z.B. nicht benötigte Protokollinformationen oder Datenpakete ausgeblendet, die Pakete nach Kriterien wie Typ, MAC-Adresse, Größe etc. sortiert, oder Pakete auf ihren Inhalt hin gesucht. Sind die gewünschten Informationen gefunden, können diese mit der von *Wireshark* angebotenen Funktionalität problemlos exportiert werden.

4 Einsatz in Forschung und Lehre

Mit einem einfachen Versuch möchten wir die Funktionsweise von *pimoto* verdeutlichen und die Einsatzmöglichkeiten in Forschung und Lehre aufzeigen.

Für diesen Versuch dient ein übersichtliches Sensornetz als Ausgangslage: Zwei *BTnode*-Sensorknoten schicken sich gegenseitig Radiopakete. Der innerhalb der Reichweite im Sensornetz liegende Monitorknoten schneidet diese mit und überträgt sie zur weiteren Auswertung an den PC, welcher sie dann an den Server leitet. Der Aufbau ist in Abbildung 2 zu sehen.

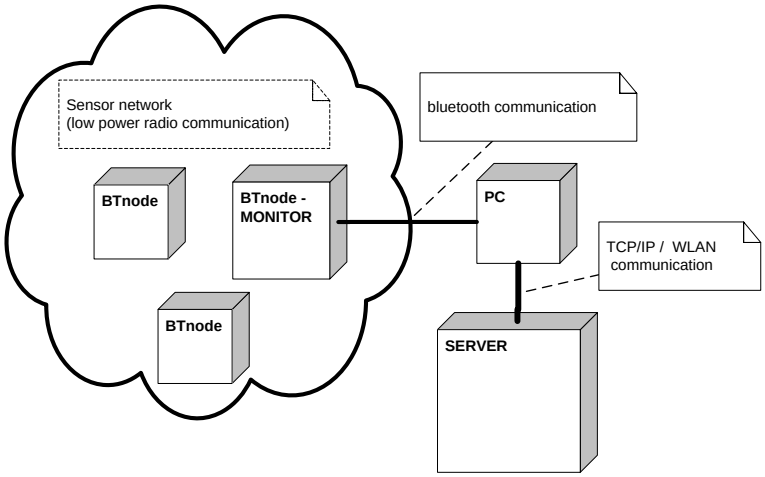

Abb. 2 Ein einfacher Versuch zur Veranschaulichung der Funktionsweise von *pimoto*

Das Ergebnis in Abbildung 3: Es werden sowohl die Pakete, die für die TCP Verbindung zwischen PC und Server nötig sind, als auch die Pakete aller Knoten im Sensornetz angezeigt.

Wireshark bietet nun die Möglichkeit, über ein Filterfeld nur den gewünschten Datenverkehr anzuzeigen. Damit lässt sich z.B. der TCP-Datenverkehr – der zum Teil nur für die Verbindung zwischen PC und Server nötig ist – aus Abbildung 3 ausblenden, so dass nur noch diejenigen TCP-Pakete angezeigt werden, die ein tatsächlich mitgeschnittenes Datenpaket aus dem Sensornetz in den Nutzdaten führen. Diese Pakete lassen sich nun weiterhin auf bestimmte Kriterien hin untersuchen. So werden z.B. mit dem Ausdruck: „btnode.typ == 3 && btnode.src == 1“ nur Pakete vom Sensorknoten mit der MAC-Adresse 1 des Typs 3 angezeigt (vgl. Abbildung 4). Jedes einzelne aufgeführte Datenpaket lässt sich nun auswählen und genau analysieren. Es werden die in Tabelle 1 genannten Werte angegeben.

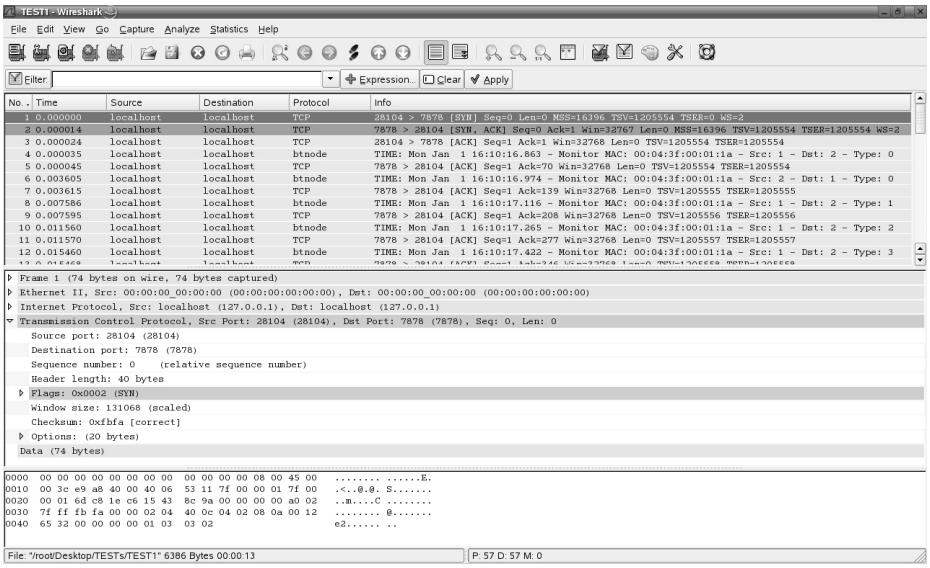

Abb. 3 Der komplette Datenverkehr auf der Benutzeroberfläche von *Wireshark*

Diese Möglichkeit, auf unkomplizierte Art und Weise den gewünschten Datenverkehr filtern und anschließend analysieren zu können, kann Anwendung beim Debuggen von Sensornetzen in der Forschung, aber auch in der Lehre finden. In folgenden Abschnitten werden wir beide Aspekte genauer betrachten.

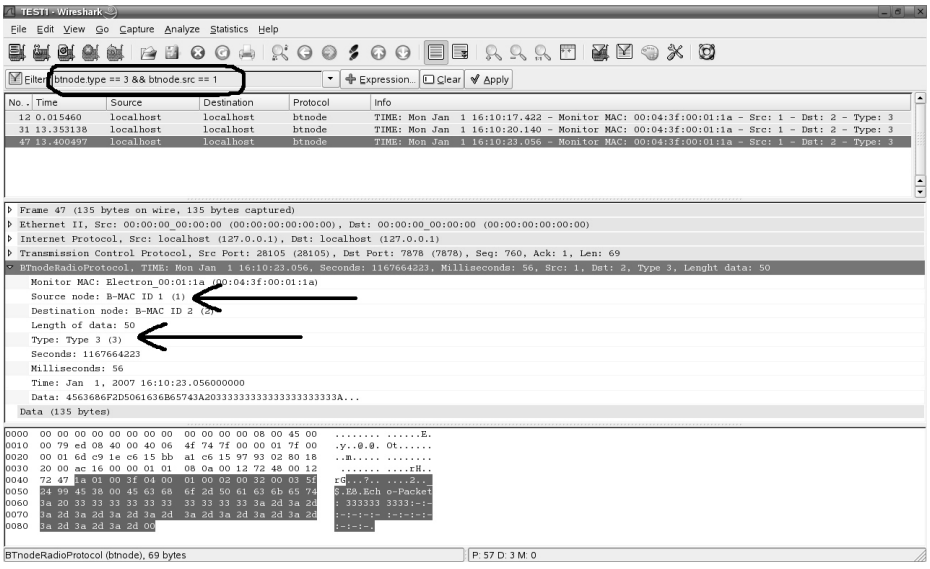

Abb. 4 Filtern nach gewünschten Datenpaketen und Anzeige der Details eines Pakets

4.1 Forschung

Das Auffinden von Fehlern in einer Software kann unter Umständen zu einem sehr langwierigen Prozess werden. Problematisch ist vor allem auch die Tatsache, dass es sich bei Sensorknoten um eingebettete Systeme handelt und deshalb auch alle damit verbundenen Schwierigkeiten beim Debuggen auftreten. Das Aufzeichnen aller Kommunikationsvorgänge sowie die Auswertung und Analyse der einzelnen Datenpakete kann eine große Hilfe beim Finden von Kommunikationsfehlern sein. Dazu werden nicht nur die Anzahl der tatsächlich ausgetauschten Datenpakete, sondern auch deren genauer Inhalt mit *Wirehark* betrachtet und bei Bedarf auch exportiert. Ungereimtheiten in der Kommunikation im Sensornetz werden erkannt und Rückschlüsse auf die implementierte Software oder angewandten Kommunikationsmethoden können gezogen werden. Eine spätere Auswertung und der Vergleich verschiedener aufgezeichneter Daten können bspw. Details über erbrachte Leistungsparameter liefern.

4.2 Lehre

Die Tatsache, dass Kommunikationsvorgänge im Sensornetz auch für Laien verständlich mit Hilfe von *Wireshark* visualisiert werden können, ermöglicht den Einsatz von *pimoto* auch in der Lehre.

Einsatzmöglichkeiten dafür finden sich in weiterführenden Schulen. Dort werden die Schüler im Fach Informatik, daneben aber auch schon in den unteren Jahrgangsstufen im Fach „Natur und Technik“ mit den grundlegenden Prinzipien und Konzepten der Informatik konfrontiert. Eines dieser grundlegenden Prinzipien ist die Kommunikation zwischen Informatiksystemen. Mit dem Einsatz dieses Tools soll es möglich sein, Teile dieser komplexen Sachverhalte den Schülern, z.B. im Informatikleistungskurs, auf anschauliche Weise nahe zu bringen.

Beispielsweise lässt sich die Funktionsweise grundlegender Kommunikationsprotokolle mit Hilfe eines sehr einfach gehaltenen Sensornetzes (z.B. aus nur drei Knoten und sehr wenigen Kommunikationsvorgängen) den Schülern auf verständliche Art erklären. Die zwischen den Sensorknoten ausgetauschten Datenpakete können durch die graphische Ausgabe von *Wireshark* auf dem Bildschirm sichtbar gemacht und folglich auch von den Schülern eingesehen und verstanden werden. Die Schüler können so die genaue Kommunikation zwischen den Knoten mitverfolgen und auch die Paketinhalte einsehen. Diese liegen in der gebräuchlichen hexadezimalen Schreibweise, aber auch in einer durch *Wireshark* interpretierten Form vor. Damit sind ein einfaches Ablesen aller Werte und eine daraus resultierende Interpretation der Vorgänge im Sensornetz möglich.

Eine weitere Einsatzmöglichkeit findet sich in den Hochschulen, denn gerade dort erfreuen sich drahtlose Sensornetze immer größerer Beliebtheit. An zahlreichen Universitäten entstehen immer mehr neue Forschungs- und Arbeitsgruppen, die sich mit diesem spannenden Thema auseinandersetzen. Übungsgruppen werden gebildet, die in regelmäßigen Treffen kreativ über neue Erkenntnisse diskutieren, die sich u.a. aus den zahlreichen Abschlussarbeiten der Studenten in diesem Fachgebiet ergeben. Das in diesem Beitrag vorgestellte Tool soll zur Unterstützung des Lehr- und Übungsbetriebs auf dem Gebiet der Sensornetze dienen. Mit diesem Tool und der Möglichkeit der einfachen Darstellung ausgetauschter Datenpakete in einem Sensornetz können nicht

nur komplizierte Kommunikationsvorgänge auf verständliche Weise dargestellt, sondern auch „neue“ Studenten an dieses Thema herangeführt und dafür begeistert werden. Der Informatik-Lehrstuhl 7 der Universität Erlangen-Nürnberg bietet speziell für diese Studenten sowie für alle Interessierten auf dem Gebiet der Sensornetze eine Arbeitsgruppe an: Sensor/Actuator Networks (SANET). In wöchentlich stattfindenden Treffen werden Themen wie z.B. ressourcenschonendes Routing in drahtlosen Sensornetzen oder experimentelle Untersuchungen mit *BTnode* Sensorknoten diskutiert und anhand von konkreten Implementierungen Lösungsmöglichkeiten für bestehende Probleme erarbeitet. Eines dieser in Sensornetzen auftretenden Probleme ist das Routing. Um nun beispielsweise die Effizienz verschiedener Routingalgorithmen den Studenten leicht verständlich zu erklären, kann das in dieser Arbeit entwickelte Werkzeug verwendet werden. Dazu werden zwei oder mehr Sensornetze, je nach Anzahl der zu vergleichenden Routingalgorithmen, in Betrieb genommen. Es gilt eine bestimmte Aufgabe innerhalb eines Netzes zu „lösen“. Die dazu nötige Kommunikation wird mit *pimoto* passiv mitgeschnitten. Eine anschließende Analyse aller ausgetauschten Datenpakete in den Sensornetzen bringt auf einfache Art und Weise Aufschluss über die Vor- und Nachteile der eingesetzten Algorithmen. Aufschluss über das Verhalten ihrer Implementierungen auf den Sensorknoten können auch die Studenten der im Sommersemester 2007 angebotenen Lehrveranstaltung Selbstorganisation in Autonomen Sensor-/Aktornetzen [SelfOrg] erhalten. In dem zur Vorlesung angebotenen Übungsbetrieb werden den Studentengruppen verschiedene Themenbereiche zur Bearbeitung zugeteilt. Die Bearbeitung beinhaltet neben einer Einarbeitung in das Thema und der Auseinandersetzung mit einem bestimmten Problem auch die Implementierung einer möglichen Lösung. Wie schon mehrfach erwähnt, ist die Kommunikation im Sensornetz eines der zentralen Themen. Aus diesem Grund werden sehr oft Lösungen für Probleme gesucht, die direkt oder indirekt von der Kommunikation im Sensornetz abhängen. Als Beispiel sei hier eine Aufgabe aus dem Sommersemester 2006 genannt: Implementierung und Analyse von Flooding und Gossiping Strategien. Für diese und alle weiteren im Übungsbetrieb gestellten Aufgaben ist dieses Tool eine zusätzliche Hilfe für das Testen und Evaluieren der von den Studenten entwickelten Anwendungen durch die Studenten selbst oder durch den Dozenten.

5 Zusammenfassung

Im vorliegenden Beitrag haben wir ein Monitorkonzept für drahtlose Sensornetze und eine Implementierung speziell für eine Architektur, bestehend aus Sensorknoten des Typs *BTnode*, vorgestellt.

Unser entwickeltes Monitorsystem besteht aus sog. Monitor-Inseln, Gateways und einem Server. Ein spezieller Monitorknoten schneidet die innerhalb einer solchen Monitor-Insel anfallende Kommunikation passiv mit und leitet diese über ein Gateway an den Server. Dieser ermöglicht mit Hilfe des Netzwerk-Tools *Wireshark* die Visualisierung und Analyse des Datenverkehrs. Dazu übernimmt ein eigens entwickeltes Plugin die Interpretation der Datenpakete aus dem überwachten Sensornetz. Weiterhin geben zusätzliche, von den Monitorknoten vorgenommene Einträge in den erlauschten

Radiopaketen, Aufschluss über Empfangszeit durch die Monitorknoten oder über die Monitor-Inseln.

Um die Funktionalität unseres Werkzeugs zu veranschaulichen, haben wir abschließend einen Versuch durchgeführt, dabei die Vorgehensweise beschrieben, die Ergebnisse erörtert und zugleich auch die Bedeutung des Tools für Forschung und Lehre aufgezeigt. Denn die Visualisierung und Filterung des Datenverkehrs ermöglicht einerseits Ursachen für Fehlverhalten des Sensornetzes ausfindig zu machen, andererseits das einfache Nachvollziehen der Kommunikationsvorgänge im Sensornetz. Es lassen sich beispielsweise grundlegende Prinzipien der Kommunikation zwischen Informatiksystemen im Informatikunterreicht veranschaulichen, oder aber auch komplexe Routingalgorithmen in der Hochschule auf ihre Effizienz hin analysieren.

Literatur

1. ETH Zürich, BTnodes – A Distributed Environment for Prototyping Ad Hoc Networks, http://www.btnode.ethz.ch
2. Wireshark: The World's Most Popular Network Protocol Analyzer, http://www.wireshark.org
3. Gillman Tolle, David Culler "Design of an application-cooperative management system for wireless sensor networks" In Proceedings of the Second European Workshop on Wireless Sensor Networks 2005, pages 121–132, Istanbul, Turkeye, Jan 2005.
4. Nithya Ramanathan, Eddie Kohler, Lewis Girod, Deborah Estrin, "Sympathy: A Debugging System for Sensor Networks" Workshop Record of the 1st IEEE Workshop on Embedded Networked Sensors (EmNetS-I), Tampa, Florida, November 2004, pages 554-555.
5. ScatterWeb, http://cst.mi.fu-berlin.de/projects/ScatterWeb/
6. V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "TWIST: A Scalable and Reconfigurable Wireless Sensor Network Testbed for Indoor Deployments", Technical University Berlin, TKN Technical Report TKN-05-008, November 2005.
7. Ratul Mahajan, Maya Rodrig, David Wetherall, John Zahorjan, "Analyzing the MAC level Behavior of Wireless Networks in the Wild", ACM SIGCOMM, Pisa Italy, 2006.