Tell me about yourself ? I’m a recent computer systems engineering graduate with a year of work experience through co-op that included programming and system administration. Both school and work gave me the opportunity to continuously learn new content quickly to do well.

4 LTE: ~ 100 Mbps

5G: 20 Gbps. Increase MIMO antennas, few thousand

Interested in this job? I really enjoyed the course I had on Software Validation, Verification and Testing and I’ve only used it on small projects. I also enjoyed a broad course on telecommunications that covered topics like cellular networks, network security, and others.

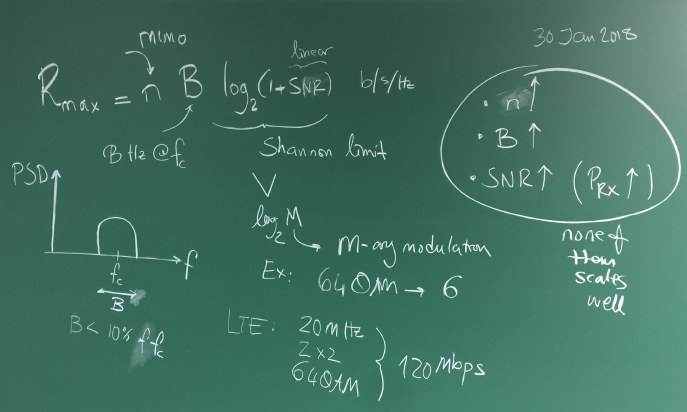
Cellular networks (slides)

Multiple Access schemes

VoIP slides

Network security: Certificates, Symmetric crypto system, public key crypto system

Why Telecom' :Why Ericsson Networks? Ericsson is one of the drivers of 5G (20 Gbps) networks and I’m interested how it going to support newer use cases and applications like Enhanced Mobile Broadband (high throughput, high res streaming), Massive Machine Type Communications Internet of things, and Ultra-Reliable Low-Latency Communications (mission critical, autonomous vehicles, and real time systems) that require high throughput with low loss.



N = multiple-input and multiple-output is the minimum number between transmitter and receive antennas

B = is the bandwidth (transfer rate) at a certain frequency (low: long distance, limited BW. High: short distance, carry more BW)

SE = spectral efficiency is the Shannon limit. Increase modulation by adding more levels

SNR = Signal to noise ratio.  power

How would you manage an issue that you have never come across before? I would try to solve it on my own, and then ask co-workers for their opinion.

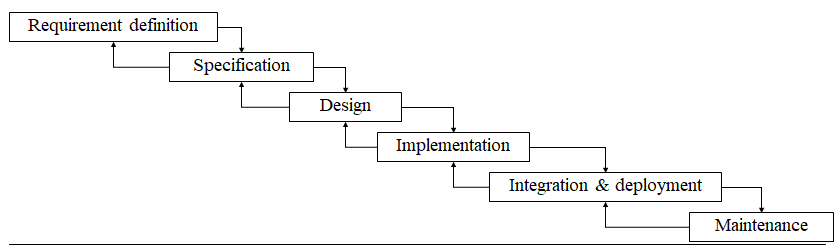
Strength? I am dependable and a good team player. During my last year of school, I had many concurrent group projects and I made sure that I completed my promised deliverables before deadlines so I could give any teammates help if they wanted it.

Weakness:

In 5 years? I would like to be working with some hardware and embedded systems.

Why should I hire you? From my experience from school and past work I have shown that I have the ability to learn the required skills quickly to do well and I’ve always met project deadlines with high quality deliverables and will use all my effort to continue to.

**Waterfall Model:** The waterfall model requires that all requirements are known at the beginning of the process and the next phase doesn’t start until the current phase is finished. The phases are:



This model was the most used in my academic projects since the requirements had to be defined at the beginning or were given.

Requirement elicitation: Gathering of Functional and non-functional requirements

Specification: Creating Scenarios, Use cases, Sequence diagrams

Design: System design diagram, class diagram.

Phase-Release Model: The Phase-release model is an incremental model, where it follows the waterfall model but it delivers a series of releases by iterating the phases of Design to Deployment to build the system piece by piece, but all the requirements were still finalized at the beginning.

**Software Verification:** Process of evaluating if a system or component satisfies the requirements at the start of a development phase, to check if it designed well. Code walkthroughs and review

**Software Validation:** Process of evaluating if a system or component satisfies the requirements during or after the development phase. Check if the system has the correct behaviour and correct outputs the stakeholders expect, requirements elicitation and analysis.

**Testing:** Evaluation of a software to provide stakeholders with information about the quality, correctness, and its performance under test. Intent of finding bugs, meets requirements, and performs correct and timely.

**Software Observability:** How easy it is to observe the behavior of a program in terms of its outputs, effects on connected hardware and software components

- Software that affects hardware devices, databases, or remote files have low observability.

**Software Controllability:** How easy it is to provide a program with the needed inputs, in terms of values, operations, and behaviours

**Test driver:** A software component or test tool that replaces a component that takes care of the control and/or the calling of a software component. The test driver executes a test case or set by executing the system with input values.

**Test stub:** Simulates the behavior of a software component that is called by the program being tested. Used to test or develop a component that calls the stub, replaces a called component. I.e. The stub may contain predetermined answers to calls made during the test, and a certain response to anything outside what’s programmed.

- Stubs and drivers enable components to be isolated from the rest of the system for testing purposes.

**Oracle:** - Assist in deciding whether a test outcome is successful or not

- Outcome of the oracle evaluation: pass/fail

- Two important tasks:

1. Deciding what the expected output is, given the test inputs of a test case. (Often done by human)

2. Comparing the actual output against the expected output (often done by software)

Tell me about:

Functionality-Based Input Domain Modeling: It is used to test a function by testing the characteristics of the inputs, for example a function requires 2 integers, say Length and width, and characteristics would be ranges of values based on boundary limits. Then you create test blocks, the range of values could have blocks of minimum, nominal, and maximum. To create the test frames the blocks are combined following a criteria like all combinations or pair wise, then the input values are chosen that fit in the test frame. Example is a length chosen to be -5 and width to be 10.

control flow criteria: It is used to test the control statements within a function or program, with a flow graph with control statements as the nodes. Following a criteria like all node or all edge criteria, the test paths are created from the initial node to final node. Finally the input values are chosen to satisfy the test paths, but some paths may prove to be infeasible so new feasible one has to be made to cover it.

data flow criteria: It is used to test the variable usage within a function or program, the first step is to create a flow graph with control statements as the nodes. (With the same graph used in control flow criteria) and then labeling where variables are initialized and used. Then following a criteria like all definition coverage or all uses coverage the test paths are created from the initial node to final node. Finally the input values are chosen to satisfy the test path, but some paths may prove to be infeasible so new feasible one has to be made to cover it.

and logic coverage: With a minimized Boolean expression that represents multiple control statement paths, we can determine the Boolean values for each condition to achieve code coverage. Criteria to follow are Predicate, combinational, active clause, Unique True point, and near false point pair coverage.

unit testing: It is testing individual functions by comparing the expected output with the actual output.

integration testing methodologies:

First the modules have to be ordered by dependencies, then there are 4 methods that determine the order that the modules are tested.

There is Big Bang Integration, where after all the modules are tested, they are all put together and the whole system is tested at once. This makes it difficult to find the root cause of any error.

There is Bottom-Up integration, where the modules with no dependencies are tested first then you connect upwards. This requires drivers with no stubs. The problem is that the lower level is more general routines and error in design at the top level may affect it.

There is Top down integration, where all the top controlling modules are tested first. Less drivers are needed, but need more stubs which are more error prone and complex.

Last is Sandwich integration, where there is a target middle layer and the bottom and top are tested. Tests both the controlling and general purpose routines, but the middle layer modules may not be sufficiently tested.

Regression testing: Testing unchanged code in the modified software to verify that it still performs the same. It requires determining which tests are now obsolete (invalid sequences), re-usable (no message changes), and re-testable (sequence valid, but message changes), then creating new tests for the modified portion.

complications of regression testing. If there are a large number of tests that will lead to over budget, we have to choose which tests to run. Could use Test minimization that discards some tests [through (or greedy – most coverage) genetic algorithms] but you run the risk of a removing tests that were critical to find some faults or covered modified part of code. Could use test prioritization by listing them by cost (time) or risk of not executing and then run tests until times up.

---

Key features of each programming language?

Java: Object oriented, needs compiler, multithreaded, and dynamic (garbage collection).

Python: Scripting, execute line by line, Can be object oriented, portable

C: Benefits of assembly like dynamic memory allocation, procedural, needs compiler, portable.

C++: Can be object oriented, more of a middle language combining low level and high level programming

Scripting vs programming: Scripting languages are programming languages that don't require an explicit compilation step.

Java and python syntax, write by hand

Steps to test software:

1. Review the Design and the Requirements of the Software to see if it meets the specifications. Also check if the usability is good.

2. Choose re-testable test cases and create tests sets for new functionality.

3. Set up the test environment that is similar to what a user would have, and then execute the tests.

4. Prepare a final report of the results of each test

Functional requirements: Defines what the system behaviour is with certain inputs described in its specifications.

Unit testing (test individual methods), integration testing (big bang, top down, bottom up, sandwich) After unit tests determine which modules to do integration testing order. System testing, acceptance testing.

Non-functional requirements: How the system operates like the quality of the system

Examples are Usability, security, performance (execution speed). (- testing)

Black box testing: Concerned with what the output is with a certain inputs to test functionality. Can be used to confirm functional and non-functional requirements, but you don’t know how much of the system was tested.

Functionality-Based Input Domain Modeling with pairwise testing and all combinations.

Boundary value analysis

White box testing: Testing the paths within the code to achieve code coverage, but doesn’t reveal missing functionality.

Control flow, data flow, logic coverage

L1 Physical Layer:

L2 Data link layer:

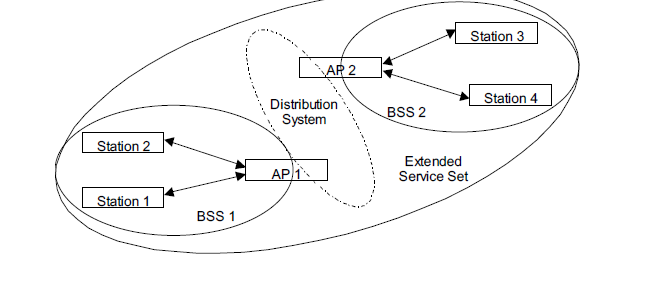
L3 Network layer:

L4: Transport layer:

802.11: Set of MAC and Physical layer specifications for implementing Wireless local area networks. Ac. Most home wireless routers are compliant with this standard.

**802.11n** - Significant bandwidth improvement from previous standards; wide support across devices and network gear.

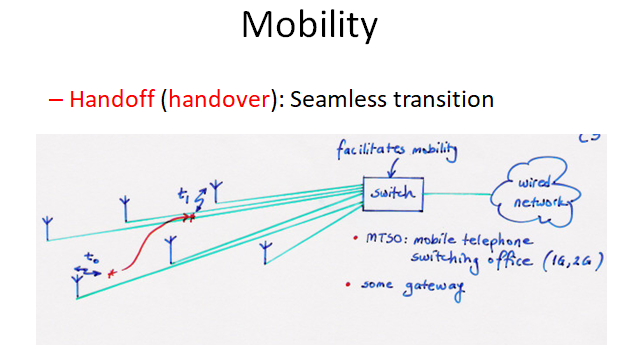
802.11 WLAN Wi-Fi. The most basic building block of 802.11 WLAN is a **station** (could be a PC, phone, etc), which contains a MAC and physical layer. Connecting stations directly or with **access points** create a **Basic service Set** (independent: no AP, infrastructure: through AP [is a [networking hardware](https://en.wikipedia.org/wiki/Networking_hardware) device that allows a [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) device to connect to a wired network. The AP usually connects to a router (via a wired network) as a standalone device]). To connect BSS’s you need a **Distribution System (**system enabling the wireless interconnection of [access points](https://en.wikipedia.org/wiki/Wireless_access_point)) is used to connect them, which then creates an **Extended Service set**.



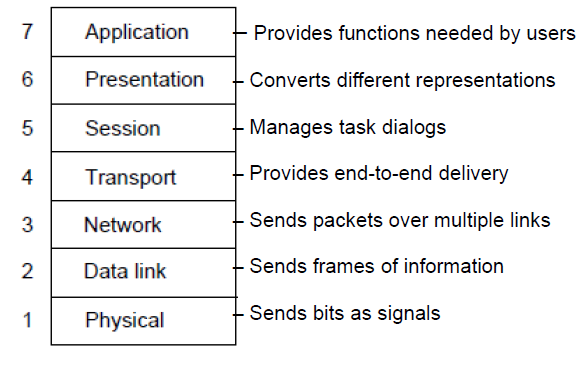
PHY layer: uses unlicensed frequency bands (900MHz, 2.4 GHz, 5, 60). Limits interference with methods like listen before transmit, limit transmit time, and limit transmit power. Uses OFDM (orthogonal frequency division multiplexing.) to share the band and since its orthogonal there is no interference from other carriers.

MAC: controls access to medium through CSMA/CA (carrier sense multiple access with collision avoidance). Provides Reliable delivery, error correction, packet fragmentation and re-assembly, mobility, authentication, and encryption.

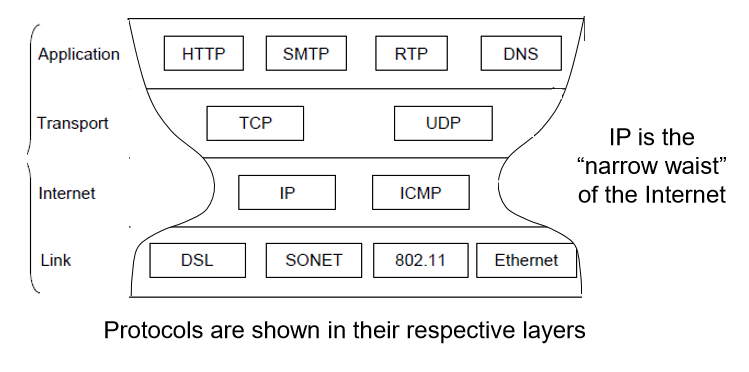
Mobility in cellular networks: There has to be seamless transitions when handing off to another base station for high Quality of service. Goal is to have the best coverage with minimum cost, fewest base stations (access point to the wired network). A problem in cellular networks is Interference due to channel re-use has to be managed with cluster size, by the increasing cluster size (more cells within a cluster) reduces co channel interference. (Less interference from another cluster where a user there is using the same channel)

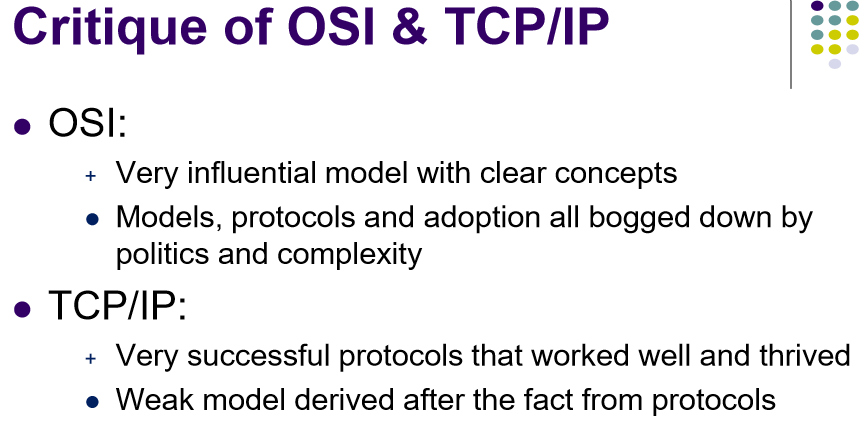


OSI Model: Clear standardized layers and encapsulation to provide interoperability.



TCP/IP: Model created after interconnectivity worked and it derived from the protocols used.





**Switch vs Router:**

- Switches create a network. Routers connect networks together and to the internet. A router acts as a dispatcher, choosing the best path for information to travel so it's received quickly.

MPLS: sets up a virtual circuit so the packets have a fixed path. It replaces the IP in the header with an MPLS label for faster lookup to forward faster

STATIC VERSUS DYNAMIC MPLS LABEL-SWITCHED PATHS (LSPS): paths are established for particular source-destination pairs. The predetermined paths that make MPLS work are called label-switched paths (LSPs).

You can create MPLS(Multiprotocol Label Switching) LSPs in one of two ways:

Static configuration: Static LSPs are a lot like static routes. You basically have to explicitly configure every **label switch router,** LSR, in an LSP manually. Because no protocols dynamically signal the LSP for you, the load on the LSRs is reduced. However, if you have changes in the topology, the paths can’t adapt to the new network. As a result, topology changes create routing black holes.

Dynamic setup: Dynamic LSPs use signaling protocols to establish themselves and propagate LSP information to other LSRs in the network. You configure the ingress router with LSP information that is transmitted throughout the network when you enable the signaling protocols across the LSRs.

Note that you have to configure the signaling protocols on all of the LSRs. If only a subset of routers is able to exchange information, the LSP isn’t established.

Because the LSRs must exchange and process signaling packets and instructions, dynamic LSPs consume more resources than static LSPs. However, dynamic LSPs can avoid the network black holes of static LSPs by detecting topology changes and outages and dynamically establishing new LSPs to move around the failure.

How LAN works: It’s a network of local computers and devices that share a common transmission medium so it uses dynamic Medium access control methods like scheduled access (polling) or random access (ALOHA). Low cost and high speed communication. Star, ring, Mesh

convert the Arabic numerals from user input to Roman numerals   by hand python (subtracting total and adding to string)

Given 10 numbers from 0-9, find which number is missing in a random array has 9 elements from this 10 numbers with best time efficiency.

(Add all the numbers then 45 – the sum.)

Fourth year project: The project was a system that helped tourist find locations of interest in a city, and my part was to create a touch screen kiosk that hosted the application so people without internet access could use it. I used a raspberry pi and created the circuits that enabled touch screen surface with infrared. To test and visualize if the correct location was touched on the screen, I wrote a visual studio application that received the intercept and showed which lines were being blocked to cause the intercept. (operation, 1 column of infrared emitters on one side of the screen and a column of infrared transistors on the other to create a touch screen area in between. The main idea is to have a single emitter on at a time and all the sensors would check if they can detect the ON emitter. If it doesn’t it means something is in the way. After cycling through the emitters, it will check if an intercept can be calculated.

TFTP: The server was multithreaded to handle multiple requests in parallel, it handled read and write requests and had error handling for incorrect packets, like wrong block number (for ack and data) or wrong transfer ID and network errors. TFTP used UDP datagrams and added reliability like handshaking and in order transfers. (Testing was with microsoft’s TFTP client to test the server, to test the whole system with the group’s client was using an error simulator in the middle that would simulate packet and network errors like (duplicate, drops).

Github: Version control system to track changes. Local and remote repositories.

Commands:

$ git init // initialize Local git repo for that folder you are in.

$ git add <file> // add the files to the staging area of the index, ready for commit

$ git add . // add all

$ git rm --cached <file> // remove

$ git status // Check status of working tree, shows difference of working directory and staging area of the index

$ git commit // Takes everything in the index and commits it to the local repo

$ git commit -m ‘comment’ // commit with message

$ touch .gitignore // put the filenames in the gitingore file to ignore when “git add .”

// log.txt /dir2 (for directory)

$ git branch <name> // new branch

$ git checkout <name> // move to the branch

$ git merge <branchName> // with current branch

// remote ---

$ git remote add origin <http://asffsf> // add remote repo

$ git push –u origin master // push to that repo

$ git push // push local repo to remote repo, need to add credentials

$ git pull // pull latest changes from remote repo

$ git clone // Clone the repo into the current folder

$ git config --global user.name ‘Jonathan chan’

$ git config –global user.email ‘@hotmail.com’