CS2105

Introduction to Computer Networks

AY2022/23 Semester 1

Notes by Jonathan Tay

Last updated on October 7, 2022

Contents

Part I

Link Layer

The **link layer** sends datagrams between *adjacent* nodes (hosts or routers) over a single link.

IP datagrams are encapsulated in **frames**.

The link layer is implemented in hardware adapters (network interface cards) or on a chip.

1 Error Detection and Correction

Links may be **error-prone**.

Error detection and correction bits (`EDC`) are appended to the **data bits** (`D`) before transmission across a link.

However, **error detection schemes** are not 100% reliable — they may not detect all errors, but a larger `EDC` increases the probability of detecting errors.

There are 3 popular error detection schemes:

1. **parity checking**:
   works well mathematically, but not in practice as errors are clustered together
2. **checksums**: used in TCP/UDP/IP
3. **cyclic redundancy checking**: used in the link layer

1.1 Parity Checking

1D parity checking

Include an additional **parity bit**.

For **even parity**, the parity bit is set to `0` if the number of 1s in the data bits is even, and `1` otherwise.

For **odd parity**, the parity bit is set to `1` if the number of 1s in the data bits is even, and `0` otherwise.

2D parity checking

If the data bits are arranged in a **2D matrix** with i rows and j columns, then we can compute each row and column parity.

In addition, we can compute the parity bit for the column and row parity bits, for a total of $i + j + 1$ parity bits.

1D parity checking can *detect* **single bit errors** (or any odd number of them), but not correct them.

2D parity checking can *detect and correct* **single bit errors**, by intersecting the error row and column.

In addition, it can *detect* **two-bit errors**, but not correct them.

1.2 Cyclic Redundancy Checking (CRC)

non-binary cyclic redundancy checking

First, some terminology:

- **`D`**: a non-binary number to transfer
- **`R`**: an r-digit error detection code
- **`G`**: an r-digit **generator** known to both sender and receiver

To transmit the new message `M`:

1. Create a new number `D'` by appending `9`, r times, to `D`.
2. Find the remainder y of `D'` divided by `G`.
3. Transmit `M = D' - y`.

Notice how `M` is divisible by `G` — the receiver can calculate the new remainder y', and discard the message if $y' \neq 0$.

binary cyclic redundancy checking

First, some terminology:

- **`D`**: binary data bits
- **`R`**: an r-bit error detection code
- **`G`**: an $r + 1$-bit **generator** known to both sender and receiver

All calculations are done **modulo 2**, to avoid carries for addition and borrows for subtraction — now identical to `XOR`.

To transmit the new message `M`:

1. Create a new number `D'` by appending `0`, r times, to `D`.
2. Find the remainder of `D'` divided by `G`, which is used as `R`.
3. Transmit `M = (D, R)`, which is `R` appended to `D`.

Now, the receiver divides `M` by `G` and checks if the remainder is `0`.

A non-zero remainder indicates an error.

CRC's error detection capabilities:

- **single bit errors**: all odd numbers of errors
- **burst errors < r+1-bits**: all such errors
- **burst errors > `r`-bits**: probability $1 - 0.5^r$

CRC is also easy to implement on hardware due to the modulo-2 arithmetic.

2 Link Access Control

Point-to-point links connect a sender and receiver directly — there is no need for multiple access control.

Broadcast links connect multiple nodes to a single shared broadcast channel.

Every node receives a copy of every broadcasted link, causing **collision** if two signals are received simultaneously.

ideal multiple access control

Given a broadcast channel of rate R bps, the muliple access control protocol should be:

1. **collision-free**
2. **efficient**: a single transmitting node should send at rate R
3. **fair**: each transmitting node among M nodes should send at an average rate R/M
4. **decentralized**: no coordination between nodes

In addition, **channel sharing coordination** *must use the channel itself* — no other out-of-band channel.

2.1 Channel Partitioning Protocols

time division multiple access (TDMA)

Let there be n nodes.

Each node is equally allocated a **time slot** of length T, spanning a **time frame** of length $n \cdot T$, during which they get access to the channel; outside of which they are **idle**.

frequency division multiple access (FDMA)

FDMA is akin to TDMA but with **frequency bands** instead of time slots.

TDMA and FDMA are *collision-free*, *perfectly fair*, *decentralized*, but **inefficient** as unusued slots are wasted.

2.2 Controlled Access Protocols

polling protocol

One node is designated as the **master node**.

The master node polls each **slave node** in turn, allowing it to transmit a pre-determined maximum number of frames.

Polling is *collision-free*, *efficient*, *perfectly fair*, but **not decentralized** as the master node results in a single point of failure.

token passing protocol

In a ring network topology, a special **token** frame, is passed between nodes sequentially.

The node possessing the token can transmit a pre-determined maximum number of frames before forwarding the token.

Token-passing is *collision-free*, *efficient*, *perfectly fair*, and **decentralized**.

However, a **token loss** and a single **node failure** can be disruptive.

2.3 Random Access Protocols

A node with data to send should be able to transmit at full channel data rate R, with **no *a priori* coordination**.

Random access protocols must *detect* and *recover* from collisions.

slotted ALOHA

Like TDMA, time is divided into slots of length L/R, and synchronized at each node.

Nodes transmit only at the beginning of a slot, re-transmitting in the event of a collision in each subsequent slot with probability p.

Slotted ALOHA is *collision-free*, *fair* and *decentralized*, but **inefficient** when many nodes are active due to collision and empty slots — resulting in a maximum 37% maximum efficiency.

pure (unslotted) ALOHA

No synchonization and time slots are needed — nodes transmit immediately at any time.

In the event of a collision, they wait for a 1-frame transmission time before re-transmitting with probability p.

Unslotted ALOHA is worse than slotted ALOHA with 18% maximum efficiency as collision probability is higher.

In general, ALOHA is flawed as transmission decision is made independently of other nodes.

CSMA

Nodes defer transmission if the channel is busy, and transmit immediately if it is idle.

Collisions still occur due to propagation delay.

Both ALOHA and CDMA are flawed, in that they do not stop transmitting when collision is detected.

CSMA/CD

Like CSMA, but abort transmission if collision is detected, and re-transmit after a random delay, determined by **binary exponential backoff**:

1. after a collision, choose $k \in \{0, 1\}$
2. wait k time units before re-transmitting
3. after another collision, choose $k' \in \{0, 1, 2, 2^2 - 1\}$
4. wait k' time units before re-transmitting
5. after m collisions, choose $k'' \in \{0, 1, 2, 2^2 - 1, \ldots, 2^m - 1\}$
6. wait k'' time units before re-transmitting

The time unit is 512-bit transmission time for Ethernet.

A **minimum frame size** (64 bytes for Ethernet) is imposed to ensure increase the chance that collision is detected.

Both CSMA and CSMA/CD *efficient*, *fair* and *decentralized*, but **not collision-free**.