BAB 1

PENDAHULUAN

1.1 Latar Belakang

Wireless Sensor Network (WSN) adalah suatu jaringan nirkabel yang terdiri dari kumpulan node sensor dengan kemampuan *sensing*, komputasi, dan komunikasi yang tersebar pada suatu tempat. Setiap sensor akan mengumpulkan data dari area yang dideteksi seperti temperatur, suara, getaran, tekanan, gerakan, kelembaban udara dan deteksi lainnya tergantung kemampuan sensor tersebut. Data yang diterima ini kemudian akan diteruskan ke *base station* untuk diolah sehingga memberikan suatu informasi. WSN dapat diimplementasikan pada berbagai bidang kehidupan manusia diantaranya bidang militer untuk deteksi musuh, bidang pertanian untuk pemantauan pertumbuhan tanaman, bidang kesehatan, deteksi bahaya dan bencana alam, bidang pembangunan dan tata kota, dan bidang pendidikan.

Terdapat dua macam arsitektur WSN, yaitu hirarkikal dan flat. Pada arsitektur hirarkikal, node sensor akan disusun secara hierarki dan memiliki peran sebagai *cluster head*, *child node*, atau *parent node*. *Cluster head* berfungsi sebagai pengatur beberapa *child node*. Sedangkan pada arsitektur flat hanya terdapat dua macam node sensor secara fungsional, yaitu *source node* dan *sink node*. Semua node sensor (*source node*) akan mengirim data ke satu tujuan akhir yaitu *sink node*.

Dalam praktiknya pengiriman data merupakan suatu hal yang penting pada WSN. Data yang didapat dari sensor harus dapat sampai ke *base station* dengan utuh dan akurat (*reliable*). Data yang *reliable* ini sangat penting karena hasil pengukuran dan tindakan selanjutnya yang akan diambil akan bergantung pada data-data tersebut. Terdapat beberapa protokol untuk memastikan transfer data *reliable* yaitu dengan protokol *Event to Sink Reliable Transport*,*Reliable Multi Segment Transport*, *Price Oriented Reliable Transport*, *Delay Sensitive Transport*, dan lain-lain.

Pada skripsi ini dibangun aplikasi untuk transfer data pada WSN. Aplikasi WSN yang dibuat juga dapat melakukan transfer data ke node sensor tetangganya hingga sampai ke node sensor yang berperan sebagai *base station*. Karena node sensor memiliki kapasitas penyimpanan yang kecil dan data yang akurat sangat dibutuhkan untuk menentukan tindakan selanjutnya, maka akan dibangun juga WSN yang memiliki sifat *reliable* tersebut.

1.2 Rumusan Masalah

- Bagaimana cara membangun aplikasi transfer data dari setiap node sensor pada *Wireless Sensor Network*?
- Bagaimana cara membangun aplikasi transfer data yang *reliable* pada *Wireless Sensor Network*?

1.3 Tujuan

- Membangun aplikasi transfer data yang *reliable* pada *Wireless Sensor Network*.

1.4 Batasan Masalah

Penelitian ini dibuat berdasarkan batasan-batasan sebagai berikut:

1. Sensor yang digunakan sebagai penelitian hanya sensor untuk mengukur temperatur, kelembapan, getaran, dan tekanan udara.
2. Arsitektur yang digunakan untuk membangun *Wireless Sensor Network* ini adalah flat dan hirarkikal.

1.5 Metodologi

Berikut adalah metode penelitian yang digunakan dalam penelitan ini:

1. Melakukan studi literatur mengenai *Wireless Sensor Network*.
2. Mempelajari protokol transfer data yang biasa pada *Wireless Sensor Network*.
3. Mempelajari prinsip *Reliable Data Transfer* pada *Wireless Sensor Network*.
4. Mempelajari pemrograman pada *Wireless Sensor Network* dengan Bahasa Pemrograman JAVA.
5. Melakukan perancangan perangkat lunak.
6. Mengimplementasi rancangan perangkat lunak pada *Wireless Sensor Network*.

1.6 Sistematika Pembahasan

Setiap bab dalam penelitian ini memiliki sistematika penulisan yang dijelaskan ke dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan yaitu membahas mengenai gambaran umum penelitian ini. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan.
2. Bab 2: Dasar Teori, yaitu membahas teori-teori yang mendukung berjalannya penelitian ini. Berisi tentang *Wireless Sensor Network*, Reliable Data Transfer di WSN, dan pemrograman pada WSN.
3. Bab 3: Analisis, yaitu membahas hasil analisis dari teori-teori yang digunakan berkaitan dengan topik skripsi.
4. Bab 4: Perancangan, yaitu membahas perancangan aplikasi WSN yang reliable untuk pengiriman data
5. Bab 5: Implementasi dan Pengujian, yaitu membahas implementasi dari hasil rancangan dan pengujian dari aplikasi WSN yang telah dibuat.
6. Bab 6: Kesimpulan, yaitu membahas kesimpulan dari hasil pengujian.

BAB 2

LANDASAN TEORI

Pada bab ini dijelaskan dasar-dasar teori mengenai *Wireless Sensor Network*, Reliable Data Transfer di WSN, dan pemrograman pada WSN.

2.1 Wireless Sensor Network

Wireless Sensor Network (WSN) merupakan jaringan nirkabel yang terdiri dari sekumpulan node sensor yang diletakan pada suatu tempat dan memiliki kemampuan untuk mengukur kondisi lingkungan sekitar(*sensing*), komputasi dan dilengkapi dengan alat komunikasi *wireless* untuk komunikasi antara node sensor. Sensor ini akan mengumpulkan data dari kondisi lingkungannya, seperti: cahaya, suara, kelembapan, getaran, gerakan, temperatur, tekanan udara, kualitas air, komposisi tanah, dan lain-lain. Data ini kemudian dapat dikirimkan langsung ke *base station* atau melalui node sensor tetangganya hingga sampai ke *base station* sebagai pusat untuk dikelola.

2.1.1 Penerapan Wireless Sensor Network

Pada awalnya *sensor network* (jaringan sensor) digunakan dalam teknologi militer untuk mendeteksi musuh di laut dan di darat. Semakin lama node sensor ini banyak dikembangkan untuk membantu berbagai bidang kehidupan manusia. Pemanfaatan WSN pada kehidupan manusia dapat dilihat pada ilustrasi Gambar 2.1. Berikut adalah beberapa penerapan WSN:

- Bidang militer
  WSN digunakan sebagai bagian dari komunikasi pada bidang militer dan deteksi target atau musuh.

- Monitoring area
  Pada *monitoring area*, node sensor akan disebar pada suatu tempat yang akan di monitoring. Saat node sensor mendeteksi kejadian(panas, tekanan, dan lain-lain) pada suatu tempat, data akan dikirimkan ke *base station* untuk ditentukan tindakan selanjutnya.

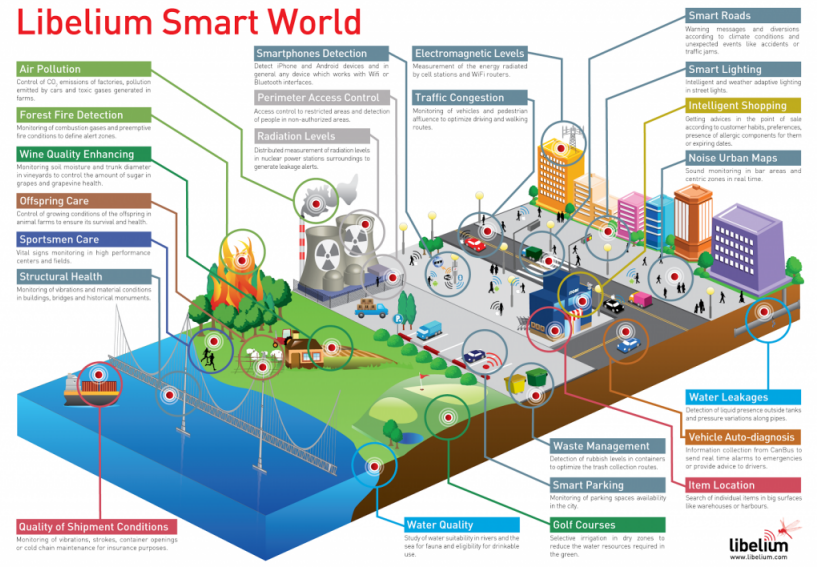
- Transportasi
  Pada bidang transportasi, WSN digunakan untuk mendeteksi lalu lintas secara aktual yang nantinya akan disampaikan kepada pengendara tentang kejadian di depan mereka seperti kemacetan lalu lintas.

- Kesehatan
  Beberapa aplikasi kesehatan seperti membantu pada disabilitas, monitoring pasien, diagnosis, pengaturan penggunaan obat, dan pelacakan dokter dan pasien di rumah sakit.

- Deteksi lingkungan
  Deteksi lingkungan yang dapat dilakuan antara lain deteksi gunung berapi, polusi udara, kebakaran hutan, efek rumah kaca, dan deteksi longsor.

- Monitoring struktur
  WSN dapat melakukan deteksi pergerakan bangunan dan infrastruktur seperti jembatan, *flyover*, terowongan dan fasilitas lain tanpa mengeluarkan biaya untuk melakukan dektesi manual dengan mendatangi tempatnya secara langsung.

- Bidang pertanian
  Pada bidang pertanian dapat membantu pengelola pertanian untuk melihat penggunaan air dalam irigasi dan mengelola buangan pertanian mereka.

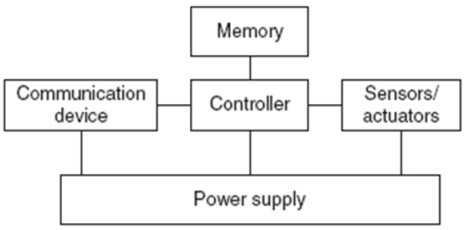


Gambar 2.1: Ilustrasi Pemanfaatan *Wireless Sensor Network*

2.1.2 Node Sensor

Struktur Node Sensor

Setiap node sensor memiliki kemampuan deteksi, komputasi dan komunikasi. Node sensor memiliki lima komponen utama yaitu *controller*, *memory*, *sensor and actuator*, *communication device*, dan *power supply*, (Gambar 2.2). Semua komponen akan bekerja secara seimbang dalam melakukan sensing, komputasi, komunikasi, dan menjaga penggunaan energi seminimal mungkin.



Gambar 2.2: Struktur Node Sensor

Controller

Controller adalah inti utama pada node sensor. *Controller* mengumpulkan data dari sensor dan memproses data tersebut hingga menentukan kapan dan kemana data tersebut dikirim. *Controller* menerima data dari node sensor lain. Pada *controller* biasanya terdapat *microcontroller* atau *microprocessor* yang mengatur dan melakukan komputasi data dari node sensor. *Microcontroller*

ini juga dapat mengurangi penggunaan energi dengan masuk pada *sleep states* yang berarti hanya bagian dari controller saja yang aktif.

Beberapa *microcontroller* yang digunakan dalam *Wireless Sensor Node*:

- Intel StrongARM (32-bit RISC, up to 206 MHz)
- Texas Instrument MSP 430 (16-bit RISC, up to 4 MHz,RAM 2-10 kB)
- Atmel Atmega 128L (8-bit)

Memory

Random Access Memory (RAM) digunakan untuk menyimpan sementara hasil yang didapat dari sensor. RAM juga menyimpan sementara paket dari node sensor lain. Jika *power supply* terjadi masalah atau energi habis maka data pada RAM ini akan hilang. Ini merupakan salah satu kekurangan dari penggunaan RAM. Maka untuk menyimpan kode program digunakanlah *Read Only Memory* (ROM). ROM ini biasa disebut *Electrically Erasable Programmable Read-Only Memory* (EEPROM) atau *Flash Memory*. *Flash memory* dapat menyimpan data jika suatu saat data pada RAM hilang atau energi sudah habis (intermediate storage).

Communication Device

Communication Device digunakan untuk bertukar data antar node sensor. Pada aplikasi WSN, *Radio Frequency(RF)* adalah media komunikasi yang paling relevan saat ini. RF-based mendukung jangkauan yang jauh, data rate yang tinggi dan tidak perlu saling mengetahui posisi antara penerima dan pengirim.

Pada node sensor dibutuhkan *transmitter* untuk mengirim data dan *receiver* untuk menerima data. Kedua hal ini dapat digabung dan disebut dengan *transceiver*. Tugas *transceiver* adalah mengubah aliran *bit* dan mengubahnya menjadi gelombang radio. Selain itu *transceiver* juga dapat mengubah gelombang radio menjadi aliran *bit*.

Sensor dan Actuator

Sensor dan Actuator adalah hal yang penting pada WSN. Tanpa sensor dan actuator maka node sensor tidak berguna dan tidak dapat digunakan. Tabel 2.1 adalah jenis - jenis sensor yang dapat dimiliki node sensor. Sensor dikategorikan menjadi tiga:

1. **Passive, omnidirectional sensors** Sensor ini dapat mengukur kualitas dari lingkungan fisik tempat node sensor tersebut tanpa mengubah lingkungannya. Beberapa sensor dikategori ini *self-powered*, sensor mendapatkan energi yang mereka butuhkan dari lingkungannya. Energi ini digunakan untuk memperkuat sinyal analog. *Omnidirectional* berarti tidak ada arah pada sensor ini. Sensor akan memancarkan sinyalnya ke segala arah. Contoh sensor ini adalah termometer, sensor cahaya, sensor getaran, mikrofon, sensor kelembapan, sensor tekanan udara, sensor deteksi asap, dan lain-lain.

2. **Passive, narrow-beam sensors** Sensor ini memiliki sifat yang sama yaitu pasif, tidak mengubah lingkungannya. Sensor ini dapat melakukan gerakan dan memiliki arah atau daerah pengukuran. Contoh dari sensor ini adalah kamera yang bisa mengukur sesuai dengan arah yang dituju.

3. **Active Sensor** Sensor ini aktif dalam memeriksa lingkungannya. Contoh dari sensor ini adalah sonar, radar atau sensor seismik. Sensor ini menghasilkan gelombang dengan ledakan kecil untuk melakukan deteksi.

Actuator jumlahnya beragam seperti sensor. *Actuator* adalah penerima sinyal dan yang mengubahnya menjadi aksi fisik. Contoh aktuator adalah LED, yang mengubah listrik menjadi cahaya dan motor (motor elektrik) juga mengubah listrik menjadi gerakan.

Tabel 2.1: Jenis - jenis sensor yang dapat dimiliki node sensor

Sensor	Penggunaan
Accelerometer	Pergerakan 2D & 3D untuk objek dan manusia
Acoustic emission sensor	Elastic Waves Generation
Acoustic sensor	Acoustic pressure vibration
Capacitance sensor	Solute Concentration
ECG	Heart Rate
EEG	Brain Electric Activity
EMG	Muscle Activity
Electrical/electromagnetic sensor	Electrical Resistivity
Gyroscope	Angular Velocity
Humidity Sensor	Mendeteksi Humidity
Infrasonic sensor	Gelombang untuk deteksi gempa dan volkanik
Magnetic sensor	Mendeteksi magnetik
Oximeter	Tekanan Oxigenation pada darah
pH sensor	Tingkat Keasaman
Photo acoustic spectroscopy	Gas Sensing
Piezoelectric cylinder	Gas Velocity
Soil moisture sensor	Mengukur tanah
Temperature sensor	Temperatur
Barometer sensor	tekanan air
Passive infrared sensor	Pergerakan infrared
Seismic sensor	Pergerakan Seismik (Gempa)
Oxygen sensor	Oksigen pada darah
Blood flow sensor	Gelombang ultrasonik pada darah

Power Supply

tPower supply atau sumber energi pada WSN bisa berasal dari dua cara yaitu ***storing energy*** dan ***energy scavenging***. *Storing energy* adalah dengan menggunakan baterai sebagai sumber energinya. Baterai yang digunakan dapat diisi ulang maupun yang tidak dapat diisi ulang. *Energy scavenging* digunakan saat membuat WSN yang akan digunakan dalam waktu yang lama. Dibutuhkan energi yang bisa dikatakan tidak terbatas. Salah satu cara *energy scavenging* adalah *photovoltaics*. *Photovoltaics* dapat disebut juga *solar cell* yang memanfaatkan cahaya matahari dan mengubahnya menjadi energi sebagai pembangkit daya. Cara lain yang bisa digunakan adalah pemanfaatan angin dan air untuk mengerakan kincir atau turbin yang akan menghasilkan listrik dan digunakan sebagai sumber energi pada node sensor.

2.1.3 Arsitektur dan Topologi Wireless Sensor Network

Pada WSN biasanya akan terdapat banyak node sensor yang disebar pada suatu tempat. Terdapat satu atau lebih *sink node* atau *base station* dalam area sensing tersebut (Gambar 2.10). *sink node* atau *base station* adalah node sensor yang bertugas untuk mendapatkan data dari node sensor lain. Dalam membuat WSN perlu diperhatikan arsitektur dan topologi yang akan digunakan. Tidak semua topologi jaringan komputer dapat digunakan untuk *Wireless Sensor Network*.

Ada banyak topologi pada jaringan sensor (*sensor network*). Pada jaringan sensor dengan menggunakan kabel, topologi yang sering digunakan adalah topologi *star*, *line*, atau *bus*. Sedangkan

pada jaringan sensor tanpa kabel (WSN), topologi yang biasa digunakan adalah *star*, *tree*, atau *mesh*.

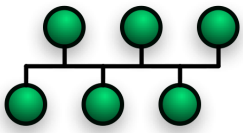
Topologi Point-to-Point

Topologi *Point-to-Point* adalah topologi yang menghubungkan dua titik (Gambar 2.3). Topologi *Point-to-Point* dibagi menjadi dua yaitu *permanent point-to-point* dan *switched point-to-point*. *Permanent point-to-point* adalah koneksi perangkat keras antara dua titik dan tidak dapat diubah. *Switched point-to-point* adalah koneksi *point-to-point* yang dapat berpindah antara node yang berbeda.

Gambar 2.3: Topologi Point-to-Point

Topologi Bus

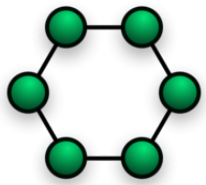
Topologi Bus seperti pada Gambar 2.4 akan terdiri dari node-node dan sebuah jalur. Setiap node akan terhubung dengan jalur yang sama. Untuk mengirim data atau komunikasi akan dilakukan bergantian antar node. Kekurangan dari topologi bus ini adalah jika suatu saat jalur / bus ini mengalami kerusakan maka setiap node tidak dapat berkomunikasi lagi.



Gambar 2.4: Topologi Bus

Topologi Ring

Pada Topologi *Ring* node akan disusun dengan bentuk melingkar (Gambar 2.5). Setiap node akan terkoneksi dengan dua node lain. Transfer data terjadi dengan cara data akan berjalan dari satu node ke node lain mengikuti jalur melingkar tersebut hingga menemukan node tujuan yang tepat. Topologi ini mudah untuk diimplementasikan tapi kekurangan dari topologi ring adalah saat ada node yang rusak maka perlu biaya lebih untuk memperbaikinya. Biasanya untuk menangani kegagalan komunikasi akibat node yang rusak, akan di atur komunikasi node tidak hanya satu arah tetapi dapat ke arah sebaliknya.



Gambar 2.5: Topologi Ring

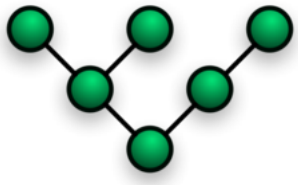
Topologi Star

Topologi *Star* terdiri dari satu node yang berada di tengah biasanya berupa *hub* atau *switch* seperti pada Gambar 2.6. Setiap node akan terkoneksi dengan node yang berada di tengah ini. Saat node akan berkomunikasi dengan node lain, node tersebut harus mengirimkan data tersebut ke node yang ada ditengah dahulu dan node yang berada ditengah ini akan meneruskan data tersebut ke node tujuan. Yang paling penting pada topologi ini adalah node yang berada di tengah, karena semua komunikasi harus melalui node tersebut. Jika node tengah mengalami kerusakan maka tidak akan terjadi komunikasi antar node pada jaringan tersebut.

Gambar 2.6: Topologi Star

Topologi Tree

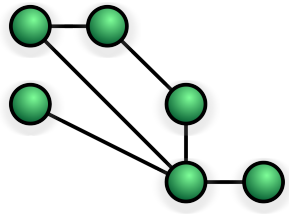
Pada Topologi Tree node-node akan disusun secara hierarki dengan satu node yang berada pada level paling atas sebagai *root node* (Gambar 2.7). *Root node* akan terhubung dengan satu atau lebih node level dibawahnya. Dengan Topologi Tree lebih mudah untuk melakukan identifikasi dan meminimalisir kesalahan, namun jika *tree* sudah sangat besar / *level tree* sudah sangat banyak maka akan sulit untuk mengaturnya.



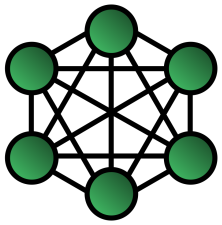
Gambar 2.7: Topologi Tree

Topologi Mesh

Topologi *Mesh* dibagi menjadi dua yaitu *partially connected mesh* dan *fully connected mesh*. Pada *partially connected mesh* (Gambar 2.8), node akan terhubung dengan lebih dari satu node. Pada *fully connected mesh* (Gambar 2.9), setiap node akan terhubung dengan semua node lain pada jaringan tersebut.

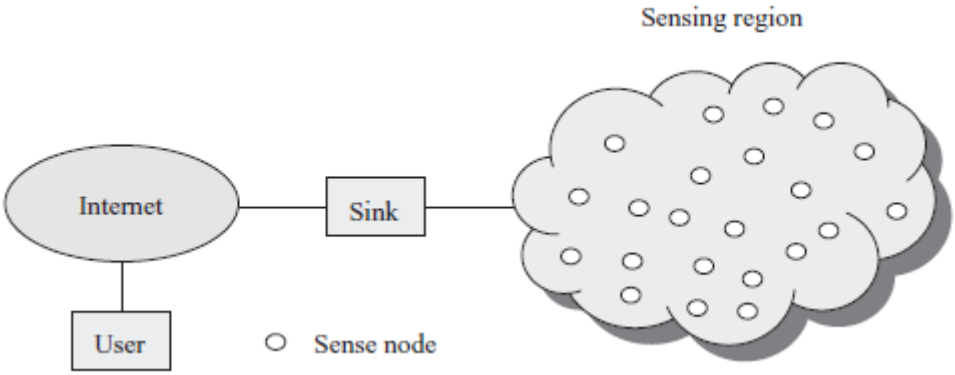


Gambar 2.8: Topologi Partially Connected Mesh



Gambar 2.9: Topologi Fully Connected Mesh

Arsitektur yang biasanya dipakai pada WSN adalah **arsitektur flat / peer-to-peer** dan **arsitektur hirarkikal**. Selain itu dalam membangun WSN perlu juga diperhatikan jalur komunikasi yang digunakan untuk menghubungkan antar node sensor saat transfer data. Untuk area *sensing* yang tidak terlalu luas dan hanya menggunakan sedikit node sensor dapat menggunakan cara komunikasi ***single hop***. Sedangkan untuk daerah yang luas dan memerlukan banyak node sensor dapat menggunakan cara komunikasi ***multi hop***.



Gambar 2.10: Arsitektur Wireless Sensor Network

Single-Hop dan Multi-Hop

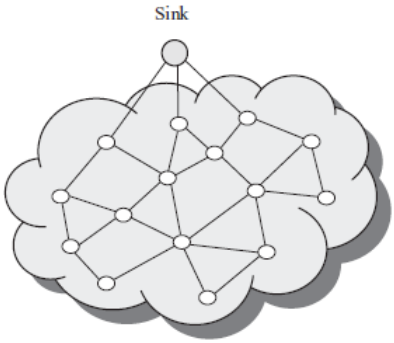
Untuk mengirim data ke *sink node* setiap node sensor dapat menggunakan *single-hop long-distance transmission*. *Single-hop long-distance* ini berarti setiap node sensor akan mengirimkan data ke sink node hanya satu kali lompatan walaupun jarak antara sink node dengan node sensor itu sangat jauh. Dalam jaringan sensor, penggunaan energi paling besar adalah saat melakukan komunikasi dibandingan saat sensing. Penggunaan energi akan semakin bertambah jika jarak sink dan node sensor semakin jauh. Untuk menangani masalah tersebut muncul protokol *multi-hop*.

Pada protokol *multi-hop*, node sensor akan disusun saling berdekatan dan terhubung dengan yang lain. Jadi saat akan berkomunikasi dengan *sink node*, node sensor harus mengirimkan data

tersebut ke node sensor tetangganya dan diteruskan hingga sampai ke *sink node*. Karena jarak yang saling berdekatan maka penggunaan energi dapat efektif. *Single-hop* dan *multi-hop* ini dapat digunakan dengan topologi flat maupun hirarkikal sesuai dengan kebutuhan sistem.

Arsitektur Flat / Peer-to-Peer

Pada arsitektur flat, setiap node sensor memiliki peran atau *role* yang sama dalam melakukan *sensing*. Secara fungsional hanya terdapat dua macam node sensor pada arsitektur flat, yaitu *source node* dan *sink node*. Untuk mendapatkan data dilakukan dengan cara *sink node* melakukan pengiriman data ke semua node sensor pada area *sensing* dengan cara *flooding* dan hanya node sensor yang sesuai yang akan merespon *sink node*. Setiap node sensor mengirimkan data ke *sink node* dengan *multi hop* dan melalui node tetangganya yang terhubung dengannya untuk meneruskan data (Gambar **??**).

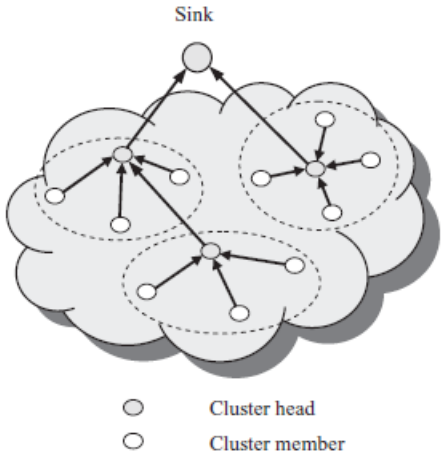


Gambar 2.11: Arsitektur flat pada *Wireless Sensor Network*

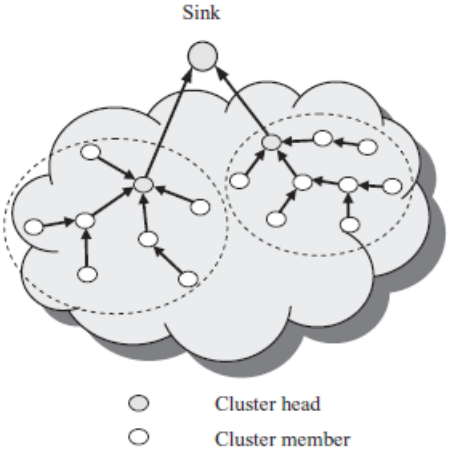
Arsitektur Hirarkikal

Pada arsitektur hirarkikal, semua node sensor dikelompokan ke dalam cluster-cluster. Terdapat *cluster head* pada setiap cluster. *Cluster head* ini yang mengumpulkan data dari setiap node sensor di bawahnya dan meneruskan data yang telah diterima ke *base station* atau *sink node*. Hal yang perlu diperhatikan pada arsitektur hirarkikal adalah pemilihan node sensor sebagai *cluster head* dan node sensor yang melakukan sensing. Penggunaan energi yang paling besar dalam WSN ini adalah saat melakukan komunikasi yaitu saat mengirimkan data ke node sensor lain. Maka untuk node sensor yang memiliki energi kecil dapat digunakan untuk *sensing*, karena node sensor ini hanya melakukan komunikasi ke *cluster head*. *Cluster head* harus memiliki energi atau daya yang lebih banyak, karena *cluster head* akan bertugas menerima hasil sensing node sensor di bawahnya dan meneruskan data ke *sink node*.

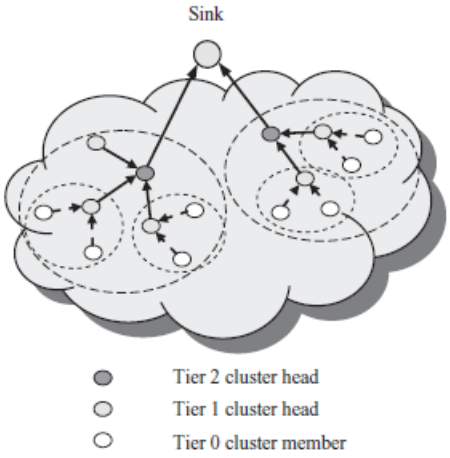
Masalah yang utama pada clustering ini adalah pemilihan *cluster head* dan bagaimana cara mengatur setiap cluster. Terdapat beberapa cara untuk membuat clustering ini. Bedasarkan jarak antara *cluster head* dengan cluster member, dapat dibuat clustering dengan *single-hop* atau *multi-hop* seperti pada Gambar **??** dan Gambar **??**. Sedangkan jika berdasarkan jumlah *tier* atau tingkat dapat dibangun *clustering single tier* atau *multi tier* (Gambar **??**).



Gambar 2.12: Arsitektur hierarki pada *Wireless Sensor Network* dengan *single hop* terhadap *Cluster Head*



Gambar 2.13: Arsitektur hierarki pada *Wireless Sensor Network* dengan *multi hop*



Gambar 2.14: Clsutering dengan multi tier

2.1.4 Sistem Operasi

Setiap node sensor memerlukan sistem operasi (OS) untuk mengontrol perangkat keras dan perangkat lunak. Sistem operasi tradisional tidak dapat digunakan pada WSN. Pada sistem operasi tradisional digunakan untuk mengatur proses, memori, CPU, dan sistem berkas. Terdapat beberapa hal yang harus ditangani oleh sistem operasi dalam WSN yaitu:

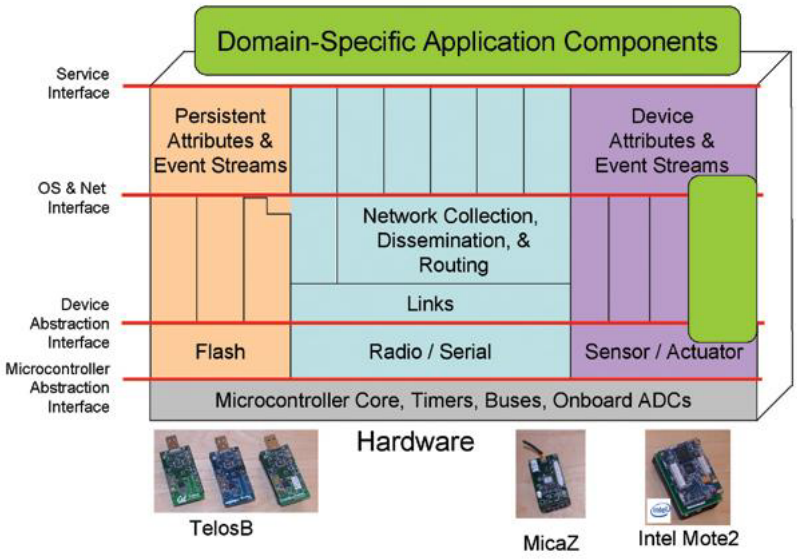
1. WSN memerlukan *real-time scheduler*. Data yang didapat harus segera dikirim atau diproses.
2. Pengaturan memori karena memori pada WSN sangat kecil.
3. Pengaturan data yang efisien terkait dengan *microprocessor* dan memori yang terbatas
4. Mendukung kode pemrograman yang efisien dan *reliable* karena dapat terjadi perubahan kode saat implementasi.
5. Mendukung pengaturan sumber daya untuk menambah waktu hidup dari node sensor dan meningkatkan performa dengan *sleep time* atau *wake up time* saat terdapat interupsi dari lingkungan.
6. Mendukung antarmuka untuk pemrograman dan antarmuka perangkat lunak.

Beberapa sistem operasi yang umum digunakan pada WSN antara lain :

1. TinyOS
2. Contiki
3. LiteOS
4. PreonVM

TinyOS

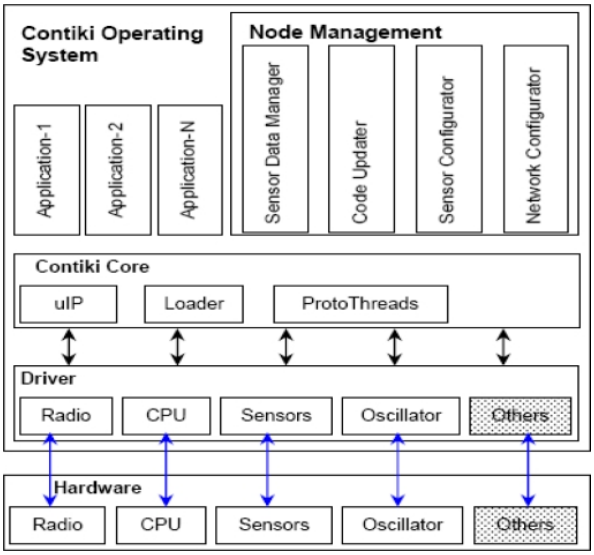
TinyOS adalah sistem operasi *open-source* yang digunakan pada WSN. TinyOS dapat menjalankan program dengan memori yang sangat kecil. Ukurannya hanya 400 Byte. Komponen *library* TinyOS terdiri dari protokol jaringan, layanan distribusi sensor, *driver sensor*, dan perangkat lunak pengamatan data sensor yang dapat digunakan untuk melakukan monitoring jaringan sensor. Gambar **??** adalah arsitektur pada TinyOS.



Gambar 2.15: Arsitektur TinyOS

Contiki

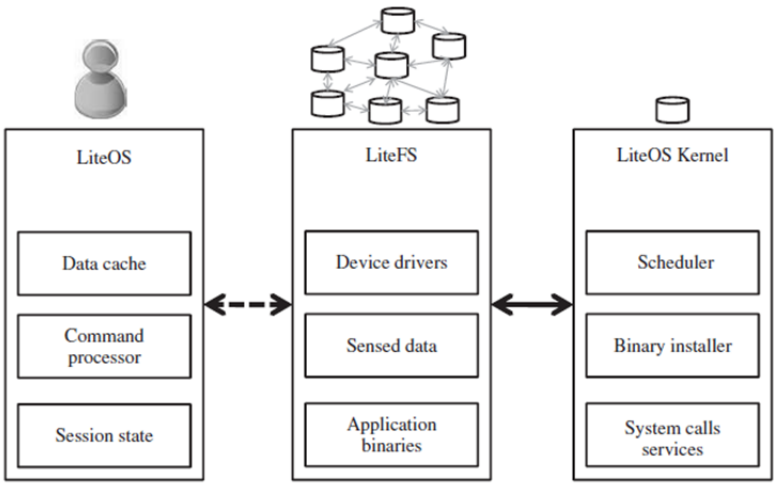
Contiki adalah sistem operasi *open-source* dengan Bahasa Pemrograman C yang digunakan pada WSN. Pengaturan Contiki hanya memerlukan 2KB dari RAM dan 40KB dari ROM. Fitur yang dimiliki oleh Contiki antara lain: *multitasking*, *multithreading*, jaringan TCP/IP, IPv6, GUI, *Web Browser*, *Web Server*, telnet, dan komputasi jaringan virtual. Gambar **??** adalah arsitektur pada Contiki



Gambar 2.16: Arsitektur Contiki

LiteOS

LiteOS adalah sistem operasi mirip UNIX yang didisain untuk WSN. Tujuan dibuat LiteOS adalah membuat sistem operasi yang mirip dengan UNIX agar lebih lebih familiar dengan paradigma pemrograman UNIX. Pada LiteOS terdapat sistem berkas yang hiearki, dan mendukung Bahasa Pemrograman LiteC++ dan UNIX Shell. LiteOS dapat digunakan untuk MicaZ yang memiliki 8 Mhz CPU, 128 byte flash, dan 4KB RAM. LiteOS memiliki tiga komponen utama yaitu: LiteShell, LiteFS, dan Kernel. Gambar **??** adalah arsitektur pada LiteOS



Gambar 2.17: Arsitektur LiteOS

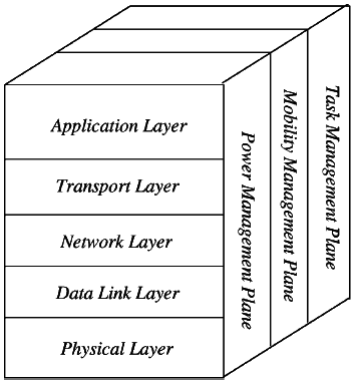
Perbandingan dari TinyOS, Contiki, dan LiteOS dapat dilihat pada Tabel **??** berikut ini:

Tabel 2.2: Tabel Perbandingan Sistem Operasi

Operating System	Programming Paradigm	Scheduling
TinyOS	Event-based	FIFO
Contiki	Predominant event-based	FIFO
LiteOS	Thread-based	Priority-based scheduling with optional round-robin support

2.1.5 Protokol Stack pada Wireless Sensor Network

WSN memiliki lima layer protokol: physical layer, data link layer, network layer, transport layer, dan application layer, seperti pada Gambar **??**.



Gambar 2.18: Layer pada *Wireless Sensor Network*

Physical Layer

Physical layer bertanggung jawab untuk mengubah bit stream dari data link layer menjadi sinyal agar bisa dilakukan transmisi melalui media komunikasi yang terdapat (transceiver). Pemilihan media dan frekuensi adalah hal yang penting dalam komunikasi antar node sensor. Salah satu cara yang bisa digunakan adalah dengan menggunakan *Radio frequency* (RF). Jaringan sensor memerlukan biaya yang kecil, ukuran yang kecil, dan penggunaan daya yang kecil untuk tranceivernya. Karena itu banyak yang menggunakan Radio Frequency (RF) untuk desain perangkat keras node sensornya.

Data Link Layer

Data link layer bertanggung jawab untuk melakukan *multiplexing* pada aliran data, membentuk data frame, mendeteksi data frame, *medium access*, dan mengatur kesalahan saat transmisi data. Fungsi paling penting data link layer adalah *Medium Access Control* (MAC). Protokol MAC menentukan kapan node sensor mengakses media untuk mengirim data, melakukan kontrol dan mengatur paket ke node sensor lain. Hal itu dilakukan agar tidak terjadi paket yang bertabrakan.

Network Layer

Network layer bertanggung jawab untuk routing dari node sensor ke sink node. Pada WSN node sensor tersebar pada suatu tempat untuk melakukan sensing. Data sensing tersebut harus dikirimkan ke *sink node* untuk diolah. Dalam mengirimkan data tersebut dapat menggunakan *single-hop* atau *multi hop*. Dalam mengirim data diperlukan protokol routing yang hemat energi.

Transport Layer

Secara umum transport layer bertanggung jawab untuk pengiriman data yang *reliable* antara node sensor dan *sink node*. Protokol transport yang biasa tidak bisa diterapkan pada WSN tanpa modifikasi. Setiap jaringan sensor memiliki fungsi khusus. Untuk aplikasi yang berbeda memerlukan kebutuhkan reliabilitas yang berbeda. Pengiriman data pada WSN dibagi menjadi dua yaitu: ***downstream*** dan ***upstream***. ***Upstream*** berarti node sensor mengirimkan hasil sensing ke sink node. ***Downstream*** berarti data berasal dari sink node contohnya, kueri, dan perintah-perintah yang dikirimkan ke setiap node sensor. Aliran data yang *reliable* untuk kedua jenis pengiriman ini mungkin berbeda. Pada *upstream*, *reliable* data dapat ditoleransi karena sensor akan melakukan sensing terus menerus dan dapat terjadi pengulangan data sehingga data yang hilang tadi dapat dikoreksi. Sedangkan pada *downstream* memerlukan 100% pengiriman data yang *reliable*.

Application Layer

Application layer meliputi berbagai macam protokol yang ada pada layer ini untuk menjalankan berbagai macam aplikasi seperti *query dissemination*, *node localization*, *time synchronization*, dan *network security*. Protokol yang ada pada layer ini antara lain:

1. Sensor management protocol (SMP) adalah protokol untuk melakukan pertukaran lokasi data, sinkornisasi node sensor, mengatur ulang node sensor, dan menyimpan status dari node sensor.
2. Sensor query and data dissemination protocol (SQDDP) adalah protokol yang mendukung antarmuka aplikasi untuk memasukan query, merespon query, dan mengumpulkan respon.
3. Sensor query and tasking language (SQTL) adalah protokol yang mendukung bahasa pemrograman pada WSN.

2.2 Reliable Data Transfer di WSN

WSN terdiri dari *sink node* dan banyak node sensor. Saat melakukan sensing dan mengirim data ke *sink node*, semua node sensor akan mengirimkan data melalui media yang sama. Hal ini dapat membanjiri jaringan dengan data tersebut dan menyebabkan *congestion* atau kemacetan pada jaringan yang berakibat *data loss* atau hilang. Untuk menangani *data loss*, maka dalam membangun WSN adalah salah satu yang diperlukan adalah protokol yang menangani *Reliability*. *Reliability* berarti memastikan data yang dikirim dari setiap node sensor diterima oleh *base station* secara lengkap dan sesuai dengan urutan pengiriman.

2.2.1 Jenis Reliability

Berdasarkan tingkat, reliability ada dua macam yaitu:

1. Packet Reliability
2. Event Reliability

Packet Reliability berarti paket yang dikirim harus sampai kepada tujuan (*base station*) secara utuh. *Packet reliability* ini membutuhkan *acknowledge* dari node sensor. Tantangan yang harus dihadapi dari *packet reliability* adalah dalam mengirim ulang paket yang hilang akan menghabiskan energi atau daya.

Event Reliability berarti hanya data hasil sensing yang akan dikirim ke *base station*. Pada *Event reliability* tidak membutuhkan *acknowledge*. Karena data yang hilang itu tidak banyak berpengaruh sehingga tidak dibutuhkan pengiriman ulang (*retransmission*) data.

Berdasarkan arah, *reliability* ada dua macam yaitu:

- Upstream Reliability
- Downstream Reliability

Upstream Reliability adalah komunikasi dari node sensor ke *sink node*. Banyak protokol yang mendukung *upstream reliability*. Pengiriman data yang dilakukan adalah *unicast* yang berarti hanya dari satu titik ke titik lain.

Downstream Reliability adalah adalah komunikasi dari *sink node* ke node sensor. Pengiriman data biasanya dilakukan dengan cara *broadcast* ke semua node sensor.

Pada protokol transport tradisional dikenal istilah TCP dan UDP, namun keduanya tidak bisa diimplementasikan pada WSN. Namun, protokol transport tersebut bisa dibuat dengan beberapa pertimbangan seperti:

- WSN membutuhkan mekanisme untuk mengembalikan paket yang hilang seperti *acknowledge*
- Proses awal dalam membangun koneksi seperti *handshake* harus disederhanakan karena akan membuang banyak daya
- Protokol harus dapat menangani *congestion*
- Protokol harus dapat adil terhadap setiap node sensor seperti pembagian penggunaan jaringan (*bandwidth*)

Retransmission dapat dilakukan dengan *End-to-End Retransmission* dan *Hop-by-Hop Retransmission (Link Level Retransmission)*. Pada ***End-to-End transmission*** dan terjadi *data loss*, maka pengirim harus mengirim ulang semua paket dan akan menghabiskan lebih banyak daya. *End-to-End Retransmission* adalah salah satu metode yang digunakan di Internet. Cara ini dapat memastikan *reliable* data tanpa harus mengetahui apa yang terjadi di tengah jaringan. Pada *End-to-End Retransmission* diperlukan *handshake* seperti pada komunikasi jaringan komputer. Pada awalnya data dikirim sebagai permintaan transfer. Jika penerima (*receiver*) memiliki cukup RAM, dan layer aplikasi dapat menerima data tersebut, maka *receiver* mengirimkan *acknowledge* untuk permintaan data tersebut. Saat koneksi sudah terbentuk, data yang sebenarnya dapat dikirim. Data tersusun dari beberapa *round*. Setiap *round*, pengirim (*sender*) mengirim paket yang hilang pada round sebelumnya. Diakhir setiap round, *receiver* mengirimkan *acknowledge* kepada *sender* yang berisi informasi paket yang hilang. *Sender* menerima *acknowledge* tersebut dan mengirim paket yang hilang tersebut. Untuk round pertama terdapat pengecualian karena semua pake pada round sebelumnya tidak ada (belum dikirim).

Sedangkan pada ***hop-by-hop transmission*** dilakukan antara node sensor dengan node sensor tetangganya. Jadi saat terjadi *data loss* maka pengiriman ulang lebih sedikit dalam menghabiskan daya. *Hop-by-hop* ini membutuhkan *buffer* atau penyimpanan sementara pada setiap node sensor dan lebih efektif dilakukan pada topologi WSN *multi-hop*. Buffer ini digunakan untuk menyimpan data sementara hingga mendapatkan *acknowledge* dari hop berikutnya.

2.2.2 Jenis - jenis Acknowledge

Dalam mencapai reliability dapat digunakan acknowledge saat melakukan pengiriman data. Terdapat empat jenis acknowledge yang dapat digunakan, yaitu:

1. Explicit Acknowledge (eACK) : Setiap berhasil mengirim pesan dibutuhkan acknowledge.

2. Negative Acknowledge (nACK) : Penerima memberitahu pengirim bahwa ada paket yang hilang
3. Implicit Acknowledge (iACK) : Setelah sender mengirim pesan, ia akan memastikan paket data dikirim ke tetangganya memberikan acknowledge
4. Selective Acknowledge (sACK) : Hanya paket yang hilang dari sebuah pesan yang akan dikirim ulang

2.2.3 Protokol Transport yang Reliable

Ada banyak protokol untuk memastikan *reliability* pada WSN. Beberapa protokol mendukung *upstream,* dan beberapa protokol mendukung *downstream.* Hanya ada sedikit protokol yang dapat mendukung keduanya. Selain itu ada protokol yang memiliki fokus utama untuk menangani *reliability* saja dan ada yang menangani *congestion* sekaligus *reliability.*

Beberapa protokol transport yang sering digunakan antara lain:

- GARUDA
- Event-to-Sink Reliable Transport (ESRT)
- Reliable Multi Segment Transport (RMST)
- Pump Slowly Fetch Quickly (PSFQ)
- Asymmetric Reliable Transport (ART)
- Price Oriented Reliable Transport (PORT)
- Delay Sensitive Transport (DST)

GARUDA

GARUDA adalah salah satu protokol *downstream* yang menggunakan pemulihan data *hop-by-hop.* GARUDA menggunakan NACK dan tidak menangani *congestion.* GARUDA menggunakan mekanisme WFP (Wait for First Packet) pulse transmission.

ESRT

ESRT adalah protokol *upstream* yang lebih banyak digunakan untuk mengirim event dibandingkan paket data. Pada ESRT tingkat pengiriman pada setiap node sensor bergantung pada tingkat reliabilitas di *sink node* dan dengan status jaringan tersebut (terdapat kemacetan atau tidak). ESRT adalah protokol pertama yang menangani *congestion control* dan *reliability* sekaligus. Pada ESRT setiap node sensor mendeteksi *congestion* berdasarkan peningkatan *buffer* lalu menambahkan N bit pada *header* sebuah paket dan meneruskannya ke sink node. Berdasarkan paket yang diterima oleh *sink node*, dapat diketahui keadaan dari jaringan dan juga *reliability* suatu paket untuk menentukan jumlah paket yang telah berhasil diterima pada periode tertentu. ESRT tidak menggunakan ACK/NACK karena menyebabkan penggunaan energi yang sangat besar.

RMST

RMST adalah protokol *upstream* yang berbasis *selective NACK.* RMST dibuat pada puncak dari suatu jalur dari node sensor ke sink node. RMST ini menggunakan *timer* untuk mendeteksi paket yang *loss.* RMST ini tidak menangani *congestion control* dan masalah terkait penggunaan energi.

PSFQ

PSFQ adalah protokol *downstream*. Cara kerja PSFQ adalah dengan memasukan paket kedalam jaringan (*Pump Operation*). Saat terjadi *data loss*, protokol ini akan menjalankan pemulihan *hop-by-hop*. PSFQ menggunakan NACK. Protokol ini tidak mendukung *congestion control* dan *paket loss* secara satuan.

ART

ART adalah protokol yang berbasis pada *event* dan *query reliability*. ART merupakan protokol *bidirectional* pertama yang sudah mendukung *congestion control*. Terdiri dari kumpulan node yang disebut *essential node* yang tersebar pada suatu area dan beberapa node yang disebut *non-essential node* yang terlibat pada pengiriman data dan *congestion control*.

PORT

PORT adalah protokol *upstream* yang berbasis pada *event reliability* dengan penggunaan energi seminimal mungkin. PORT mendukung mekanisme dengan energi yang efisien disertai *congestion control*. Selain itu PORT juga adalah protokol yang mendukung komunikasi *End-to-End*. PORT membutuhkan *sink node* untuk mengatur aliran data. Kekurangannya adalah tidak ada pemulihan paket yang hilang.

DST

DST adalah tambahan dari ESRT. Tujuan dari DST adalah dicapai *reliability* pada *sink node*. Pada DST terdapat aturan *Time Critical Event First* (TCEF). Pada TCEF, data paket dengan deadline minimum diberikan prioritas untuk melakukan retransmission. DST dapat bekerja dengan baik pada satu event, namun pada banyak event akan menjadi lebih kompleks.

Lebih singkat setiap protokol tersebut dapat dilihat pada Tabel **??**

Tabel 2.3: Tabel Perbandingan Protokol

Protokol	Arah Reliability	Tingkat Reliability	Retransmission Mechanism	Type of ACK	Congestion M
GARUDA	Downstream	Packet	Hop-By-Hop	NACK	No
ESRT	Upstream	Event	End-to-End	-	Yes
RMST	Upstream	Packet	Hop-By-Hop	NACK	No
PSFQ	Downstream	Packet	Hop-By-Hop	NACK	No
ART	Both	Event	End-to-End	-	Yes
PORT	Upstream	Event	Hop-By-Hop	-	Yes
DST	Upstream	Event	End-to-End	-	No

2.3 Pengembangan Pemrograman pada Wireless Sensor Network

2.3.1 Tabel

Berikut adalah contoh pembuatan tabel. Penempatan tabel dan gambar secara umum diatur secara otomatis oleh LaTeX, perhatikan contoh di file bab2.tex untuk melihat bagaimana cara memaksa tabel ditempatkan sesuai keinginan kita.

Perhatikan bawa berbeda dengan penempatan judul gambar gambar, keterangan tabel harus diletakkan di atas tabel!! Lihat Tabel **??** berikut ini:

Tabel 2.4: Tabel contoh

	v_{start}	$\mathcal{S}_1$	v_{end}
τ_1	1	12	20
τ_2	1		20
τ_3	1	9	20
τ_4	1		20

Tabel **??** dan Tabel **??** berikut ini adalah tabel dengan sel yang berwarna dan ada dua tabel yang bersebelahan.

Tabel 2.5: Tabel bewarna(1)

	v_{start}	$\mathcal{S}_2$	$\mathcal{S}_1$	v_{end}
τ_1	1	5	12	20
τ_2	1	8		20
τ_3	1	2/8/17	9	20
τ_4	1			20

Tabel 2.6: Tabel bewarna(2)

	v_{start}	$\mathcal{S}_1$	$\mathcal{S}_2$	v_{end}
τ_1	1	12	5	20
τ_2	1		8	20
τ_3	1	9	2/8/17	20
τ_4	1			20

2.3.2 Kutipan

Berikut contoh kutipan dari berbagai sumber, untuk keterangan lebih lengkap, silahkan membaca file referensi.bib yang disediakan juga di template ini. Contoh kutipan:

- Buku: [**?**]
- Bab dalam buku: [**?**]
- Artikel dari Jurnal: [**?**]
- Artikel dari prosiding seminar/konferensi: [**?**]
- Skripsi/Thesis/Disertasi: [**?**] [**?**] [**?**]
- Technical/Scientific Report: [**?**]
- RFC (Request For Comments): [**?**]
- Technical Documentation/Technical Manual: [**?**] [**?**] [**?**]
- Paten: [**?**]
- Tidak dipublikasikan: [**?**] [**?**]
- Laman web: [**?**]
- Lain-lain: [**?**]

2.3.3 Gambar

Pada hampir semua editor, penempatan gambar di dalam dokumen LaTeX tidak dapat dilakukan melalui proses *drag and drop*. Perhatikan contoh pada file bab2.tex untuk melihat bagaimana cara menempatkan gambar. Beberapa hal yang harus diperhatikan pada saat menempatkan gambar:

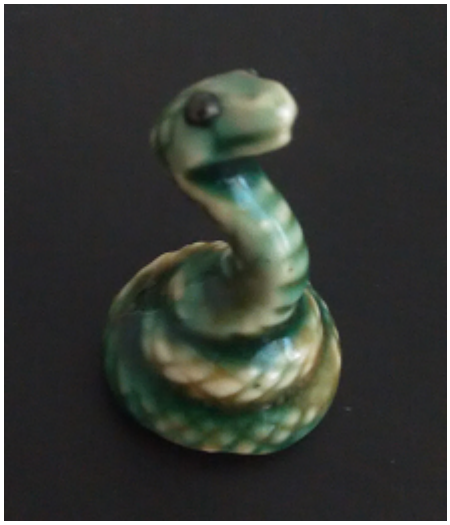
- Setiap gambar **harus** diacu di dalam teks (gunakan *field* LABEL)
- *Field* CAPTION digunakan untuk teks pengantar pada gambar. Terdapat dua bagian yaitu yang ada di antara tanda [dan] dan yang ada di antara tanda { dan }. Yang pertama akan muncul di Daftar Gambar, sedangkan yang kedua akan muncul di teks pengantar gambar. Untuk skripsi ini, samakan isi keduanya.
- Jenis file yang dapat digunakan sebagai gambar cukup banyak, tetapi yang paling populer adalah tipe PNG (lihat Gambar **??**), tipe JPG (Gambar **??**) dan tipe PDF (Gambar **??**)
- Besarnya gambar dapat diatur dengan *field* SCALE.
- Penempatan gambar diatur menggunakan *placement specifier* (di antara tanda [dan] setelah deklarasi gambar. Yang umum digunakan adalah **H** untuk menempatkan gambar **sesuai** penempatannya di file .tex atau **h** yang berarti "kira-kira" di sini.
  Jika tidak menggunakan *placement specifier*, LaTeX akan menempatkan gambar secara otomatis untuk menghindari bagian kosong pada dokumen anda. Walaupun cara ini sangat mudah, hindarkan terjadinya penempatan dua gambar secara berurutan.
  - Gambar **??** ditempatkan di bagian atas halaman, walaupun penempatannya dilakukan setelah penulisan 3 paragraf setelah penjelasan ini.
  - Gambar **??** dengan skala 0.5 ditempatkan di antara dua buah paragraf. Perhatikan penulisannya di dalam file bab2.tex!
  - Gambar **??** ditempatkan menggunakan *specifier* **h**.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris



Gambar 2.19: Gambar *Serpentes* dalam format png

at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Gambar 2.20: Ular kecil

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Gambar 2.21: *Serpentes* jantan

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

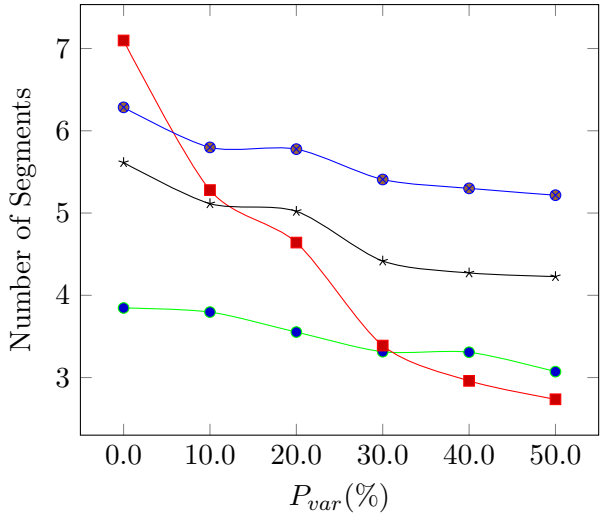
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

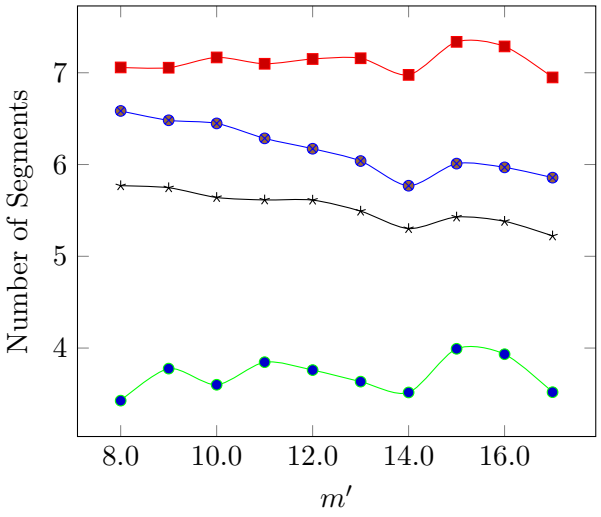
LAMPIRAN B

HASIL EKSPERIMEN

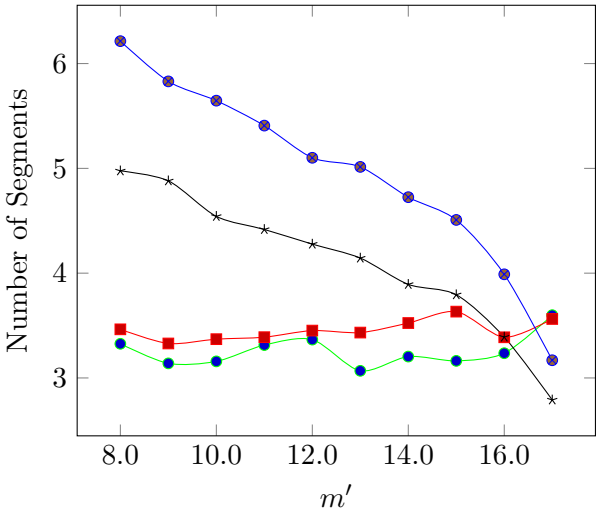
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



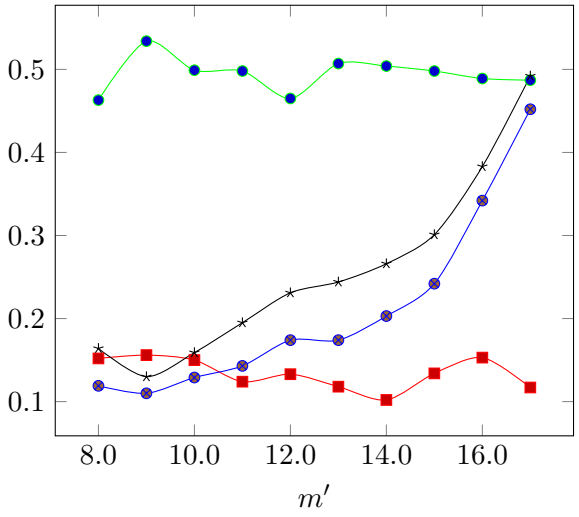
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4