Bedside Patient Monitoring Network

Abdulrazak Alghabra[1], Derico Pratama[2], Karan Mehta[3]

School of Computation, Information and Technology, Technische Universität München

[1]abdulrazak.alghabra@tum.de

[2]derico.pratama@tum.de

[3]karan.mehta@tum.de

Abstract— **Due to frequent understaffing in hospitals, there is a strong requirement for efficient patient monitoring systems, which are vital in decreasing adverse events and mortality rates. Our system uses heart rate sensors and motion sensors to monitor the conditions of the patients. Low-latency communication is the objective of our network design, given the nature of our application. Finally, our system also has a graphical user interface that includes the overview of the patients' conditions and also the network system.**

I. Introduction

Patient monitoring systems play a vital role in healthcare by ensuring the continuous and accurate assessment of patient's health conditions. Understaffing is one of the prevalent issues frequently encountered by hospitals. With a nurse-to-patient ratio of 1:4 [1], these systems become crucially important in ensuring quality care.

They detect and monitor vital signs like heart rate, blood pressure, oxygen saturation, and respiratory rate. By providing real-time data and alerts, these systems enable prompt intervention, enhance patient safety, and improve healthcare outcomes. Additionally, patient monitoring systems optimize resource allocation by prioritizing patients based on severity. With the ability to track vital signs and provide timely alerts, these systems contribute to high-quality and personalized patient care in various healthcare settings.

Patient monitoring systems give healthcare workers the ability to closely monitor patients, spot early indications of deterioration or complications, and take immediate action.

II. Implementation

A. Environmental Conditions

For our demonstration the environment in which the system operates is considered mostly stationary. The patient is assumed to be on the bed for most of the observation period. Therefore, the sensors are also considered stationary.

The sensors used for the demonstration include:

1. Pulse Sensor
2. Motion Sensor

The system assumes a static environment without frequent changes in the network topology.

The foremost concern of this application is latency. An ideal system would have low latency, enabling timely data transmission. As higher latency will cause delays in assisting the patient. Power consumption is not a major concern, as it is an indoor system and connected to the power source, allowing us to focus on other aspects of system design and performance optimization.

III. Network Description

As mentioned before, low latency communication is highly crucial for this application when it comes to emergency situations. The network architecture, network discovery, failure recovery and MAC configurations of this system is designed to meet that requirement.

A. Network Architecture

The system implements a hierarchical and heterogeneous network. It is hierarchical in a sense that the nodes are classified into three groups: Patient (P), Intermediate (I) and User (U) nodes. Each class of nodes has different capabilities and functionalities, hence its heterogeneity. Patient nodes collect the data from the sensors, pack them into a specific data type and send them to the Intermediate nodes. Patient nodes only talk to the Intermediate nodes and do not talk to each other. They have smaller transmission power (0 dBm), and subsequently smaller range.

Intermediate nodes, on the other hand, can connect to every other node, including other Intermediate nodes, since their function is to relay messages until they reach the destination (User node). They always have a maximum transmission power (7 dBm). Sensor data from the Patient nodes is collected here first before being sent outwards every one second (plus small random delay to avoid network overload).

Finally, the User node (the destination node) functions as a gateway to the terminal (computer). They are connected with a serial communication link.

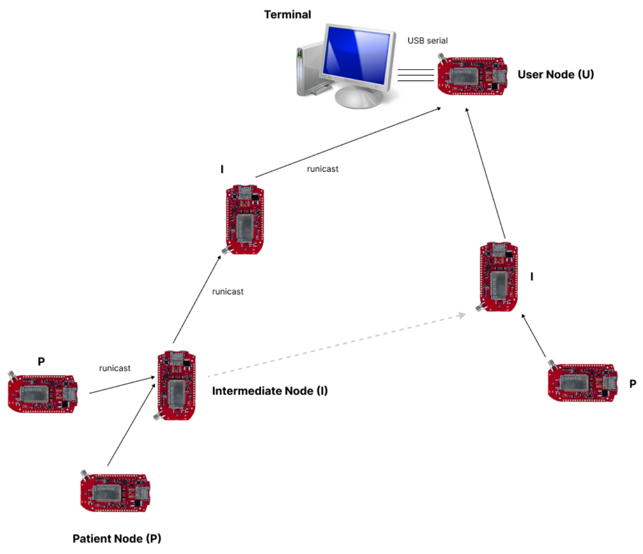
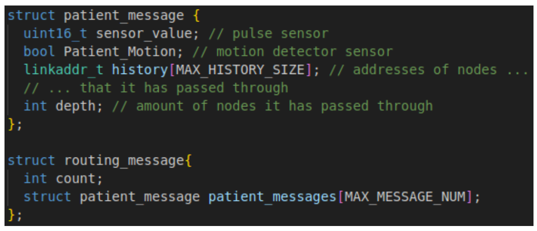


Fig. 1. Network Topology Example

B. Messages

The messages being exchanged in the system can be classified into two groups: Payload messages and Maintenance messages.

1) *Payload messages:* include all messages that contain the sensor data. In the Patient nodes, the sensor readings are packaged into *patient_message*. The *patient_message* of each Patient node is then collected by the common Intermediate node as *routing_message*, which is sent towards the User node. Payload messages are always transmitted using runicast protocol during the data transmission phase.

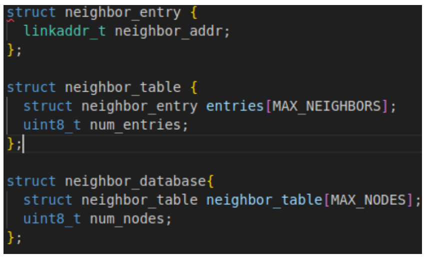


```
struct patient_message {
  uint16_t sensor_value; // pulse sensor
  bool Patient_Motion; // motion detector sensor
  linkaddr_t history[MAX_HISTORY_SIZE]; // addresses of nodes ...
  // ... that it has passed through
  int depth; // amount of nodes it has passed through
};

struct routing_message{
  int count;
  struct patient_message patient_messages[MAX_MESSAGE_NUM];
};
```

Fig. 2. Structure of Payload Messages

2) *Maintenance messages:* include all messages that are exchanged during the network discovery phase. Each Intermediate node and the User node has its own *neighbor_table*, containing the addresses of its neighbors. The *neighbor_table* is then exchanged with the neighbors to build the node's own *neighbor_database*. Maintenance messages are always transmitted using broadcast protocol during the network discovery phase.



```
struct neighbor_entry {
  linkaddr_t neighbor_addr;
};

struct neighbor_table {
  struct neighbor_entry entries[MAX_NEIGHBORS];
  uint8_t num_entries;
};

struct neighbor_database{
  struct neighbor_table neighbor_table[MAX_NODES];
  uint8_t num_nodes;
};
```

Fig. 3. Structure of Maintenance Messages

C. Network Discovery and Failure Recovery

Given the heterogeneity of the system, each class of nodes have different network discovery and failure recovery protocols. To begin with, each class has its own address signature with which it can be easily identified. The last byte of the Patient nodes are always smaller than 0x80, while for Intermediate nodes they are always larger than or equal to 0x80. Meanwhile the address of the user node is always 0xFFFF.

1) *Network Discovery*
Patient nodes initiate the network discovery process by broadcasting and receiving messages from and to any Intermediate nodes. When the Patient node receives a broadcast or a reply message from an Intermediate node, it stops broadcasting and considers the neighbor discovery done.
During the network discovery phase, the Intermediate and User nodes are essentially performing the same protocols. First, they broadcast a message to identify their neighbors. This message reaches a node and the source address is then put into the receiving node's *neighbor_table*. Then, they all broadcast and receive the *neighbor_table* from their neighbors and input them to their own *neighboring_database*. To put it simply, every Intermediate node and User node knows every neighbor of their neighbors through *neighboring_database*. These processes are repeated ten times each to ensure every node gets all the data from their neighbors.
Once the neighbor discovery is done, the routing algorithm is executed. When the shortest path is built, the system stops using any broadcast messages and starts using runicast to send the *routing_messages*. The network discovery is rerun whenever there is a transmission failure, and also every 15 seconds regardless of failure to ensure the freshness of the network topology.

2) *Failure Recovery*
The system runs on runicast during the transmission phase. Runicast includes ACK messages, therefore any transmission failure can be detected swiftly. Once failure is detected in a node, it initiates the network discovery protocol again and floods the network with

a "reset message" to notify the other nodes for network rediscovery..

For Patient nodes, in addition to the protocols mentioned in *1*, they also perform an increment of the transmission power from 0 dBm to 7 dBm gradually to find a new Intermediate node.

D. *MAC Configurations*

For this application, low latency data transmission is the most important criteria that needs to be met. According to [2], for medium load systems, contention-based MAC provides the lowest latency. For that reason, CSMA MAC is implemented here. In addition, since it is an indoor system, power consumption is not an important issue. Therefore, NullRDC (where nodes are always on) is implemented to further reduce the latency. As mentioned in before, the system uses a combination of broadcast and runicast protocols for data transmission, where broadcast is used during the network discovery phase and runicast is used during the data transmission phase.

IV. Routing Techniques

As the network is mostly static, the network discovery is infrequent. It happens either every 2 minutes or in case the node fails to transmit the data.

Path finding is only done while performing network discovery. Initially all the nodes create a database, in which the information about neighbouring nodes is stored. After that, each node shares these databases with their neighbouring nodes. The final data has each node with their neighbouring nodes.

The next part is to decide the next hop for every node. Initially every node looks for the User node in its database. If it is directly connected to it, the next hop is directly to the User node. If it is not directly connected to the user node then, it recursively tries to find the User node in its neighbouring nodes. Once it finds the User node it traces it back and jumps out of the loop. And it saves the immediate neighbour as the next hop value. This next hop is calculated every time the network discovery takes place. For every node this hop value is generated and is fixed until the next network discovery.

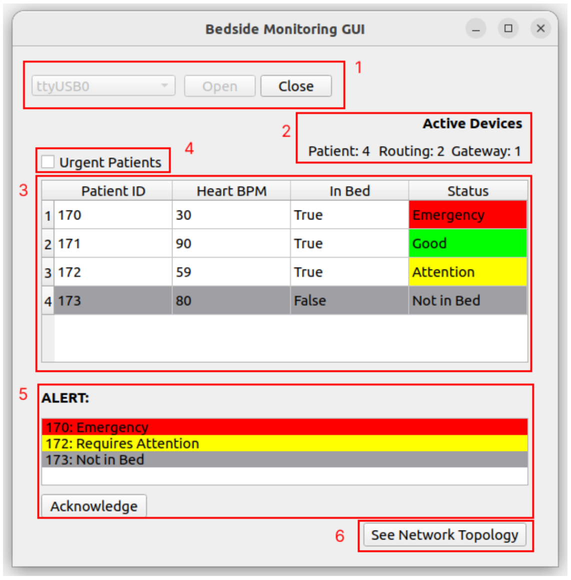
V. Graphical user interface

The graphical user interface (GUI) is created using Qt Creator (C++). The GUI application does not only function as a display terminal for the received sensor data, but also as a final processing point. For instance, the process of classifying which patients are in need of assistance takes place in the GUI application. That way, the computational load on the capacity-limited motes is reduced.

The GUI application consists of two windows: the main application window and the network topology window.

A. *Main Application Window*

The main application window connects to the gateway mote to receive data and displays it as a table. This window is the interface that users (doctors and nurses) will mostly use.



	Patient ID	Heart BPM	In Bed	Status
1	170	30	True	Emergency
2	171	90	True	Good
3	172	59	True	Attention
4	173	80	False	Not in Bed

ALERT:
170: Emergency
172: Requires Attention
173: Not in Bed

Fig. 5. GUI: Main Application Window

The main application window consists of the following features:

1) *Serial port connections*: by implementing the library *qextserialport*, the application connects to the gateway via USB serial communication link. Since the exchange messages are custom data types, they have to be serialized in the gateway and then de-serialized when reaching the application.
2) *Status of connected devices*: this feature helps users quickly check the number of patients and monitor the health of the network.
3) *Data table*: displays the patient data. The Patient ID is obtained from the source address of the message. Instead of being displayed in hexadecimal like the original address, it is translated to decimal, taking only the last byte of the address which corresponds to the room and bed number of the patient. This makes it easy for the users to quickly respond to an emergency situation. The status of the patients is determined using the decision tree in *Figure 6* and is color-coded.

   To avoid the patient data from piling up in the table, a time-to-live (TTL) is introduced in each data. The TTL is given by *number_of_nodes * 4* incoming messages after the last received message that includes the data of that particular patient node. For a system with 5 patient nodes, the application waits until 20 consecutive messages without the data of the patient before removing it from the table.

4) *Data filtering*: filters the data table to exclude patients in “Good” conditions.
5) *Alert notification*: shows a list of patients that might need personal assistance. The list is sorted by urgency (“Emergency” → “Not in Bed”). At the bottom, there is also an “Acknowledgement” button to clear a notification that has been noticed or handled.
6) *Access button to network topology*: opens the network topology window. The main window remains open.

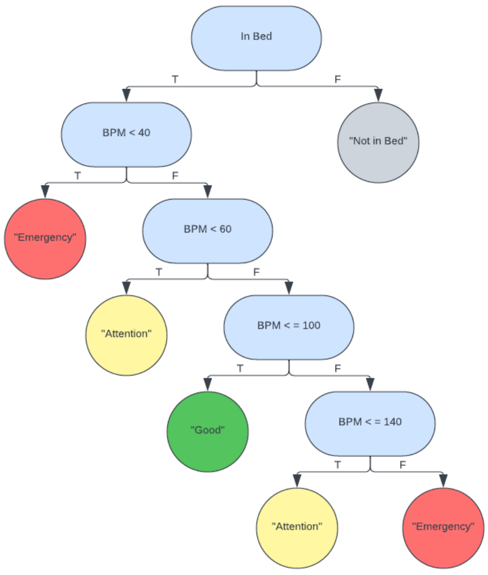
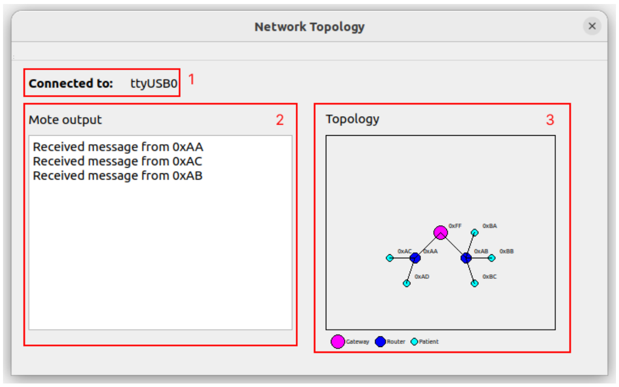


Fig. 6. Decision Tree for Patient Status

B. Network Topology Window

The network topology window is mainly used for maintenance, network monitoring and is mostly accessed by the network administrator.



Received message from 0xAA
Received message from 0xAC
Received message from 0xAB

Fig. 7. GUI: Network Topology Window

It consists of the following features:

1) *Connection status label*: shows whether it is connected to a certain device.
2) *Mote output*: notifies about incoming messages. Here, the original addresses of the motes (in hexadecimal) are used to make debugging easier for the network administrator.
3) *Topology*: draws the network topology. The data of the node connections is obtained from the *history* attribute of each message.

In addition to the GUI application, LEDs on the motes can be used to indicate the ongoing processes. In the Patient nodes, the LED indicators has the following meanings:

- Blue: message transmission for neighbor discovery.
- Green: *patient_message* transmission.
- Purple: Intermediate node not found.

Meanwhile, the Intermediate nodes has the following LED indicators:

- Green: *routing_message* transmission.
- White: *neighbor_database* transmission.
- Blue: neighbors discovered.
- Purple: received “initialization” message from the Patient node.
- Red: received “reset message” during failure recovery.
- Yellow: received *routing_message*.

The User node has the same LED indicators as the Intermediate nodes, except that it does not have a Green LED since it does not transmit *routing_message*.

References

[1] S. K. Sharma and R. Rani, “Nurse-to-patient ratio and nurse staffing norms for hospitals in India: A critical analysis of national benchmarks,” Journal of family medicine and primary care, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7491754/ (accessed Jul. 6, 2023).

[2] A. Bachir, M. Dohler, T. Watteyne and K. K. Leung, "MAC Essentials for Wireless Sensor Networks," in IEEE Communications Surveys & Tutorials, vol. 12, no. 2, pp. 222-248, Second Quarter 2010, doi: 10.1109/SURV.2010.020510.00058.