Alma Mater Studiorum – Università di Bologna
Corso di Laurea in Informatica

Un approccio cross-layer all'affidabilità guidata dalle applicazioni

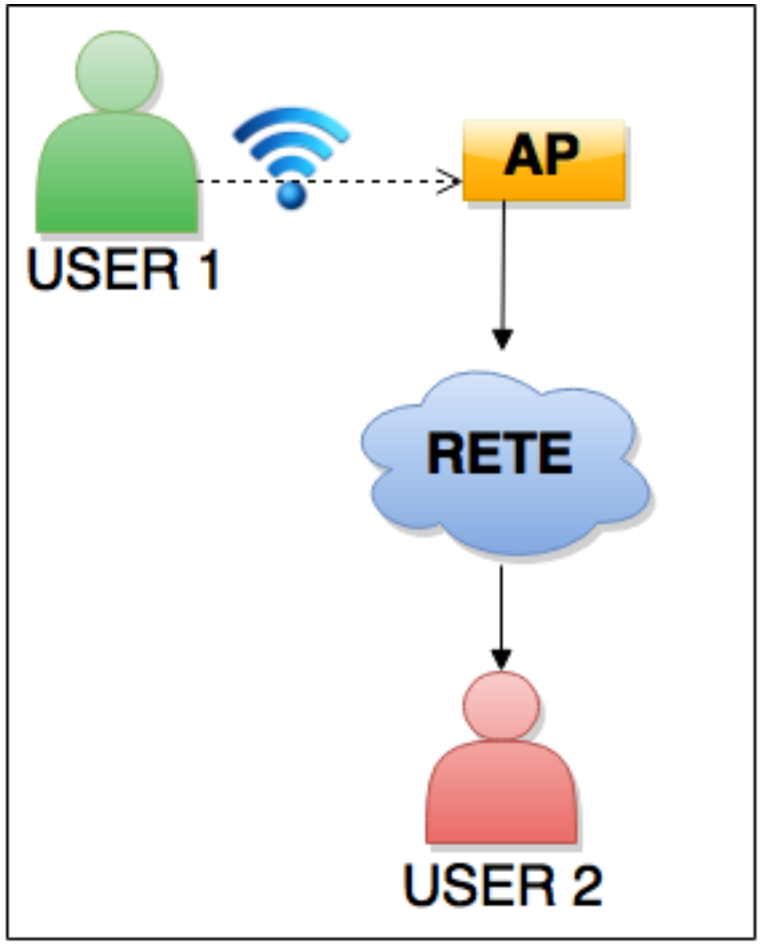
Relatore:
Chiar.mo Prof. Vittorio Ghini

Presentata da:
Alessandro Mengoli

SCENARIO

- Trasmissione VoIP tramite protocollo Wi-Fi

- Dispositivo mobile con più interfacce di rete, di cui almeno una Wi-Fi



OBIETTIVO

TED (Transmission Error Detector)

- Fa parte di ABPS
- Analizza la qualità del segnale
- Indica pacchetti non ricevuti dall'Access Point

Implementato per kernel Linux 4.0.

Verranno spiegate API fornite e la fase di sperimentazione.

API

Fornite API per:

- Invio del pacchetto: `sendmsg` modificata
- Ricezione della notifica: `recvmsg` modificata

API *SENDMSG*

Per sfruttare meccanismo TED, l'applicazione deve utilizzare la syscall `sendmsg`:

- Settando il campo della struttura `struct cmsghdr cmsg_type` al nuovo valore `ABPS_CSMG_TYPE`.
- Specificando, negli ancillary data (sequenza di `struct msghdr`) del messaggio, un puntatore ad una variabile intera.
  - La syscall riempirà la variabile con id del pacchetto inviato

API *RECVMSG*

Nella `recvmsg` è necessario inserire il flag `MSG_ERRQUEUE` per ricevere messaggi di errore.

`Recvmsg` restituisce struttura dati con informazioni su esito trasmissione:

- ACK/NACK da parte di Access Point
- Tempi ricezione notifica
- Retry count, numero tentativi invio pacchetto

APPLICAZIONE

Realizzata applicazione che permette di:

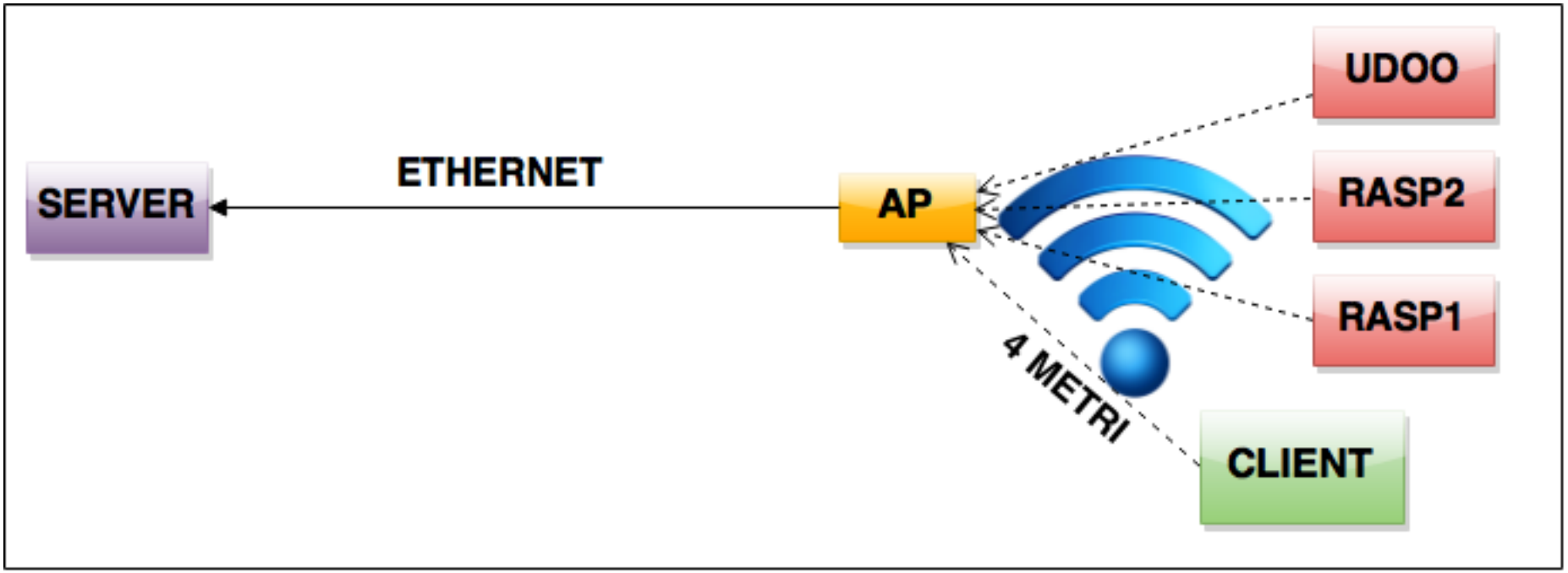
- Rilevare ACK pacchetto
- Conoscere tempo di ricezione notifica da Access Point
- Sapere numero di retry count

Senza modifica kernel non sarebbe stato possibile.

Applicazione e test realizzati:

- Salvando dati in JSON
- Analizzando risultati con scripts Python

TEST-CONFIGURAZIONI



- Trasmissione in movimento
- Presenza di ostacoli tra AP e dispositivo
- Presenza di alto traffico di rete
- Dimensione pacchetti

TEST - PARAMETRI

Parametri valutati:

- ACK/NACK
- Tempo ricezione notifica
- Retry count per ogni pacchetto
- Versione IP
- Quality of Service (QoS) VoIP
- Velocità trasmissione degli altri hosts, misurata tramite applicazione specificatamente creata
- Frammentazione pacchetti inviati

TEST - QOS

Linee guida della qualità del segnale VoIP impongono:

- Ritardo massimo ricezione pacchetto 150 ms
- Percentuale pacchetti persi < 3%

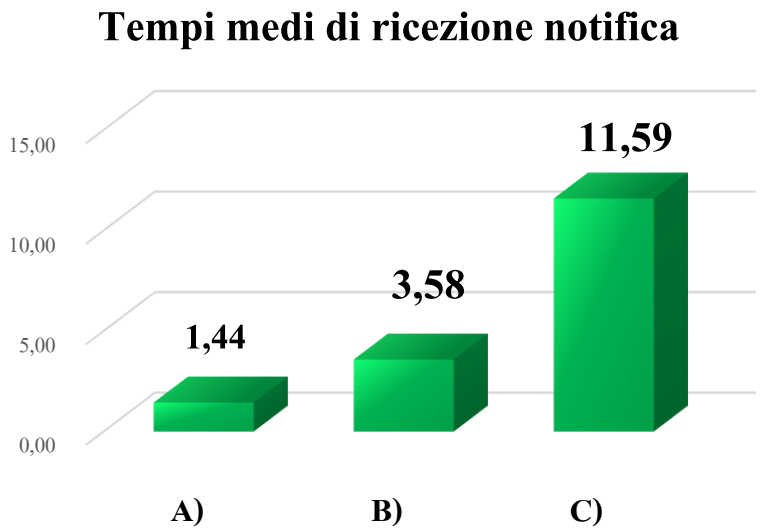
Analizzata come stima la qualità della trasmissione con AP, da questa si può capire andamento QoS VoIP

TEST - NACK

Numero NACK consecutivi importante per trasmissione in movimento.

Elevato numero può comportare perdita di informazioni.

RISULTATI TRAFFICO



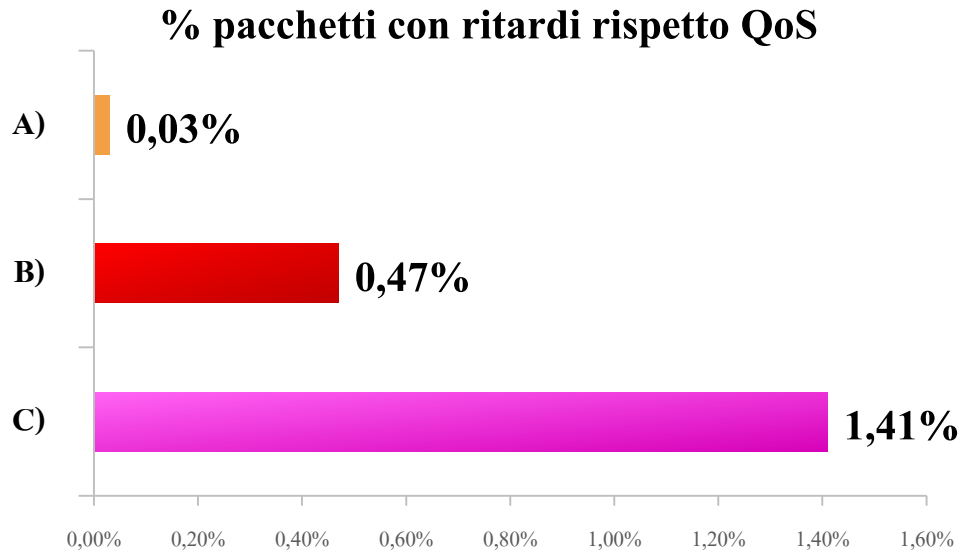
A) Client da solo nella rete, trasmette 5000 pacchetti da 1096B l'uno.

B) Aggiunto Raspberry che trasmette continuamente file da 71MB alla velocità di 31 Mb/s.

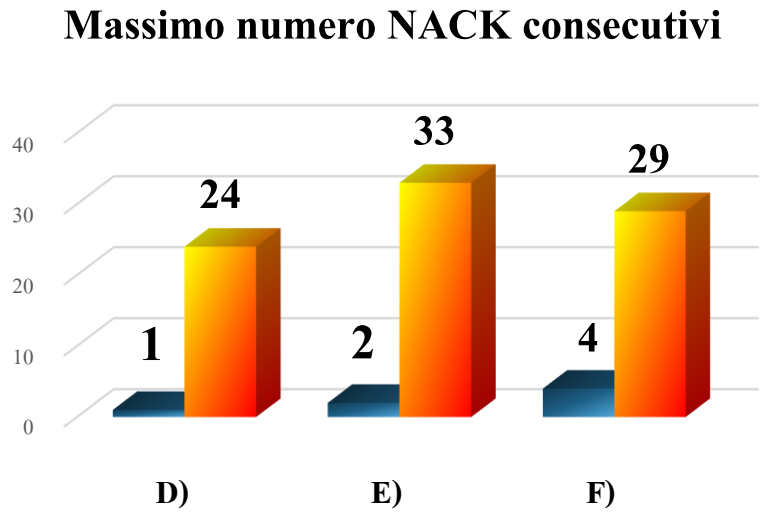
C) Raspberry crea traffico a velocità di 5Mb/s.

Analizzati:

- Media numero retry
- Media tempi notifica
- QoS VoIP



RISULTATI MOVIMENTO



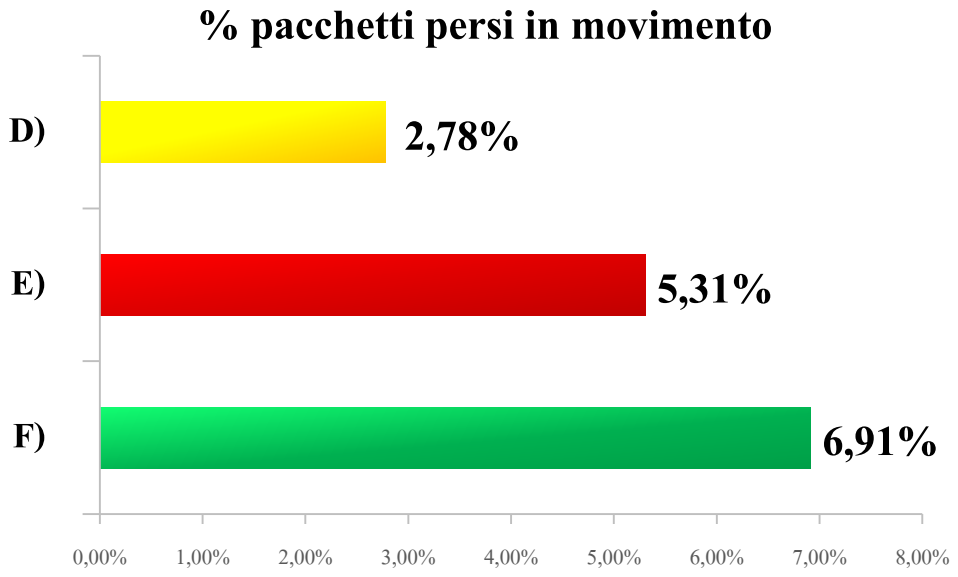
D) Client solo nella rete.

E) Aggiunta di un Raspberry che trasmette continuamente file da 71MB a velocità di 15Mb/s.

F) Aggiunta di un altro Raspberry con le stesse configurazioni.

Analizzati:

- Numero NACK consecutivi
- Media tempi notifica
- QoS VoIP



CONCLUSIONI E SVILUPPI FUTURI

Realizzato porting su kernel Linux versione 4.0

Fornite le API e realizzato un prototipo di applicazione.

Eseguiti test per verificare il funzionamento e capire l'andamento della comunicazione.

Sviluppi futuri

- Porting su Android.
- Realizzazione app per gestione frammentazione.

RECVMSG

```
for(cmsg = CMSG_FIRSTHDR(message); cmsg; cmsg = CMSG_NXTHDR(message, cmsg))
 {
        if((cmsg->cmsg_level == IPPROTO_IPV6) && (cmsg->cmsg_type == IPV6_RECVERR))
        {
                first_hop_transmission_notification = (struct sock_extended_err *) CMSG_DATA(cmsg);

                switch (first_hop_transmission_notification->ee_origin)
                {
                        /* new origin type introduced  */
                        case SO_EE_ORIGIN_LOCAL_NOTIFY:
                        {
                            if(first_hop_transmission_notification->ee_errno == 0)
                            {
                          uint32_t identifier = ted_message_identifier_from_notification(first_hop_transmission_notification);

                           printf("Just got a new notification for a message marked with identifier \%" PRIu32 ". \n", identifier);

                            /* perform some cool actions with the data fetched from TED notification*/.
                               }
                        }
                 }
        }
}
```

SENDMSG

```
uint32_t identifier;

uint32_t *pointer_for_identifier = &identifier;

char ancillary_buffer[CMSG_SPACE(sizeof (pointer_for_identifier))];

struct iovec iov[3]; /* Scatter-Gather I/O */

struct msghdr message_header;

struct cmsghdr *cmsg;

char buffer[MESSAGE_LENGTH];

memset(buffer,0, MESSAGE_LENGTH);

strncpy(buffer,"Hello, from an app built on top of TED!", MESSAGE_LENGTH);


iov[0].iov_base = (void *) buffer;
iov[0].iov_len = strlen(buffer);

/* struct sockaddr_in for destination host */
message_header.msg_name = (void *) &destination_address;
message_header.msg_namelen = sizeof (destination_address);

message_header.msg_iov = iov;          /* message content*/
message_header.msg_iovlen = 1;
```

SENDMSG

```
message_header.msg_control = ancillary_buffer;
message_header.msg_controllen = sizeof(ancillary_buffer);

/* get first struct cmsg from message_header */
cmsg = CMSG_FIRSTHDR(&message_header);

cmsg->cmsg_level = SOL_UDP;

/* new type introduced for cmsg_type field defined in socket.h */
cmsg->cmsg_type = ABPS_CMSG_TYPE;

cmsg->cmsg_len = CMSG_LEN(sizeof(pointer_for_identifier));

/* accessing cmsg_data field */
pointer = (char *) CMSG_DATA(cmsg);

/* copying pointer to variable in cmsg_data field*/
memcpy(pointer, &pointer_for_identifier, sizeof(pointer_for_identifier));

message_header.msg_controllen = cmsg->cmsg_len;


/* Send the message. */
result_value = sendmsg(file_descriptor, &message_header, MSG_NOSIGNAL);
```