Contents

1 Terminale Nascosto, da whatsapp

1.1 Problema

abemus A, B, e C A comunica con B, C vuole comunicare con B C è fuori dalla portata di A, e non sa che A sta parland con con B C manda il suo pacchetto a B, che riceve sia da A a che da C, collisione su B, che non capisce un cazzo e perde i pacchi

1.2 Soluzione

imposta in tutti i nodi una three way handshake un pacchetto RTS e uno CTL

quindi adesso quando A manda a B prima deve

- mandargli un RTS
- e se B è libero, B riceve l'RTS, e manda a tutti i terminali nel suo range un CTL
- A, ricevendo il CTS, sa che B è libero, e manda

- C, che ha ricevuto il CTS senza aver mandato un RTS, sa che B è occupato, e aspetterà il suo turno

2 Terminale Esposto, da whatsapp

la soluzione al problema del terminale nascosto porta alla creazione del cosiddetto *problema del terminale esposto*, vale a dire

2.1 Problema

C riceve il CTS da B, e se dovesse ricevere un RTS da D, altro terminale a cazzo allora questi due, (RTS di D e CTS di B) si annullano e C non manda un cazzo visto che D sta apsettando un CTS per mandare, e su C collisione a quanto pare, allora D non riceve il CTS da C e non manda un cazzo pure lui

2.2 Soluzione

questo si risolve impostando un tempo di back-off che poi viene messo nel NAV di ogni nodo

3 Terminale Nascosto, Fantacci

procedura di handshake (riscritta in C++ perchè i diagrammi di flusso mi fanno gottare)

```
void Terminale::aspetta_alla_cazzo() {
    this->wait_time(IFS);
    int r = rand();
    this->wait_time(r);
}

void Terminale::manda_e_aspetta(Canale* c) {
    this->send_frames_on_channel(c);
    this->wait_time(SIFS);
}

Paccketto Terminale::ricevi_ack(Canale* c) {
    this->manda_e_aspetta(c);
    return this->wait_for_package(c);
```

```
}

void Terminale::accesso_canale(Canale* c) {
    int volte_che_non_mhanno_cagato = 0; // k
    this->wait_for_free_channel(c);
    this->wait_time(DIFS);
    if(c->is_free()) {
        this->aspetta_alla_cazzo();
    }
    this->manda_e_aspetta(c);
    if(this->ricevi_ack(c) == Pacchetto::due_di_picche) {
        volte_che_non_mhanno_cagato ++ ;
    }
    else {
        this->boh_apro_connessione(c);
        return;
    }

    /* e ora comincia il ciclo di non farsi cagare */
    while(volte_che_non_mhanno_cagato < 15) {
        this->aspetta_alla_cazzo();
        this->manda_e_aspetta();
        if(t->ricevi_ack() == Paccketto::due_di_picche) {
            volte_che_non_mhanno_cagato ++;
        }
        else {
            this->boh_apro_connessione(c);
            return;
        }
    }
    std::cout<<"mai na gioia, mi arrendo"<<std::endl;
    this->piangi_in_un_angolino();
    return;
}
```

3.1 handshake

la fase di handshake ha tre fasi

- conclusa la fase di accesso, se il canale è ancora libero, A manda un messaggio RTS richiesta di collegamento a B

- se B riceve l'RTS correttamente allora aspetta $SIFS$ e manda un messaggio di conferma (CTS) ai vicini
- A riceve il CTS da B e, atteso un tempo $SIFS$, sa che adesso può mandare roba a B.

conclusa la fase di handshake la gestione delle successive fasi di accesso avvengono secondo normalissimo manda e ACK

4 Terminale Esposto, Fantacci

4.1 Contesto

nel problema del terminale nascosto

l'invio del messaggio RTS viene visto da tutti i terminali nel raggio di A, che vedono un gigantesto "A vorrebbe parlare con B" e pensano "ok, eviterò di farmi i cazzi di A o B per ora"

il messaggio "ciao sono A vorrei parlare con B" non viene visto da C visto che C non è nel raggio di A.

quello che C vede sarà il CTS di "ciao sono B e mi metto a parlare con A", con cui sa che "ok, non mi farò i cazzo di B"

4.2 Problema

il ricevimento da parte di C del messaggio CTS gli impedisce di inviare o ricevere per un certo tempo (quello nel campo duration id). ci si accorge subito che utilizzare un handshake abbia effetti sull'efficienza della rete. si riducono le collisioni ma ci vuole più tempo a cominciare a mandare roba, per limitare l'efficienza o meno della rete occorre che i frame da mandare siano abbastanza lunghi da rendere trascurabile i tempi di setup (e/o teardown)

4.2.1 Inibizione dei pacchetti post CTS

ok, abbiamo un terminale che ha ricevuto un CTS uscito da una handshake che non lo riguardava e ora sta zitto per evitare che i terminali provino ad attivare fasi di attivazione canale quando questo è già attivo è stato introdotto in ogni terminale un ulteriore contatore a decremento chiamato `Network Allocation Vector`[1]

si è visto che il header MAC contiene un campo che indica la durata della trasmissione un duration id, appena un terminale vede invia un RTS o CTS

[1] il contatore si chiama vettore, ma vaffanculo

, tutti i nodi nella sua area di copertura leggono questo campo e impostano il loro NAV, come "timer de li cazzi tua", al duration id del header mac.

poi solito di prima, C riceve il "fatti li cazzi tua", D non è in range, prova a contattare C, e C sta aspettando il CTS quindi non si accorge dell'RTS di D, tutti stanno male, amen.

4.2.2 Monopolio

Un altro problema che può sorgere è che certi canali poi abbiano un accesso esclusivo al canale per troppo tempo a discapito di altri nodi della rete

lo standard prevede quindi la possiblità di assegnare a un nodo anche un tempo massimo di trasmissione, indicato come Transmission Opportunity