Remote Data Collection for CAN-based Distributed Systems

RoSES

Robust Self-Configuring
Embedded Systems

Ernie Pusateri
Chris Martin
Pratish Halady
Aditi Bajoria

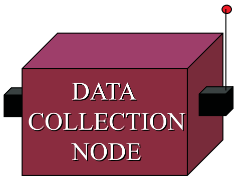
- **Focus**
  - Design and optimize a system that remotely reads data from an automotive CAN bus in real-time
- **Motivation**
  - Collect information from a car's internal distributed system for component evaluation and fault reporting

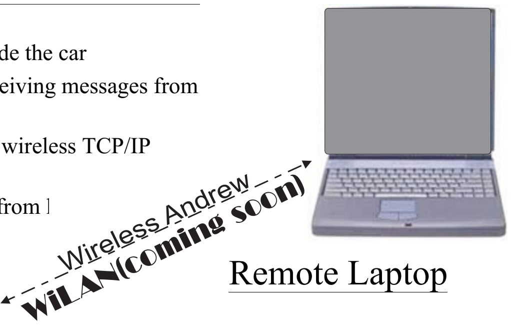
PC modeling car distributed system

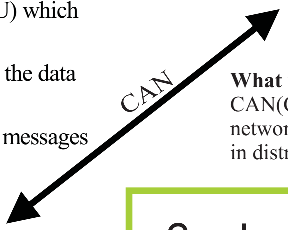
- Simulates "electrical control units" (ECU) which are connected by a CAN bus.
- ECUs use CAN bus to send messages to the data collection system.
- System can also react to incoming CAN messages from data collection system.

PC modeling data collection node

- Resides on the CAN bus inside the car
- Collects data from car by receiving messages from CAN bus.
- Interacts with remote PC via wireless TCP/IP network.
- Can be configured remotely from l





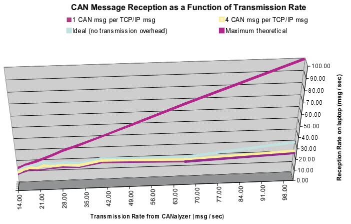


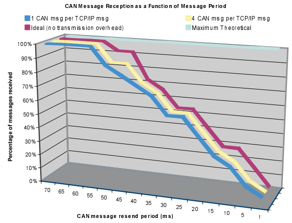
Remote Laptop

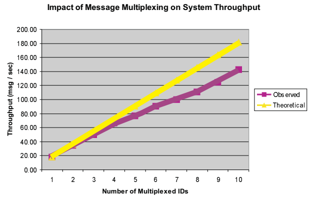
- Send and receive data from the Data Collection Node over a wireless TCP/IP connection.
- Sends messages to configure the Data Collection Node, telling it which messages to receive from CAN bus.
- Sends messages to put CAN messages onto the bus.

What is CAN?
CAN(Controller Area Network) is a network protocol designed for use in distributed embedded systems.

System Analysis and Optimization: Using Bandwidth Efficiently







First Approach: CAN message Buffering

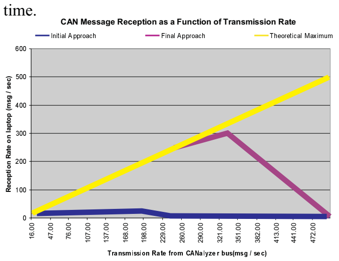
- **Approach description:** Buffering places multiple CAN messages inside of a single TCP/IP message**.**
- **Reason to try this approach:** If the system bottleneck is in the TCP/IP connection, buffering will increase throughput by dividing the TCP/IP header overhead over many CAN messages.
- **Results**: Buffering provided no significant improvement in throughput.
- **Conclusion**: The TCP/IP connection is not the bottleneck in the system.

Second Approach: CAN Message ID Multiplexing

- **Approach description:** ID Multiplexing rotates message IDs in a round-robin fashion when transmitting data on the CAN bus.
- **Reasons to try this approach**: Decreasing the period between CAN messages causes message loss, limiting the data collection rate. Dividing the overhead of service time over multiple mailboxes increases throughput.
- **Results**: Approach resulted in near linear increase in throughput.
- **Conclusions**: Results showed CAN device driver would read EVERY "full" mailbox each time ANY mailbox was polled.

Final Approach: More efficient use of Driver

- **Approach description:** Optimize hardware driver to more efficiently service incoming messages by reducing the time spent waiting for incoming control messages.
- **Reasons to try this approach:** Previous results indicated that system was handling message reception inefficiently. Drastic improvements using multiplexing indicated that the mailbox service algorithm had a large amount of associated overhead.
- **Results:** Dramatic throughput increase superior to previous approaches.
- **Conclusion:** System bottleneck determined to be control message waiting time.



Outcome:
Demonstrated feasibility of real-time remote monitoring of a CAN bus via a wireless link using a CAN to TCP/IP gateway for automotive applications.

BOSCH

GM

Institute for Complex Engineered Systems

Electrical & Computer ENGINEERING