Informatik aktuell

Herausgeber: W. Brauer
im Auftrag der Gesellschaft für Informatik (GI)

Peter Holleczek
Birgit Vogel-Heuser (Hrsg.)

Mobilität und Echtzeit

Fachtagung der GI-Fachgruppe
Echtzeitsysteme (real-time)
Boppard, 6./7. Dezember 2007

Springer

Herausgeber

Peter Holleczek
Regionales Rechenzentrum
der Universität Erlangen-Nürnberg
Martensstraße 1, 91058 Erlangen
holleczek@rrze.uni-erlangen.de

Birgit Vogel-Heuser
Universität Kassel
Fachgebiet Eingebettete Systeme
Fachbereich Elektrotechnik/Informatik
Wilhelmshöher Allee 73, 34121 Kassel
vogel-heuser@uni-kassel.de

Bibliographische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

CR Subject Classification (2001): C3, D.4.7

ISSN 1431-472-X
ISBN 978-3-540-74836-6 Springer Berlin Heidelberg New York

Springer Berlin Heidelberg New York
Springer ist ein Unternehmen von Springer Science+Business Media

springer.de

Printed in Germany

Satz: Reproduktionsfertige Vorlage vom Autor/Herausgeber
Gedruckt auf säurefreiem Papier SPIN: 12119943 33/3180-543210

Mobilität und Echtzeit – ein Vorwort

Der Begriff der Mobilität liegt mitten im hart umkämpften Spannungsfeld zwischen purer Notwendigkeit und Ressourcenverbrauch. Man kommt nicht darum herum: Deutschland als europäisches Transitland, als automobile Nation mit Fahrzeugherstellern im Spitzensegment und mit einem sich eben erst entspannenden Arbeitsmarkt, ist davon besonders betroffen.

Einerseits muss man mobil sein, um einen Arbeitsplatz zu gewinnen oder zu sichern, andererseits bringt der übermäßige Ressourcenverbrauch uns alle einer Klimakatastrophe näher und unser Land in außenpolitische Zwickmühlen. Insofern ist klar, dass man heute bei "Mobilität" in erster Linie ans Auto und an andere Verkehrsmittel denkt. Dazu kommt noch die Sorge, wie es mit der persönlichen Mobilität im Alter, bei Gebrechen z.B., aussieht: werden wir selbst einmal robotergestützte Mobilitätshilfen brauchen?

Muss man sich bei so vielen philosophischen Gedanken fragen, ob der Fachgruppe nicht ihr technisches Selbstverständnis abhanden gekommen ist?

Nein, ganz im Gegenteil: Bewegung bzw. deren Kontrolle ist eine zuinnerst echtzeittypische Problemstellung. Wer denkt schon daran, dass er "beim Gasgeben" auf ein Potentiometer tritt? Umweltschonende Antriebe lassen sich nur noch mit ausgeklügeltem Einsatz von Steuerelektronik verwirklichen. Harte Anforderungen an Zeit und Sicherheit sind gefragt, denn es geht um Hilfestellung für Menschen. Die Aufgabe der Fachgruppe ist hier zu beobachten, ob und wie alles technisch Mögliche zur Aufrechterhaltung und Verbesserung der Mobilität getan wird.

Insofern wundert es nicht, dass auf breiter Front Beiträge eingegangen sind, die sich leicht zu einem Programm zusammenfügen. Insbesondere freut uns, dass sich mit Audi ein renommierter Kfz-Hersteller in besonderer Weise engagiert hat.

Die eingegangenen Beiträge beziehen sich auf diverse Aspekte der Mobilität, vordringlich natürlich im Kraftfahrzeugbereich, z.B.

- Kommunikation im Auto (Feldbus, Flexray)
- Steuergeräte
- Flottenkommunikation
- Fahrassistenz
- Satellitenkompass in der Binnenschifffahrt
- Wetterdaten für den Flugbetrieb
- zweibeinige Roboter.

Der Technikeinsatz muss natürlich durch Grundsatzuntersuchungen vorbereitet und abgesichert werden. Daher freuen wir uns, Ihnen Grundsatzbeiträge über Echtzeitsysteme, wie z.B.

- Verhaltensbeschreibung
- kontrollflussunabhängige Beschreibung
- Sensornetze
- Simulationen
- Modelltransformation zwischen Simulation und Steuerung

präsentieren zu können.

So ist das Programm wieder eine runde Sache geworden. Wir wünschen den Tagungsteilnehmern informative Stunden und einen regen Gedankenaustausch. Mit dem Springer-Verlag verbindet uns eine lange Tradition der Zusammenarbeit, von der wir auch in diesem Jahr profitieren. Zum Gelingen beigetragen haben die Firmen ARTiSAN, ESD und Werum, bei denen wir uns herzlich bedanken.

Für die Fachgruppen-Leitung und das Redaktionskollegium

Peter Holleczek Birgit Vogel-Heuser

September 2007

Inhaltsverzeichnis

Systeme und Netze

Entwicklung (1)

Entwicklung (2)

Praxis und Ausbildung

Fahrzeuge und Verkehr

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.

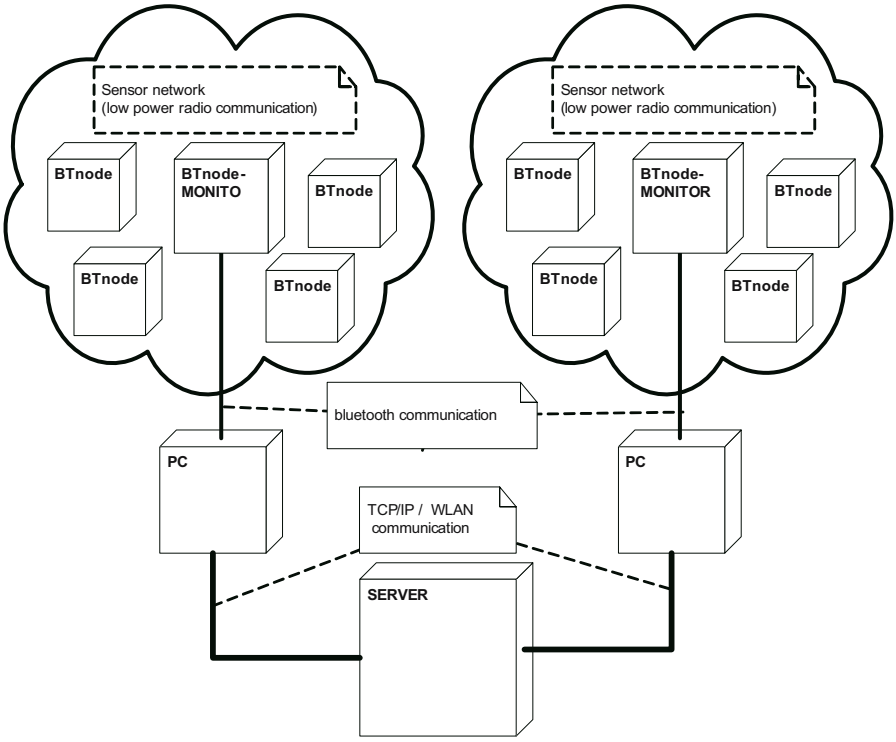

Abb. 1 Hierarchische Struktur: Architektur mit zwei „Monitor Inseln“

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 Voraussetzung 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.

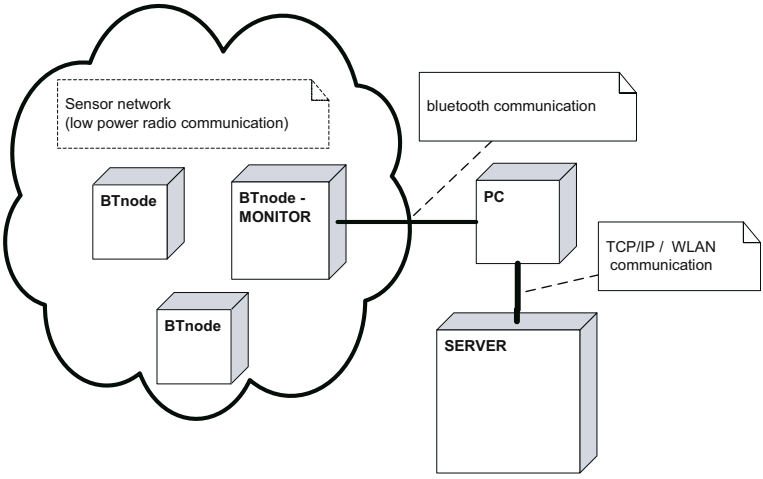

Abb. 2 Ein einfacher Versuch zur Veranschaulichung der Funktionsweise von *pimoto*

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.

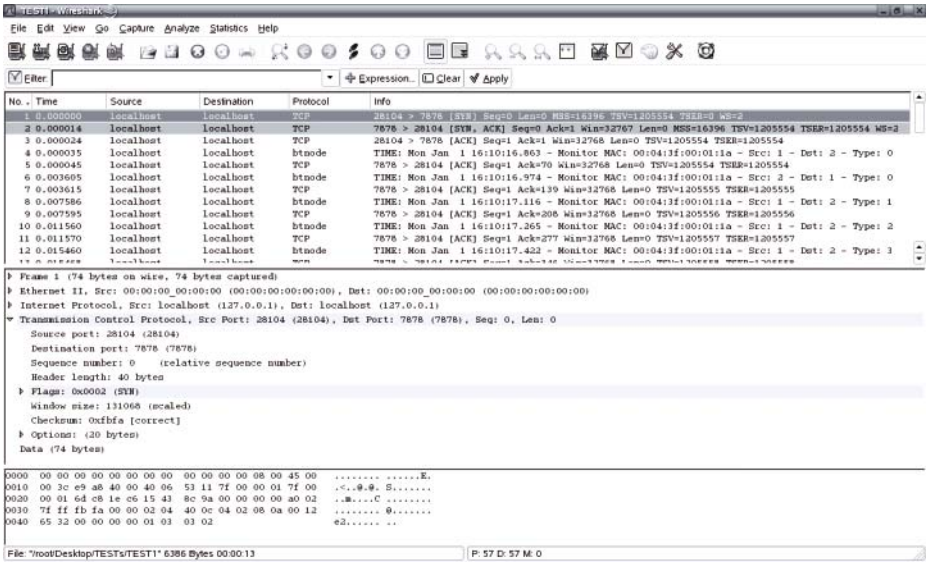

Abb. 3 Der komplette Datenverkehr auf der Benutzeroberfläche von *Wireshark*

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.

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.

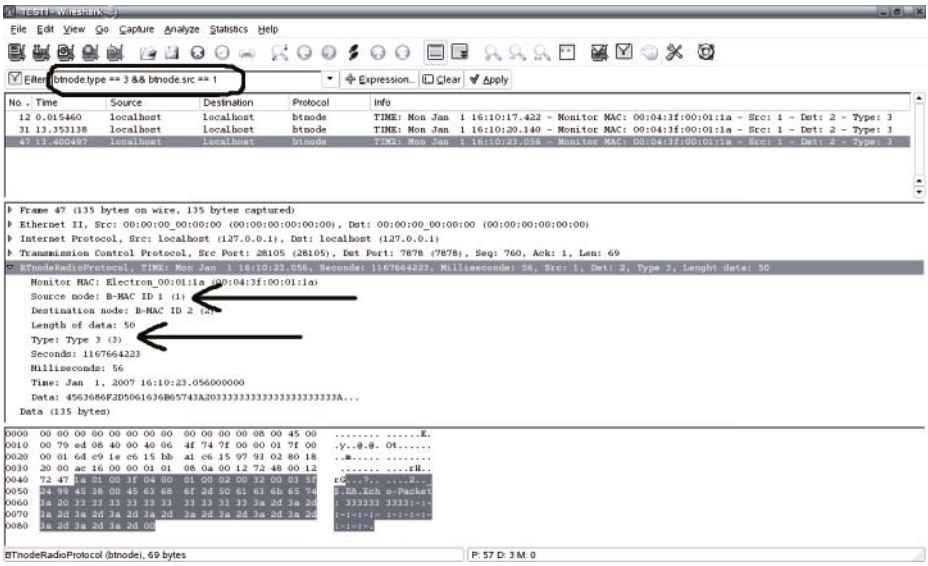

Abb. 4 Filtern nach gewünschten Datenpaketen und Anzeige der Details eines Pakets

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