CG4002

Computer Engineering Capstone Project

Lecture

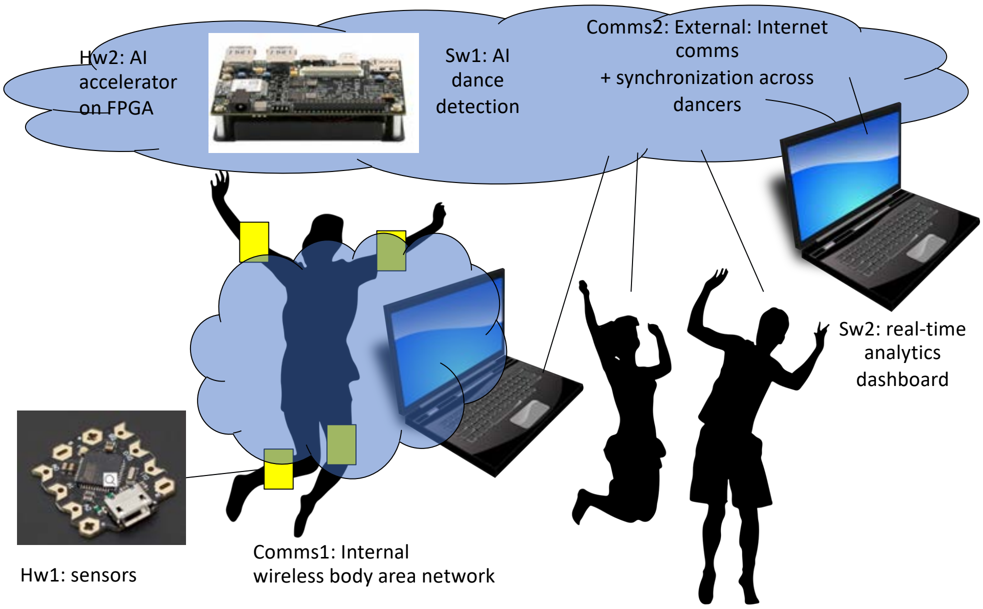
External communications: Secure wireless Internet communications and clock synchronization

Peh Li Shiuan, Professor, Computer Science &
Electrical and Computer Engineering (Courtesy)
peh@nus.edu.sg

NUS National University of Singapore
School of Computing

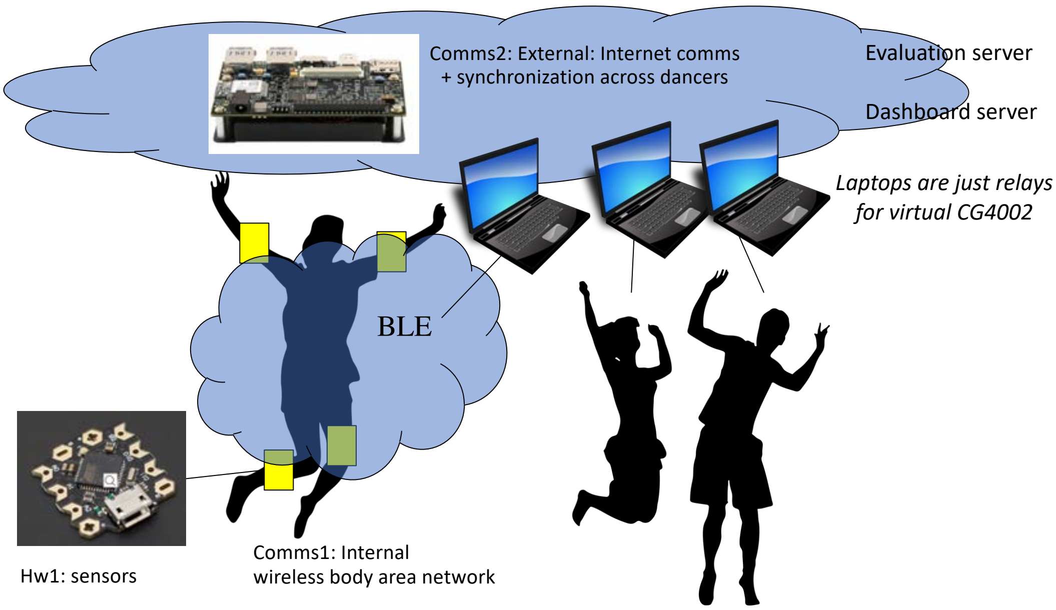
1

Virtual CG4002 in 6 parts:



2

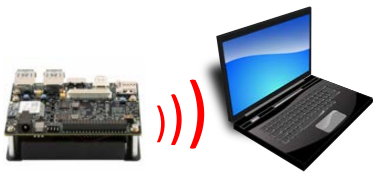
Comms External:



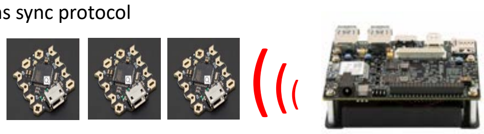
3

Week 7: Individual subcomponent test (20% indiv)

- Comms2: External
  - Walkthrough and demo secure socket comms between laptop and Ultra96, and Ultra96 and evaluation server



  - Walkthrough protocol for estimating dancer synchronization delay (Start of dance move between fastest and slowest dancer)
    - Demo at laptop locally with dummy processes
    - After Week 7, before Week 11:
      - Work with Comms Internal to incorporate comms sync protocol



4

Secure wireless communications between system and server

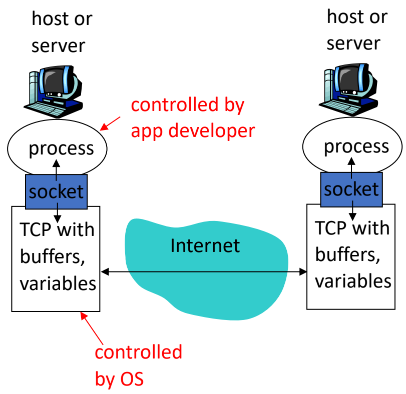
5

Introduction to sockets

- Interface between the application layer and transport layer
- OS-controlled interface (a "door") into which application process can both send and receive messages

- Addressing
  - Host address + process identifier
  - Eg. IP address + port number
    - Eg HTTP – port 80, SMTP – port 25, ftp – port 21

[Slide from CS3103, Dr Anand Bhojan]

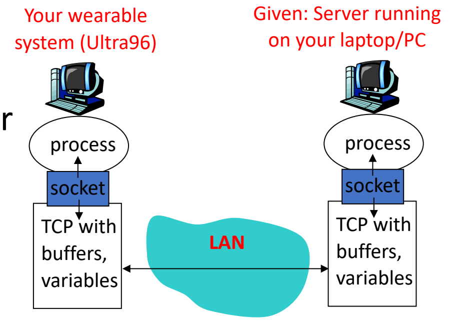


Socket: endpoint for comm. A combination of host IP address and port number.

Image Source: Chapter 2 - Kurose& Ross

6

Sockets in CG3002



- Interface between the application layer and transport layer
- OS-controlled interface (a "door") into which application process can both send and receive messages

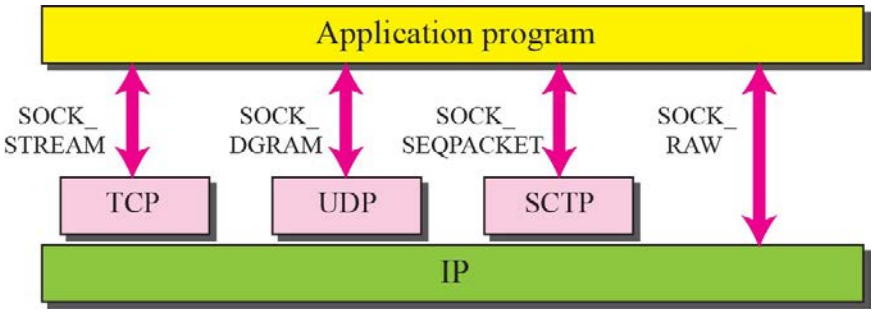
- Addressing
  - Host address + process identifier
  - Eg. IP address + port number
    - IP address: IP address of server on LAN
    - Port number: You define

Socket: endpoint for comm. A combination of host IP address and port number.

Image Source: Chapter 2 - Kurose& Ross

7

Socket Types

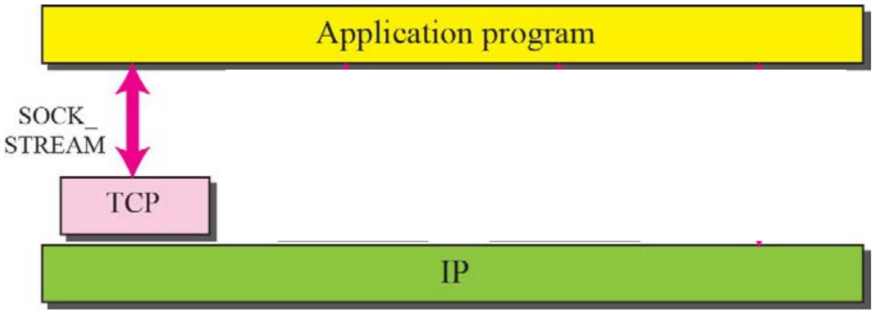


- Types of transport service via socket:
  - unreliable datagram
  - reliable, byte stream-oriented
- Lower level protocols and network interfaces can be accessed through 'RAW Socket'.
- The new SCTP socket provides multiple types of service

[Slide from CS3103]

8

Socket Types [in CG4002]



- TCP: **reliable, byte stream-oriented**
- **S**erver process must be running first
- Server must have created socket (door) that welcomes client's contact
- Client creates client-side local TCP socket specifying IP address, port number of server process to bind to the server
- When client creates socket: client TCP establishes connection to server TCP

9

Socket API in Python3 (laptop/Ultra96 – you)

- TCP socket: SOCK_STREAM
- Server IP address and socket number
  - Depends on the network you run
- Socket library: import socket
- Creating a socket: sock = socket.socket(…)
- Connecting to a socket: sock.connect()
- Sending using a socket: sock.sendall()
- Receiving from a socket: sock.recv()
- Closing a connection: sock.close()

10

Socket API in Python (on evaluation server – provided, run on your laptop/PC)

- Socket library: import socket
- Creating a socket: sock = socket.socket(…)
- Binds server address to socket: sock.bind(server_address)
- Listens to the socket for client messages: sock.listen()
- Accepts client connection: sock.accept()
- Receives data from socket: data = connection.recv()
- Sends status after handshaking: connection.send(status.encode())

11

The big bad NUS wolf/firewall ☺

Your Ultra96 FPGA boards can be accessed remotely:

1. You need to ssh into sunfire (for students) :

ssh -l nusnet_id sunfire.comp.nus.edu.sg

2. From Sunfire, you can access the boards:

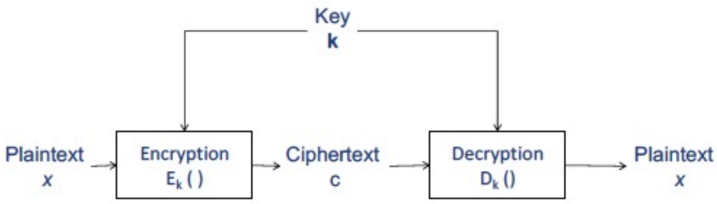
ssh -l xilinx <IP address of your group's board>

- How do you tunnel through the NUS firewall so you can communicate between laptop and Ultra96 FPGA board?

12

Encryption: Flashback from CS2103

An *encryption scheme* (also known as *cipher*) consists of two algorithms:
encryption and decryption



Correctness: For any plaintext x and key k,
$D_k (E_k (x)) = x$

Security: From the ciphertexts, it is "difficult" to derive useful information of the key k, and the plaintext x. The ciphertexts should resemble sequences of random bytes. (There are many refined formulations of security requirements, e.g. semantic security. In this module, we will not go into details).

[Slide from CS2103, Prof. Chang Ee Chien]

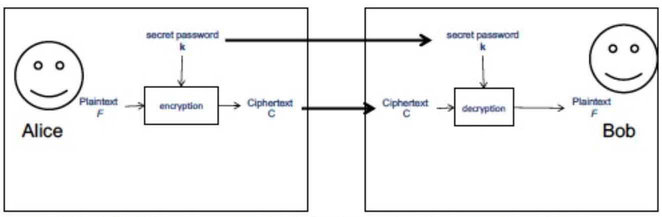
13

Encryption: Flashback from CS2103

A simple application scenario.

Alice has a large file ***F*** (say info on her bank accounts and financial transactions in Excel). She "encrypted" the file ***F*** using winzip with a password "13j8d7wjnd" and obtained the ciphertext ***C***. Next, she called Bob to tell him the 10-character password, and subsequently, she sent the ciphertext to Bob via email attachment. Later, Bob received ***C*** and decrypted the ciphertext with the password to obtain the plaintext ***F***.

Anyone who has obtained C, without knowing the password, is unable to get any information on ***F***. Although C indeed contains info of F, the information is "hidden". To someone who doesn't know the secret, C is just a sequence of random bits.



Remark: Winzip is ***not*** an encryption scheme. It is an application that employs standard encryption schemes such as AES.

[Slide from CS2103, Prof. Chang Ee Chien]

14

Why do we need encryption in our system?

- Open wireless networks
- Personal data privacy
- Authentication

- Key
  - Your choice – Tell us during evaluation so we can decrypt

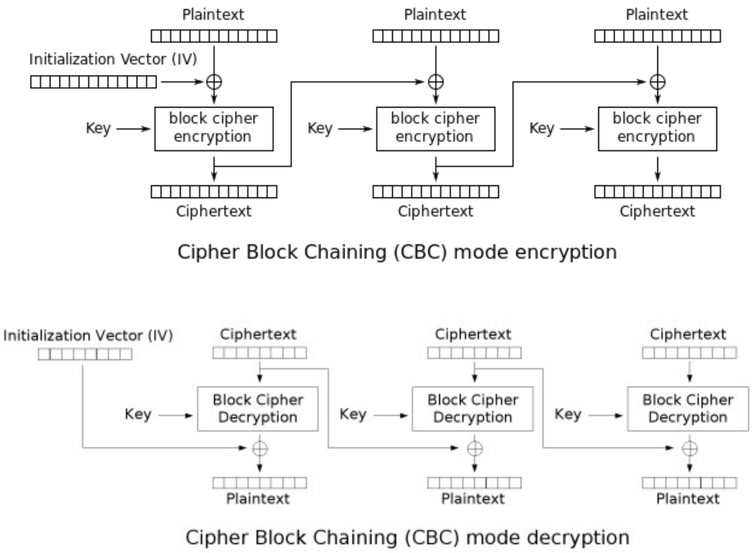
15

Using the AES Encryption scheme

- AES: Popular and widely adopted symmetric encryption standard
- Cryptodome Cipher library in python
  - from Crypto.Cipher import AES
- AES
  - mode CBC
  - Base 64
  - Secret key
  - Initial value: random
  - Padding

16

AES



17

Authentication: Server (provided)

- decodedMSG = base64.b64decode(encodedMsg)
- iv = decodedMSG[:16]
- cipher = AES.new(secret_key,AES.MODE_CBC,iv)
- decryptedText = cipher.decrypt(decodedMSG[16:]).strip()

18

Authentication: Client (You! ☺)

- iv = Random.new().read(AES.block_size)
- cipher = AES.new(secret_key,AES.MODE_CBC,iv)
- encoded = base64.b64encode(iv + cipher.encrypt(msg))

19

Server python code provided (on Luminus)

- eval_server.py
  - Server expects a secret key
  - Server expects message string of this format: '#position|action|syncdelay|'
  - E.g. '#2 1 3|muscle|1.87|'
  - Syncdelay (in ms)
  - AES expects base64 encoded message of 128-bits initial value + message
  - AES expects padding
  - Server returns previous correct positions so you can recalibrate
- Tips
  - Test your wireless comms client on your laptop first, localhost
  - Use a wireless hotspot so Ultra96 and laptop are on the same wireless LAN, rather than NUS WiFi
  - Test socket comms and encryption/decryption separately

20

Clock and network synchronization

21

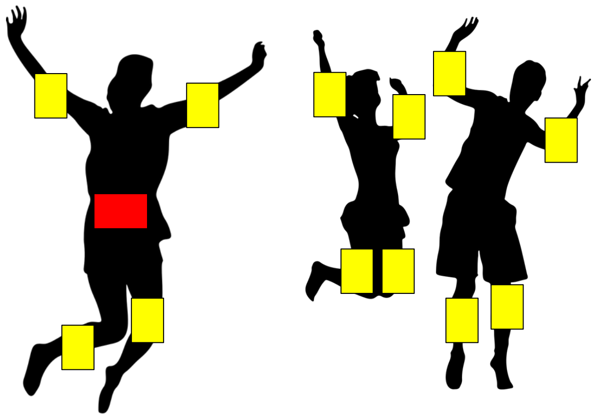
Problem: Estimate asynchrony between dancers

- What's the ground truth?
  - Human view of time
  - When do dancers 1, 2, 3 start a dance move
- How will we measure ground truth?
  - Video

22

Why's this challenging?

- When's the start of a dance move??
  - Work with hardware sensor person to set special flag packet to mark start
- Multiple clocks of different accuracy
  - Many beetles and laptops and Ultra96
  - Beetles only have Atmel 8-bit MCU with oscillator
- Communications is not instantaneous
  - Beetles only have BLE, not WiFi



23

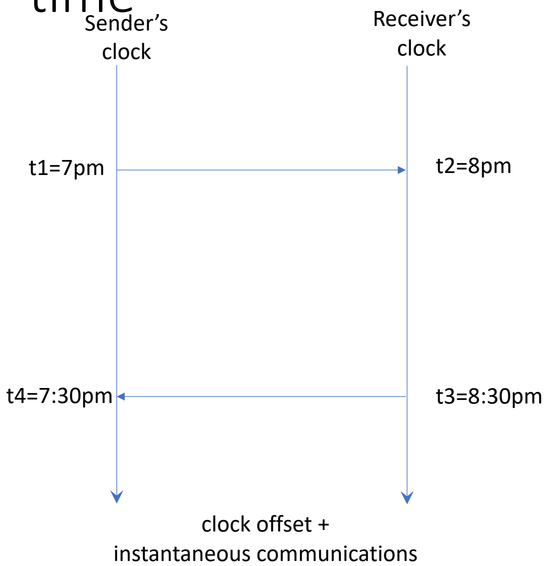
Let's learn from Network Time Protocol

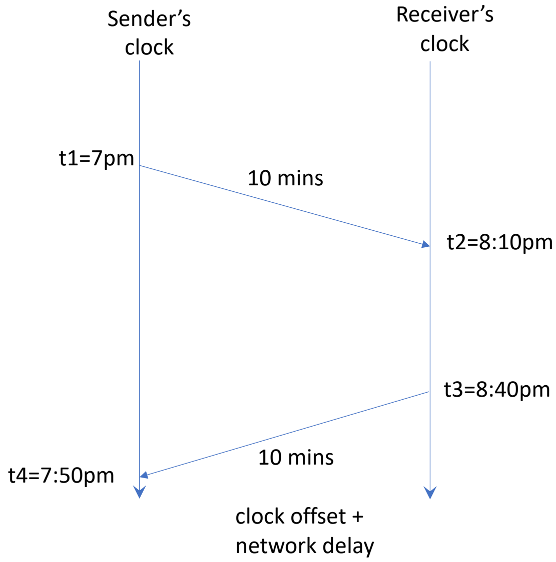
- Reference clocks
  - UTC: measures quantum resonance of a cesium atom
  - Atomic clocks
- Highly scalable
  - The entire Internet
  - Hierarchical
- Highly accurate
  - milliseconds
- Robust
  - Clock failure, link failure

- Reference clocks
  - Ultra96's clock
- Scalable -- hierarchy?
  - Beetles and Ultra96
- Accuracy
  - Human perception
- Robust
  - Clock failure?
  - Link failure?

24

NTP estimation of clock offset and round trip time





25

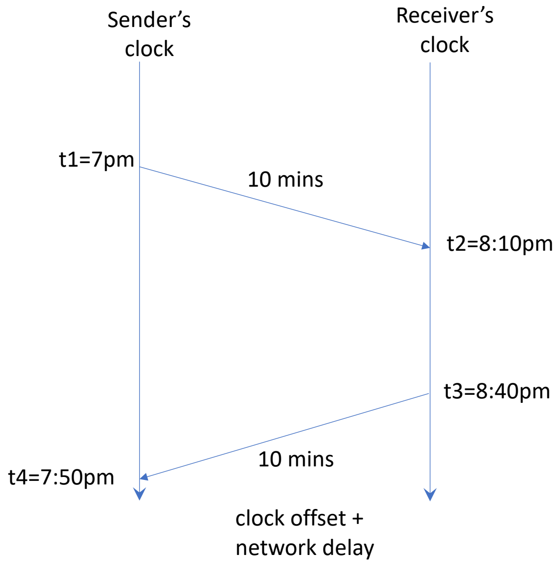
NTP estimation of clock offset and round trip time

RTT = t2-t1 (70mins) - t3-t4 (50 mins) = 20 mins

clock offset = t2 – t1 – RTT/2
(70-10=60 mins)

or

Or clock offset = t3 – t4 + RTT/2
(50+10=60 mins)



26

Now, your system

- Need relative time difference between start of moves
  - Set flag at start of a move
  - Send flag packet with local timestamp
  - Relative time difference between flag packets from different dancers
- Estimate clock offset of Beetles from Ultra96 clock
  - Timestamps on both devices (t1, t2, t3, t4)
- Estimate one-way communication delay

For each dancer vs. Ultra96

- Calculate actual time difference between flag packets, taking into account one-way communication delay and clock offset.

27

Evaluation of synchronization delay

- Video recording of 3 dancers
  - Ground truth determined manually: Time delay between fastest and slowest dancers' starting a dance move
  - Error: Difference between your system's predicted delay and ground truth
  - Average error across multiple moves
- How can you automatically evaluate and optimize your synchronization algorithm's accuracy?
- What are the tradeoffs?

28