„pimoto - Ein System zum verteilten passiven Monitoring von Sensornetzen “

Rodrigo Nebel

Institut für Informatik
Lehrstuhl für Rechnernetze und Kommunikationssysteme (Informatik 7)
Friedrich-Alexander-Universität Erlangen-Nürnberg

Überblick

- **Hintergrund und Motivation dieser Arbeit**
  - Was sind drahtlose Sensornetze?
  - Motivation für diese Arbeit
  - Zielsetzung
- **Überlegungen zur Architektur**
  - Entwicklung
  - Anforderungen an die einzelnen Komponenten
- **Implementierung**
- **Zusammenfassung**
- **Einsatz in Forschung und Lehre**

Drahtlose Sensornetze

- **Netzwerke, bestehend aus vielen autonomen Sensorknoten**
  - Kommunikation drahtlos (Funk)
  - Aufnahme von Informationen aus der Umwelt über Sensoren
  - Vorverarbeitung/Weiterleitung der Daten zur Auswertung
- **Besonderheiten**
  - Begrenzte Energie- und Speicherkapazität
  - Geringe Funkreichweite
  - Die „Stärke“ liegt im Verbund
- **Einsatzgebiete: Überwachung der Umwelt**
  - Militär
  - Katastrophenschutz

Drahtlose Sensornetze

- **Forschungsgebiete**
  - Routingalgorithmen für solche Ad-Hoc Netze
  - Effiziente Energie- und Speicherverwaltung
  - Fehlertoleranz, Selbstorganisation
- **Notwendigkeit der Entwicklung und des Testens neuartiger Lösungsansätze**
- **Motivation für diese Arbeit**
  - *Unterstützung des Debuggingprozesses* in Sensornetzen: Sensorknoten sind eingebettete und verteilte Systeme zugleich
  - Interessierten Personen den *Einstieg in dieses Gebiet* durch Visualisierung und Interpretation der Kommunikation innerhalb von Sensornetzen zu erleichtern

Zielsetzung meiner Arbeit

- ***Entwicklung*** **eines Konzepts zum passiven Monitoring von drahtlosen Sensornetzen**

- ***Implementierung*** **für eine Architektur, bestehend aus Sensorknoten des Typs BTnode**

Anforderungen

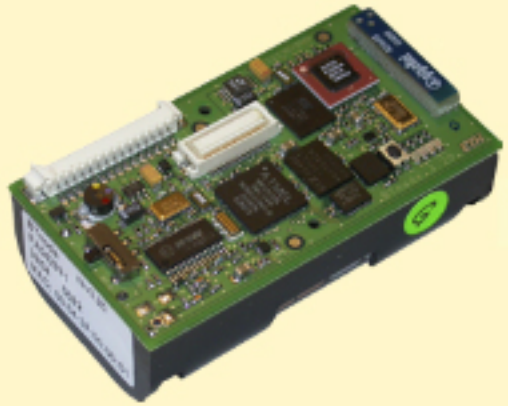
- **Passives Mitschneiden der im Sensornetz anfallenden Kommunikation**
- **Aufbereitung der aufgezeichneten Datenpakete**
- **Weiterleitung des Datenverkehrs zur Auswertung an eine externe Instanz**
- **Dekodierung, Interpretation und Visualisierung der Datenpakete über ein graphisches Frontend**

Der Sensorknoten BTnode rev3

- **Entwicklung an der ETH Zürich**
- **Betriebssystem: BTnut Systemsoftware**
- **Radio und Bluetooth Schnittstelle**

BTnode rev3 features at a glance

- Microcontroller: Atmel ATmega 128L (8 MHz @ 8 MIPS)
- Memories: 64+180 Kbyte RAM, 128 Kbyte FLASH ROM, 4 Kbyte EEPROM
- Bluetooth subsystem: Zeevo ZV4002, supporting AFH/SFH
- Scatternets with max. 4 Piconets/7 Slaves, BT v1.2 compatible
- Low-power radio: Chipcon CC1000 operating in ISM band 433-915 MHz
- External Interfaces: ISP, UART, SPI, I2C, GPIO, ADC,Timer, 4 LEDs
- Standard C Programming, TinyOS compatible



Integration der Monitor-Software

1. Vorhandener Knoten

- Keine zusätzliche HW nötig

- Störung des „natürlichen“ Verhaltens durch zusätzliche Software auf dem Knoten

- Speicherverbrauch

2. Zusätzlicher Knoten

- Mehr Speicher zur Zwischenspeicherung und Aufbereitung der Datenpakete

- Unabhängigkeit des Monitorsystems

- zusätzliche Hardware nötig: Monitorknoten im Sensornetz

Wahl der Architektur

Erster Ansatz:

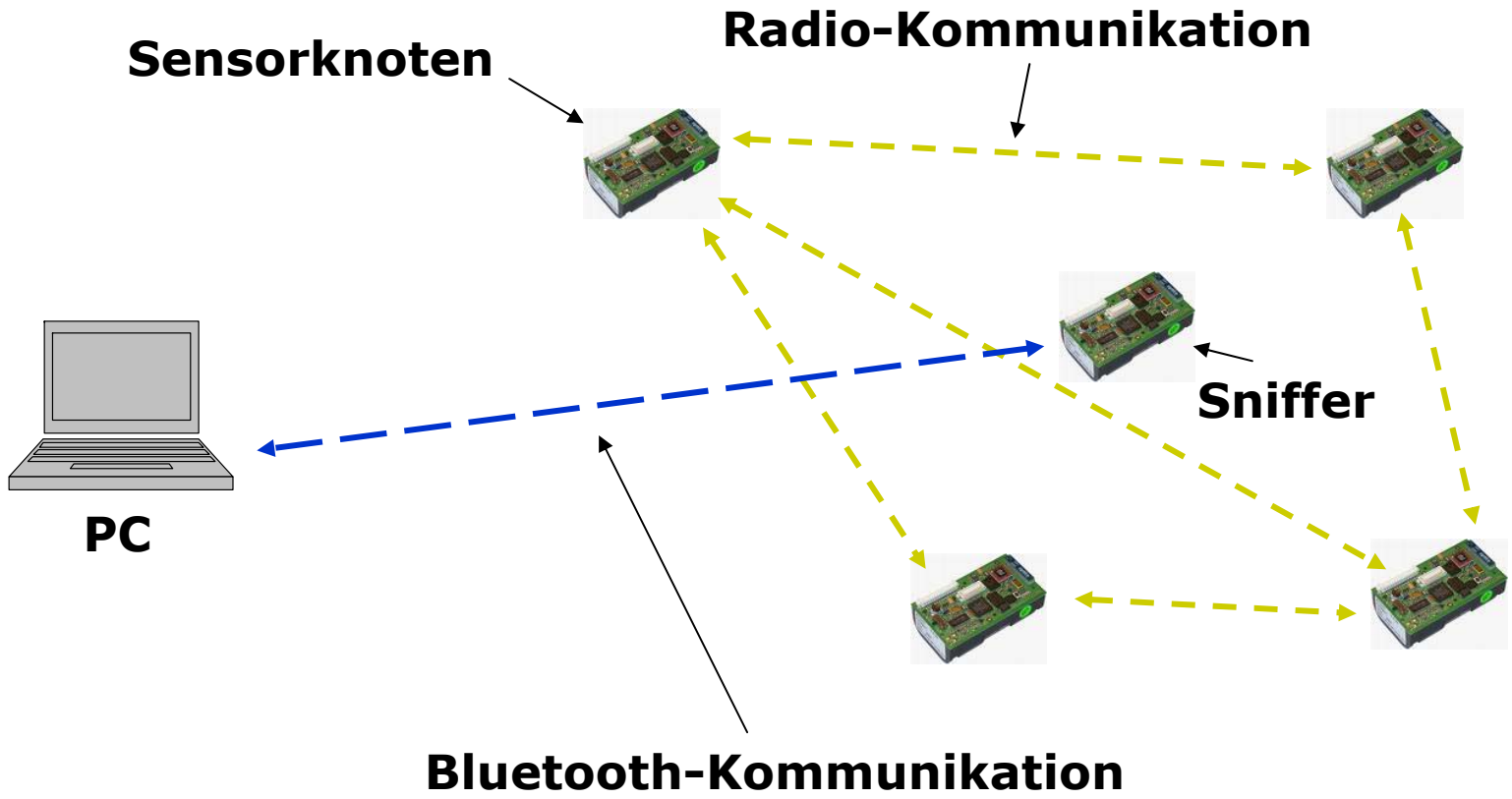
Architektur, bestehend aus einem Snifferknoten und einer externen Auswertungsinstanz (PC).

- **Einfachheit, geringer Installationsaufwand**

ABER:

- **Flaschenhals: Bluetooth Reichweite zw. Sniffer und PC**
- **Erweiterung aufwendig, keine Modularität**

Wahl der Architektur: Erster Ansatz



Wahl der Architektur

Zweiter Ansatz:

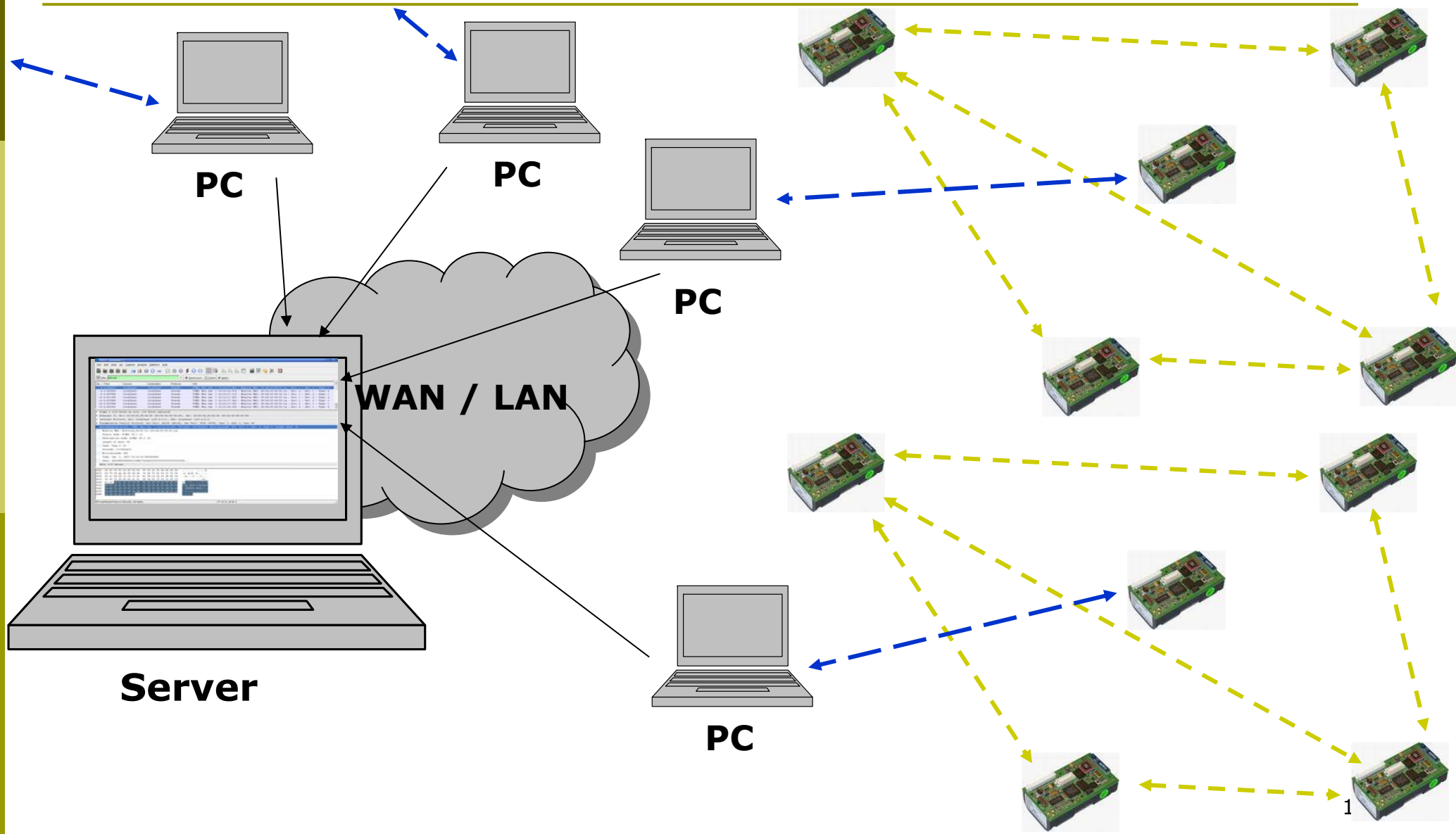
Erweiterung der vorgestellten Architektur um einen Server

- **Unabhängigkeit, Modularität**
- **Erweiterbarkeit**
  (Integration mehrerer sog. „Monitor-Inseln“)

ABER:

- **Zusätzliche Hard- und Software nötig**

Wahl der Architektur: Zweiter Ansatz



Möglichkeiten der Datenauslieferung

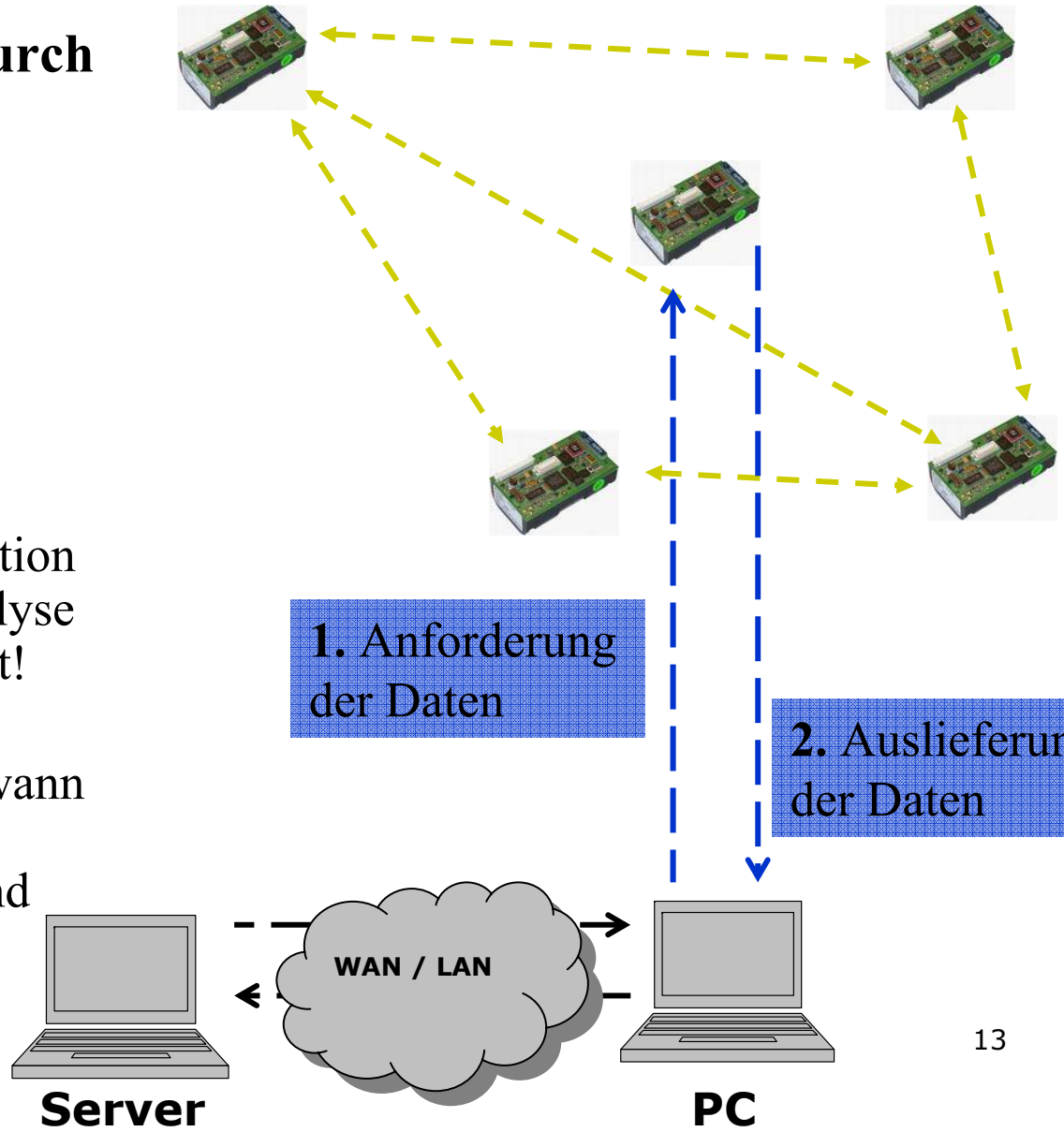
1. *Pull* – explizite Nachfrage durch den Verbraucher (PC, bzw. Server)

- Eignung bei niedriger bzw. unregelmäßiger Anfragerate

Aber

- Kontinuierliche Kommunikation im Sensornetz, zeitnahe Analyse des Datenverkehrs erwünscht!

- PC bzw. Server weis nicht, wann neue Informationen (Datenpackete) verfügbar sind



Möglichkeiten der Datenauslieferung

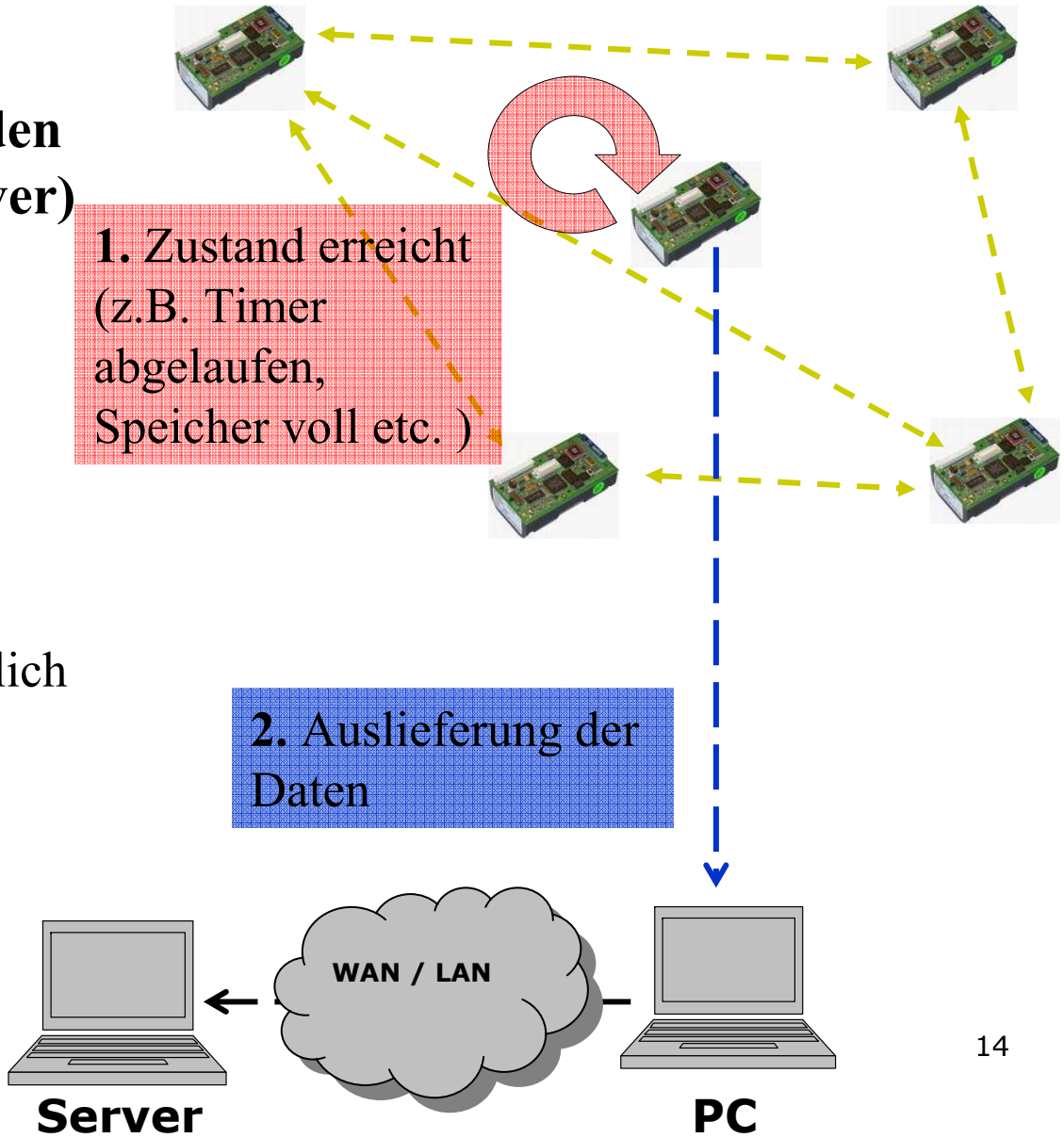
2. *Push* – automatische Auslieferung der Daten an den Verbraucher (PC, bzw. Server)

- Unmittelbares Verschicken neuer Daten nach Erhalt/Erzeugung

- Analyse des anfallenden Datenverkehrs zeitnah möglich

Aber:

- Bedingungen für die Datenauslieferung nötig



Analyse der erlauschten Datenpakete

1. Neue Anwendung

- Exakt auf die Anforderungen zugeschnitten

- Großer Programmieraufwand für die Grundfunktionalität

- Zusätzliche Implementierung jeder einzelnen Analysefunktion

2. Plugin

- Grundfunktionalität durch Anwendung schon gegeben

- Nutzung aller Analyse-funktionen der bereits implementierten Anwendung

- Aufwendige Installation

Anforderungen an die Komponenten

1. BTnode Sniffer

- Aufzeichnen der Pakete
- Erstellung von Zeitstempeln
- Weiterleitung an den PC

2. PC

- Empfangen der Pakete
- Berechnung der Empfangszeit
- Weiterleitung an Server

3. Server

- Empfangen von Paketen

4. Wireshark Plugin

- Interpretation der Kommunikation zw. PC und Server

BTnode Monitor-Software

- **Mitschneiden der Pakete im Sensornetz**
  - „promiscuous mode“ auf dem Monitorknoten
  - Modifikation der BTnut-Quellen
    - MAC Protoll (bmac.c)
    - Packethandler (ccc.c)
- **Übertragung der Pakete an den PC**
  - Einsatz des Bluetooth Radios
  - Verbindungsorientiertes „rfcomm“ Protokoll garantiert die unverfälschte Übertragung
  - Löschen übertragener Pakete
  - Einstellbares Zeitintervall zwischen den Übertragungen

BTnode Monitor-Software

- **Zeitsynchronisation**
  - Gewünscht: Empfangszeit für jedes erlauschte Paket (für spätere Analyse von Interesse)
  - Problem: BTnodes besitzen keine Uhr, nur einen Zähler
  - Lösung: PC übernimmt die Berechnung der Empfangszeiten

- **Erweiterung jedes erlauschten Datenpakets um ein weiteres 4 Byte großes Feld („systemtime“)**

BTnode Monitor-Software

- **Zeitsynchronisation - Idee**
  - Berechnung mit Hilfe des internen BTnode Zählers
    - Bei Paketempfang wird der aktuelle Wert dieses Zählers (Millisekunden) in diesem neuen Feld „systemtime“ gespeichert
    - An den PC wird mit jedem Datenpaket die Differenz aus dem aktuellen Zählerwert und „systemtime“ übertragen
    - Der PC subtrahiert diesen Wert von seiner aktuellen Uhrzeit und erhält die Empfangszeit des Pakets
  - Verfolgungszeitraum: ca. 49 Tage, danach Überlauf
  - Bei mehreren PCs: Synchronisation zwischen diesen nötig (z.B. mit NTP)

PC Software

Aufgabe: Entgegennahme und Weiterleitung der Pakete an den Server

- Threads zur Realisierung von Nebenläufigkeit
  - Entgegennahme von Verbindungen u.U. mehrerer Monitorknoten
  - Weiterleiten empfangener Pakete an den Server
- Erweiterung des Datenpakets um die Bluetooth MAC- Adresse des Monitorknotens (für die spätere Zuordnung der Pakete zu den entsprechenden Monitorknoten)
- Anpassen des TCP Sendepuffers, um genau ein „erlauschtes“ Paket in den Nutzdaten eines TCP Pakets zu verschicken

Server Software

<u>Aufgabe:</u> Verbindungen zulassen und Pakete u.U. von mehreren PCs empfangen

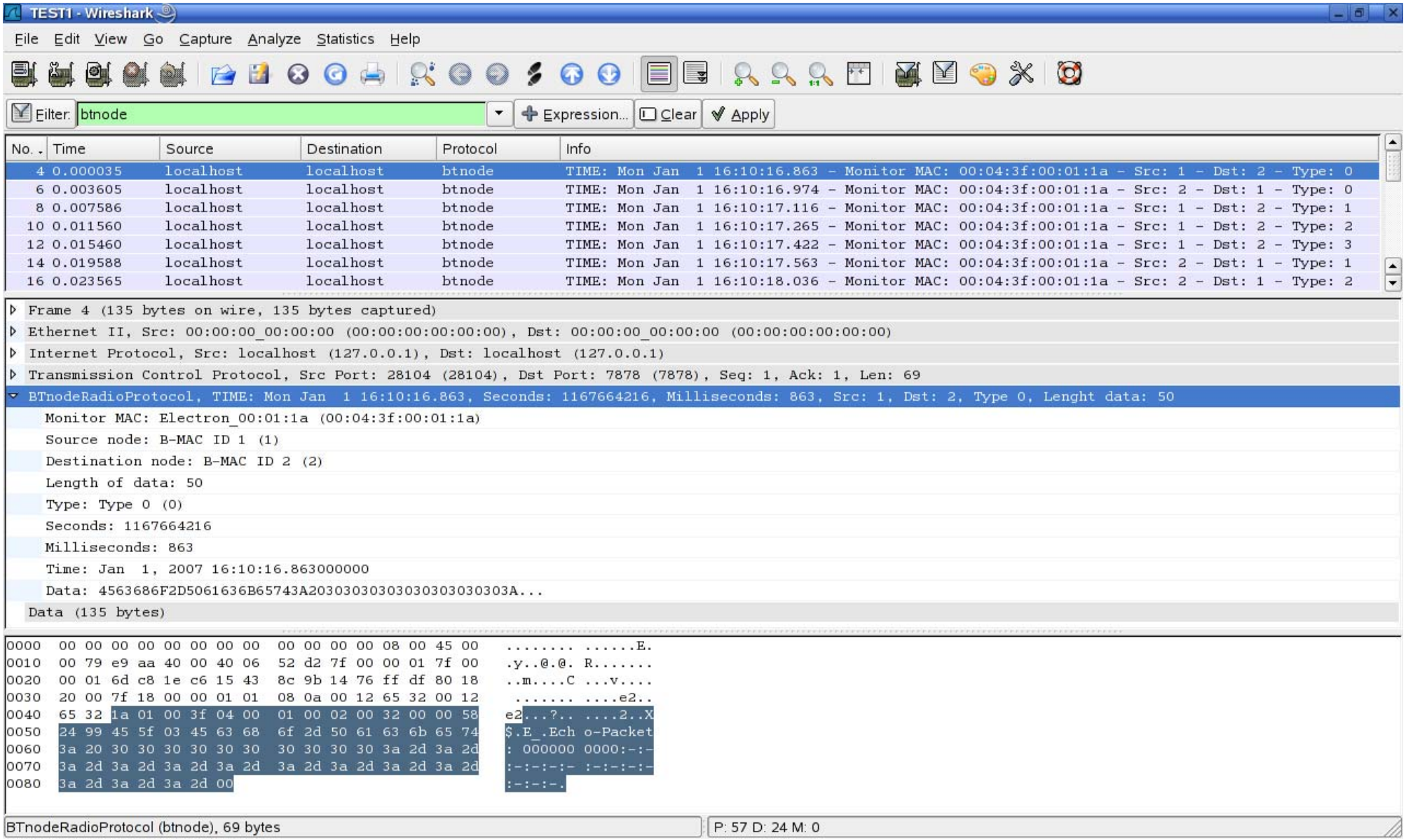
- Empfangene Daten werden nicht benötigt und verworfen

Wireshark Plugin

<u>Aufgabe:</u> Dekodierung und Interpretation der zwischen PC und Server ausgetauschten Datenpakete

- Grundfunktionalität durch Wireshark bereits gegeben
  - **Graphische Benutzeroberfläche**
  - **Speicher- und Exportfunktionen**
  - **Anzeigefilter**
  - **etc.**

Benutzeroberfläche Wireshark



Funktionsweise und Eigenschaften

- **Tool zum verteilten Monitoring in Sensornetzen**
  - Passives Mitschneiden des anfallenden Datenverkehrs
  - Zusätzlicher Monitorknoten im Sensornetz
  - Keine Beeinflussung des „natürlichen“ Verhaltens der zu überwachenden Sensorknoten im Sensornetz
  - Hierarchische Architektur
- **Modularität**
  - Erweiterung der Architektur um weitere sog. Monitorinseln
- **Pushlösung**
  - Selbstständiges „Ausliefern“ des erlauschten Datenverkehrs

Funktionsweise und Eigenschaften

- **Zentrale Zusammenführung aller erlauschten Pakete**
- **Einfaches Analysieren, Speichern, Exportieren etc. über die graphische Benutzeroberfläche von Wireshark**
- **Plugin für Wireshark übernimmt Dekodierung und Interpretation der Pakete**

Einsatz in Forschung und Lehre

- **Forschung**
  - Debugging in Sensornetzen
  - Bewertung und Analyse von Kommunikationsmethoden
- **Lehre**
  - Weiterführende Schulen
    - aktuell in Bayern: Informatik als Pflichtfach im “G8”
    - Kommunikation zwischen Informatiksystemen als zentraler Schwerpunkt in der 12. Klasse
  - Universität
    - Unterstützung des Lehr- und Übungsbetriebs
      - Veranschaulichung geeigneter Sachverhalte aus den Vorlesungen
      - Visualisierung der Kommunikation im Sensornetz als “Feedback” beim Testen eigener Sensorknotensoftware

Fragen?

Vielen Dank für Ihre Aufmerksamkeit!