BAB 2
LANDASAN TEORI

Bagian ini memaparkan teori-teori yang secara langsung mendasari tentang permasalahan penelitian, dengan penjelasan yang dibatasi hanya pada teori-teori esensial serta didukung oleh sitasi sumber pustaka yang jelas pada setiap literatur yang digunakan.

2.1 *Cable Based Tsunameter*

Cable-Based Tsunameter (CBT) merupakan sistem deteksi tsunami yang menggunakan jaringan sensor dasar laut (*Ocean Bottom Unit*/OBU) yang saling terhubung melalui kabel serat optik bawah laut [2]. Sistem ini dikenal sebagai *Indonesia Cable Based Tsunameter* (INA-CBT), dan salah satu implementasinya terdapat di Labuan Bajo dan Rokatenda sejak tahun 2020 [4]. Setiap OBU dilengkapi dengan dua jenis sensor utama, yaitu *Accelerometer* (Acc) untuk mengukur perpindahan tiga dimensi, serta *Bottom Pressure Recorder* (BPR) untuk mengukur tekanan air laut yang berkorelasi dengan tinggi kolom air [1]. Data dari sensor-sensor ini dikumpulkan di *Landing Station* (LS) dan kemudian dikirimkan ke *Read Down Station* (RDS) pusat [4].

2.1.1 *Bottom Pressure Recorder*

Bottom Pressure Recorder (BPR) adalah perangkat pengukur tekanan dasar laut yang dirancang untuk memantau fluktuasi tekanan air laut secara presisi di kedalaman yang signifikan. Alat ini umumnya ditempatkan di dasar laut dan digunakan untuk mendeteksi gelombang panjang yang disebabkan oleh fenomena oseanografi seperti tsunami, pasang surut, dan gelombang infragravitasi [4]. BPR merupakan sensor tekanan dengan sensitivitas tinggi yang mengirim data dengan frekuensi 1 Hz dan ukuran data *payload* yang dihasilkan 80 byte. Tekanan yang tercatat dapat dikonversi menjadi estimasi ketinggian kolom air di atas instrumen, sehingga informasi ini menjadi dasar analisis dinamika laut dalam [7]. Ketika terjadi peningkatan mendadak dalam tekanan dasar laut, sistem dapat menginterpretasikan sinyal tersebut sebagai kemungkinan terjadinya gelombang tsunami. BPR sering sekali digunakan sebagai referensi dalam validasi model

numerik gelombang laut serta pengamatan jangka panjang terhadap perubahan iklim laut [4].

2.1.2 *Accelerometer*

Accelerometer (Acc) adalah perangkat sensor yang digunakan untuk mengukur percepatan suatu objek dalam satu atau lebih arah [4]. Acc memiliki sensitivitas tinggi terhadap gerakan kecil dan mampu memberikan data percepatan dalam bentuk vektor. Informasi ini sangat berguna dalam analisis dinamika struktur laut, seperti kabel bawah laut, platform pengeboran, atau sistem pemantauan tsunami berbasis dasar laut. Acc dilengkapi dengan *digital signal processing* yang memungkinkan filtrasi data secara real-time untuk mengurangi gangguan dari gerakan non-seismik. Keunggulan dari Acc terletak pada kemampuannya untuk mendeteksi pergerakan sangat kecil dengan tingkat ketelitian tinggi [8].

Oleh karena itu, alat ini juga digunakan dalam sistem *early warning* gempa bumi dan pemantauan deformasi struktur geologi bawah laut. Data dari Acc dapat dikalibrasi dan dikorelasikan dengan data seismik lainnya supaya membangun model prediksi pergerakan tanah yang lebih komprehensif. Secara teknis, Acc mengirim data dengan frekuensi 125 Hz dan untuk penelitian ukuran data *payload* Acc sebesar 800 byte yang lebih besar dari ukuran data *payload* BPR[4].

2.2 *Message Queuing Telemetry Transport*

Message Queuing Telemetry Transport (MQTT) adalah protokol *messaging* berbasis model *publish/subscribe* yang diciptakan supaya memenuhi kebutuhan komunikasi antar perangkat dengan sumber daya terbatas dalam jaringan yang seringkali tidak stabil [4]. Protokol ini dikembangkan untuk aplikasi yang membutuhkan *overhead* data yang rendah, sehingga cocok untuk aplikasi IoT dan sistem sensor terdistribusi. Secara konseptual, MQTT memungkinkan perangkat *(clients)* untuk saling berkomunikasi melalui perantara tanpa harus saling mengetahui keberadaan secara langsung [9].

Setiap pesan yang dikirim disusun dalam format yang ringan, sehingga penggunaan *bottleneck bandwidth* dapat diminimalisir. Dalam penerapan praktis, MQTT tidak hanya menghemat sumber daya perangkat tetapi juga meningkatkan kecepatan dan keandalan komunikasi, menjadikan protokol ini ideal untuk aplikasi

seperti smart home, monitoring lingkungan, dan sistem pengelolaan energi [10].

2.2.1 *Publisher*

Publisher bertugas untuk menghasilkan dan mengirim pesan ke broker. *Publisher* tidak melakukan pengecekan langsung terhadap pesan yang dikirim ke *subscriber* yang akan menerima pesan tersebut [11]. *Publisher* mengirimkan pesan ke broker dengan menentukan topik untuk pesan. Pesan tersebut dapat berupa data sensor, instruksi kontrol, atau informasi lainnya. Pesan-pesan tersebut biasanya dikemas dalam bentuk *payload* yang ringan untuk meminimalkan penggunaan *bottleneck bandwidth* [12].

2.2.2 *Broker*

Broker bertindak sebagai pusat komunikasi yang mengelola seluruh pesan yang masuk dan mendistribusikannya kepada *subscriber* yang sesuai [13]. Pada model ini, *broker* memastikan bahwa komunikasi antar perangkat berjalan dengan andal dan efisien. Setelah menerima pesan dari *publisher*, *broker* akan memeriksa topik dan menyesuaikan dengan daftar *subscriber* yang ada. *Broker* kemudian mendistribusikan pesan tersebut kepada semua *subscriber* yang terdaftar pada topik yang sesuai [4]. Broker MQTT dapat diartikan sebagai server perantara yang mengelola proses publish dan subscribe antar perangkat IoT untuk menjamin pengiriman pesan sesuai dengan tingkat Quality of Service (QoS), dan memastikan komunikasi data berlangsung secara real-time, andal, serta efisien [14].

2.2.3 *Subscriber*

Subscriber berfungsi untuk mendaftar pada satu atau lebih topik yang diminati dan menerima pesan yang dikirimkan oleh *publisher* melalui broker. Proses pendaftaran ini memungkinkan *subscriber* untuk hanya menerima informasi yang relevan, sehingga mengoptimalkan penggunaan sumber daya [11]. *Subscriber* tidak mengetahui penerbit pesan, hanya topik yang telah didaftarkan yang menjadi acuan penerimaan pesan. *Subscriber* mengirimkan permintaan kepada broker untuk mendaftar pada satu atau beberapa topik tertentu . Permintaan ini memungkinkan *broker* untuk mengetahui pesan-pesan yang harus diteruskan kepada *subscriber* tersebut [12].

2.3 *Quality of Service*

Quality of Service (QoS) adalah salah satu fitur unggulan dalam protokol MQTT yang menangani keandalan pengiriman pesan. QoS menentukan pesan yang dikirim dan diakui oleh broker dan client. MQTT mendukung tiga level QoS yang masing-masing menawarkan jaminan yang berbeda terkait pengiriman pesan [15]. Perbedaan tingkat QoS ini berimplikasi langsung terhadap kemungkinan terjadinya *data loss*. QoS 0 memiliki risiko kehilangan pesan yang lebih besar, sedangkan QoS 1 dan 2 dirancang untuk meminimalkan atau menghilangkan kehilangan data melalui mekanisme pengakuan dan pengiriman ulang pesan [16].

2.3.1 QoS 0

Pada QoS level 0, pesan yang dikirimkan hanya sekali tanpa adanya proses konfirmasi atau pengulangan [4]. Ini berarti bahwa jika terjadi kegagalan dalam pengiriman pesan, pesan tersebut tidak akan dikirim ulang dan penerima tidak akan menerima pesan tersebut. Penggunaan QoS level 0 cocok untuk aplikasi yang toleran terhadap kehilangan data seperti streaming sensor dimana update terjadi secara periodik dan kehilangan satu pesan tidak mengganggu keseluruhan sistem [1].

2.3.2 QoS 1

Pada QoS level 1, pengirim akan mengirim pesan dan menunggu konfirmasi dari broker bahwa pesan telah diterima [4]. Jika tidak ada konfirmasi dalam batas waktu tertentu, pesan akan dikirim ulang hingga konfirmasi diterima. QoS level 1 ideal untuk aplikasi yang membutuhkan jaminan bahwa pesan sampai, meskipun dengan kemungkinan terjadi duplikasi [1].

2.3.3 QoS 2

QoS level 2 merupakan tingkat tertinggi dari jaminan pengiriman pesan dan data yang dikirimkan secara berurutan [4]. Maka mekanisme ini memastikan bahwa setiap pesan hanya diterima satu kali oleh penerima, sehingga kemungkinan terjadinya duplikasi dapat dihindari. Penggunaan QoS level 2 sangat direkomendasikan karena pesan terjamin sampai dan urutan pesan akan tetap sama [1].

2.4 *Bottleneck Bandwidth*

Bottleneck bandwidth merupakan kapasitas terendah pada lintasan komunikasi jaringan yang menjadi faktor pembatas utama terhadap performa transmisi data secara *end-to-end* [4]. Pada sistem INA-CBT, *bottleneck bandwidth* digunakan untuk memodelkan keterbatasan kapasitas jaringan antara LS dan RDS, yang secara langsung mempengaruhi latensi pengiriman data sensor berbasis MQTT [1]. Dalam penelitian berbasis *virtual testbed*, *bottleneck bandwidth* direalisasikan dengan membatasi kapasitas link jaringan menggunakan pengaturan *bottleneck bandwidth* pada router untuk mensimulasikan kondisi jaringan terbatas. Penurunan *bottleneck bandwidth* dapat meningkatkan latensi pengiriman pesan, terutama pada *payload* berukuran besar dan pada penggunaan QoS level yang lebih tinggi [4].

2.5 *Network Simulator 3*

Network Simulator 3 (NS-3) adalah sebuah perangkat lunak simulasi jaringan berbasis *discrete-event* yang banyak digunakan untuk keperluan penelitian, pengembangan, dan pengujian protokol serta arsitektur jaringan [4]. NS-3 dikembangkan sebagai penerus dari NS-2 dengan peningkatan pada struktur modularitas, efisiensi simulasi, serta dukungan terhadap bahasa pemrograman C++ dan *Python* [17]. Sebagai simulator *open-source*, NS-3 menawarkan fleksibilitas tinggi dalam membangun skenario jaringan yang kompleks, seperti jaringan nirkabel, jaringan sensor, hingga simulasi protokol komunikasi modern yang digunakan pada sistem IoT. Dengan memanfaatkan NS-3, dapat mengidentifikasi kendala komunikasi MQTT pada jaringan dengan keterbatasan *bottleneck bandwidth* maupun kondisi dinamis, sehingga mendukung proses optimasi protokol untuk kebutuhan IoT. Oleh karena itu, NS-3 menjadi salah satu alat penting dalam penelitian komunikasi IoT berbasis MQTT, karena mampu merepresentasikan kondisi jaringan secara terkontrol dan terukur sebelum diterapkan pada lingkungan nyata [6].

2.5.1 *Container*

Container merupakan teknologi virtualisasi ringan yang memungkinkan isolasi lingkungan eksekusi aplikasi secara efisien tanpa memerlukan sistem operasi penuh seperti halnya *virtual machine*. Dalam konteks simulasi dan pengembangan

IoT, penggunaan *container* seperti *Mosquitto* memberikan keuntungan signifikan dalam hal portabilitas, dan kemudahan dalam pengujian lintas platform. Teknologi ini memungkinkan penyusunan lingkungan simulasi MQTT yang konsisten dan terstandarisasi, serta mendukung integrasi dengan simulator jaringan seperti NS-3. Dengan demikian, pengujian performa sistem komunikasi IoT dapat dilakukan secara lebih fleksibel dan mendekati kondisi operasional sebenarnya [18].