PERTEMUAN 6
ERROR DETECTION

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini akan dijelaskan mengenai *error detection.* Setelah mempelajari materi ini mahasiswa diharapkan mampu untuk:

1. Memahami fungsi *error detection*
2. Menjelaskan macam-macam metode error detection

B. URAIAN MATERI

1. Memahami Fungsi Error Detection

Kesalahan bit terkadang dimasukkan ke dalam frame. Ini terjadi, misalnya, karena gangguan listrik atau gangguan termal. Meskipun kesalahan jarang terjadi, terutama pada tautan optik, beberapa mekanisme diperlukan untuk mendeteksi kesalahan tersebut sehingga tindakan korektif dapat dilakukan. Jika tidak, pengguna akhir akan bertanya - tanya mengapa program C yang berhasil dikompilasi beberapa saat yang lalu sekarang tiba - tiba memiliki kesalahan sintaks di dalamnya, padahal semua yang terjadi sementara itu disalin melalui sistem file jaringan.

Ada sejarah panjang sebuah teknik untuk menangani kesalahan bit dalam sistem komputer, sejak tahun 1940-an. Kode Hamming dan Reed-Solomon merupakan dua contoh penting yang dikembangkan untuk digunakan dalam pembaca kartu punch, saat menyimpan data pada disk magnetik, dan dalam memori inti awal. Bagian ini menjelaskan beberapa teknik deteksi kesalahan yang paling umum digunakan dalam jaringan.

Prinsip dasar deteksi kesalahan adalah agar pemancar menghitung karakter cek berdasarkan konten pesan asli. Ini dikirim ke penerima di akhir pesan dan penerima mengulangi perhitungan yang sama pada bit yang diterimanya. Jika karakter pemeriksaan yang dihitung tidak cocok dengan yang dikirim, maka akan dianggap sedang terjadi kesalahan. Bentuk paling sederhana dari pemeriksaan kesalahan dalam sistem asinkron adalah dengan memasukkan bit paritas, yang mungkin genap atau ganjil. Paritas genap membutuhkan jumlah

total bit data pada logika 1 ditambah bit paritas agar sama dengan bilangan genap. Perangkat keras komunikasi di ujung transmisi menghitung paritas yang diperlukan dan menetapkan bit paritas untuk memberikan jumlah logika 1 bit yang genap. Paritas ganjil bekerja dengan cara yang sama seperti paritas genap, kecuali bahwa bit paritas disesuaikan sehingga jumlah total bit logika 1, termasuk bit paritas, sama dengan angka ganjil.

Perangkat keras di ujung penerima menentukan jumlah total logika 1 bit dan melaporkan kesalahan jika itu bukan bilangan genap atau ganjil yang sesuai. Perangkat keras penerima juga mendeteksi pembengkakan penerima dan kesalahan bingkai. Secara statistik, penggunaan bit paritas hanya memiliki peluang sekitar 50% untuk mendeteksi kesalahan pada sistem kecepatan tinggi. Metode ini dapat mendeteksi jumlah bit yang salah dan tidak akan mendeteksi jumlah bit yang salah. Bit paritas biasanya dihilangkan jika ada skema pemeriksaan kesalahan yang lebih canggih.

Mendeteksi kesalahan hanyalah salah satu bagian dari masalah. Bagian lainnya mengoreksi kesalahan setelah terdeteksi. Dua pendekatan dasar dapat diambil ketika penerima pesan mendeteksi kesalahan. Salah satunya adalah memberi tahu pengirim bahwa pesan tersebut rusak sehingga pengirim dapat mengirimkan kembali salinan pesan tersebut. Jika kesalahan bit jarang terjadi, maka kemungkinan besar salinan yang dikirim ulang akan bebas dari kesalahan. Alternatifnya, beberapa jenis algoritma deteksi kesalahan memungkinkan penerima untuk menyusun kembali pesan yang benar bahkan setelah pesan tersebut rusak; algoritme semacam itu mengandalkan kode koreksi kesalahan.

Ide dasar dibalik skema deteksi kesalahan adalah menambahkan informasi yang berlebihan ke bingkai yang dapat digunakan untuk menentukan apakah kesalahan telah diperkenalkan. Secara ekstrim, kita bisa membayangkan mentransmisikan dua salinan lengkap data. Jika kedua salinan identik di penerima, maka mungkin keduanya benar. Jika berbeda, maka kesalahan dimasukkan ke salah satu (atau keduanya), dan harus dibuang. Ini adalah skema deteksi kesalahan yang buruk karena dua alasan. Pertama, ia mengirimkan n bit redundan untuk pesan n-bit. Kedua, banyak kesalahan yang tidak akan terdeteksi oleh kesalahan apapun yang terjadi merusak posisi bit yang sama disalinan pertama dan kedua dari pesan tersebut. Secara umum, tujuan dari kode pendeteksi kesalahan adalah untuk memberikan probabilitas yang tinggi untuk

mendeteksi kesalahan yang dikombinasikan dengan jumlah bit redundan yang relatif rendah.

Bit ekstra yang dikirim merupakan hal yang mubazir, karena tidak menambahkan informasi baru ke dalam pesan. Sebaliknya, pesan tersebut diturunkan langsung dari pesan asli menggunakan beberapa algoritme yang ditentukan dengan baik. Baik pengirim dan penerima tahu persis apa algoritma itu. Pengirim menerapkan algoritme ke pesan untuk menghasilkan bit yang redundan. Kemudian mengirimkan pesan dan beberapa bit ekstra itu. Ketika penerima menerapkan algoritma yang sama ke pesan yang diterima, itu seharusnya (jika tidak ada kesalahan) muncul dengan hasil yang sama dengan pengirim. Ini membandingkan hasil dengan yang dikirim oleh pengirim. Jika cocok, itu dapat menyimpulkan (dengan kemungkinan tinggi) bahwa tidak ada kesalahan yang diperkenalkan dalam pesan selama transmisi. Jika tidak cocok, dapat dipastikan bahwa pesan atau bit yang berlebihan telah rusak, dan harus mengambil tindakan yang tepat, yaitu, membuang pesan atau memperbaikinya jika memungkinkan.

Satu catatan tentang terminologi bit ekstra ini. Secara umum, mereka disebut sebagai kode pendeteksi kesalahan. Dalam kasus tertentu, ketika algoritma untuk membuat kode didasarkan pada penjumlahan, mereka dapat disebut checksum. Kita akan melihat bahwa checksum Internet dinamai dengan tepat: Ini adalah pemeriksaan kesalahan yang menggunakan algoritma penjumlahan. Sayangnya, kata checksum sering digunakan secara tidak tepat untuk mengartikan segala bentuk kode pendeteksi kesalahan, termasuk CRC. Ini bisa membingungkan, jadi kami mendorong Anda untuk menggunakan kata checksum hanya untuk diterapkan ke kode yang benar-benar menggunakan penambahan dan menggunakan kode pendeteksi kesalahan untuk merujuk ke kelas umum kode yang dijelaskan di bagian ini.

2. Menjelaskan Macam-Macam Metode Error Detection

a. Parity Checking

Parity Checking adalah salah satu metode yang digunakan untuk memeriksa apakah data telah berubah atau rusak setelah transmisi dari satu perangkat atau media ke perangkat atau media lain. Sebuah bit data, misalnya, dialokasikan sebagai bit paritas. Ini dialokasikan sebelum transmisi

terjadi. Sistem yang menggunakan Paritas Genap memiliki bilangan genap 1-bit, sedangkan sistem yang menggunakan Paritas Ganjil memiliki bilangan ganjil 1-bit.

Tabel 4. Parity Bit

	1	1	0	1	1	0	0

Metode sederhana untuk mendeteksi kesalahan adalah dengan menambahkan bit redundan yang disebut bit paritas ke setiap karakter. Metode ini disebut pemeriksaan paritas dan umumnya digunakan untuk karakter ASCII di mana tujuh bit digunakan untuk pengkodean karakter aktual dan bit kedelapan untuk paritas. Nilai bit paritas (yaitu, bit kedelapan) dipilih untuk membuat bilangan 1 menjadi bilangan genap (untuk paritas genap) atau bilangan ganjil (untuk paritas ganjil).

Sebagai contoh paritas genap, asumsikan bahwa urutan bit 1001101 akan dikirim; pesan yang sebenarnya dikirim adalah 10011010 - jumlah pertama adalah 4 (genap), jadi bit 0 ditambahkan setelah urutan aslinya. Masalah dengan skema ini adalah jika, misalnya dalam paritas genap, kesalahan dalam jumlah genap terjadi, skema pemeriksaan paritas akan gagal mendeteksi fakta bahwa frame harus dibuang. Misalnya, jika kami mengirimkan 10011010 dan menerima 10011000, pemeriksaan paritas akan gagal karena bit kedua hingga terakhir rusak, dan itu bagus. Tetapi jika urutan bit 11011000 diterima, pemeriksaan paritas akan lolos, meskipun bit kedua dan bit kedua hingga terakhir rusak saat transit.

Mungkin bentuk paling sederhana dari deteksi kesalahan adalah penggunaan bit paritas tunggal. Misalkan informasi yang akan dikirim “D”, maka memiliki d bit. Dalam skema paritas genap, pengirim hanya menyertakan satu bit tambahan dan memilih nilainya sedemikian rupa sehingga jumlah total 1 dalam bit (informasi asli ditambah bit paritas) adalah genap. Untuk skema paritas ganjil, nilai bit paritas dipilih sedemikian sehingga terdapat angka ganjil 1. Operasi penerima d + 1 juga sederhana dengan bit paritas tunggal. Penerima hanya perlu menghitung jumlah 1 dalam bit yang diterima. Jika bilangan ganjil dari bit bernilai 1 ditemukan dengan skema paritas genap, penerima mengetahui bahwa setidaknya satu kesalahan bit telah terjadi. Lebih tepatnya, ia mengetahui bahwa beberapa kesalahan bit ganjil telah terjadi. Jika kesalahan bit berjumlah bilangan genap, maka harus

meyakinkan diri sendiri bahwa ini akan menghasilkan kesalahan yang tidak terdeteksi. Jika probabilitas kesalahan bit kecil dan kesalahan dapat diasumsikan terjadi secara independen dari satu bit ke bit berikutnya, kemungkinan kesalahan beberapa bit dalam paket akan sangat kecil.

Jika bit ini menggunakan paritas genap, maka bit paritas harus 0 karena sudah ada bilangan genap 1 bit. Jika yang digunakan paritas ganjil, maka bit paritas harus 1 agar jumlah bilangan ganjil 1 bit.

Oleh karena itu, bit sebelum transmisi adalah:

Tabel 5. Parity Bit (Even)

Paritas genap

0	1	1	0	1	1	0	0

Tabel 6. Parity Bit (Odd)

Paritas ganjil

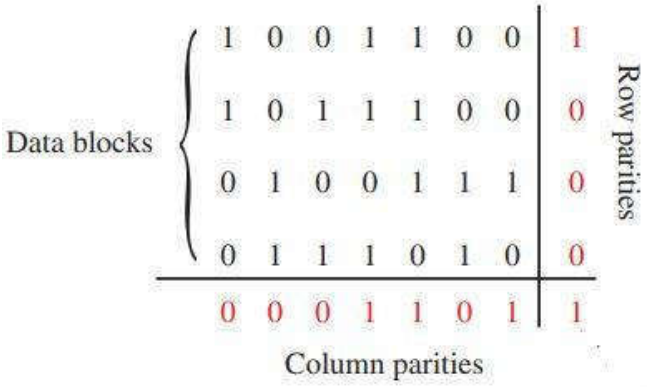
1	1	1	0	1	1	0	0

Sebelum data ditransfer, kesepakatan dibuat antara pengirim dan penerima mengenai mana dari dua jenis paritas yang digunakan.

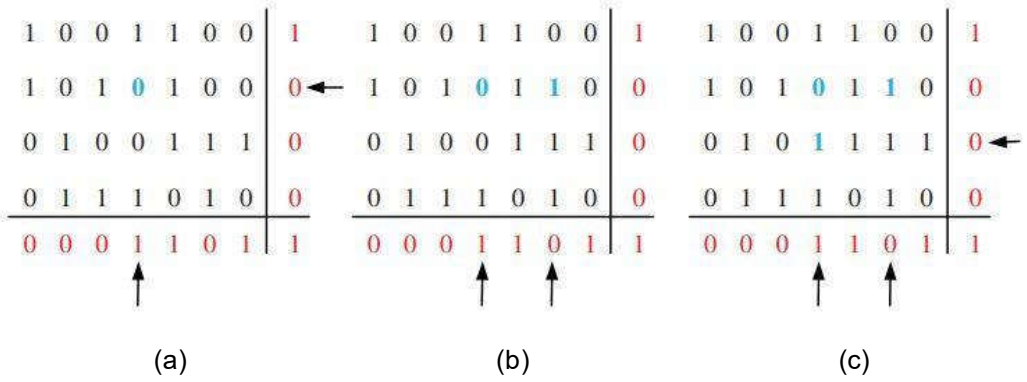
b. Two-Demensional Parity

Dalam pemeriksaan paritas dua dimensi, blok data diatur sebagai larik dua dimensi. Secara khusus, setiap baris dari larik adalah blok data yang akan dikirim. Bit paritas ditambahkan ke setiap baris berdasarkan apakah paritas genap atau ganjil digunakan. Bit paritas untuk setiap kolom dihitung dengan cara yang sama. Ini diilustrasikan pada Gambar 1, yang memiliki empat blok data dan bahkan paritas digunakan pada paritas baris dan kolom.

Seperti yang ditunjukkan pada Gambar 2 (a), ketika tepat satu kesalahan terjadi, skema dapat mendeteksi lokasi dengan kegagalan pemeriksaan horizontal dan paritas yang terkait dengan bit itu. Kemudian dapat membalik bit sehingga memperbaiki kesalahan. Ia dapat mendeteksi konfigurasi kesalahan tertentu, terutama jika tidak terjadi pada saat yang sama



Gambar 11. *Contoh Two-Demensional Parity*



Gambar 12. Beberapa konfigurasi yang terjadi kesalahan (a) 1 Kesalahan; (b) 2 Kesalahan; dan (c) 3 Kesalahan.

baris atau kolom yang tidak dapat dideteksi. Namun, itu tidak dapat memperbaiki kesalahan seperti yang diilustrasikan oleh Gambar 2 (b). Demikian pula, ia dapat mendeteksi tiga kesalahan, seperti yang diilustrasikan pada Gambar 2 (c). Akhirnya, itu tidak dapat mendeteksi keempat konfigurasi kesalahan.

c. Checksumming

Salah satu metode checksumming sederhana adalah dengan menjumlahkan bilangan bulat k-bit ini dan menggunakan jumlah yang dihasilkan sebagai bit deteksi kesalahan. Checksum Internet didasarkan pada pendekatan suatu bit data diperlakukan sebagai integer 16-bit dan dijumlahkan. Komplemen pertama dari jumlah ini kemudian membentuk

checksum Internet yang dilakukan di header segmen. Penerima memeriksa checksum dengan mengambil komplemen pertama dari jumlah data yang diterima (termasuk checksum) dan memeriksa apakah hasilnya semua 1 bit. Jika salah satu bit adalah 0, maka akan menunjukan suatu kesalahan. Dalam protokol TCP dan UDP, checksum Internet dihitung di semua bidang (termasuk bidang header dan data). Dalam IP, checksum dihitung melalui header IP (karena segmen UDP atau TCP memiliki checksumnya sendiri). Dalam protokol lain, misalnya, XTP, satu checksum dihitung melalui header dan checksum lain dihitung di seluruh paket.

Metode checksumming membutuhkan overhead paket yang relatif kecil. Misalnya, checksum di TCP dan UDP hanya menggunakan 16 bit. Namun, mereka memberikan perlindungan yang relatif lemah terhadap kesalahan dibandingkan dengan pemeriksaan redundansi siklik, yang dibahas di bawah ini dan yang sering digunakan dalam lapisan tautan. Pertanyaan umum pada saat ini adalah, Mengapa checksumming digunakan pada lapisan transport dan pemeriksaan redundansi siklik digunakan pada lapisan tautan? Ingatlah bahwa lapisan transport biasanya diimplementasikan dalam perangkat lunak di host sebagai bagian dari sistem operasi host. Karena deteksi kesalahan lapisan transportasi diterapkan dalam perangkat lunak, penting untuk memiliki skema deteksi kesalahan yang sederhana dan cepat seperti checksumming. Di sisi lain, deteksi kesalahan pada lapisan tautan diimplementasikan pada perangkat keras khusus pada adaptor, yang dapat dengan cepat melakukan operasi CRC yang lebih kompleks.

d. Cyclic Redundancy Check

Teknik deteksi kesalahan yang digunakan secara luas di jaringan komputer saat ini didasarkan pada kode *Cyclic Redundancy Check* (CRC). Kode tersebut juga dikenal sebagai kode polinomial, karena dimungkinkan untuk melihat string bit yang akan dikirim sebagai polinomial yang koefisiennya adalah nilai 0 dan 1 dalam string bit, dengan operasi pada string bit diinterpretasikan sebagai aritmatika polinomial.

Untuk menjalankan Kode *Cyclic Redundancy Check* (CRC) yaitu dengan memperrtimbangkan potongan data d-bit. D, yang ingin dikirim oleh node pengirim ke node penerima. Pengirim dan penerima pertama-tama harus menyetujui pola bit, yang dikenal sebagai generator atau tunjukkan

sebagai G. Kemudian akan meminta bit G yang paling signifikan (paling kiri) adalah 1. Untuk bagian data tertentu, D, pengirim akan memilih r bit tambahan, R, dan menambahkannya ke D sehingga pola bit yang dihasilkan (diinterpretasikan sebagai bilangan biner) tepat habis dibagi oleh G (yaitu, tidak memiliki sisa) menggunakan 2 modul aritmatika. Proses pengecekan kesalahan dengan CRC sangat sederhana, dengan cara penerima membagi bit yang diterima dengan G. Jika sisanya bukan nol, penerima mengetahui bahwa kesalahan telah terjadi; jika tidak, data tersebut dianggap benar.

Setiap standar CRC dapat mendeteksi kesalahan ledakan kurang dari r + 1 bit. (Ini berarti bahwa semua kesalahan bit berturut-turut dari r bit atau lebih sedikit akan dideteksi.) Selanjutnya, dengan asumsi yang sesuai, ledakan dengan panjang lebih besar dari r + 1 bit dideteksi dengan probabilitas 1 = 0,5r. Selain itu, setiap standar CRC dapat mendeteksi kesalahan bit ganjil.

Tujuan utama dalam merancang algoritma deteksi kesalahan adalah untuk memaksimalkan kemungkinan mendeteksi kesalahan dengan hanya menggunakan sejumlah kecil bit yang berlebihan. Pemeriksaan redundansi siklik menggunakan beberapa matematika yang cukup kuat untuk mencapai tujuan ini. Misalnya, CRC 32-bit memberikan perlindungan yang kuat terhadap kesalahan bit umum dalam pesan yang panjangnya ribuan bit. Landasan teoritis dari pemeriksaan redundansi siklik berakar pada cabang matematika yang disebut medan hingga. Meskipun ini mungkin terdengar menakutkan, ide dasarnya dapat dengan mudah dipahami. Untuk memulai, misalkan pesan (*n* + 1) bit yang diwakili oleh polinomial derajat n, yaitu, polinomial yang suku orde tertingginya adalah (*xn*). Pesan diwakili oleh polinomial dengan menggunakan nilai setiap bit dalam pesan sebagai koefisien untuk setiap suku dalam polinomial, dimulai dengan bit paling signifikan untuk mewakili suku orde tertinggi.

C. SOAL LATIHAN/ TUGAS

1. Jelaskan definsi error detction yang Anda ketahui!
2. Jelaskan apa yang dimaksud dengan parity cracking!
3. Jelaskan apa yang dimaksud dengan Two demensional parity!
4. Jelaskan apa yang dimaksud dengan cyclic redundancy check!

D. REFERENSI

Kurose, James F.; Rose, Keith W. (2017), *Computer Networking: A Top-Down Approach* (Seventh Edition.), Pearson Education.

Ibe, Oliver C.(2017), Fundamentals of Data Communication Network, Willey.

Peterson, Larry L.; Davie, Bruce S. (2012), Computer Network: A Systems Approach (Fifth Edition), Morgan Kaufmann.

Watson, David; Williams, Helen. (2014), *Computer Science*, Hodder Education.