

Παρουσίαση αποτελεσμάτων

Εργασία δικτυακού προγραμματισμού : Java socket programming

Ο κώδικας

Για τη συγγραφή του κώδικα της εφαρμογής βασιστήκαμε στα ενδεικτικά τμήματα κώδικα που υπάρχουν στην εκφώνηση, μετά από την απαραίτητη έρευνα στο διαδίκτυο για να κατανοήσουμε τις κλάσεις της Java που χρησιμοποιούνται στην εφαρμογή(DatagramSocket, DatagramPacket, SourceDataLine κ.λ.π.).

Η συνάρτηση getEcho() αναλαμβάνει την λήψη echo packets από των server, με η χωρίς τεχνητή καθυστέρηση. Ταυτόχρονα υπολογίζει τον χρόνο απόκρισης του server καθώς και το bandwidth της σύνδεσης χρησιμοποιώντας την τεχνική του moving average. Δηλαδή η τιμή του bandwidth τη χρονική στιγμή t εξαρτάται από ένα χρονικό παράθυρο προηγούμενων χρονικών στιγμών.

Η συνάρτηση getImage() κατεβάζει και αποθηκεύει μια εικόνα από τον server κατά τα γνωστά από την προηγούμενη εργασία με τη διαφορά ότι αυτή τη φορά χρησιμοποιούνται sockets.

Οι συναρτήσεις getDPCM() και getAQDPCM() αναλαμβάνουν τη λήψη ήχου αντίστοιχης αποκωδικοποίησης από το server. Παρατηρώντας ότι η κωδικοποίηση DPCM είναι στην ουσία μια ειδική περίπτωση της AQDPCM με σταθερά μ και β κατασκευάσαμε μια κοινή συνάρτηση decodeAQDPCM η οποία χρησιμοποιείται για την αποκωδικοποίηση και των δύο. Τα παραγόμενα bytes αναλαμβάνει να αναπαραγάγει η συνάρτηση playAudio().

Η συνάρτηση ithakiCopter() λαμβάνει από το σερβερ μετρήσεις σχετικές με την κατάσταση της πλατφόρμας ithakiCopter και τις αποθηκεύει σε logfile ώστε να είναι δυνατή η εξαγωγή διαγραμμάτων. Ο χειρισμός του ithakiCopter γίνεται από το java applet που είναι διαθέσιμο στο σερβερ του μαθήματος.

Τέλος η συνάρτηση getCarDiagnostics() αναλαμβάνει τη λήψη και την αποκωδικοποίηση μετρήσεων σχετικά με την λειτουργία ενός οχήματος σε μορφή OBD-II. Το OBD-II είναι ένα ιδιαίτερα ισχυρό εργαλείο για τη άμεση διάγνωση προβλημάτων σχετικών με τη λειτουργία του αυτοκινήτου και είναι ενσωματωμένο σε όλα σχεδόν τα σύγχρονα οχήματα. Με την αγορά του κατάλληλου αναγνώστη OBD ο οποίος συνδέεται στη αντίστοιχη θύρα που υπάρχει(συνήθως καλά κρυμμένη) στο αυτοκίνητο, μπορούν να εξαχθούν οι μετρήσεις που εμείς λαμβάνουμε από τον σερβερ. Στη βιβλιογραφία παραθέτουμε ένα παράδειγμα μιας τέτοιας συσκευής η οποία συνδέεται μάλιστα με χρήση bluetooth στο smartphone του χρήστη και καθιστά δυνατή την άμεση λήψη και γραφική προβολή των διαγνωστικών στοιχείων.

Υλοποίηση του bandwidth moving average

Έχει ενδιαφέρον να παρουσιαστεί εδώ συνοπτικά ο τρόπος που επιλέξαμε να υλοποιήσουμε τον moving average. Φτιάξαμε ένα array packetsRxInSecond[] στο οποίο κρατάμε για κάθε

δευτερόλεπτο i , πόσα πακέτα λάβαμε. Για να το πετύχουμε αυτό, ελέγχουμε κάθε φορά που λαμβάνουμε ένα πακέτο, αν έχουμε “αλλάξει” δευτερόλεπτο, αν δηλαδή έχουν περάσει δευτερόλεπτα περισσότερα από i . Τότε αυξάνουμε τον δείκτη i κατά 1 και στη συνέχεια υπολογίζουμε την τιμή του bandwidth για το κάθε ζητούμενο timeframe. Το μήκος του timeframe προσαρμόζεται κατάλληλα όταν βρισκόμαστε στην αρχή των μετρήσεων.

Υλοποίηση της αποκωδικοποίησης AQDPCM

Η αποκωδικοποίηση γίνεται από τη συνάρτηση `decodeAQDPCM()` με τον εξής τρόπο. Το κάθε byte χωρίζεται καταρχάς, με χρήση shift, στα δύο μέρη του D1, D2 αποτελούμενα το καθένα από 4 bit. Από το κάθε μέρος αφαιρούμε τις 8 μονάδες που έχουν προστεθεί κατά την κωδικοποίηση και πολλαπλασιάζουμε επί τη διασπορά β ώστε να «αποκανονικοποιήσουμε» τις τιμές μας. Στη συνέχεια το D1 προστίθεται στην προηγούμενη τιμή ήχου που έχουμε υπολογίσει (`previousNibble`) και στη νέα τιμή προστίθεται το D2. Στις δύο νέες τιμές που προέκυψαν προσθέτουμε τον μέσο όρο μ , αφού ανανεώσουμε τη μεταβλητή `previousNibble` και τέλος τις αποθέτουμε στο ArrayList εξόδου.

Μετρήσεις και σχολιασμός Αποτελεσμάτων

Echo Packets

Session 1

Στο Session 1 πήραμε χρόνους απόκρισης echo γύρω στα 1000ms δηλαδή throughput γύρω στα 250kbps. Οι χρόνοι απόκρισης είναι σχετικά σταθεροί αφού περιορίζονται από τον server του εργαστηρίου. Τα πακέτα χωρίς τεχνητή καθυστέρηση είχαν χρόνους απόκρισης της τάξης των 100ms

Session 2

Στο Session 2 παρατηρήσαμε σαφώς μεγαλύτερους χρόνους απόκρισης κοντά στα 1500ms και ανάλογα μειωμένο throughput. Τα πακέτα χωρίς τεχνητή καθυστέρηση είχαν χρόνους απόκρισης μικρότερους από 1000ms που είναι μια τάξη μεγαλύτερη από αυτούς του session 1

Audio Packets

Οι κυματομορφές ήχου είναι όπως τις περιμέναμε, με σταθερή συχνότητα για τη γεννήτρια συχνότητας. Τα μουσικά κομμάτια περιέχουν πολλές συχνότητες και μάλιστα σε διαφορετικά πλάτη η κάθε μία. Για μεγαλύτερες τιμές του β παρατηρούμε αύξηση του κέρδους έντασης του κομματιού δηλαδή έχουμε στην ουσία πιο δυνατό σήμα.

IthakiCopter


Οι μετρήσεις που παίρνουμε από το ελικόπτερο σε πραγματικό χρόνο είναι ακριβής. Δεν παρατηρούμε κάποιο θόρυβο η σφάλμα. Οι μετρήσεις αυτές θα μπορούσαν να χρησιμοποιηθούν για τη δημιουργία ενός απομακρυσμένου closed-loop controller. Παρατηρήσαμε ότι το ελικόπτερο ανταποκρίνεται ταχύτερα σε εντολές ανόδου παρά καθόδου.

Car Diagnostics

Τα διαγνωστικά στοιχεία OBD-II έχουν παρασταθεί σε ένα διάγραμμα με υπέρθεση. Παρατηρούμε ότι μπορούμε να παρακολουθήσουμε πως μεταβάλλονται διάφορα στοιχεία της μηχανής ανάλογα με την ταχύτητα του αυτοκινήτου και έτσι να εξάγουμε εύκολα συμπεράσματα για πιθανές βλάβες η και να βελτιώσουμε την απόδοση της μηχανής.

Port Forwarding

Για να μπορέσουμε να επικοινωνήσουμε με το server και πιο συγκεκριμένα για να μπορέσει ο server να στείλει δεδομένα στην εφαρμογή μας, ανοίξαμε από τις ρυθμίσεις του router τα κατάλληλα ports και τα ανακετευνάμε στην τοπική IP του υπολογιστή μας.


ZXV10 H201L

Status
Network
Security
Application
VoIP
DDNS
DMZ Host
UPnP
UPnP Port Mapping
DNS Service
QoS
SNTP
IGMP
Port Trigger
SOHO Service
SOHO Service
SOHO Address Mapping
SOHO DMZ
Administration

Path: Application-SOHO Service-SOHO Address Mapping
Logout

Enable ☒

Name

Protocol

WAN Start Port (1 ~ 65535)

WAN End Port (1 ~ 65535)

External IP Address

LAN Host Start Port (1 ~ 65535)

LAN Host End Port (1 ~ 65535)

LAN Host IP Address

| Enable | Name | WAN Start Port | LAN Host Start Port | External IP Address | Modify | Delete |
|-------------------------------------|----------|----------------|---------------------|---------------------|--------|--------|
| | Protocol | WAN End Port | LAN Host End Port | LAN Host IP Address | | |
| <input checked="" type="checkbox"/> | java2 | 48000 | 48000 | | | |
| | TCP AND | 48900 | 48900 | 192.168.1.5 | | |

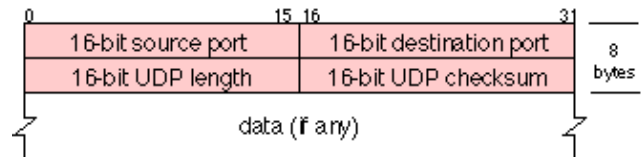
Copyright © 2011 ZTE Corporation. All rights reserved.

Βιβλιογραφική Αναφορά: Το πρωτόκολλο UDP και διεθνή πρότυπα audio streaming

UDP

Το udp είναι ένα πρωτόκολλο επικοινωνίας υπολογιστών που έχει οριστεί για χρήση πάνω στο πρωτόκολλο IP. Σε αντίθεση με το ευρέως χρησιμοποιούμενο TCP, το UDP δεν είναι πλήρως αξιόπιστο, δεν εγγυάται δηλαδή πάντα την επιτυχή μεταφορά όλων των πακέτων χωρίς σφάλματα, αλλά κάνει μόνο περιορισμένους ελέγχους(best-effort) και αφήνει την διαχείριση σφαλμάτων η χαμένων μηνυμάτων σε ανώτερα πρωτόκολλα. Με τον τρόπο αυτό το UDP ελαχιστοποιεί τις καθυστερήσεις ελέγχου και επανεκπομπών.

Η απλότητα αυτή του UDP είναι και η δύναμη του που το κάνει ιδιαίτερα χρήσιμο σε κάποιες εφαρμογές όπως το live audio streaming, όπου αν κάποια πακέτα δεν παραληφθούν επιτυχώς, ίσως να μην έχει νόημα η επανεκπομπή τους, αφού το streaming έχει ήδη προχωρήσει. Το λογισμικό αναπαραγωγής αναλαμβάνει να κάνει τις απαραίτητες προσαρμογές για τα χαμένα bytes ώστε ο τελικός χρήστης να μην παρατηρεί διακοπές.



Ένα άλλο χαρακτηριστικό του UDP είναι η δυνατότητα broadcasting/multicasting δηλαδή η αποστολή πακέτων σε πολλούς υπολογιστές ταυτόχρονα. Στην περίπτωση του multicasting μάλιστα μπορούν να επιλεγούν μόνο κάποιο από τους υπολογιστές που είναι συνδεδεμένοι στο δίκτυο. Με τον τρόπο αυτό μια ροή ήχου/εικόνας μπορεί να μεταδοθεί ταυτόχρονα και γρήγορα σε πολλούς παραλήπτες.

Λόγω των παραπάνω χαρακτηριστικών το UDP χρησιμοποιείται πολύ συχνά σε εφαρμογές VOIP, Audio/Video Streaming αλλά ακόμα και σε υπηρεσίες όπως DNS και το Bittorrent. Στις υπηρεσίες αυτές είναι πολύ πιο σημαντική η ταχύτητα από την απόλυτα αξιόπιστη μετάδοση των δεδομένων.

Audio Streaming

Η οικογένεια RTP/RTCP/RTSP

Τα τρία αυτά πρωτόκολλα χρησιμοποιούνται συνήθως μαζί. Το Real Time Transport Protocol(RTP) τρέχει πάνω από το UDP, αν και θεωρείται και αυτό μέρος του transport layer. Είναι αυτό που αναλαμβάνει τη μεταφορά δεδομένων streaming video/audio. Είναι στενά συνδεδεμένο με το Real Time Control Protocol (RTCP), το οποίο τρέχει στο session layer και παρέχει feedback σχετικά με την ποιότητα της μετάδοσης δεδομένων, έτσι ώστε να είναι δυνατή η διόρθωση σφαλμάτων με την αλλαγή του ρυθμού μετάδοσης. Τέλος το Real Time Streaming Protocol, αν και είναι ανεξάρτητο από τα άλλα δυο, χρησιμοποιείται συχνά μαζί τους. Είναι αντίστοιχο του HTTP, τρέχει στο presentation layer και αναλαμβάνει την εκτέλεση ενεργειών όπως «αναπαραγωγή», «παύση», «εγγραφή» κ.λ.π.

Τα τρία αυτά πρωτόκολλα εκτελούνται σε διαφορετικά ports το καθένα και όταν χρησιμοποιούνται και τα τρία αναφερόμαστε σε αυτά ως RTSP stack. Θα πρέπει να σημειωθεί ότι το πρωτόκολλο UDP πάνω στο οποίο τρέχει συνήθως το RTSP stack δεν υποστηρίζεται από τους web browser, ούτε από τους media players που βρίσκονται προεγκατεστημένοι στα περισσότερα λειτουργικά συστήματα, οπότε συνήθως απαιτείται η εγκατάσταση τρίτων εφαρμογών όπως το flash.

RTMP

Το Real Time Messaging Protocol αναπτύχθηκε από την Adobe για χρήση κυρίως με την τεχνολογία Flash της εταιρίας. Με αυτή την έννοια είναι ένα «κλειστό»(proprietary) πρωτόκολλο, παρόλου που η εταιρία έχει δημοσιεύσει ένα specification, το οποίο όμως έχει σημαντικά κενά. Συνήθως χρησιμοποιείται πάνω από το TCP. Η δημοτικότητα του αναμένεται να πέφτει όσο μειώνεται και η δημοτικότητα του Adobe Flash.

HTTP Live Streaming

Αποτελεί μάλλον το πιο απλό και δημοφιλές streaming πρωτόκολλο που υλοποιείται πάνω στο επίσης δημοφιλές HTTP. Αυτά τα δύο θεωρητικά δεν ταιριάζουν αφού το HTTP χρησιμοποιεί το TCP/IP το οποίο δίνει έμφαση στην ακρίβεια των δεδομένων περισσότερο από τη συνεχή ροή, όμως στην πράξη, με την αυξημένη ταχύτητα και αξιοπιστία των συνδέσεων στις μέρες μας αυτό δεν αποτελεί σημαντικό πρόβλημα.

HLS

Το HTTP Live Streaming(HLS) της Apple αναπτύχθηκε από την Apple για χρήση στο iOS και έχει δημοσιευθεί το specification του ως draft, δεν παύει όμως να είναι ένα proprietary πρωτόκολλο. Το πρωτόκολλο υποστηρίζεται μερικώς από τον Chrome for Android αλλά όχι από τους γνωστούς desktop browsers. Το HLS «σπάει» το stream σε μικρά κομμάτια τα οποία μεταδίδονται ως HTTP downloads. Μια λίστα M3U περιέχει τα metadata σχετικά με τα διαφορετικά bitrates που είναι διαθέσιμα και ο χρήστης μπορεί να επιλέγει από αυτά ανάλογα με τη σύνδεση του.

Και άλλες εταιρίες όπως η Adobe και οι Microsoft έχουν αναπτύξει τα δικά τους HTTP πρωτοκόλλα χωρίς ιδιαίτερη επιτυχία.

DASH

Το Dynamic Adaptive Streaming over HTTP μοιάζει πολύ με το HLS αλλά αποτελεί ένα διεθνές standard. Είναι coded agnostic δηλαδή μπορεί να χρησιμοποιηθεί με οποιαδήποτε κωδικοποίηση όπως H.265, H.264, VP9 κ.λ.π. Όπως και στο HLS, το stream χωρίζεται σε μικρότερα κομμάτια τα οποία μεταφέρονται μέσω του HTTP, ενώ κάθε φορά ο client επιλέγει το κομμάτι με το μεγαλύτερο bitrate που προβλέπεται ότι θα μεταφερθεί στην ώρα του. Έτσι το DASH μπορεί να προσαρμόζεται σε μεταβαλλόμενες συνθήκες δικτύου χωρίς ο χρήστης να παρατηρεί διακοπές και καθυστερήσεις

Βιβλιογραφία

https://en.wikipedia.org/wiki/On-board_diagnostics

<http://www.ebay.com/itm/OBD-2-Android-Scanner-CAN-Diagnostics-Reader-Wireless-Bluetooth-Tool-For-HONDA-/262773194360?hash=item3d2e80be78:g:ilIAOSwZQRYW2y->

<http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/udp.html#Anchor-3800>

<https://el.wikipedia.org/wiki/UDP>

<http://www.garymcgath.com/streamingprotocols.html>

<https://stackoverflow.com/questions/30184520/whats-the-best-protocol-for-live-audio-radio-streaming-for-mobile-and-web>

https://en.wikipedia.org/wiki/Streaming_media#Protocols

https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol

https://en.wikipedia.org/wiki/RTP_Control_Protocol

<http://tools.ietf.org/html/draft-pantos-http-live-streaming-11>

<http://www.dveo.com/Streaming-Video-HTTP-RTSP-Flash-IPTV/HLS-live-streaming.html>