



Temi per attività di progetto

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Corso di Reti di Telecomunicazioni C, a.a. 2004/2005
<http://www.tlc.unipr.it/veltri>



Note

- Qui di seguito vengono riportati alcuni possibili temi per le attività di progetto all'interno del corso di Reti di Telecomunicazioni C
- Tali argomenti sono intesi come tracce e rimangono aperti a modifiche e nuove proposte
- I temi proposti possono risultare di differente difficoltà, questo non deve essere un problema ma solo una opportunità
- Per comodità l'elenco è stato diviso per argomenti:
 - **Multicast**
 - **IPv6**
 - **IP Telephony**
 - **QoS**



Multicast

- Realizzazione di una porzione rete con supporto di IP multicast
 - studio del supporto IP multicast su piattaforma Linux
 - realizzazione di una rete IP con supporto IP multicast (IGMP + Multicast routing protocol)
 - **come:**
 - utilizzo di 1 distribuzione live-cd linux su più PC del laboratorio didattico
 - **requisiti:**
 - conoscenza di linux e configurazione di rete
- Studio di protocolli di trasporto multicast (reliable multicast)
 - studio dello stato dell'arte delle proposte per protocolli di trasporto affidabili per comunicazioni multicast
- Sviluppo di un applicativo in Java o in C/C++ che utilizza IP multicast
 - studio del supporto per IP multicast in Java e/o tramite socket C/C++
 - sviluppo di una applicazione peer-to-peer punto-multipunto basata su IP multicast
 - **come:**
 - utilizzo di JDK Java o di ambiente di sviluppo C/C++ (e.g. GCC)
 - **requisiti:**
 - conoscenza base della programmazione C, C++, o Java



IPv6

- Meccanismi di transizione IPv4-IPv6 in ambiente Linux e/o Microsoft
 - studio del supporto IPv6 e relativi meccanismi di transizione su piattaforma Linux e/o Microsoft Win2K
 - realizzazione di uno scenario di rete IPv6 e interlavoro con nodi IPv4 (tunneling e/o translation)
 - e.g. 6 over 4, 6to4, static tunnels, NAT-PT
 - **come:**
 - utilizzo di 1 distribuzione live-cd linux in laboratorio (preferibile), o PC Windows 2000
 - **requisiti:**
 - conoscenza di linux o MS Windows e configurazione di rete
- Approfondimento delle problematiche di transizione da IPv4 a IPv6
 - studio dei meccanismi di transizione e del porting degli applicativi da IPv4 a IPv6
- Sviluppo di 1 applicativo in Java o in C/C++ nativo IPv6
 - studio dei socket C/C++ e/o Java per IPv6
 - sviluppo di una applicazione client-server nativa in IPv6
 - **Come:**
 - utilizzo di JDK Java o di ambiente di sviluppo C/C++ (e.g. GCC)
 - **Requisiti:**
 - conoscenza base della programmazione C, C++, o Java

RTP/RTCP

- Sviluppo di un Mixer (Elab. Segnali Audio e Video)
 - **analisi delle caratteristiche/funzionalità di un mixer audio**
 - time equalization, buffering, sum
 - audio level adjustment
 - only best N flows
 - eventualmente transcoding
 - RTP packetizer
 - **sviluppo di un mixer con alcune delle funzionalità individuate**
 - a partire da N flussi di pacchetti RTP/IP in ingresso si deve generare un unico flusso in uscita opportunamente costruito (somma eventualmente pesata degli N flussi)
 - gli indirizzi di sorgente e destinazione sono configurati manualmente (no segnalazione)
 - il funzionamento del mixer potrà essere provato tramite applicativi audio standard (e.g. RAT)
 - **come:**
 - tramite programmazione diretta a livello di socket (in C/C++ o Java)
 - oppure, tramite apposite librerie RTP in C/C++ (da cercare) o Java (JMF o MMAPI)
 - **requisiti:**
 - conoscenza di almeno un CODEC audio
 - buona conoscenza della programmazione C, C++ o Java

5

RTP/RTCP (cont.)

- Sviluppo di un Transcoder (Elab. Segnali Audio e Video)
 - **analisi delle caratteristiche/funzionalità di un transcoder audio**
 - transcoding
 - RTP packetizer
 - **sviluppo di un transcoder tra due codec (e.g. PCM to GSM)**
 - da un flusso di pacchetti RTP/IP in ingresso si deve generare un flusso in uscita opportunamente transcodificato
 - gli indirizzi di sorgente e destinazione sono configurati manualmente (no segnalazione)
 - il funzionamento del transcoder potrà essere provato tramite applicativi audio standard (e.g. RAT)
 - **come:**
 - tramite programmazione diretta a livello di socket (in C/C++ o Java)
 - oppure, tramite apposite librerie RTP in C/C++ (da cercare) o Java (JMF o MMAPI)
 - **requisiti:**
 - conoscenza di almeno due CODEC audio (uno può essere il PCM)
 - buona conoscenza della programmazione C, C++ o Java

6

RTP/RTCP (cont.)

- Sviluppo di un media sender/receiver (audio o video)
 - **sviluppo di una coppia di audio o video sender/receiver su piattaforma Linux, MS Windows, o Java**
 - **come:**
 - tramite programmazione diretta in codice nativo C/C++ o Java
 - oppure, tramite apposite librerie RTP in C/C++ (da cercare) o Java (JMF o MMAPI)
 - **requisiti:**
 - conoscenza della programmazione C, C++ o Java
- Secure RTP
 - **Studio dei meccanismi di sicurezza per flussi RTP audio/video**
 - **implementazione di un semplice codificato decodificatore RTP**
 - **come:**
 - tramite programmazione diretta a livello di socket (in C/C++ o Java)
 - oppure, sfruttando apposite librerie RTP in C/C++ (da cercare) o Java (JMF o MMAPI)
 - algoritmi di crittografia disponibili in apposite librerie
 - **requisiti:**
 - conoscenza dei meccanismi base di crittografia
 - conoscenza della programmazione C, C++ o Java

7

IP Telephony

- Sviluppo di un SIP Registrar con location service
 - **sviluppo di un registrar SIP con location service tramite DB locale**
 - **come:**
 - utilizzo di un SIP Registrar già disponibile in Java
 - utilizzo di un DB su piattaforma Linux (e.g. MySQL) o MS Windows
 - **requisiti:**
 - programmazione Java (può essere acquisita anche durante il progetto)
- Sviluppo di un SIP registrar con autenticazione
 - **sviluppo di un registrar SIP con supporto della autenticazione tramite info locale o tramite backend authentication server (e.g. server Radius)**
 - **come:**
 - utilizzo di un SIP registrar già disponibile in Java
 - eventualmente, utilizzo di un server Radius
 - **requisiti:**
 - conoscenza delle basi di autenticazione (sufficiente HTTP digest)
 - programmazione Java (può essere acquisita anche durante il progetto)

8

IP Telephony (cont.)

- Studio delle problematiche relative al passaggio di servizi di IP Telephony attraverso firewall e NAT router
 - **Studio dei vari problemi introdotti da middlebox nella realizzazione di servizi di IP Telephony**
 - **Studio delle possibili soluzioni**
 - NAT/ALG
 - STUN/TURN
 - Media Gateway
 - altro
- IP Telephony attraverso firewall o NAT router
 - **analisi dei possibili meccanismi**
 - **sviluppo di uno di essi (Media Gateway o controllo delle politiche di filtraggio/NAT)**
 - **come:**
 - utilizzo di un SIP Proxy Java (già esistente)
 - sviluppo dell'interfaccia tra questo e il packet filter di Linux (iptables)
 - oppure, sviluppo di un Media GW (un semplice relay UDP)
 - **requisiti:**
 - programmazione in java (ma non è indispensabile)

9

IP Telephony (cont.)

- Studio delle problematiche relative all'interlavoro tra SIP/H.323
 - **Studio dei vari problemi nella realizzazione di un GW SIP/H.323**
 - **proposta di un semplice schema funzionale a blocchi**
 - **in alternativa, configurazione e utilizzo di un GW SIP/H.323, se disponibile**
- Sviluppo di un semplice Call generator
 - **sviluppo di un semplice generatore di chiamate SIP (solo segnalazione)**
 - **tale generatore dovrà generare un certo numero di chiamate SIP al secondo in accordo ad un opportuno modello di chiamata (tempi di interarrivo, tempo di risposta, durata, probabilità di rifiuto)**
 - **come:**
 - utilizzo di libreria SIP già disponibile (in Java) e relative API
 - utilizzo di un esempio di UA già disponibile
 - **requisiti:**
 - basi di programmazione

10

IP Telephony (cont.)

- Sviluppo di una applicazione di messaging basata su SIP
 - **sviluppo di una semplice applicazione di messaging/chat basata su SIP**
 - **come:**
 - utilizzo di libreria SIP già disponibile (in Java) e relative API
 - utilizzo di un ambiente di programmazione visuale (e.g. Borland JBuilder)
 - **requisiti:**
 - programmazione in Java (eventualmente acquisita durante il progetto)
- Studio dei meccanismi di presence previsti in SIP per servizi di messaging a conference
 - **modello Subscribe/Notify**
 - **applicazione a scenari di messaging**
 - **e/o applicazione a scenari di conference**
- Analisi di meccanismi di presence proprietari quali MSN Messenger e/o ICQ
 - **analisi del funzionamento di meccanismi di presence commerciali esistenti, quali MSN Messenger e ICQ**
 - **come:**
 - studio di documentazione eventualmente disponibile
 - utilizzo di protocol analyzer (e.g. Ethereal)

11

QoS

- Realizzazione di un Audio Quality Tester "analogico" (Elab. Segnali Audio e Video)
 - **realizzazione di tester della qualità audio con input/output "analogico" attraverso l'ingresso (mic) e uscita (speaker) di un PC**
 - **l'AQT genera un flusso audio digitale (sintetico o estratto da file) e lo invia alla porta audio del sistema**
 - **la scheda audio provvede alla conversione D/A e spedito alla presa dello speaker o headphones**
 - **l'audio viene ritornato in ingresso nella presa del mic**
 - **la scheda audio provvede alla conversione A/D**
 - **l'AQT deve leggere il flusso in ingresso e provvedere all'opportuna correlazione con quello inviato**
 - **l'AQT estrae le dovute misure e statistiche**
 - e.g. ritardo, attenuazione, degradazione, altro
 - **come:**
 - qualsiasi linguaggio di programmazione su piattaforma Linux, MS Windows o Java
 - **requisiti:**
 - programmazione

12

QoS (cont.)

- Realizzazione di un Audio Quality Tester "digitale"
 - realizzazione di tester della qualità audio con input/output "digitale" attraverso l'invio e ricezione di flussi IP (pacchetti raw IP, UDP/IP, o RTP/UDP/IP)
 - l'AQT genera un flusso audio digitale (sintetico o estratto da file), lo packetizza e lo invia in rete
 - l'audio viene ritornato di nuovo in ingresso (eventualmente dopo il transito in rete)
 - l'AQT riceve il flusso in ingresso e provvedere all'opportuna correlazione con quello inviato
 - l'AQT estrae le dovute misure e statistiche
 - e.g. ritardo, degradazione..
 - come:
 - qualsiasi linguaggio di programmazione su piattaforma Linux, MS Windows o Java
 - utilizzo di opportune librerie RTP o tramite implementazione diretta
 - requisiti:
 - programmazione

13

QoS (cont.)

- Supporto per la QoS in ambiente Linux
 - studio dei supporti per la QoS (traffic controller, filtering, etc.) in ambiente Linux
 - sviluppo di una semplice configurazione di rete con classificazione e differenziazione del traffico (architettura Diffserv)
 - come:
 - supporto nativo di Linux
 - eventualmente, moduli aggiuntivi
 - eventualmente, utilizzo di una distribuzione live-cd
- Realizzazione di una architettura integrata SIP/QoS
 - studio di una soluzione per il supporto di QoS per applicazioni SIP <draft-veltri-sip-qsip-01.txt>
 - realizzazione di un dimistratore in laboratorio
 - come:
 - utilizzo di una implementazione di QSIP Java già disponibile
 - sviluppo dell'interfaccia con il supporto nativo di Linux per la QoS (Traffic Controller)
 - requisiti:
 - programmazione

14

Altro (cont.)

- Studio delle problematiche di balancing e ridondanza per server di segnalazione SIP
 - studio di soluzioni di load balancing e di ridondanza
 - valutazione di vantaggi e svantaggi
- Studio dei meccanismi di descrizione/programmazione della logica di servizio
 - studio del CPL
 - esempio di realizzazione in CPL di un possibile servizio di chiamata avanzato
 - facoltativamente, schema di implemmentazione delle primitive di servizio
- Istallazione di un Media server
 - installazione e configurazione di un Media server
 - analisi del funzionamento
 - come:
 - utilizzo di SW open source o di MS Media Server
 - analisi del suo funzionamento tramite un protocol analyzer (e.g. Ethereal)

15