Penerapan Algoritma Pohon Biner pada RFID untuk Efisiensi *Anti-Collision*

Michelle Theresia 13518050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518050@itb.ac.id

***Abstract*—Sistem RFID (Radio Frequency Identification) adalah sistem untuk melacak barang dengan lebih cepat dan efisien. Sistem RFID dapat mendeteksi beberapa data di satu tempat yang sama. Akan tetapi, karena kemampuan tersebut, dapat terjadi *collision* pada data. Solusi dari masalah tersebut disebut protokol atau algoritma *anti-collision*. Protokol atau algoritma yang digunakan tentunya harus efisien. Makalah ini membahas penerapan algoritma Pohon Biner untuk *anti-collision* pada RFID yang efisien.**

***Keywords*—RFID, *collision*, *anti-collision*, algoritma Pohon Biner.**

I. INTRODUCTION

Sebagian besar produk pada zaman sekarang menggunakan sistem *Universal Product Code* (UPC) *bar code*. *Barcode* digunakan untuk melacak produk yang ada dalam inventori suatu produksi dan untuk mempercepat proses pemeriksaan saat produk keluar dari produsen atau distributor ke tangan konsumen. Barcode harus dipindai satu per satu dengan jarak yang cukup dekat agar hasil pindai yang berupa data produk dapat disimpan ke dalam perangkat keras penyimpanan. Akan tetapi, penduduk dunia yang semakin modern cenderung lebih memilih untuk membeli produk daripada membuat produk sehingga pergerakan produk dari produsen ke konsumen semakin cepat dan banyak. Proses pembacaan *barcode* harus dilakukan satu per satu secara dekat ke pemindai *barcode*. Akibatnya antrian di toko menjadi panjang dan membuang banyak waktu. Tentu hal tersebut menjadi tidak efektif baik untuk produsen maupun konsumen karena pergerakan produk menjadi tidak maksimum. Untuk mengatasi masalah tersebut, RFID pun mulai digunakan oleh beberapa produsen di dunia.

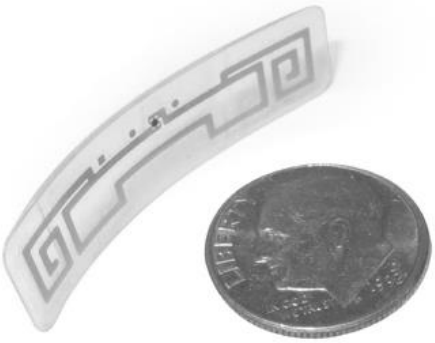
Radio Frequency Identification (RFID) adalah teknologi ADC (*Automated Data Collection*) yang menggunakan frekuensi radio untuk mentransfer data. Sistem RFID terdiri dari Tag RFID, Antena RFID, internet, dan komputer untuk menyimpan data yang ditransfer. Tag RFID ditempelkan pada produk bergerak, umumnya digunakan sama seperti barcode, untuk melacak produk tersebut dari manufaktur hingga sampai ke tangan konsumen. RFID mengirimkan data biner dengan ukuran 64 bit melaui internet, kemudian pemindai RFID akan membaca data dari Tag RFID dan melakukan enkripsi terhadap data tersebut kemudian membandingkan data tersebut dengan data yang telah tersimpan di penyimpanan perangkat keras. Data yang disimpan di Tag RFID dapat diubah, diperbarui, atau dikunci yang merupakan kelebihan dari RFID. Data pada barcode bersifat *read-only* atau tidak dapat diubah. Pemindai RFID dapat mendeteksi Tag RFID bergantung frekuensi yang dipancarkan Tag RFID dan jenis Tag RFID. Umumnya berkisar dari 6 meter hingga 100 meter. Akan tetapi hal tersebut akan menyebabkan pemindai RFID mendeteksi beberapa Tag RFID di suatu tempat, sehingga ada kemungkinan *collision* pada pemanggilan data. *Collision* tersebut dapat dicegah dengan menerapkan algoritma Pohon.

Makalah ini akan membahas tentang penerapan implementasi Pohon pada RFID untuk mencegah *collision* tag.

II. *COLLISION* TAG PADA RFID

Sistem RFID terdiri dari tag RFID, antena RFID, internet, dan host (komputer) untuk menyimpan data yang ditransfer. Tag RFID dapat diklasifikasi menjadi dua jenis, yaitu tag aktif dan tag pasif. Klasifikasi tersebut berdasarkan adanya daya listrik pada tag. Secara singkat, tag aktif merupakan sistem dengan daya baterai, sementara tag pasif tidak memiliki sumber daya tetapi mendapat daya dari energi elektromagnetik yang dipancarkan oleh pemindai RFID. Tag aktif umumnya digunakan sebagai suar untuk melacak secara akurat dan langsung (*real-time*) lokasi dari produk atau aset yang bergerak sangat cepat. Tag aktif dapat dideteksi dari jarak yang lebih jauh dari juga Active tag memiliki dua frekuensi utama, yaitu 433 MHz dan 915 MHz. Frekuensi yang digunakan oleh suatu active tag bergantung penggunaan tag tersebut. Semakin besar frekuensi berarti semakin jauh tag tersebut dapat dipindai suatu RFID reader. Akan tetapi harga tag aktif juga lebih mahal. Tag pasif umumnya digunakan untuk kontrol akses, pelacak file, manajemen supply chain, label pintar, dan masih banyak lagi. Kelemahannya dari tag pasif yaitu dapat dideteksi oleh pemindai RFID dari jarak yang lebih dekat dari tag aktif, tetapi ukuran tag pasif lebih kecil dari tag pasif dan mudah ditempel di barang apa pun. Oleh sebab itu harga tag pasif lebih murah sehingga lebih sering dipakai oleh perusahaan-perusahaan.

Gambar 1 Tag RFID aktif
(sumber: https://blog.atlasrfidstore.com/active-rfid-vs-passive-rfid)



Gambar 2 Tag RFID pasif
(sumber: https://www.indiamart.com/proddetail/passive-steel-rfid-tag-18929735962.html)

Tag RFID dapat dideteksi oleh pemindai RFID ketika pemindai mendapat perintah dari host dan memancarkan sinyal carrier yang kemudian mengenai tag. Tag menerima dan memodifikasi sinyal tersebut lalu mengirim kembali sinyal yang telah dimodulasi. Antena menangkap sinyal yang dikirimkan tag dan mengirimnya ke pemindai RFID yang akan menerjemahkan sinyal tersebut menjadi data yang dapat diterima oleh host.

Antena RFID atau pemindai RFID berfungsi untuk memberi daya pada tag pasif, menginisiasi hubungan data secara dua arah, menyimpan data tag dan menyaring hasilnya, dan melakukan komunikasi dengan server internet. Pemindai RFID dapat memindai 100-300 tag per detik. Bentuk pemindai RFID juga bervariasi bergantung kebutuhan.

Gambar 3 Pemindai RFID
(sumber: http://www.rfidsolutionsinc.com/invengo-xc-rf850-uhf-rfid-reader)

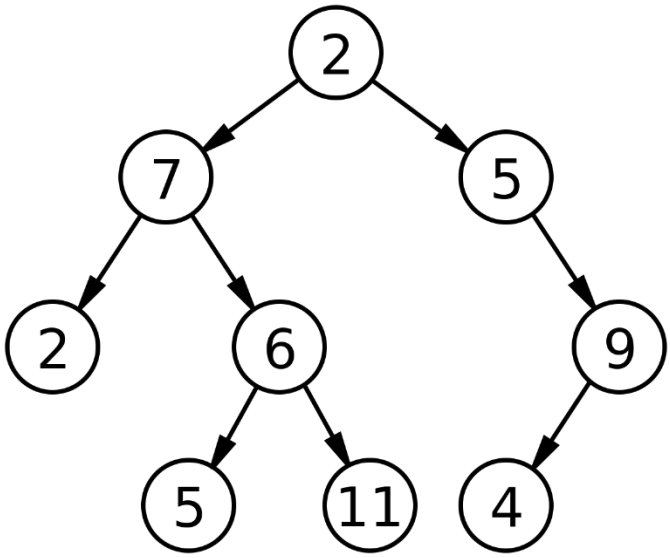
Gambar 4 Pemindai RFID
(sumber: https://www.marketlab.com/asset-management/p/Invisi-Tag-Handheld-RFID-Reader)

Pemindai RFID dapat mengidentifikasi beberapa tag sekaligus. Tag akan membalas saat pemindai RFID mentransmisikan data dalam bentuk angka ‘0’ atau ‘1’ secara acak melalui frekuensi radio. Tag yang jumlahnya sama dengan ‘0’ akan mentransmisikan ID-nya ke pemindai, sementara yang lainnya akan diam. Tag tidak mungkin berkomunikasi satu sama lain dan tag pasif tidak bisa membedakan apakah pemindai sedang sibuk atau tidak, sehingga ada kemungkinan terjadi *collision* pada pemindai RFID. *Collision* terjadi saat ada dua atau lebih tag dengan jumlah ‘0’ mentransmisikan datanya secara bersamaan.

Collision dapat diatasi dengan menggunakan protokol yang biasa disebut protokol *anti-collision*. Contoh protokol: algoritma pohon, *memoryless*, *contactless*, dan I-code.

III. Pohon Biner

Pohon Biner adalah jenis khusus dari Pohon yang memiliki struktur tambahan yang membuatnya sangat berguna sebagai struktur data. Setiap simpul di dalam Pohon Biner memiliki maksimum 2 anak. Anak kiri dan anak kanan pada Pohon Biner dibedakan sehingga Pohon Biner adalah pohon terurut.



Gambar 5 Binary Tree

Ada beberapa istilah dalam Pohon, yaitu:

1. Anak dan Orangtua
   7 dan 5 adalah anak dari 2
   2 adalah orangtua dari 7 dan 5
2. Lintasan
   Lintasan dari 2 ke 4 adalah 2, 5, 9, 4
   Panjang lintasan adalah 3
3. Simpul
   Angka pada Pohon merupakan simpul
4. Daun
   daun adalah simpul yang tidak memiliki anak

Pohon Biner dimanfaatkan dalam banyak hal. Contoh pemanfaatan Pohon Biner yaitu Pohon Ekspresi, parsing, pembacaan token, dan lain-lain.

IV. PROTOKOL *ANTI-COLLISION*

A. Definisi

Protokol *anti-collision* harus memiliki karakteristik:

1. Pemindai harus dapat mengidentifikasi semua tag di dalam jarak kemampuan pemindai
2. Algoritma *anti-collision* harus mempunyai mekanisme yang mampu memverifikasi semua tag yang teridentifikasi
3. Protokol harus meminimalisir waktu yang dibutuhkan untuk mengidentifikasi tag dan *collision*. Semakin lama waktu yang dibutuhkan untuk mengidentifikasi tag maka semakin sulit pemindai mengidentifikasi objek yang bergerak cepat.

Protokol *anti-collision* terklasifikasi menjadi dua jenis pendekatan berdasarkan cara protokol menentukan waktu transmisi, yaitu pendekatan probabilistik dan deterministik. Skema tag *anti-collision* probabilistik berdasarkan ALOHA, yaitu skema dasar dari kontrol akses. Pada ALOHA, tiap tag menghasilkan nomor secara acak dan menunggu waktu transmisi bergantung nomor yang terpilih. Jika data yang ditransmisikan oleh tag tidak diinterferensi oleh data lain maka pemindai dapat mengidentifikasi tag. Tag terus menerus melakukan hal yang sama untuk waktu yang acak. Jika dalam waktu yang sama ada dua tag atau lebih metransmisikan data maka akan terjadi *collision*. Untuk menyelesaikan sebagian masalah *collision*, waktu transmisi dibagi menjadi selang yang diskrit di dalam slot ALOHA. Semua tag berusaha untuk mentransmisikan data setelah back-off (tertahan) secara acak. Jika tidak ada *collision* parsial, ALOHA akan menggandakan jumlah saluran. Slot pada ALOHA dapat digabung menjadi *frame*. Tiap *frame* terdiri dari beberapa slot transmisi. Tags menentukan slot transmisi saat mereka menerima pesan inisialisasi dari pemindai berdasarkan nomor acak yang dihasilkan oleh tag.

Faktor penting yang sangat memengaruhi perfoma adalah hubungan antara nomor tag dengan rentang nomor pada ruang kosong di frame, jumlah maksimum dari pengatur waktu back-off. Jika rentang nomor acak yang dihasilkan oleh tag lebih besar dari rentang nomor tag yang dapat diterima pemindai, maka mungkin akan terjadi banyak slot yang mengalami *collision*. Sebaliknya, jika rentang tag lebih kecil maka frame akan memiliki banyak slot kosong.

Berdasarkan jumlah transmisi tag pada suatu frame, slot dapat dibedakan menjadi 3 tipe:

1. Readable: tepat 1 tag mentransmisikan data dan pemindai berhasil mengidentifikasi tag
2. Collided: dua atau lebih tag mentransmisikan data dan pemindai tidak dapat mengidentifikasi keduanya
3. Idle: tidak ada tag yang mentransmisikan data

Dalam skema ALOHA, ukuran frame berarti ukuran dari ruang kosong di frame ALOHA. Ukuran frame mudah diubah setiap frame mulai. Ada banyak skema yang dapat mengoptimalkan frame ALOHA. Ukuran frame optimum saat nomor di tag dan ukuran frame tepat sama.

Skema tag *anti-collision* deterministik, tag menentukan titik transmisi saat menerima pesan dari pemindai dan membuat proses dari pesan. Pemindai membagi tag menjadi dua grup. Pemindai membagi salah satu grup tersebut menjadi dua lagi. Pemindai harus dapat membedakan tiap grup yang telah dibagi. Proses membagi tag terus berlanjut sampai salah satu grup mengandung satu tag. Proses pembagian grup tersebut terus berlanjut hingga pemindai mengidentifikasi semua tag.

B. Algoritma Pohon Biner dalam Pembagian

Algoritma pohon biner merupakan skema probabilistik dalam pencarian pohon. Asumsi pemindai bisa mengatasi *collision* saat angka *collision* tidak lebih dari N, maka penyelesaian tersebut menggunaka algoritma Pohon Biner. Berikut adalah *flowchart* dari algoritma Pohon Biner pada penyelesaian *collision* tag.

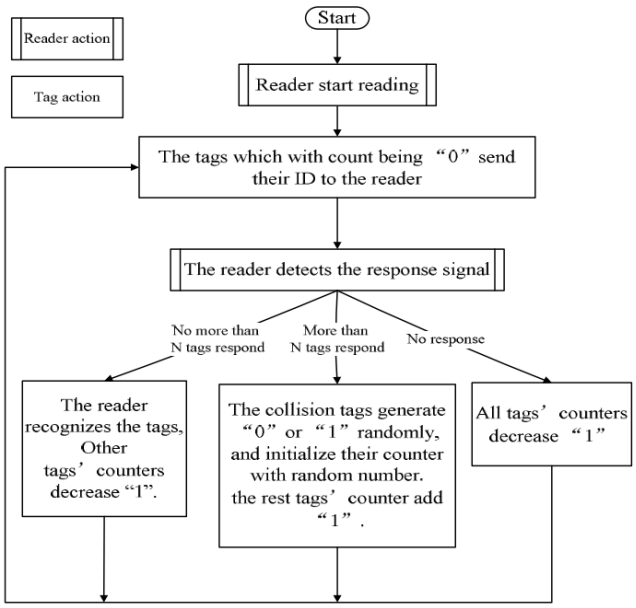


Figure 6 Flowchart pohon biner sebagai anti-collision

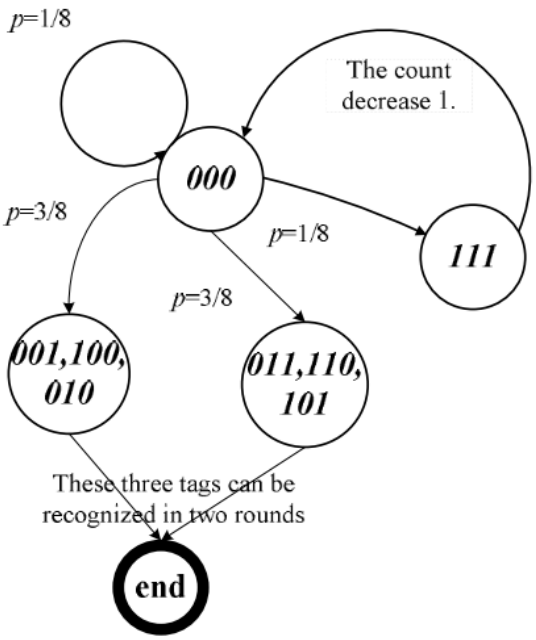
V. PERHITUNGAN EFISIENSI

Asumsi pemindai RFID dapat mengatasi *collision* dua tag. Perhitungan efisiensi dimulai dari nomor *collision* 3 dan 4. Kemudian, metode rekursi digunakan untuk menyimpulkan efisiensi identifikasi untuk nomor tag yang berubah-ubah. Berikut adalah persamaan untuk efisiensi.

$$eff_n = \frac{n}{E(T_n)} \quad (1)$$

Keterangan:
eff_n = efisiensi system
n = nomor tag
Ti = pembulatan pemindai untuk mengenali nomor pada tag i
E(X) = nilai yang dieskpektasi dari X

1. n = 3, tiga tag
   Nilai mula-mula di tiap penghitung tag adalah '0', pemindai akan mendeteksi *collision* untuk pertama kalinya. Setelah *collision* pertama, tiap tag akan menghasilkan '0' atau '1' secara acak dan dicatat di penghitung. Nyatakan nilai penghitung dari ketiga tag sebagai 'c1 c2 c3'. Delapan kasus S1. S2, S3, S4, S5, S6, S7, S8 mungkin terjadi saat proses mengidentifikasi dua tag dari kasus S seperti berikut.
   S = {initial state, 'c1 c2 c3' = '000' };
   S1= {dari kasus *S*, setelah *collision*, 'c1 c2 c3' = '000'};
   S2= {Dari kasus *S*, setelah *collision*, 'c1 c2 c3' = '001'};
   S3= {Dari kasus *S*, setelah *collision*, 'c1 c2 c3' = '010'};
   S4= {Dari kasus *S*, setelah *collision*, 'c1 c2 c3' = '100'};
   S5= {Dari kasus *S*, setelah *collision*, 'c1 c2 c3' = '011'};
   S6= {Dari kasus *S*, setelah *collision*, 'c1 c2 c3' = '101'};
   S7= {Dari kasus *S*, setelah *collision*, 'c1 c2 c3' = '110'};
   S8= {Dari kasus *S*, setelah *collision*, 'c1 c2 c3' = '111'};
   Transisi keadaan mengidentifikasi proses dapat dideskripsikan seperti gambar 7.



Gambar 7 Flowchart transisi keadaan proses identifikasi

Urutan angka dalam lingkaran mewakili jumlah dalam penghitung tiga tag. Panah mewakili transisi keadaan 'akhir' berarti pembaca mengenali semua tag dan identifikasi proses selesai. p adalah probabilitas transisi negara. Jelas bahwa {S} = {S1, S2, S3, S4, S5, S6, S7, S8} didefinisikan T, T1, T2, T3, T4, T5, T6, T7, T8 sebagai banyak pengulangan pemindaian dalam mengenali dua tag ini dari kasus S, S1, S2, S3, S4, T5, T6, T7, T8. Kemudian hitung nilai waktu identifikasi yang diharapkan sebagai berikut.

E(T) = E(T1)P(T1) + E(T2)P(T2) + E(T3)P(T3) + E(T4)P(T4) + E(T5)P(T5) + E(T6)P(T6) + E(T7)P(T7) + E(T8)P(T8)

dimana,

P(T1) = P(T2) = P(T3) = P(T4) = P(T5) = P(T6) = P(T7) = P(T8) = 1/8

dari kasus S ke kasus S2, S3, S4, S5, S6, atau S7, itu membutuhkan 3 banyak pengulangan pemindaian untuk selesai mengidentifikasi 3 tag berikut sehingga E(T2) = E(T3) = E(T4) = E(T5) = E(T6) = E(T7) = 3; E(T) sama dengan perkiraan T_3, maka E(T) = E(T_3); E(T1) membutuhkan satu lagi banyak pengulangan pemindaian dari E(T) dan E(T8) membutuhkan dua lagi banyak pengulangan pemindaian dari E(T). Oleh karena itu, terbentuk tabel berikut yang mencantumkan probabilitas setiap nilai dan banyak pengulangan pemindaian yang diperlukan untuk mengenali kedua tag.

$$\begin{cases} 000 & 1/8 & (1+T_3) \\ 011,101,110 & 3/8 & (3) \\ 001,100,010 & 3/8 & (3) \\ 111 & 1/8 & (2+T_3) \end{cases} \quad (2)$$

Kemudian dapat dibentuk,

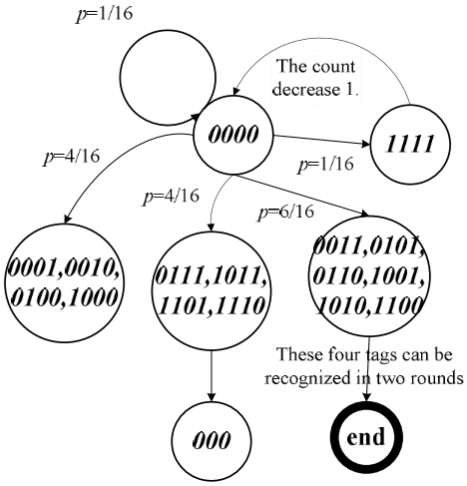
$$E(T_3) = \frac{1}{8}(1+E(T_3)) + \frac{3}{8}\times 3 + \frac{3}{8}\times 3 + \frac{1}{8}(2+E(T_3)) \quad (3)$$

$$\Rightarrow E(T_3) = \frac{7}{2} = 3.5$$

Maka dapat disimpulkan efisiensi dari tiga tag adalah

$$eff_3 = \frac{3}{E(T_3)} = 0.8571 \quad (4)$$

2. n = 4, empat tag
   Proses analisisnya sama dengan kasus di atas. Setelah *collision* pertama, penghitungan di penghitung empat tag ini memiliki nilai yang memungkinkan '0000', '0001', '0010', '0100', '1000', '0011', '0101', '1001', ' 0110 ',' 1010 ',' 1100 ',' 0111 ',' 1011 ',' 1101 ',' 1110 ',' 1111 '. Proses mengidentifikasi ketiga tag ini dapat digambarkan sebagai berikut.



Gambar 8 Transisi perpindahan untuk proses identifikasi empat tag

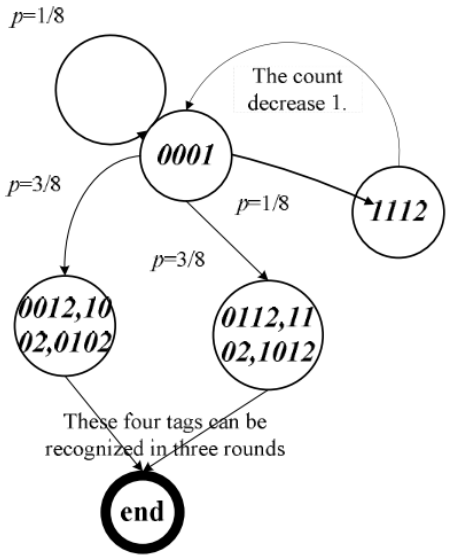
Probabilitas setiap nilai dan banyak pengulangan pemindaian diperlukan untuk mengidentifikasi semua tags adalah

$$\begin{cases} 0000 & 1/16 & (1+T_4) \\ 0001,0010,0100,1000 & 4/16 & (1+T_{0001}) \\ 0011,0101,1001,0110,1010,1100 & 6/16 & (3) \\ 0111,1101,1011,1110 & 4/16 & (2+T_3) \\ 1111 & 1/16 & (2+T_4) \end{cases} \quad (5)$$

T001 menunjukakan banyak pengulangan pemindaian dari ‘0001’ yaitu empat dalam penghitung. Keempat perhitungan tersebut menunjukkan kemungkinan nilai 0002, 0012, 0102, 1002, 0112, 1012, 1102, 1112 dari keadaan ‘0001’. Proses mengidentifikasi empat tag dari ‘0001’ dapat dijelaskan seperti yang diberikan gambar

Probabilitas setiap nilai dan banyak pengulangan pemindaian dari ‘0001’ terdaftar sebagai (6).

$$\begin{cases} 0002 & 1/8 & (2+T_{0001}) \\ 0112,1012,1102 & 3/8 & (4) \\ 0012,1002,0102 & 3/8 & (4) \\ 1112 & 1/8 & (2+T_{0001}) \end{cases} \quad (6)$$



Gambar 9 Transisi perpindahan untuk proses mengidentifikasi empat tag

Nilai yang diekspektasi T_{001}, $E(T_{001})$ dihitung sebagai

$$E(T_{0001}) = \frac{1}{8}\times(2+E(T_{0001})) + \frac{3}{8}\times 4 + \frac{3}{8}\times 4 + \frac{1}{8}\times(2+E(T_{0001})) \quad (7)$$

$$\Rightarrow E(T_{0001}) = \frac{14}{3}$$

Nilai yang diekspektasi dari T_4, $E(T_4)$ dihitung sebagai

$$E(T_4) = \frac{1}{16}(1+E(T_6)) + \frac{4}{16}(1+E(T_{0001})) + \frac{4}{16}\times 3 + \frac{4}{16}(2+E(T_3)) + \frac{1}{16}(2+E(T_4)) \quad (8)$$

$$\Rightarrow E(T_4) = \frac{65}{14} = 4.64$$

Efisiensi system dengan 4 tag yaitu

$$eff_4 = \frac{4}{E(T_4)} = 0.8615$$

3. Untuk n tag
   Misal terdapat $E(T_k)(k = 1,2,3\ldots n-1)$ saat nomor tag sama dengan 1,2,3…n-1. Untuk k = n, jumlah di penghitung tag memiliki perbedaan kasus 2^n setelah *collision* pertama. Probabilitas tiap nilai dan jumlah pengulangan harus dapat mengidentifikasi semua tag dari kasus berbeda yang terdaftar sebagai (10).

$$\begin{cases} 00.....0 & 1/2^n & (1+T_n) \\ \underbrace{000..0}_{x}\underbrace{11..1}_{y} & C_n^x/2^n & 1+T_{\underbrace{00..0}_{x}\underbrace{11..1}_{y}} \\ ..... & & \\ 11.....1 & 1/2^n & (2+T_n) \end{cases} \quad (10)$$

   Dimana,
   Tn: jumlah pengulangan pemindaian yang mengenali sebanyak n tag
   $T_{\underbrace{00..0}_{x}\underbrace{11..1}_{y}}$: jumlah pengulangan pemindaian dari jumlah

tag $\underbrace{00..0}_{x}\underbrace{11..1}_{y}$', $x+y=n; 1 \le x \le n-1$;

Misal banyak pegulangan pemindaian dari jumlah di penghitung tag $(\underbrace{00..0}_{n-1},1)$ adalah T_n'. Jumlah $(\underbrace{00..0}_{n-1},1)$ mungkin berubah bergantung kasus yang terdaftar sebagai (11).

$$\begin{cases} & probability \quad reading \ round \\ 00.....0,2 & 1/2^{n-1} \quad (2+T_n') \\ \underbrace{000..0}_{l}\underbrace{11..1}_{m},2 & C_{n-1}^{l}/2^{n-1} \quad (1+T_{l+1}'-1+T_{m+1}') \\ & l+m=n\text{-}1, 1 \le l \le n-2 \\ ..... & \\ 11.....1,2 & 1/2^{n-1} \quad (2+T_n') \end{cases} \tag{11}$$

Kemudian $E(T_n')$, ekspetasi nilai T_n' bisa dihitung sebagai berikut

$$E(T_n') = \frac{1}{2^{n-1}}(2+E(T_n')) + \sum_{l=1}^{n-2}\frac{C_{n-1}^{l}}{2^{n-1}}(E(T_{l+1}')+E(T_{m+1}')) + \frac{1}{2^{n-1}}(2+E(T_n'))$$

Atur persamaannya akan didapatkan

$$E(T_n') = \frac{\frac{4}{2^{n-1}} + \sum_{l=1}^{n-2}\frac{C_{n-1}^{l}}{2^{n-1}}(E(T_{l+1}')+E(T_{n-l}'))}{(1-\frac{1}{2^{n-2}})} \tag{12}$$

Dengan (12), akan didapatkan $E(T_3') = \frac{14}{3}$, yang sama dengan nilai yang telah dihitung di (7).

(12) dapat diserdahanakan menjadi

$$(2^{n-1}-2)E(T_n') = 4 + \sum_{l=1}^{n-2}C_{n-1}^{l}E(T_{l+1}') + \sum_{l=1}^{n-2}C_{n-1}^{l}E(T_{n-l}') \tag{13}$$

Karena l+m – n-l, formula $\sum_{l=1}^{n-2}C_{n-1}^{l}E(T_{n-l}')$ bisa disederhanakan menjadi:

$$\sum_{l=1}^{n-2}C_{n-1}^{l}E(T_{n-l}') = \sum_{l=1}^{n-2}C_{n-1}^{n-1-l}E(T_{n-l}') = \sum_{m=1}^{n-2}C_{n-1}^{m}E(T_{m+1}') = \sum_{l=1}^{n-2}C_{n-1}^{l}E(T_{l+1}') \tag{14}$$

Substitusi (14) ke (13), maka akan didapatkan (15).

$$E(T_n') = \frac{(2+\sum_{l=1}^{n-2}C_{n-1}^{l}E(T_{l+1}'))}{(2^{n-2}-1)} \tag{15}$$

Saat jumlah adalah $\underbrace{000..0}_{x}\underbrace{11..1}_{y}$', x+y = n, jumlah pengulangan pemindaian adalah $T_{\underbrace{00..0}_{x}\underbrace{11..1}_{y}} = T_{x+1}' - 1 + T_y$ adalah

$$E(T_n) = \frac{1}{2^n}(1+E(T_n)) + \frac{1}{2^n}(2+E(T_n)) + \sum_{x=1}^{n-1}\frac{C_n^x}{2^n}(1+E(T_{x+1}')-1+E(T_{n-x})) \tag{16}$$

$$\Rightarrow E(T_n) = \frac{\frac{3}{2^n}+\sum_{x=1}^{n-1}\frac{C_n^x}{2^n}(E(T_{x+1}')+E(T_{n-x}))}{\frac{2^{n-1}-1}{2^{n-1}}}$$

didapatkan nilai ekspektasi $E(T_n)$. Efisiensi sistem dapat dihitung menjadi (17).

$$eff_n = \frac{n}{E(T_n)} = \frac{n}{\frac{\frac{3}{2^n}+\sum_{x=1}^{n-1}\frac{C_n^x}{2^n}(E(T_{x+1}')+E(T_{n-x}))}{\frac{2^{n-1}-1}{2^{n-1}}}} \tag{17}$$

Disimpulkan efisiensi identifikasi sistem adalah (17). Semakin kuat kemampuan *anti-collision*, semakin kecil nilai pertama $E(T_n)$ dan $E(T_{n'})$ yang menuju peningkatan substansial dalam efisiensi sistem. Apabila nilai n menjadi sangat besar maka dapat menggunakan sistem identifikasi menggunakan algoritma pohon biner yang telah disederhanakan:

$$eff = \lim_{n\to\infty}\frac{n}{E(T_{n+1}')} \tag{18}$$

Berdasarkan simulasi yang dilakukan oleh Choi J. dan Lee W., efisiensi *anti-collision* dengan Pohon Biner sebesar 85,7%.

VI. CONCLUSION

RFID dapat memindai lebih dari satu tag RFID dan tag RFID akan merespon dengan mentransmisikan data berbentuk '0' atau '1'. Apabila jumlah data sama dengan '0', maka data dari tag RFID akan diterima oleh pemindai. Akan tetapi, apabila ada dua atau lebih tag RFID mentransmisikan data berjumlah '0' secara bersamaan, maka akan terjadi *collision*. *Collision* tersebut dapat diatasi dengan *anti-collision*, yang lebih efisien menggunakan algoritma Pohon Biner. Buktinya, efisiensi algoritma Pohon Biner sebagai *anti-collision* mencapai 85,7%.

REFERENCES

[1] https://cse.buffalo.edu/~rapaport/191/S09/whatisdiscmath.html (Diakses tanggal 29 November 2019)

[2] https://blog.atlasrfidstore.com/active-rfid-vs-passive-rfid (diakses tanggal 30 November 2019)

[3] https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/rfid3.htm (diakses tanggal 29 November 2019)

[4] https://www.cse.iitb.ac.in/~sri/talks/rfid-05.pdf (diakses tanggal 29 November 2019)

[5] Choi J., Lee W. (2007) Comparative Evaluation of Probabilistic and Deterministic Tag Anti-collision Protocols for RFID Networks. In: Denko M.K. et al. (eds) Emerging Directions in Embedded and Ubiquitous Computing. EUC 2007. Lecture Notes in Computer Science, vol 4809. Springer, Berlin, Heidelberg

[6] Kenneth H. R, *Discrete Math in Computer Science*. McGraw-Hill, 2003, ch. 4.

[7] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf (diakses tanggal 30 November 2019)

[8] Y. Cui, "System efficiency of collision recover binary tree algorithm in RFID," *2012 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*, Nice, 2012, pp. 408-412.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Desember 2019

Michelle Tberesia 13518050