Δίκτυα Υπολογιστών 30/10 5ο

Άλλα μηνύματα αιτήματος HTTP

POST method:  
▪ ιστοσελίδα συνήθως περιλαμβάνει φόρμα εισόδου

HEAD method:  
▪ ζητά τις κεφαλίδες (μόνον) που θα επιστρέφονταν *εάν* το συγκεκριμένο URL είχε ζητηθεί με μία μέθοδο HTTP GET

PUT method:  
▪ ανεβάζει νέο αρχείο (αντικείμενο) στον εξυπηρέτη  
▪ αντικαθιστά πλήρως το αρχείο που υπάρχει στο καθορισθέν URL με το περιεχόμενο του σώματος του μηνύματος αιτήματος POST HTTP

▪ είσοδος χρήστη αποστέλλεται από πελάτη σε

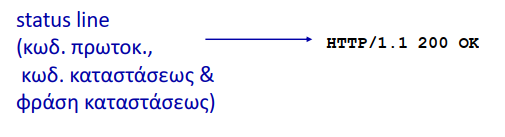
εξυπηρέτη στο σώμαμηνύματος αιτήματος HTTP POST

GET method (αποστολή δεδομένων  
προς εξυπηρέτη):

▪ συμπερίληψη δεδομένων χρήστη στοπεδίο URL του

μηνύματος αιτήματος HTTP GET (μετά από χαρακτήρα ‘?’):  
**www.somesite.com/animalsearch?monkeys&banana**

Μήνυμα απόκρισης HTTP



Κωδικοί καταστάσεως απόκρισης HTTP

* ο κωδικός κατάστασης εμφανίζεται στην 1η γραμμή του μηνύματος απόκρισης

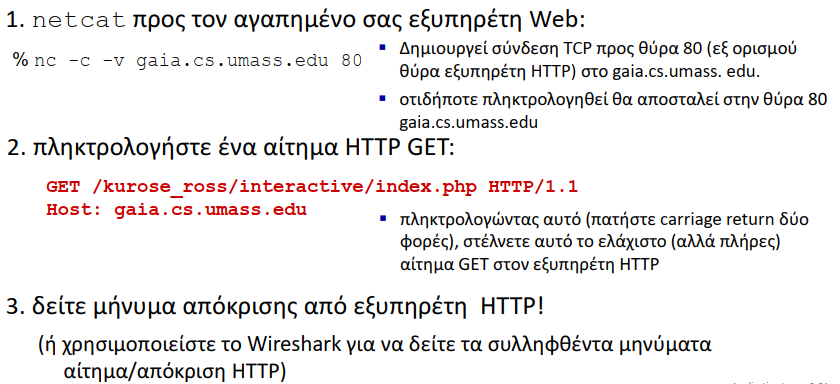
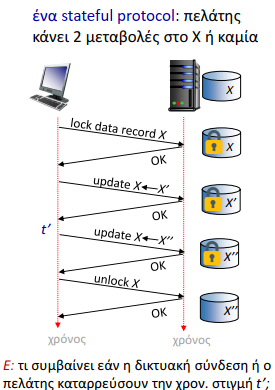
εξυπηρέτη-προς-πελάτη.

* μερικοί τέτοιοι κωδικοί:

200 OK  
• αίτημα επιτυχές, αιτηθέν αντικείμενο παρακάτω στο μήνυμα αυτό

301 Moved Permanently  
• αιτηθέν αντικείμενο μετακινήθηκε, νέα τοποθεσία παρακάτω στο μήνυμα αυτό (στο “Location: <πεδίο>”)  
400 Bad Request  
• μήνυμα αιτήσεως ακατάληπτο από τον εξυπηρέτη  
404 Not Found  
• αιτηθέν έγγραφο δεν βρέθηκε σε αυτόν τον εξυπηρέτη  
505 HTTP Version Not Supported

Δοκιμάζοντας το HTTP (πλευρά πελάτη)



Τηρώντας κατάσταση χρήστη/εξυπηρέτη: cookies

Θυμηθείτε: η διάδραση HTTP GET/απόκριση είναι *stateless*

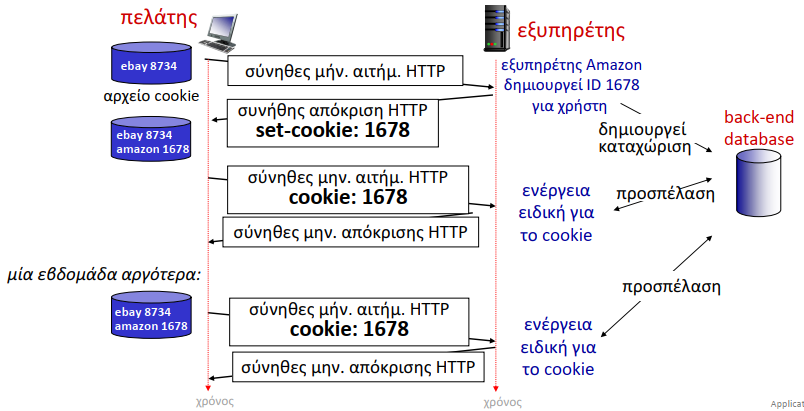
* καμία αντίληψη πολλαπλών ανταλλαγών μηνυμάτων HTTP για να ολοκληρωθεί μία “συναλλαγή” Web
* καμία ανάγκη πελάτη/εξυπηρέτη για παρακολούθηση “κατάστασης” ανταλλαγής πολλαπλών βημάτων
* όλα τα αιτήματα HTTP ανεξάρτητα το κάθε ένα από το άλλο
* καμία ανάγκη πελάτη/εξυπηρέτη να “ανανήψει” από μια εν μέρει αλλά όχι πλήρως ολοκληρωμένη συναλλαγή

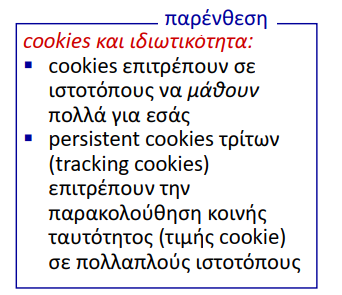
Κατάσταση χρήστη-εξυπηρέτη: cookies

Παράδειγμα:  
▪ Η Susan χρησιμοποιεί browser στο  
laptop, επισκεπτόμενη συγκεκριμένο  
site e-commerce για πρώτη φορά  
▪ Όταν αρχικά αιτήματα HTTP  
φθάνουν στην τοποθεσία, η  
τελευταία δημιουργεί:  
• μοναδικό ID (“cookie”)  
• καταχώρηση για το ID στην back-end  
database  
• ακόλουθα αιτήματα HTTP απόSusan  
προς αυτό το site θα περιέχουν τιμή  
cookie ID, επιτρέποντας στο site να  
“αναγνωρίσει” την Susan

ιστότοποι και browser πελατών  
χρησιμοποιούν *cookies* για να τηρούν  
κάποια κατάσταση μεταξύ συναλλαγών  
*τέσσερα συστατικά:*1) γραμμή κεφαλίδος cookie του μηνύματος απόκρισης HTTP  
2) γραμμή κεφαλίδος cookie στο επόμενο  
μήνυμα αιτήματος HTTP  
3) αρχείο cookie τηρείται στον Η/Υ  
χρήστη, διαχειζόμενο από τον browser  
του χρήστη  
4) back-end database στον ιστότοπο

Τηρώντας κατάσταση χρήστη/εξυπηρέτη: cookies



cookies HTTP: σχόλια

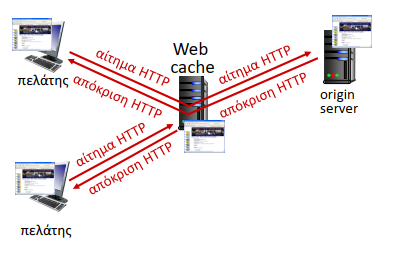
*για τι μπορούν να χρησιμοποιηθούν τα cookies:*

▪ εξουσιοδότηση  
▪ καλάθια αγορών  
▪ προτάσεις αγορών  
▪ κατάσταση συνόδου χρήστη (Web e-mail)

*Πρόκληση: πώς τηρούμε “κατάσταση”;*▪ *στα τερματικά σημεία πρωτοκόλλου:* τήρηση κατάστασης σε πελάτη/εξυπηρέτη για πολλαπλές συναλλαγές  
▪ *στα μηνύματα:* cookies σε μηνύματα HTTP μεταφέρουν κατάσταση

Web caches

*Στόχος:* ικανοποίηση αιτημάτων πελάτη χωρίς εμπλοκή εξυπηρέτη  
προέλευσης (origin)

▪ χρήστης ρυθμίζει browser να δείχνει προς μία (τοπική) *Web cache*▪ browser στέλνει όλα τα αιτήματα HTTP στην cache

• *if* αντικείμενο στην cache: cache επιστρέφει αντικείμενο στον πελάτη

• *else* cache ζητά αντικείμενο από εξυπηρέτη προέλευσης, caches ληφθέν αντικείμενο, και μετά  
επιστρέφει αντικείμενο στο πελάτη

Web caches (εξυπηρέτες proxy)

▪ Μία web cache δρα και ως  
πελάτης και ως εξυπηρέτης

*Γιατί* web caching;

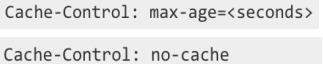
▪ μειώνει χρόνο απόκρισης για  
αίτημα πελάτη

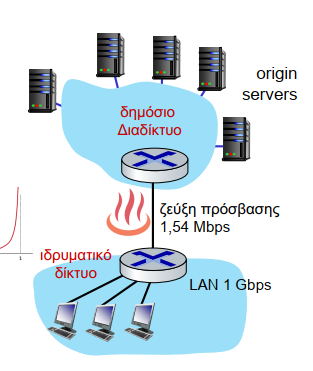
•cache πλησιέστερα στον πελάτη  
▪ μειώνει κυκλοφορία στην ζεύξη  
πρόσβασης ενός ιδρύματος  
▪ Διαδίκτυο βρίθει από caches

• καθιστά “φτωχούς” παρόχους  
περιεχομένου ικανούς να παραδίδουν  
περιεχόμενο πιο αποδοτικά

• ως εξυπηρέτης για πελάτη που  
ζητά κάτι για πρώτη του φορά

• ως πελάτης προς origin server

▪ εξυπηρέτης λέει στην cache για το  
επιτρεπόμενο caching του  
αντικειμένου στην κεφαλίδα  
απόκρισης:

Παράδειγμα caching

*Σενάριο:*▪ ρυθμός ζεύξης πρόσβασης: 1,54 Mbps  
▪ RTT γειτονικού δρομ/τή - origin server: 2 sec  
▪ μέγεθος αντικειμένου web: 100K bits  
▪ μέσος ρυθμός αιτημάτων από browsers προς  
origin servers: 15/sec  
▪ μέσος ρυθμ. δεδ. προς browsers: 1,50 Mbp

*Απόδοση:*

*πρόβλημα:* μεγάλη  
αναμονή σε ουρά,  
όταν high utilization!

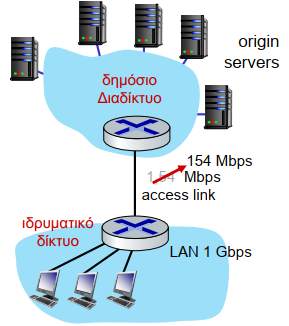
▪ access link utilization = 0,97

▪ LAN utilization: 0,0015

▪ καθυστ. από-άκρο-σε-άκρο = Internet delay +access link delay + LAN delay

= 2 sec + λεπτά + μsecs

Επιλογή 1: μίσθωση ταχύτερης ζεύξης πρόσβασης

*Σενάριο:*▪ ρυθμός ζεύξης πρόσβασης: 154 Mbps  
▪ RTT γειτονικού δρομ/τή - origin server: 2 sec  
▪ μέγεθος αντικειμένου web: 100K bits  
▪ μέσος ρυθμός αιτημάτων από browsers προς  
origin servers: 15/sec  
▪ μέσος ρυθμ. δεδ. προς browsers: 1,50 Mbp

*Απόδοση:*

▪ access link utilization = 0,0097

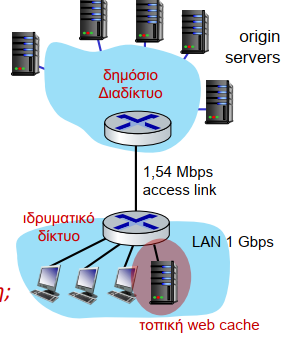
▪ LAN utilization: 0,0015

▪ καθυστ. από-άκρο-σε-άκρο = Internet delay +access link delay + LAN delay

= 2 sec + msecs+ μsecs

*Κόστος:* ταχύτερο access link (ακριβό!)

*Επιλογή 2: εγκατάσταση μίας web cache*

*Σενάριο:* ▪ ρυθμός ζεύξης πρόσβασης: 1,54 Mbps  
▪ RTT γειτονικού δρομ/τή - origin server: 2 sec  
▪ μέγεθος αντικειμένου web: 100K bits  
▪ μέσος ρυθμός αιτημάτων από browsers προς  
origin servers: 15/sec  
▪ μέσος ρυθμ. δεδ. προς browsers: 1,50 Mbp

*Κόστος:* web cache (φθηνή!)

*Απόδοση:*

▪ access link utilization = ;

▪ LAN utilization: ;

▪ καθυστ. από-άκρο-σε-άκρο = ;

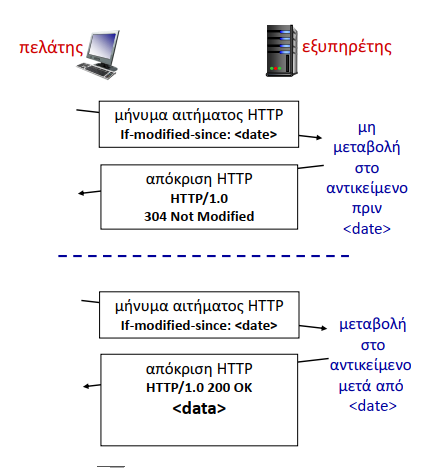
Υπολογίζοντας access link utilization, καθυστέρηση από-άκρο-σε-άκρο με χρήση cache:

έστω cache hit rate είναι 0,4:  
▪ 40% αιτημάτων ικανοποιούνται από cache, με χαμηλή (msec) καθυστέρηση  
▪ 60% αιτημάτων ικανοποιούνται σε origin

• ρυθμός δεδομένων προς browsers μέσω access link = 0,6 \* 1,50 Mbps = 0,9 Mbps

• access link utilization = 0,9/1,54 = 0,58 άρα χαμηλή (msec) αναμονή σε ουρά στο access link

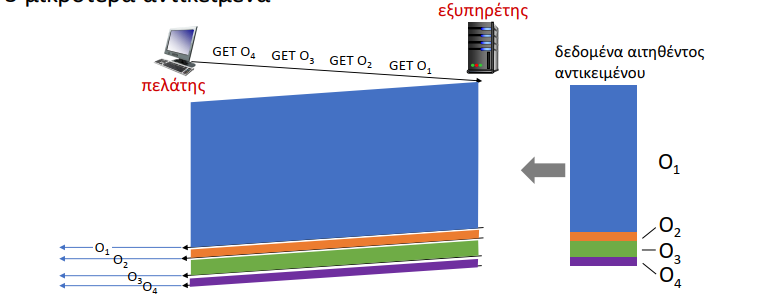
▪ μέση καθυστέρηση από-άκρο-σε-άκρο : = 0,6 \* (καθυστέρηση από origin servers) + 0,4 \* (καθυστέρηση όταν cache) = 0,6 (2,01) + 0,4 (~msecs) = ~ 1,2 secs  
*μέση καθυστέρηση από-άκρο-σε-άκρο < από ζεύξη 154 Mbps (και φθηνότερη!)*

Conditional GET  
*Στόχος:* μη στέλνεις αντικείμενο εάν η  
cache έχει ενημερωμένη έκδοση  
• δεν υπάρχει καθυστέρηση για μετάδοση  
αντικειμένου (ή χρήση δικτυακών πόρων)  
▪ *πελάτης:* προσδιόρισε ημερομηνία  
αντιγράφου στην cache στο αίτημα HTTP  
**If-modified-since: <date>**▪ *εξυπηρέτης:* απόκριση δεν περιέχει  
αντικείμενο εάν το αντικείμενο στην  
cache είναι ενήμερο χρονικά:  
**HTTP/1.0 304 Not Modified**

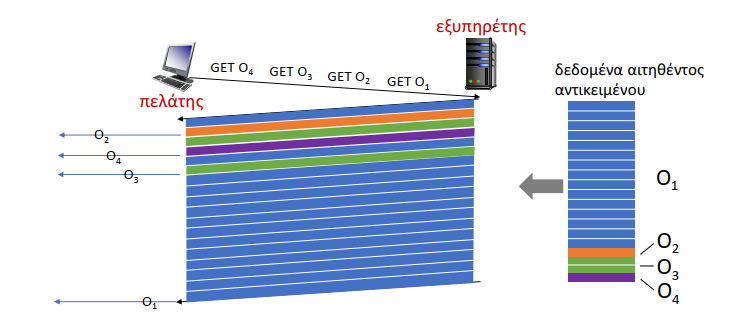
HTTP/2

*Βασικός στόχος:* μείωση καθυστέρησης σε αιτήματα HTTP για πολλαπλά αντικείμενα  
***HTTP1.1:***εισήγαγε πολλαπλά, pipelined GETs μέσω μίας σύνδεσης TCP  
▪ εξυπηρέτης αποκρίνεται *με-την-σειρά* (προγραμματισμός FCFS: firstcome-first-served) σε αιτήματα GET  
▪ με FCFS, μικρό αντικείμενο μπορεί να χρειασθεί να περιμένει για μετάδοση (head-of-line (HOL) – *κορυφή της γραμμής* – blocking) πίσω από μεγάλο(α) αντικείμενο(α)  
▪ ανάνηψη από απώλεια (επαναμετάδοση απολεσθέντων τμημάτων TCP) καθυστερεί μετάδοση αντικειμένων  
***HTTP/2: [RFC 7540, 2015]*** αυξημένη ευελιξία στον *εξυπηρέτη* για αποστολή αντικειμένων στον πελάτη:  
▪ μέθοδοι, κωδικοί κατάστασης, περισσότερα πεδία κεφαλίδος αμετάβλητα από HTTP 1.1  
▪ σειρά μετάδοσης αιτηθέντων αντικειμένων βάσει προτεραιότητας που ορίζεται από τον πελάτη (άρα, όχι κατ΄ ανάγκην FCFS)  
▪ *push* των μη αιτηθέντων αντικείμενων προς τον πελάτη  
▪ διαίρεση αντικειμένων σε πλαίσια, προγραμματισμός πλαισίων για άμβλυνση του HOL blocking

HTTP/2: άμβλυνση του HOL blocking

HTTP 1.1: πελάτης αιτείται 1 μεγάλο αντικείμενο (π.χ., αρχείο βίντεο) και  
3 μικρότερα αντικείμενα

*αντικείμενα παραδίδονται με την σειρά που ζητήθηκαν: O2, O3, O4 περιμένουν πίσω από O1*

HTTP/2: αντικείμενα διαιρούνται σε πλαίσια (frames), μετάδοση πλαισίων με διάπλεξη (interleaving)

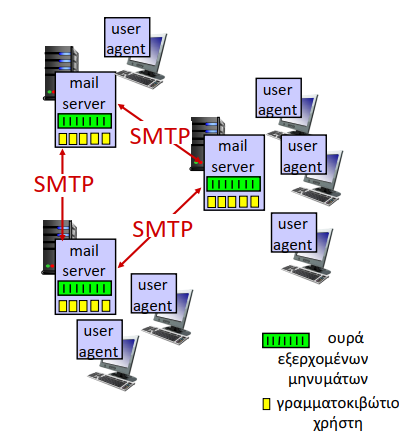
*O2, O3, O4 μεταδίδονται σύντομα, O1 καθυστερεί ελαφρά*

Από το HTTP/2 προς το HTTP/3  
HTTP/2 μέσω μίας σύνδεσης TCP σημαίνει:  
▪ ανάνηψη από απώλεια πακέτων συνεχίζει ακόμα να καθυστερεί όλες τις μεταδόσεις αντικειμένων

• όπως στο HTTP 1.1, οι browsers έχουν κίνητρο να δημιουργούν πολλαπλές παράλληλες συνδέσεις TCP για μείωση καθυστερήσεων και αύξηση συνολικής διαμεταγωγής (throughput)

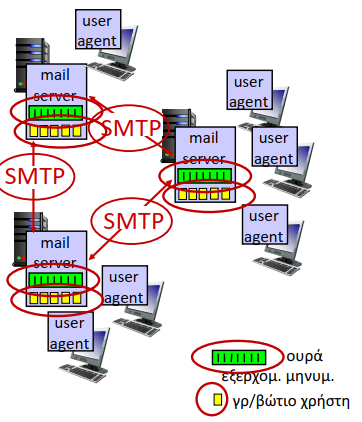
▪ Καμία ασφάλεια σε συμβατική σύνδεση TCP  
▪ HTTP/3: προσθέτει ασφάλεια, έλεγχο σφαλμάτων ανά αντικείμενο και έλεγχο συμφόρησης (με περισσότερο pipelining) μέσω UDP

• περισσότερα για το HTTP/3 στο επίπεδο Μεταφοράς

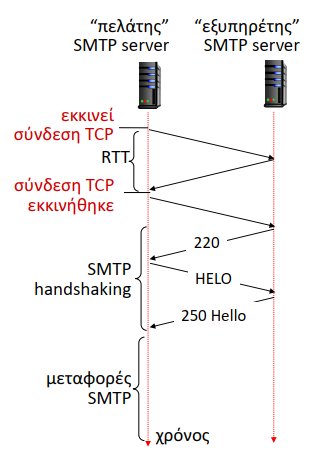
E-mail, SMTP, IMAP

E-mail

Τρία κύρια συστατικά:  
▪ user agents  
▪ mail servers  
▪ simple mail transfer protocol: SMTP  
User Agent  
▪ δλδ. “mail reader”  
▪ συνθέτει, τροποποιεί, διαβάζειμηνύματα mail  
▪ π.χ., Outlook, iPhone mail client  
▪ εξερχόμενα, εισερχόμενα μηνύματα αποθηκεύονται στον εξυπηρέτη

E-mail: εξυπηρέτες mail

εξυπηρέτες mail:  
▪ *mailbox* περιέχει εισερχόμενα μηνύματα για τον χρήστη  
▪ *ουρά μηνυμάτων* εξερχομένων (προς αποστολή)  
πρωτόκολλο SMTP μεταξύ  
εξυπηρετών mail για αποστολή  
μηνυμάτων email  
▪ πελάτης: στέλνει σε εξυπηρέτη mail  
▪ “server”: εξυπηρέτης mail – λήπτης

SMTP RFC (5321)

▪ χρησιμοποιεί TCP για αξιόπιστη μεταφορά email  
από πελάτη (εξυπηρέτης mail εκκινεί δημιουργία σύνδεσης) προς εξυπηρέτη, θύρα 25

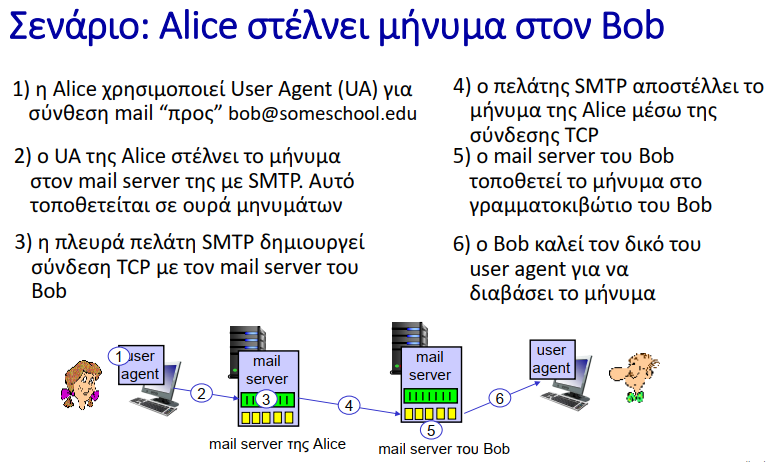
▪ απ’ ευθείας μεταφορά: αποστέλλων εξυπηρέτης  
(ενεργώντας ως πελάτης) προς λαμβάνοντα εξυπηρέτη

▪ τρεις φάσεις στην μεταφορά

• χειραψία (SMTP handshaking (χαιρετισμός))  
• μεταφορά μηνυμάτων SMTP  
• κλείσιμο SMTP

▪ διάδραση τύπου εντολή/απόκριση (όπως HTTP)

• εντολές: κείμενο ASCII  
• απόκριση: κωδικός κατάστασης και φράση



SMTP: παρατηρήσεις

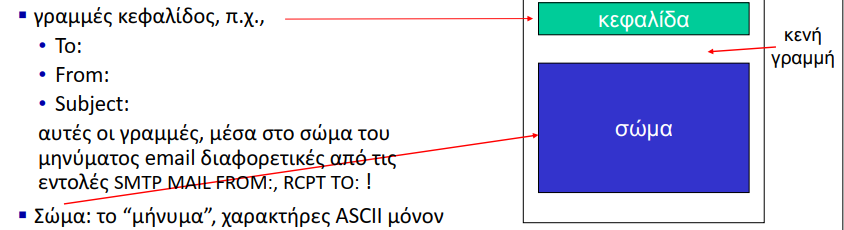
*σύγκριση με HTTP:*▪ HTTP: client pull (δημιουργεί σύνδεση περιοδικά, παίρνει αυτό που θέλει και μετά την κλείνει)  
▪ SMTP: client push (δημιουργεί σύνδεση που κρατά ανοικτή διαρκώς, ενώ ο server στέλνει ό,τι νεότερο έχει)  
▪ αμφότερα έχουν διάδραση τύπου εντολή/ απόκριση σε ASCII, κωδικούς κατάστασης  
▪ HTTP: κάθε αντικείμενο ενθυλακώνεται στο δικό του μήνυμα απόκρισης  
▪ SMTP: αποστέλλονται πολλαπλά αντικείμενα σε μήνυμα με πολλαπλά μέρη

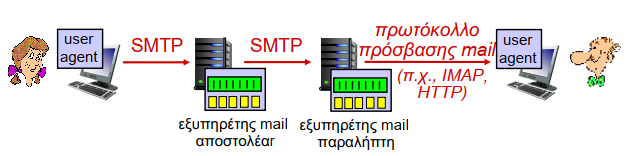
▪ το SMTP χρησιμοποιεί persistent (επίμονες/παραμένουσες) συνδέσεις  
▪ το SMTP απαιτεί το μήνυμα (κεφαλίδα & σώμα) να είναι σε ASCII 7-bit  
▪ ο εξυπηρέτης SMTP χρησιμοποιεί τους χαρακτήρες <CRLF> <CRLF> για να προσδιορίσει το τέλος του μηνύματος (οπτικά κενή γραμμή)

Μορφή μηνύματος mail

* SMTP: πρωτόκολλο για ανταλλαγή μηνυμάτων email, ορισμένο στο RFC 5321

(όπως RFC 7231 ορίζει το HTTP)

* RFC 2822 ορίζει το *συντακτικό* για τα ίδια τα μηνύματα e-mail (όπως το
* HTML ορίζει το συντακτικό για έγγραφα web)

Ανακτώντας email: πρωτόκολλα πρόσβασης mail

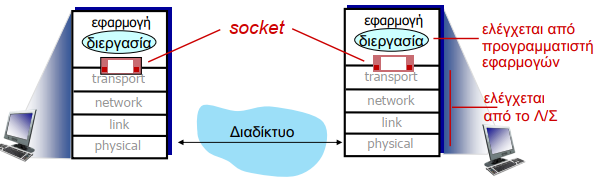
▪ SMTP: παράδοση/αποθήκευση στον εξυπηρέτη του παραλήπτη  
▪ πρωτόκολλο πρόσβασης mail: ανάκτηση από εξυπηρέτη

• IMAP: Internet Mail Access Protocol [RFC 3501]: μηνύματα αποθηκεύονται στον  
εξυπηρέτη, IMAP παρέχει ανάκτηση, διαγραφή, φακέλους αποθηκευμένων  
μηνυμάτων στον εξυπηρέτη

▪ HTTP: gmail, Hotmail, Yahoo!Mail, κλπ. παρέχει διεπαφή βασιζόμενη στο web επάνω  
από STMP για αποστολή, IMAP (ή POP) για ανάκτηση μηνυμάτων e-mail

**Προγραμματισμός socket με UDP και TCP**

Προγραμματισμός Socket

***στόχος****:* να μάθετε πώς να δημιουργείτε εφαρμογές πελάτη/εξυπηρέτη που επικοινωνούν μέσω sockets  
***socket****:* πόρτα μεταξύ διεργασίας εφαρμογής και πρωτοκόλλου μεταφοράς από-άκρο-σε-άκρο

Δύο τύποι socket για δύο υπηρεσίες μεταφοράς:

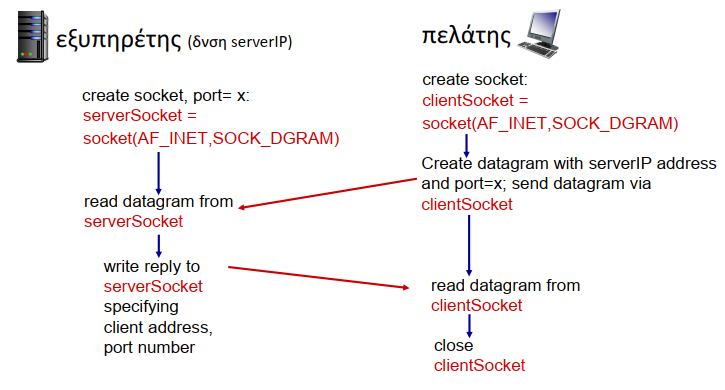
▪ *UDP:* αναξιόπιστο δεδομενόγραμμα

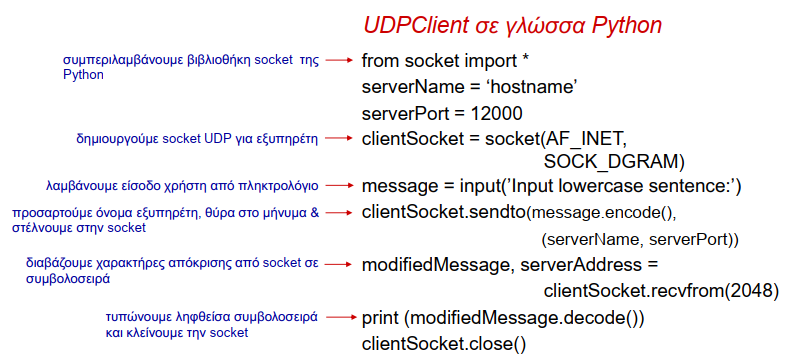
▪ *TCP:* αξιόπιστο, προσανατολισμένο σε ρεύμα από byte

Παράδειγμα Εφαρμογής:  
**1.** πελάτης διαβάζει μία γραμμή χαρακτήρων (δεδομένα) από το πληκτρολόγιό του και τα αποστέλλει στον εξυπηρέτη  
**2.** εξυπηρέτης λαμβάνει τα δεδομένα και μετατρέπει τους χαρακτήρες σε κεφαλαίους  
**3.** Εξυπηρέτης αποστέλλει τροποποιημένα δεδομένα στον πελάτη  
**4.** πελάτης λαμβάνει τροποποιημένα δεδομένα και εμφανίζει γραμμή στην οθόνη του

Προγραμματισμός Socket με UDP

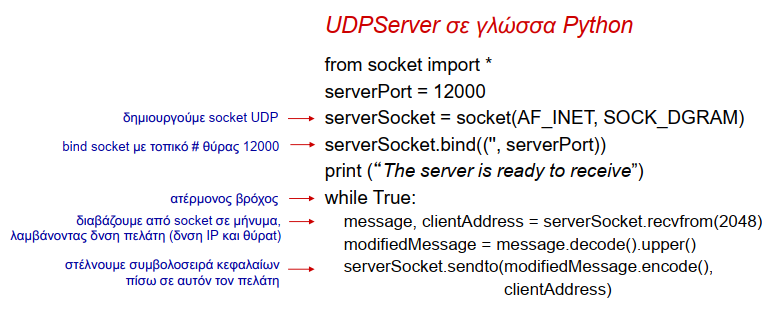
UDP: καμία “σύνδεση” μεταξύ πελάτη & εξυπηρέτη:  
▪ καμία συμφωνία (handshaking) πριν αποστολή δεδομένων  
▪ αποστολέας προσαρτά ρητά διεύθυνση IP και #θύρας προορισμού σε κάθε πακέτο  
▪ Παραλήπτης εξάγει διεύθυνση IP και #θύρας αποστολέα από το ληφθέν πακέτο  
UDP: μεταδοθέντα δεδομένα μπορεί να χαθούν ή να ληφθούν εκτός σειράς  
Σκοπιά εφαρμογής:  
▪ UDP παρέχει *αναξιόπιστη* μεταφορά ομάδων byte (“δεδομενογράμματα”) μεταξύ διεργασιών πελάτη και εξυπηρέτη

Διάδραση socket πελάτη/εξυπηρέτη: UDP

 Παράδειγμα εφαρμογής: πελάτης UDP

2.AF\_INET:εκδοση 4

Παράδειγμα εφαρμογής: εξυπηρέτης UDP

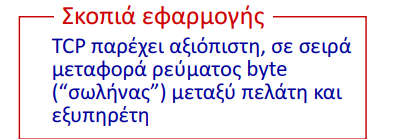


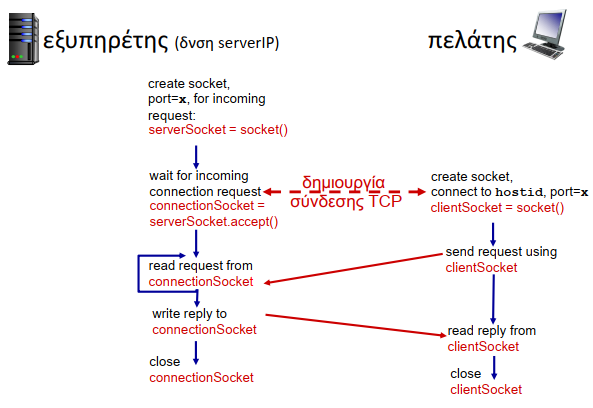
Προγραμματισμός Socket με TCP

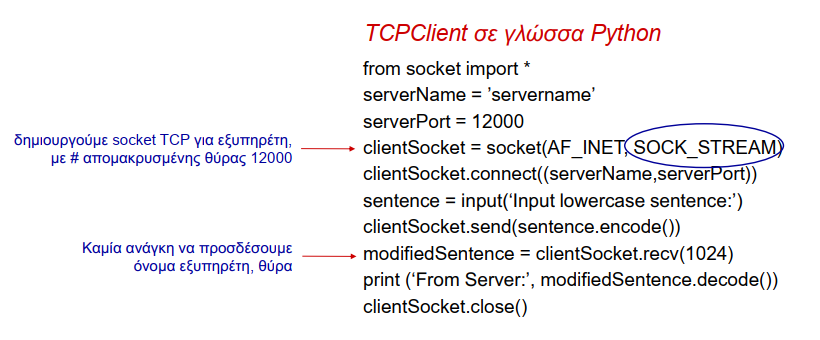
πελάτης πρέπει να επικοινωνήσει με εξυπηρέτη  
▪ διεργασία εξυπηρέτη πρέπει πρώτα να τρέχει  
▪ εξυπηρέτης πρέπει να έχει δημιουργήσει socket (πόρτα) για να καλωσορίζει επαφές από πελάτες  
πελάτης έρχεται σε επαφή με εξυπηρέτη μέσω:  
▪ δημιουργίας socket TCP, προσδιορίζοντας δνση IP, αριθμό θύρας διεργασίας εξυπηρέτη  
▪ *όταν πελάτης δημιουργήσει socket:* TCP πελάτη εγκαθιστά σύνδεση με TCP εξυπηρέτη

▪ όταν έρχεται σε επαφή με πελάτη, *TCP εξυπηρέτη δημιουργεί μία νέα socket* ώστε η διεργασία εξυπηρέτη να επικοινωνήσει με τον συγκεκριμένο πελάτη

• επιτρέπει στον εξυπηρέτη να μιλά με πολλαπλούς πελάτες

• *αριθμοί θύρας αποστολέα χρησιμοποιούνται για να διακρίνει μεταξύ πελατών (περισσότερα στο Κεφ. 3)*

Διάδραση socket πελάτη/εξυπηρέτη: TCP

Παράδειγμα εφαρμογής: πελάτης TCP

Παράδειγμα εφαρμογής: εξυπηρέτης TCP

