Distributed Systems Group 14

Final Report

Environmental Monitoring System in Python

Group 14: Asya Karabulut, Kristian Paulic, Mario Müller
GitHub Repository: https://github.com/kristian-p7/Distributed-Systems
February 1st February 2024

1 Introduction

The aim of the project is to develop a distributed system for monitoring ocean wave height. This system represents a significant leap in maritime technology, focusing on a decentralized, resilient and scalable approach that seamlessly adapts to the dynamic and unpredictable nature of the ocean. At the heart of the project is a peer-to-peer architecture, a structure carefully chosen for its robustness and adaptability. Unlike traditional centralized models, this system uses multiple wave sensors. These sensors can be dynamically added or removed, providing unparalleled flexibility and responsiveness to changing ocean conditions. This feature is particularly important for real-time monitoring, ensuring that the system is not only operational but also highly effective with as few as three active sensors. The innovative spirit of the project is fur-

ther exemplified by its fault-tolerance capabilities. Ocean monitoring poses significant challenges due to the harsh and unpredictable marine environment. The system's fault tolerance ensures uninterrupted functionality, a critical factor in scenarios where reliable data is essential for timely decision making. One of the most notable innovations of this project is the implementation of a voting mechanism to select a leader sensor. This mechanism is at the heart of the decentralized decision-making process. The leader-sensor, once elected, takes on a central role. It is responsible for compiling and calculating average wave heights from the data collected by the network of sensors.

The information gathered by these sensors is highly valuable, contributing significantly to our understanding of the marine environment and helping to formulate effective maritime policies and strategies. By harnessing the power of peer-to-peer architectures and innovative sensor technology, this project stands as a beacon of progress in our ongoing endeavor to understand and protect our marine ecosystems.

2 Project Requirements

2.1 Dynamic Discovery

To establish communication for any network an essential step is the identification of participants. Therefore, the network of the wave-monitoring system must support dynamic discovery. This concept allows new participants to join the network without knowing the already connected peers without central coordination. Furthermore, the participants must be able to be removed from the session without central coordination too. The technological concept enabling that is the use of UDP Broadcast.

Therefore, the system must be able to manage different contexts. In case the joining peer is the first in the session, it must become the leader at least until another peer joins. The second possible context is the peer is joining an already existing session, in this case must establish a connection to the network without central coordination of the network. The last possible context is the reaction of an already connected peer in the network to a new peer. The new peer must be acknowledged and its position among the other peers must be defined.

To provide all these functions, the system must continuously listen to messages from the other participants, if a new one is announcing itself. The joining participants must send this standardized announcement to be acknowledged. For their identification, the message must contain its IP-Adress, which is necessary for the list of the participants and the creation of a UUID.

2.2 Crash Fault Tolerance

The network must be able to maintain the session even if one of the different participants in the system, the wave-sensors, leaves the session, crashes or does anything else which results in a loss of connection. Furthermore, the other participants must acknowledge that one of the other participants lost the session. The network must update the list of participants and implement the change. Because the participants in the network each know their neighbors, the ones next to the missing participant must acknowledge that their neighbors changed.
To provide this function, the neighboring wave-sensors are monitoring each other's status by exchanging heartbeats using TCP Unicast. A ring of sensors is formed by each sensor individually searching for its left and right neighbors based on a UUID in a sorted list of active sensors in the system. The fault tolerance mechanism requires at least 3 participants to be operational to calculate the average wave height and form a ring. If the number is less than 3, the system waits for a new participant to join. A broadcast message is sent from the neighboring sensors of the departing sensor to every other participant, so that they can update their active participant list.

2.3 Voting

The voting mechanism is necessary for the determination of a leader among the participants. The mechanism must be executed without central organization and be able to elect a leader any time a new leader is needed. In the wave-monitoring-system the leader is responsible for the aggregation of wave values and the calculation of averages. At the beginning of the session, the leader is always the first sensor,
that does initiate the broadcast and starts the system. The list must be updated every time a new participant joins the network.

A Broadcast message is sent to all participants, every time the list of active participants is updated, telling the sensors, who is the leader. In the event of the leader's departure from the system, the system must acknowledge the absence of the old leader and initiate an election, where the new leader can be elected. Furthermore, it is essential for the new leader to confirm the election. Therefore, it must send a message confirming it has acknowledged its new status as a leader. The message must be received from any other participant, so that the election process can be finished. Essential for the implementation of the process is its reliability and safety. This means that as the result of the process the right leader is chosen and confirmed. Also, all participants must take part in the election and make a vote. It is important that no other participant crashes during the election.

3 Architecture Design

The environmental monitoring system is conceived to have a decentral organization to facilitate operational reliability even in case of missing participants. For that, the system is based on a “Peer-to-Peer Network”. This implies that the sensors used for the wave-surveillance are the participants of the network, called peers. The peers are ordered in ring-formation, as shown in figure 1. Each peer has its own unique ID to be identified among the others. Every peer joining the network is assigned a new ID. This ID is derived from the total number of existing peers in the network, incremented by 1. Due to the architecture of the network, it supports Dynamic Discovery, which allows seamless addition or removal of peers without central coordination. This functionality of the system is based on the concept of “UDP Multicast”.

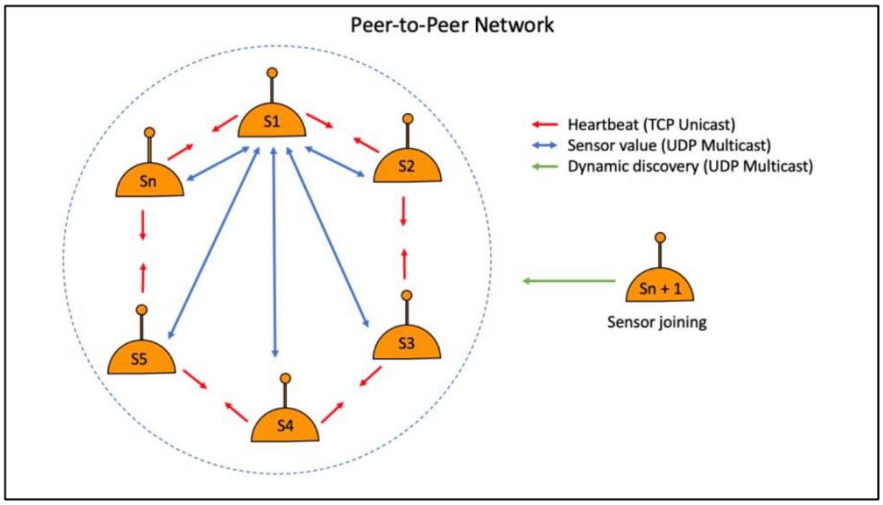

Fig. 1: Peer-to-Peer Network example: diagram environmental monitoring

The peers are sending their wave-values measured by the sensors to the leader in constant exchange. The monitoring program collects the wave-values of the different sensors and calculates values averages. To ensure that a constant connection is maintained, the Heartbeat is utilized. This is a signal that peers within the circular formation send to their immediate neighbors on the left and right at regular intervals of 5 seconds. The Heartbeat is based on TCP Unicasts.

4 Implementation

4.1 UDP Broadcast

The ‘BroadcastListener’ class, is designed for handling broadcast messages in a networked application. Upon initialization, it sets up a UDP socket, configured for broadcast communication, and binds it to a predefined broadcast port 59073. The class continuously listens for incoming UDP packets in its run method. When data is received, it decodes the message and processes it based on the command type in the ‘handleMessage’ method. For a ‘NEW_SENSOR’ command, the class adds the new peer's IP and UUID to the peers list if it's not already present. Conversely, for a ‘LOST_SENSOR’ command, it removes the peer with the specified UUID. Running in its own thread, ‘BroadcastListener’ allows asynchronous listening for broadcast messages, facilitating efficient network communication management in the application.
The ‘BroadcastSender’ class is designed to send broadcast messages over a network. It initializes with a unique UUID for the sender and determines the broadcast IP (bcip) and port (bcport) to use. The class creates a new UDP socket for each message to send, configures it for broadcasting, and sends a message containing a command, the sender's UUID, IP address, and the actual message content. The message is encoded before sending. This class enables the application to communicate with other networked entities using broadcast messages. The UDP Sender is not only used when a new sensor is joining the peer-to-peer network of sensors, but also when the active participant list is updated (e.g. lost sensor event).

4.2 Heartbeat based on TCP Unicast

In the intricate design of a peer-to-peer network, the process of onboarding new peers and maintaining the stability of the network's ring topology has a high importance. This process is adeptly handled within the provided codebase through the orchestrated efforts of several specialized classes.

The BroadcastListener class plays a critical role by listening for announcements from newly joined peers. Instead of an INIT message typically used in many systems, the BroadcastListener looks out for a "NEW_SENSOR" command. Upon detecting such a message, it adds the new sensor's information to the global peers list. This action ensures that the network is consistently updated with the latest members, thereby maintaining a comprehensive registry of all active nodes.

Concurrently, the BroadcastSender class functions as the communication beacon of the network. It broadcasts the presence of a new peer by sending out a

"NEW_SENSOR" message periodically. This repetitive broadcasting acts as a pulse, keeping the entire network informed and synchronized with the arrival of new peers or the loss of existing ones.

In addition, the TCPListener class handles heartbeat messages, which are essential for checking the liveliness of each node.. This class confirms the operational status of peers by sending back an acknowledgment upon receiving a heartbeat, thus playing a vital role in the fault tolerance mechanism of the system.
In addition, the TCPSender class is the active participant that manages the sending of heartbeats to adjacent nodes. If a heartbeat goes unanswered, signaling a potential node failure, the TCPSender takes action by notifying the rest of the network through the BroadcastSender.

Collectively, these components—BroadcastListener, BroadcastSender, TCPListener, and TCPSender—form the backbone of the network's ring topology. They facilitate the seamless integration of new peers, uphold continuous monitoring through Heartbeats, and enable swift reconfiguration of the network in response to dynamic changes

The Heartbeat Mechanism:

In the sophisticated design of the wave height monitoring system, the TCPListener and TCPSender classes synergize to form the backbone of the network's heartbeat communications. Each class, integral to the system's function, plays a unique role in ensuring the robustness and resilience of the peer-to-peer network.

The TCPListener stands on its designated port, attentively scanning for in-coming heartbeat messages from peers. When such messages are detected, the TCPListener affirms the network's vitality by sending back timely acknowledgments, confirming the ongoing operational status of the peers. This confirmation signals the active presence of a peer. Once acknowledgments are dispatched, the TCPListener continues its watchful stance.

Concurrently, the TCPSender class proactively extends the heartbeat rhythm across the network. It continuously identifies and communicates with the nearest neighbors, defined by the find_neighbors method—to verify their presence. This consistent heartbeat exchange is crucial for the network's ability to quickly detect and adapt to changes, whether it's the addition of new peers or the unexpected loss of existing ones.
If a peer does not respond, the TCPSender and BroadcastSender work together to determine the peer's failure and share this information across the network using the sendLostPeerMessage method to improve the network's structure.

The BroadcastListener class helps to handle the consequences of a peer's failure. It initiates a new voting process when the leader is compromised, ensuring that the network's leadership is current and representative.

The system's design is centered around its ring structure. This structure requires that each peer has two neighbors, one to the right and one to the left. It is important to note that sorting is crucial. The TCPSender class ingeniously manages the spatial arrangement of peers within the network, including identifying each peer's immediate neighbors, using the find_neighbors method. The text has been improved by simplifying the vocabulary, using shorter sentences, and ensuring grammatical correctness. The method uses the modulo operation to circulate through the list of peers smoothly, ensuring continuous and consistent network connectivity. For example, (index + 1) % len(sorted_uuids) calculates the right neighbor by wrapping around to the beginning of the peer list when the end is reached, maintaining an unbroken chain of communication across the network.

This dynamic heartbeat protocol, enriched by the modulo-based navigation through the peer list, exemplifies the system's commitment to fault tolerance. It allows the network to dynamically adjust to both the addition of new peers and the unexpected departure of existing ones, underscoring the system's capacity to sustain its critical role in real-time ocean wave height monitoring.
https://en.wikipedia.org/wiki/Peer-to-peer

4.3 Leader Election / Voting (LCR Algorithm)

An essential part of the system is the Leader Election. This process serves the function to elect one of the peers as the leader of the system. The leader has a higher responsibility in the environmental monitoring system and is running the monitoring functions such as the calculation functions. The leader is evaluating the values sent by the peers and sends the calculated results in return.

To elect the leader the system is following a certain process:
The voting is initialized by the system, when a new peer joins the system. Voting and leader election is done only when at least 3 sensors are in the system and every time a new sensor is joining or is lost. In case of the wave-monitoring-system, our implementation contains information, which is relevant for the voting-process and the calculation-process at the same time. The calculation-process is described in the next chapter.

The process starts with the actual leader sending a message to its right neighbor via TCP Unicast. This message contains always the UUID of the sender.
The procedure is kept as long as the same unchanged message of one peer has circulated one time around the whole ring of peers containing the same UUID and returning to its sender. After the circulation, the peer getting its own message with its own

ID back is elected as the new leader. This process guarantees that the peer with the highest UUID becomes the leader.

To confirm the election, the new leader sends a new message for election acknowledgement (message leader elected = true) to its right neighbor, which sends it to its right neighbor again. After the election acknowledgement message was sent around the whole circle, the election is confirmed, and the process is finished with a new elected leader.

4.4 Wave Monitoring

As already mentioned, the continuously sent heartbeats contain the values of the wave-heights measured by the sensor. This implies that each sensor is sending a wave-value continuously to the leader. The leader itself is receiving the wave values from each of the peers. It runs the calculation-operations, where it evaluates the averages of the wave heights. The average is calculated by the accumulated value of the single wave-values divided trough the number of participants in the network. The number of sensors is generated through the function, which is recognizing the length of the list of peers in the network.

In our system, the calculation of average wave height is facilitated by the use of our defined TCPListener and TCPSender classes. These classes collectively represent a process in which wave height values are exchanged between sensors (peers) and the network leader.

- Generation of Wave Height Values:

Each sensor in the network generates random wave height values between 0 and 10 meters using the get_my_wave_height() function. This function simulates the measurement of wave height by a sensor.

- Transmission of Wave Height Values by the Leader:

During the election process, when a peer is chosen as the leader, it begins collecting wave height values from all other peers. This is done through the handle_message method in the TCPListener, which responds to VOTE messages containing wave height values.

- Accumulation and Average Calculation:

Once the leader is elected and receives wave height values from the peers, it accumulates these values and calculates the average. This logic is implemented in the leader's TCPListener once it determines that it is the leader (self.isLeader = True).

In summary, this implementation enables dynamic leader detection within the network and coordinated calculation of average wave height based on individual sensor measurements. This process underscores the importance of the leader election and efficient data communication within the peer-to-peer network for our solution for the implementation of the wave-monitoring system.

5 Conclusion and Limitations

The development of a monitoring-system for ocean wave height is a good example for the practical use of a distributed system. This system, built on a peer-to-peer architecture, provides a decentralized, resilient, and scalable approach to real-time ocean monitoring. The requirements of the project, particularly the use of a voting mechanism for leader selection and the dynamic integration of sensors, are crucial for the effective functioning of the system.

The fault tolerance mechanism is engineered to ensure uninterrupted operation and maintain operability. Furthermore, the role of the leader sensor in gathering and analyzing data from a network of distributed sensors highlights the system's capability for efficient data aggregation and analysis. This strategic compilation and calculation of average wave heights are instrumental in the system's objective to deliver precise and timely identification of critical oceanic events, such as high waves.

Despite the project's features, there are inherent limitations that need acknowledgment and consideration for future development:

- Sensor Coverage and Accuracy: The effectiveness of the system is heavily dependent on the density and accuracy of the deployed sensors. Large oceanic areas may require a significant number of sensors to ensure comprehensive coverage and data reliability.
- Communication Delays and Losses: The peer-to-peer architecture is resilient, but it can be affected by communication delays and losses, particularly in adverse weather conditions that may impact real-time data processing and timely alert issuance.
- Security and Data Privacy: The decentralized nature of the system requires robust security measures to prevent unauthorized access and ensure the integrity of the data transmitted across the network.

In conclusion, the ocean wave height monitoring system stands as a testament to the potential of distributed sensor networks in environmental monitoring. Despite its limitations, the foundation laid by this project provides a solid basis for future innovations. Further research and development are essential to address the identified limitations and fully realize the system's potential in contributing to marine safety.

3109 words