Μια μικρή εισαγωγή στο TCP/IP

Μέρος 1ο
Το IP και τα βοηθητικά πρωτόκολλα
(v0.9)

mickflemm 'at' gmail 'dot' com

ΠΕΡΙΕΧΟΜΕΝΑ

Επειδή η γνώση πρέπει να είναι ελεύθερη, όπως μας δόθηκε, το παρόν κείμενο διαδίδεται βάσει της άδειας Creative Commons Attribution-non commercial-share alike v3 την οποία μπορείτε να βρείτε στα Ελληνικά (είναι draft ακόμα) εδώ ->
http://www.ebusinessforum.gr/teams/teamsall/view/inner/index.php?language=el&ctn=97&moduleid=9&label=51
και το επίσημο κείμενο εδώ ->
http://creativecommons.org/licenses/by-nc-sa/3.0/

Μπορείτε να χρησιμοποιήσετε οποιοδήποτε μέρος του tutorial σε κάποιο δικό σας tutorial (γι' αυτό και δεν χρησιμοποιώ την non-derivatives άδεια) αλλά όχι ολόκληρο το tutorial χωρίς να επικοινωνήσετε πρώτα μαζί μου. Επίσης αν θέλετε το Tutorial σε μορφή Open Document Format μπορείτε επίσης να επικοινωνήσετε μαζί μου. Η τελευταία έκδοση του tutorial θα βρίσκεται στο forum του awmn (www.awmn.net). Όλο το tutorial φτιάχτηκε χρησιμοποιώντας ελεύθερο λογισμικό και συγκεκριμένα το OpenOffice και το GIMP.

Οι εικόνες δεν είναι δικές μου και άρα δεν καλύπτονται από την άδεια χρήσης (έχουν άλλους όρους χρήσης), εκτός από το σχεδιάγραμμα στην παράγραφο 2.1, στην παράγραφο 3.2.3 και τα screenshots στο παράρτημα β όπου καλύπτονται κανονικά. Ποιο συγκεκριμένα...

α) Η εικόνα 1 στην παράγραφο 1 είναι του Joseph A. Carr (http://joecarr.ca/copyright.htm) ©1975

β) Οι χάρτες του ARPANET (εικόνα 2 στην παράγραφο 2.3 και η εικόνα στην παράγραφο 3.2.2.1) υπάρχουν σε διάφορα μέρη στο διαδίκτυο. Η δεύτερη του 1982 είναι του Jon Postel (του πρώτου RFC editor) και δεν έχει copyright αφού ανήκει στο DARPA και άρα στο public domain όπως λέει και η wikipedia -> http://en.wikipedia.org/wiki/Image:Internet_map_in_February_82.jpg
Τη πρώτη εικόνα την βρήκα εδώ ->
http://archive.computerhistory.org/resources/still-image/Arpanet/
και δεν συνοδεύεται ούτε αυτή από κάποιο copyright.

γ) Παράγραφος 2.5 ->
Η μεγάλη εικόνα με τα layers είναι απ' τη wikipedia και θα τη βρείτε σε διάφορα tutorials στο διαδίκτυο (δεν ξέρω γιατί την έβγαλαν απ΄ τη wikipedia).
Η μικρότερη εικόνα με τα layers είναι από εδώ ->
http://www.techexams.net/technotes/ccna/osimodel.shtml
και είναι copyrighted αλλά δεν μπόρεσα να βρω κάποιο license και η σελίδα δεν λέει “all rights reserved” οπότε δεν είναι σαφές το πώς προστατεύεται. Το copyright δεν παραβιάζεται εφόσον παραπέμπω στη σελίδα.

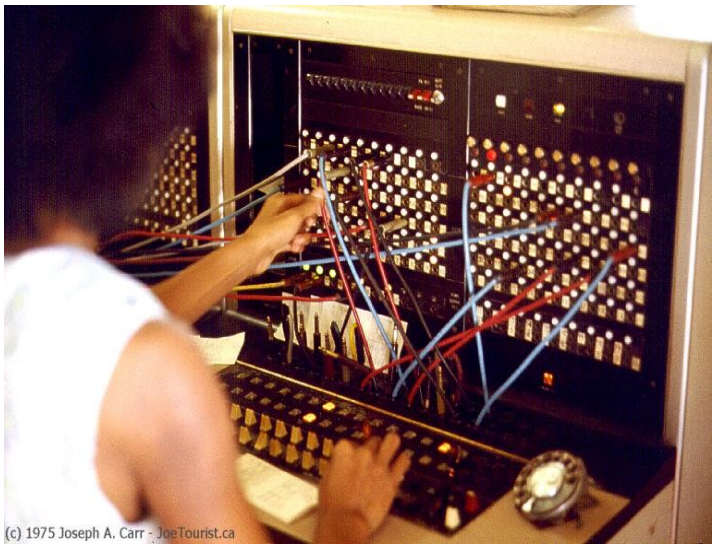
Τις υπόλοιπες εικόνες τις βρήκα σε διάφορα μέρη ψάχνοντας στο google και δεν βρήκα κάποιο copyright. Αν βρείτε κάποια από αυτές τις εικόνες με copyright παρακαλώ ειδοποιήστε με για να το συμπεριλάβω.

Αφιερωμένο στα παιδιά του AWMN, τους γνωστούς και τους άγνωστους, τα παιδιά του Hellug γιατί είναι υπέροχοι και τέλος σε όλους όσους επιμένουν να είναι ρομαντικοί παρά τη μιζέρια γύρω τους. Επίσης το αφιερώνω στα παιδιά που δουλέψαμε τον τελευταίο καιρό μαζί για κάτι καλύτερο γύρω μας, τον Αστέρη, τον Δημήτρη, τον Αντώνη, τον Άγγελο, την Αίγλη, την Κατερίνα, την Κωνσταντίνα, τον Τάσσο, τη Μαρία, τον Κυριάκο, την Πέπη, τον Μιχάλη, τον Θόδωρο, τον Γιάννη, τον Γιώργο, τον Θανάση, τη Σοφία, τα φιλαράκια μου που δεν βρήκα καθόλου χρόνο να τους δω, τον Βασίλη, τον Ιάσονα, τον Θάνο, τον Φώτη και τον Σωτήρη και όλους όσους ξεχνώ και δεν με ξεχνούν. Να 'στε όλοι καλά !

1. ΕΙΣΑΓΩΓΗ

Η ιστορία μας ξεκινάει στα τέλη του 60 με αρχές 70, μέχρι τότε τα διάφορα δίκτυα υπολογιστών που υπήρχαν ήταν πολύ εξειδικευμένα και χρησιμοποιούνταν βασικά για έρευνα., παρείχαν δε μόνο επικοινωνία μεταξύ των διαφόρων τερματικών τους με συνδέσεις σημείου-προς-σημείο (point-to-point). Όταν λέμε σημείου-προς-σημείο εννοούμεότι για κάθε σύνδεση μεταξύ των τερματικών χρησιμοποιούνταν ένα ολόκληρο δεσμευμένο κύκλωμα, όπως ήταν το παλιό τηλεφωνικό δίκτυο που βλέπουμε καμιά φορά στις ταινίες με τους "operators" να αλλάζουν καλώδια σε έναν πίνακα στον οποίο καταλήγουν τα διάφορα τερματικά είχαμε δηλαδή το λεγόμενο "circuit switching" (εναλλαγή κυκλωμάτων). Για να το καταλάβουμε αυτό λίγο καλύτερα ας δούμετι έκανε ο "operator" στο παλιό τηλεφωνικό δίκτυο.

Ο "operator" λοιπόν ήταν ο άνθρωπος που καθόταν στο "κέντρο" και ανάλογα με το που του ζήταγε ο κάθε χρήστης, συνέδεε το άκρο του τηλεφώνου του χρήστη με κάποιο άλλο άκρο, με αποτέλεσμα τελικά να σχηματίζεται ένα κύκλωμα μεταξύ των δύο άκρων που ήθελαν να μιλήσουν. Έτσι λέμε ότι υπήρχε ένα κανάλι επικοινωνίας μεταξύ τους το οποίο ήταν συνεχώς δεσμευμένο και δεν μπορούσε να χρησιμοποιηθεί από κάποιον άλλο. Αργότερα την θέση του ανθρώπου πήρε ένα μηχάνημα που αντί να του λέμε με ποια πόλη πχ. να μας συνδέσει, πληκτρολογούσαμε έναν χαρακτηριστικό αριθμό (στην Ελλάδα τον λέγαμε και "αυτόματο" κάποτε κι αυτή η δομή των αριθμών υπάρχει ακόμα και σήμερα, το 210 πχ. είναι ο "αυτόματος" για την Αθήνα και το 2810 ο "αυτόματος" για το Ηράκλειο κλπ -παρόλο που πλέον το τηλεφωνικό δίκτυο δεν δουλεύει έτσι).



Εικόνα 1: Ένα "switchboard", ο πίνακας στον οποίο ο operator άλλαζε τα κυκλώματα

Η τεχνική αυτή μπορεί να φαίνεται πολύ λογική όταν αναφερόμαστε στο τηλεφωνικό δίκτυο γιατί όταν μιλάμε στο τηλέφωνο δεν έχει νόημα να χρησιμοποιεί και άλλος το ίδιο κύκλωμα (εκτός αν κρυφακούει) αλλά για ένα δίκτυο υπολογιστών δεν είναι καθόλου καλή πρακτική. Αρχικά το να παραμένει το κύκλωμα δεσμευμένο ουσιαστικά το καθιστά άχρηστο για κάποιον άλλο να το χρησιμοποιήσει, ακόμα κι αν δεν μεταφέρονται εκείνη την στιγμή δεδομένα. Στο τηλέφωνο δεν έχει νόημα την ώρα που περιμένεις κάποιον να έρθει να σου μιλήσει, να αρχίσουν δυο άλλοι να μιλάνε στην γραμμή, σε ένα δίκτυο υπολογιστών όμως έχει νόημα και μάλιστα είναι αναγκαίο. Σκεφτείτε το πλήθος των κυκλωμάτων που θα έπρεπε να φτιάξουμε για να επικοινωνούν όλοι οι υπολογιστές ενός δωματίου μεταξύ τους ταυτόχρονα, θα έπρεπε ο κάθε υπολογιστής να έχει ένα κύκλωμα με καθέναν απ' τους υπόλοιπους, για ένα δίκτυο 5 υπολογιστών δηλαδή θα χρειαζόμασταν 10 κυκλώματα (4 κυκλώματα ο κάθε υπολογιστής /2) στο ίδιο δωμάτιο (μιλάμε για πολλά καλώδια, φανταστείτε τι γίνεται δε όσο το δίκτυο μεγαλώνει, για 6 υπολογιστές πχ. θέλουμε ξαφνικά 15 κυκλώματα !!).

Επίσης σε ένα δίκτυο υπολογιστών χρειάζεται κάποια σταθερότητα, μπορεί εκ πρώτης όψεως να φαίνεται ότι ένα "δεσμευμένο κύκλωμα" είναι κάτι αρκετά σταθερό αλλά στην πράξη αν αυτό το κύκλωμα διακοπεί για διαφόρους λόγους (πχ. να κοπεί ένα καλώδιο, να πέσει κάπου στιγμιαία το ρεύμα, να ρίξει κάποιος τον καφέ του εκεί που δεν έπρεπε κλπ), τότε θα πρέπει να αναδημιουργηθεί το κύκλωμα απ' την αρχή, κάτι που έχει ως αποτέλεσμα την απώλεια δεδομένων αλλά και την διακοπή κάθε επικοινωνίας μεταξύ των άκρων μέχρι να στηθεί το νέο κύκλωμα. Αυτό ίσως το θυμάστε αρκετοί από τις dial-up συνδέσεις όπου ουσιαστικά καλώντας τον ISP σας δημιουργούταν ένα (εικονικό, γιατί

σχηματιζόταν από τα ήδη υπάρχοντα κυκλώματατου τηλεφωνικού δικτύου -θα το δείτε και ως virtual circuit-) δεσμευμένο κύκλωμα μεταξύ του modem σας και του modem του ISP. Αν αυτό το κύκλωμα "έπεφτε" τότε διακόπτονταν και η σύνδεσή σας και έπρεπε να ξανακάνετε dial. Αν δε σηκώνατε το τηλέφωνο ή κάποιος σας καλούσε, αν είχε πολύ υγρασία κλπ η σύνδεση έπεφτε και πάλι αφού δημιουργούνταν παρεμβολέςμε αποτέλεσμα να δημιουργείται στο κύκλωμα "θόρυβος" και η επικοινωνία να διακόπτεται.

Για να λυθούν τα παραπάνω προβλήματα έπρεπε να αναθεωρήσουμε τον τρόπο με τον οποίο μεταφέρονταν τα δεδομένα. Για να γίνουν τα κυκλώματα κοινόχρηστα και να μην παραμένουν δεσμευμένα, τα δεδομένα έπρεπε να μεταφέρονται ασυνεχώς, έτσι ώστε να δίνεται η ευκαιρία σε όλους να χρησιμοποιήσουν το κύκλωμα. Επίσης για να υπάρχει σταθερότητα έπρεπε τα δεδομένα να μπορούν να ακολουθούν διαφορετικές διαδρομές ανάλογα με τις συνθήκες, να δρομολογούνται δηλαδή αυτόματα μέσα από ένα πλήθος υπαρχόντων κυκλωμάτων προκειμένου να φτάσουν τον προορισμό τους.

Τελικώς επικράτησε ένα μοντέλο που αντί για το τηλεφωνικό δίκτυο της εποχής έμοιαζε ποιο πολύμε το ταχυδρομείο. Τα δεδομένα λοιπόν οργανώθηκαν σε "μονάδες πληροφορίας" (που μπορούμε να τις παρομοιάσουμε με ταχυδρομικούς φακέλους)οι οποίες μεταφέρονταν μέσω των διαφόρων κυκλωμάτων και δρομολογούνταν κατάλληλα ώστε να φτάσουν από τον αποστολέα στον παραλήπτη τους κι έτσι από το "circuit switching" φτάσαμε στην τεχνολογία του "packet switching"...

Προηγουμένως μιλάγαμε απλά για τα άκρα ενός κυκλώματος, όταν όμως μιλάμε για αποστολέα και παραλήπτη το πράγμα γίνεται ποιο ενδιαφέρον...

Όταν παίρνουμε τηλέφωνο κάποιον στην πραγματικότητα η επικοινωνία μας γίνεται σεδύο "επίπεδα" απ' την μία δημιουργείται μια σύνδεση μεταξύ των τηλεφωνικών συσκευών μας κι απ' την άλλη γίνεται μια "σύνδεση" μεταξύ των ατόμων που μιλάν στο τηλέφωνο. Όταν "παίρνουμε τηλέφωνο τον Γιάννη" το τηλεφωνικό δίκτυο δεν έχει ιδέα ούτε ποιος είμαστε εμείς, ούτε ποιος είναι ο Γιάννης, το μόνο που ξέρει είναι ότι η τηλεφωνική συσκευή του σπιτιού μας συνδέεται με την τηλεφωνική συσκευή του σπιτιού του Γιάννη και ότι κάποιος σήκωσε το τηλέφωνο. Τώρα στο επίπεδο της ανθρώπινης "σύνδεσης" όταν παίρνουμε τηλέφωνο τον Γιάννη μπορεί να το σηκώσει ο ίδιος ή οποιοσδήποτε άλλος βρίσκεται στο σπίτι του, μπορεί να θέλουμε να μιλήσουμε με τον Γιάννη, τον αδερφό του Γιάννη ή και όλη την οικογένεια (με ανοιχτή ακρόαση πχ.) και πάλι το τηλεφωνικό δίκτυο δεν έχει καμία σχέση με το ποιος χρησιμοποιεί το τηλέφωνο και πώς.

Άρα θα μπορούσαμε να πούμε ότι έχουμε δυο δίκτυα, το δίκτυο δεδομένων και το δίκτυο πληροφοριών, το δίκτυο δεδομένων στο παραπάνω παράδειγμα είναι το τηλεφωνικό δίκτυο και το δίκτυο πληροφοριών είναι το δίκτυο των ανθρώπων που επικοινωνούν χρησιμοποιώντας το τηλέφωνο. **Το δίκτυο δεδομένων δεν ξέρει τι μεταφέρει παρά μόνο το πώς θα το μεταφέρει, ενώ το δίκτυο πληροφοριών ξέρει τι μεταφέρει και δεν το ενδιαφέρει πώς θα το μεταφέρει** (μπορεί να πάρουμε τον Γιάννη στο κινητό πχ. και πάλι για εμάς δεν θα αλλάξει τίποτα παρόλο που το τηλεφωνικό δίκτυο άλλαξε εντελώς, δεν μας ενδιαφέρει μέσω ποιου τηλεφωνικού δικτύου θα μιλήσουμε με τον Γιάννη αλλά το τι θα του πούμε γιατί εμείς και ο Γιάννης ανήκουμε στο δίκτυο πληροφοριών).

Σημείωση:
Για λόγους συντομίας από εδώ και πέρα θα λέμε το δίκτυο δεδομένων "**φυσικό δίκτυο**" και το δίκτυο πληροφοριών "**λογικό δίκτυο**".

Πριν προχωρήσουμεστο κυρίως θέμα του tutorial ας δούμε λίγο καλύτερα την ιδέα αυτή των "επιπέδων".

2.ΤΑ ΕΠΙΠΕΔΑ (LAYERS)

2.1.Τα πρωτόκολλα

Ας προσπαθήσουμε να τυποποιήσουμε λίγο την παραπάνω διαδικασία και να την δούμε βήμα προς βήμα...

Οι λειτουργίες που έγιναν στο επίπεδο του λογικού δικτύου ήταν οι εξής:
α) Ταυτοποίηση του αποστολέα και του παραλήπτη, μέσω του ονόματός τους (Γιάννης, Γιώργος κλπ).
γ) Ταίριασμα μεταξύ του χρήστη και του αριθμού τηλεφώνου του (το τηλέφωνο του Γιάννη).
δ) Μεταφορά πληροφορίας με τη μορφή φωνής.

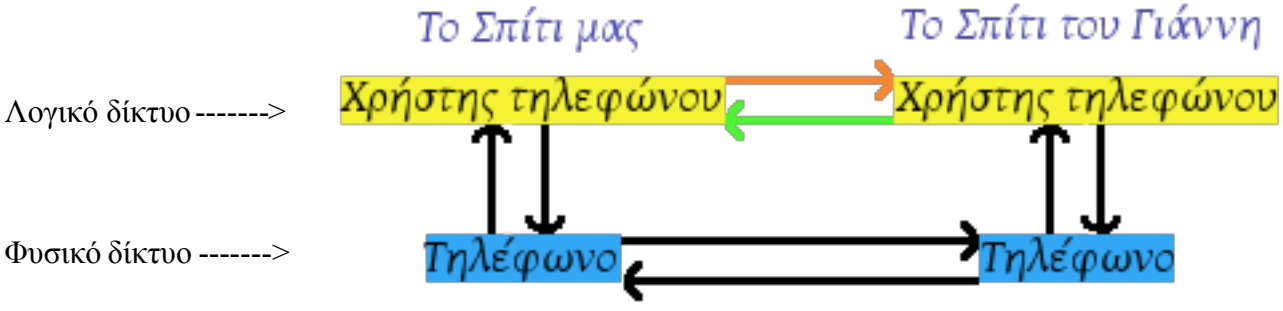
Και οι λειτουργίες που έγιναν στο επίπεδο του φυσικού δικτύου:
α) Ταυτοποίηση συσκευής Α και συσκευής Β, μέσω του τηλεφωνικού αριθμού
β) Σύνδεση μεταξύ των συσκευών
γ) Μεταφορά δεδομένων ήχου.

Παρατηρούμε ότι ενώ το κάθε επίπεδο είναι αυτόνομο, αναλαμβάνει δηλαδή συγκεκριμένη δουλειά, δεν είναι και ανεξάρτητο. Για να επικοινωνήσουμε με τον Γιάννη χρειαζόμαστε τον αριθμό τηλεφώνου του (χαρακτηριστικό του φυσικού δικτύου δηλαδή), καθώς και τον ίδιο το φυσικό δίκτυο, αφού χωρίς τηλέφωνο δουλειά δεν γίνεται.

Έτσι λέμε ότι σε κάθε επίπεδο γίνεται μια συγκεκριμένη λειτουργία η οποία ορίζεται από ένα συγκεκριμένο **πρωτόκολλο επικοινωνίας**, ένα σύνολο κανόνων δηλαδή που περιγράφουν την μορφή που θα έχει η κάθε "**μονάδα πληροφορίας**" στο συγκεκριμένο επίπεδο (φράση στο λογικό δίκτυο, ηλεκτρικά σήματα στο φυσικό δίκτυο) και τις διαδικασίες για την κωδικοποίηση και την αποκωδικοποίηση αυτής της μονάδας..

Μέχρι τώρα λοιπόν έχουμε περιγράψει δύο πρωτόκολλα επικοινωνίας, αυτό που χρησιμοποιείται απ' το τηλεφωνικό δίκτυο (φυσικό δίκτυο) κι αυτό που χρησιμοποιείται απ' τους ανθρώπους (λογικό δίκτυο). Το πρωτόκολλο του τηλεφωνικού δικτύου παρέχει κάποιες υπηρεσίες στο πρωτόκολλο επικοινωνίας των ανθρώπων και το πρωτόκολλο επικοινωνίας των ανθρώπων χρησιμοποιεί αυτές τις υπηρεσίες.

Μπορούμε έτσι να ορίσουμε μια ιεραρχία στα πρωτόκολλα, ξεκινώντας απ' το φυσικό δίκτυο και προχωρώντας στο λογικό...



Αυτό που έχει σημασία είναι **οι υπηρεσίες που παρέχει το κάθε επίπεδο στο επόμενο και η μορφή των πληροφοριών που δίνει το κάθε επίπεδο στο προηγούμενο**, για παράδειγμα για να δουλέψει το πρωτόκολλο "Χρήστης Τηλεφώνου" χρειάζεται την "υπηρεσία μεταφοράς δεδομένων ήχου". Αν αυτή την υπηρεσία μπορεί να του την δώσει ένα άλλο πρωτόκολλο επικοινωνίας, όπως πχ. το "Κινητό Τηλέφωνο" τότε το πρωτόκολλο "Κινητό Τηλέφωνο"μπορεί να αντικαταστήσειτο "Τηλέφωνο" και η επικοινωνία με το πρωτόκολλο "Χρήστης Τηλεφώνου" να μπορεί να γίνει κανονικά. Αντίστοιχα για να δουλέψει το πρωτόκολλο "Τηλέφωνο" πρέπει να του δίνουμε πληροφορίες με τη μορφή ήχου, έτσι αν υπάρχει ένα άλλο πρωτόκολλο πχ. "Σήματα Μορς" που μεταφέρει πληροφορία με τη μορφή ήχου, μπορεί να χρησιμοποιηθεί αντί του πρωτοκόλλου "Χρήστης Τηλεφώνου".

2.2.Το Encapsulation

Ας δούμε τώρα τι θα γίνει αν πάνω απ' το επίπεδο του λογικού δικτύου βάλουμε κι ένα άλλο επίπεδο επικοινωνίας. Έστω λοιπόν ότι εκεί που μιλάμε με τον Γιάννη στο τηλέφωνο, η μητέρα μας μας λέει να πούμε στον Γιάννη να ρωτήσει την μητέρα του πώς φτιάχνει κέικ σοκολάτα (υποθέτουμε εδώ ότι η μητέρα μας εκτός από υπολογιστές δεν ξέρει και τίποτα από κέικ οπότε θέλει ανάλυση). Την μητέρα μας δεν την νοιάζει ούτε ο διάλογος που θα κάνουμε με τον Γιάννη (πχ. πόσο καλό είναι το κέικ κλπ), ούτε πώς θα μιλήσουμε με τον Γιάννη (αν θα μιλήσουμε μέσω τηλεφώνου ή αν θα πάμε απ' το σπίτι να του το πούμε -και να έχουμε και μία καλύτερη άποψη του κέικ-), το μόνο που την νοιάζει είναι να της φέρουμε την συνταγή, να δημιουργήσουμε δηλαδή μια "σύνδεση" μεταξύ αυτής και της μητέρας του Γιάννη.

Έτσι μπορούμε να πούμε ότι το νέο επίπεδο που δημιουργείται, θα είναι πάνω απ' το λογικό δίκτυο αφού προκειμένου να υπάρξει επικοινωνία σε αυτό το επίπεδο, χρειάζεται να μεσολαβήσουμεμείς. Η υπηρεσία που παρέχεται σε αυτό το επίπεδο για να δουλέψει είναι "η μεταφορά συνταγής" και τα δεδομένα που αυτό το επίπεδο περνάει στο κατώτερο επίπεδο είναι διάφορες πληροφορίες, μπορεί η μητέρα του Γιάννη να του κάνει νοήματα για τις δόσεις στο κέικ, να του δείχνει τις φόρμες, εικόνες στον Τσελεμεντέ κλπ.

Επειδή η παραγωγή πληροφορίας και η μετατροπή της σεκατάλληλη μορφή για να μεταφερθεί είναι δυο διαφορετικές λειτουργίες (δεν είναι δουλειά του Γιάννη που δεν έχει και ιδέα από μαγειρική να προσπαθεί να μας περιγράψει τις διάφορες εικόνες και τα νοήματα της μητέρας του) και επειδή σε κάθε επίπεδο πρέπει να επιτελείται μία συγκεκριμένη λειτουργία, θα σπάσουμε το επίπεδο που πάμε να φτιάξουμε στα δυο. Έτσι η μητέρα μας και η μητέρα του Γιάννη θα είναι το επίπεδο "**επεξεργασίας πληροφορίας**" και μεταξύ εμάς και της μητέρας μας (και του Γιάννη και της μητέρας του) θα υπάρχει ένα επίπεδο "**μεταφοράς πληροφορίας**" (έστω ο πατέρας μας και ο πατέρας του Γιάννη), στο οποίο η συνταγή θα μετατρέπεται σε μονάδες πληροφορίας που καταλαβαίνουμε εμείς και ο Γιάννης (πχ. γραπτό κείμενο) και μπορούμε να μεταφέρουμε όπως ξέρουμε.

Έτσι την υπηρεσία "μεταφορά συνταγής"θα την παρέχει στις μητέρες μας το επίπεδο μεταφοράς πληροφορίας (όχι εμείς άμεσα), στο οποίο θα υπάρχει ένα πρωτόκολλο που θα περιγράφει πχ. το πώς θα λέει η μητέρα του Γιάννη τη συνταγή στον πατέρα του, πως αυτός θα οργανώνει τις πληροφορίες σε κατάλληλη μορφή για να μεταφερθούν και πώς ο πατέρας μας θα κάνει την αντίστροφη διαδικασία. Εμείς δεν έχουμε καμία ιδέα από μαγειρική και άρα δεν μπορούμε να ελέγξουμε την ορθότητα των πληροφοριών που μεταφέρουμε, επειδή όμως το επίπεδο μεταφοράς (οι πατεράδες μας) πρέπει να παρέχει σωστά την υπηρεσία "μεταφορά συνταγής", θα χρειαστεί να έχει έναν μηχανισμό ελέγχου των πληροφοριών που μεταφέρονται. Έτσι πχ. θα σπάει την συνταγή σε ερωτήσεις/ απαντήσεις επειδή θα είναι μικρές σε μέγεθος θα είναι λιγότερες οι πιθανότητες να τις μεταφέρουμε λάθος και θα κάνει τον έλεγχο αν έφτασαν σωστά (αν δεν έφτασαν θα μας τις ξαναδίνει γραπτώς).

Και πάλι κάθε επίπεδο δουλεύει αυτόνομα, στο επίπεδο "επεξεργασίας πληροφορίας" η μητέρα μας θα

έχει ουσιαστικά έναν κανονικό διάλογο με την μητέρα του Γιάννη χωρίς να την ενδιαφέρει πώς οι πληροφορίες μεταφέρονται, από ποιόν και σε ποιον, πώς οργανώνονται κλπ ενώ η μονάδα πληραφορίας που θα χρησιμοποιείται σε αυτό το επίπεδο είναι η συνταγή του κέικ. Το επίπεδο μεταφοράς πληροφορίας απ' την άλλη μετατρέπει την μονάδα πληροφορίας "συνταγή" στην δικιά του μονάδα πληροφορίας, τις "γραπτές ερωτήσεις/ απαντήσεις" και δεντο νοιάζει πώς θα φτάσουν στον προορισμό τους, το μόνο που το νοιάζει είναι να μεταφερθούν σωστά για να μπορεί να δώσει την συνταγή σωστά στο παραπάνω επίπεδο.

Βλέπουμε λοιπόν ότι **το κάθε επίπεδο έχει την δικιά του μονάδα πληροφορίας**, στο επίπεδο επεξεργασίας έχουμε την συνταγή, στο επίπεδο μεταφοράς έχουμε τις γραπτές ερωτήσεις/ απαντήσεις, στο επίπεδο του λογικού δικτύου έχουμε τις φράσεις και στο επίπεδο του τηλεφώνου έχουμε τα ηλεκτρικά σήματα.

Οι μονάδες πληροφορίας μεταφέρονται οριζόντια, απ' το επίπεδο του ενός άκρου (το σπίτι μας) στο αντίστοιχο επίπεδο του άλλου άκρου (το σπίτι του Γιάννη) και κατακόρυφα μεταξύ των διαφορετικών επιπέδων στο κάθε άκρο(από την μητέρα μας στο τηλέφωνο και το ανάποδο).

Αν τώρα παρομοιάσουμε τις μονάδες πληροφορίας με φακέλους, τότε μπορούμε να πούμε ότι ο πατέρας του Γιάννη βάζει τη συνταγή σε έναν κατάλληλο φάκελο προς τον πατέρα μας (οριζόντια επικοινωνία μεταξύ των άκρων στο ίδιο επίπεδο όπως είπαμε), τον οποίο ο Γιάννης παίρνει και βάζει σε έναν φάκελο που προορίζεται για εμάς, μετά γνωρίζοντας την διεύθυνση του σπιτιού μας (όπως αντίστοιχα ξέραμε εμείς τον αριθμό του τηλεφώνου του), ο Γιάννης πάει στο ταχυδρομείο και να τον στείλει στην διεύθυνση του σπιτιού μας. Ο φάκελος αυτός μετά θα μπει σε έναν ταχυδρομικό σάκο, θα φτάσει στο σπίτι μας (άρα σχηματίζεται ένα φυσικό δίκτυο, αυτό του ταχυδρομείου αντί του τηλεφώνου), εμείς θα πάρουμε τον φάκελο που μας έστειλε ο Γιάννης (άρα σχηματίζεται και πάλι το λογικό δίκτυο μεταξύ εμάς και του Γιάννη), θα βγάλουμε από μέσα τον φάκελο που προορίζεται για τον πατέρα μας (το επίπεδο μεταφοράς) και αυτός θα βγάλει την συνταγή από μέσα και θα την δείξει όπως αυτός ξέρει στην μητέρα μας (το επίπεδο επεξεργασίας).

Την διαδικασία αυτή του "φακελώματος" την λέμε **encapsulation** και ουσιαστικά περιγράφει τοπως μια μονάδα πληροφορίας πρωτοκόλλου (Protocol Data Unit ή PDU) μπορεί να "φακελωθεί" μέσα στην μονάδα πληροφορίας ενός άλλου πρωτοκόλλου και το ανάποδο (στην περίπτωσή μας η μονάδα πληροφορίας του πρωτοκόλλου που περιγράφει τη συνταγή, "φακελώνεται" στη μονάδα πληροφορίας του πρωτοκόλλου του επιπέδου μεταφοράς, περιέχεται δηλαδή στις ερωτήσεις/ απαντήσεις που μεταφέρουν οι πατεράδες μας). **Κάθε PDU αποτελείται από μια κεφαλίδα (το τι γράφει πάνω ο φάκελος) + την PDU του ανώτερου επιπέδου (τα δεδομένα δηλαδή που "φακελώνει"). Η κεφαλίδα είναι κι αυτό που χαρακτηρίζει το κάθε πρωτόκολλο**.

Προφανώς όσο κατεβαίνουνε επίπεδα, τόσο οι φάκελοι αυτοί μεγαλώνουν καθότι εκτός απ' την πληροφορία που παίρνουν απ' τα ανώτερα επίπεδα και πρέπει να μεταφέρουν, έχουν κι επιπλέον πληροφορίες. Έτσι πχ. η μονάδα πληροφορίας που μεταφέρουν οι πατεράδες μας περιέχει την συνταγή + τις ερωτήσεις/ απαντήσεις, η μονάδα πληροφορίας που μεταφέρουμε εμείς περιέχει τα προηγούμενα + το όνομα του παραλήπτη, η μονάδα πληροφορίας που μεταφέρεται μέσα απ' το φυσικό δίκτυο τέλος περιέχει τα προηγούμενα + την διεύθυνση του σπιτιού του παραλήπτη. Έτσι γίνεται σαφές ότι το encapsulation γίνεται μόνο από ένα κατώτερο επίπεδο σε κάποιο ανώτερο, δεν μπορούμε πχ. να βάλουμε τον φάκελο με την διεύθυνση του σπιτιού του Γιάννη μέσα στον φάκελο με το όνομά του γιατί το ταχυδρομείο δεν θα ξέρει που να το στείλει.

Ήρθε πλέον η ώρα να πάμε στα δίκτυα υπολογιστών και να αφήσουμε όλη την ιστορία με το κέικ...

2.3.Η αρχή του ARPANET - μια πρώτη εφαρμογή των layers

Είπαμε στην εισαγωγή πως φτάσαμε απ' την εναλλαγή κυκλωμάτων (circuit switching) στην εναλλαγή πακέτων (packet switching) και αναφέραμε ως βασικό λόγο την καλύτερη εκμετάλλευση των κυκλωμάτων και την σταθερότητα του δικτύου. Το βασικό θέμα όμως εκείνη την εποχή, αυτό που έδωσε την αφορμή για να ξεκινήσει το στήσιμο ενός packet switching δικτύου ήταν η της εξάλειψης της ανομοιογένειας των διαφόρων πρωτοκόλλων που υπήρχαν .

Όπως είπαμε λοιπόν τα δίκτυα υπολογιστών τότε ήταν point-to-point, εξειδικευμένα κλπ. Στο Defense Advanced Research Projects Agency (DARPA) του Υπ. αμύνης των Η.Π.Α. κατέληγαν 3 τερματικά, από αντίστοιχα 3 τέτοια εξειδικευμένα δίκτυα, το καθένα με τις δικές του ιδιαιτερότητες (εντελώς διαφορετικό τρόπο επικοινωνίας, διαφορετικό hardware και software δηλαδή). Κρίθηκε λοιπόν αναγκαίο να βρεθεί ένας κοινός τρόπος "συνεννόησης" μεταξύ τους, έτσι ώστε να αντικατασταθούν από ένα τερματικό το οποίο να μπορεί να "μιλάει" και με τα τρία διαφορετικά δίκτυα.

Με το packet switching λυνόταν αυτό το πρόβλημα, αφού όλα τα τερματικά στο δίκτυο θα χρησιμοποιούσαν μια κοινή μονάδα πληροφορίας (ένα κοινό πρωτόκολλο δηλαδή) κι έτσι θα μπορούσαν να μιλήσουν όλοι με όλους (θα εξαλείφονταν η ανομοιογένεια μεταξύ των διαφορετικών πρωτοκόλλων που χρησιμοποιούσαν τα διάφορα δίκτυα), ενώ δεν θα χρειάζονταν να συνδεθούν όλοι με όλους (με δεσμευμένα κανάλια) και θα έφευγαν τα point-to-point links. Έτσι λοιπόν ανατέθηκε στην εταιρία BBN (Bolt, Barenek and Newman) να φτιάξουν το πρώτο packet switching δίκτυο στον κόσμο, το ARPANET και δόθηκε χρηματοδότηση σε διάφορα πανεπιστήμια να το δοκιμάσουν και να το επεκτείνουν. Το packet switching μεταξύ των διαφόρων τερματικών το έκανε ένα μηχάνημα γνωστό ως IMP (Interface Message Processor) το οποίο βρισκόταν σε καθένα από τα διαφορετικά σημεία πρόσβασης και αποτελούσε την πύλη στο ενιαίο δίκτυο. Τα τερματικά (τότε μεγάλα mainframes) συνδεόντουσαν πάνω στο τοπικό IMP και μέσα από το ενιαίο δίκτυο μπορούσαν να μεταφέρουν ταυτόχρονα όλοι δεδομένα. Το κάθε IMP για αρχή μπορούσε να έχει μέχρι 4 τερματικά και να συνδέεται με το πολύ 6 άλλα IMP. **Στην αρχή λοιπόν το ARPANET συνέδεε τερματικά μεταξύ τους**, σε κάθε εγκατάσταση, τα τερματικά συνδέονταν πάνω στο IMP και κατόπιν τα διάφορα IMP συνδέονταν μεταξύ τους.

Στην αρχή το 1969 φτιάχτηκε το πρωτόκολλο "1822" που αποτελούσε τη βάση του ARPANET. Το "1822" περιελάμβανε χοντρικά τις λειτουργίες των 2 πρώτων επιπέδων (τότε η λογική των επιπέδων δεν είχε επικρατήσει ακόμα γι' αυτό και είχαν βάλει όλες τις λειτουργίες σε ένα πρωτόκολλο, δηλαδή στο ίδιο επίπεδο), ήταν δηλαδή ταυτόχρονα πρωτόκολλογια το λογικό και το φυσικό δίκτυο. Η διεύθυνση δε ήταν κοινή για το λογικό και το φυσικό δίκτυο (ένας χαρακτηριστικός αριθμός για κάθε τερματικό). Το κάθε τερματικό λοιπόν, βάσει του 1822, έφτιαχνε μια PDU στην κεφαλίδα της οποίας έβαζε την διεύθυνση του παραλήπτη και τον τύπο του μηνύματος και έδινε την μονάδα αυτή στο IMP μέσω του "1822 interface" (είπαμε ότι το 1822 ήταν ολόκληρος ο μηχανισμός το πρωτόκολλο δηλαδή όριζε και το πως θα είναι η κάρτα δικτύου, τα καλώδια κλπ). Το IMP κατόπιν δρομολογούσε την μονάδα πληροφορίας (το πακέτο δηλαδή που πήρε απ' το ανώτερο επίπεδο) μέσα απ' το υπόλοιπο δίκτυο για να φτάσει στον προορισμό της. Τέλος το 1822 είχε κι έναν μηχανισμό ελέγχου, όταν το πακέτο έφτανε στον προορισμό του, το IMP έστελνε ένα μήνυμα στο τερματικό ότι μπορεί να στείλει το επόμενο κλπ (είχε δηλαδή και την λειτουργία τουεπιπέδου μεταφοράς).

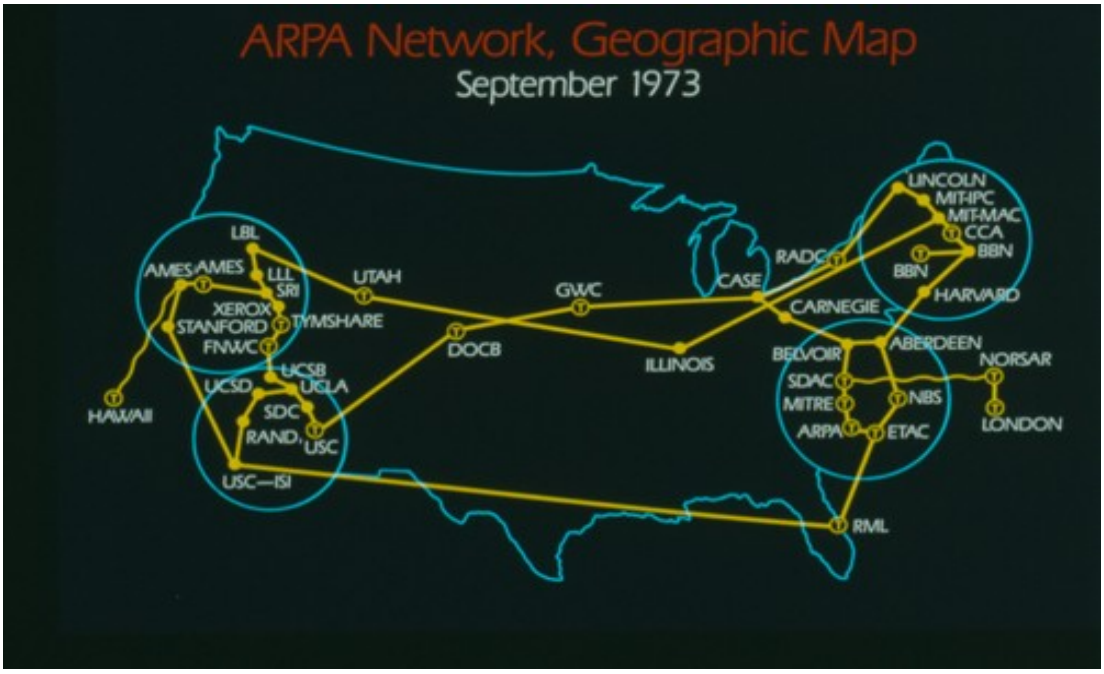
Έτσι σε πρώτη φάση εξαλείφθηκεη ανομοιογένεια σε επίπεδο δικτύου (φυσικού + λογικού) αφού όλα τα τερματικά ακολουθούσαν το ίδιο πρωτόκολλο επικοινωνίας και το μόνο που έπρεπε να γίνει ήταν να φτιαχτούν για τα διάφορα τερματικά κατάλληλες "κάρτες δικτύου" (τότε σειριακά καλώδια) που να ακολουθούν το 1822 καθώς και να γραφτεί το κατάλληλο λογισμικό για τα διάφορα λειτουργικά συστήματα (unix εννοείται) της εποχής.

Με την εξέλιξη του δικτύου άρχισε να φαίνεται ότι ο έλεγχος για το αν η πληροφορία έφτασε σωστά (error detection) ή όχι, καθώς και το flow control (ο έλεγχος της ροής των πληροφοριών, πχ. πότε

ξεκινάει ένας "διάλογος" μεταξύ των τερματικών και πότε σταματάει) δεν μπορούσαν να προσδιοριστούν σωστά για διαφορετικές εφαρμογές που δουλεύουν ταυτόχρονα, έπρεπε δηλαδή να υπάρξει ένα ενδιάμεσο πρωτόκολλο για να εξαλείψει την ανομοιογένεια των εφαρμογών αυτή τη φορά. Έτσι το κενό αυτό ήρθε να καλύψει το Network Control Program (NCP) που ουσιαστικά ήταν **ένα πρωτόκολλο ανάμεσα στις εφαρμογές και το δίκτυο, που λειτουργούσε αυτόνομα** και έδινε πλέον την δυνατότητα διαφορετικών εφαρμογών να μεταφέρουν ταυτόχρονα διαφορετικού τύπου δεδομένα (με διαφορετική δομή κλπ) μέσα απ' το δίκτυο. Ένα νέο επίπεδο δηλαδή.

Μετά κι από αυτή την εξέλιξη άρχισαν να βγαίνουν στο προσκήνιο και οι πρώτες εφαρμογές, βασισμένες στο καινούριο πρωτόκολλο. Το 1971 φτιάχτηκε το e-mail κι αργότερα το FTP, το TELNET κλπ.

Φτάσαμε λοιπόν σε ένα μοντέλο 3 επιπέδων, το επίπεδο των εφαρμογών (αυτό που πριν λέγαμε παραγωγής/ επεξεργασίας πληροφορίας) στο οποίο υπήρχε το e-mail, το FTP κλπ, το επίπεδο ανάμεσα στις εφαρμογές και το δίκτυο (το επίπεδο μεταφοράς που λέγαμε) στο οποίο υπήρχε το NCP και το επίπεδο δικτύου που υπήρχε το 1822 (που περιελάμβανε ουσιαστικά το λογικό και το φυσικό δίκτυο).



Εικόνα 2: Το ARPANET όπως ήταν το 1973 (τα κυκλάκια είναι IMPs)

2.4.Το DoD (Department Of Defense) Model και πώς φτάσαμε στο TCP/IP

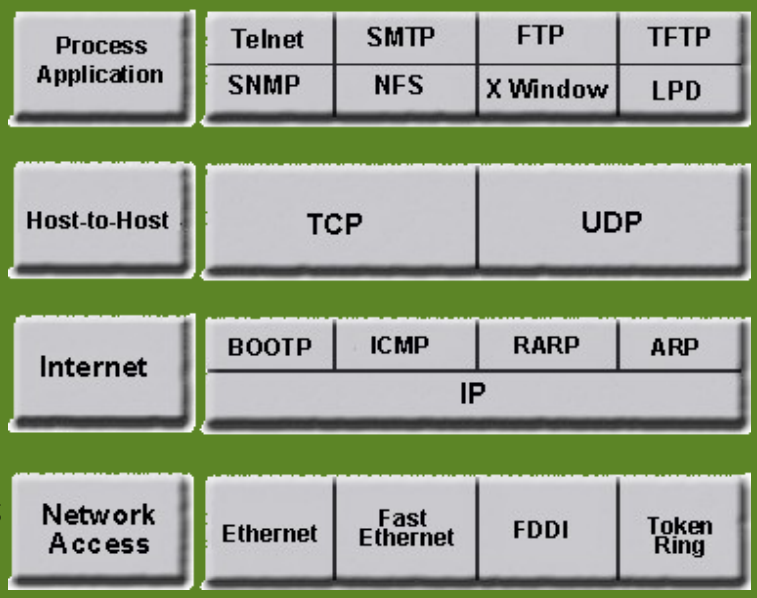
Όσο το ARPANET μεγάλωνε τόσο τα προβλήματα του υπάρχοντος μοντέλου (NCP + 1822) ερχόντουσαν στην επιφάνεια. Εν το μεταξύ γύρω από τα διάφορα τερματικά που απάρτιζαν το ARPANET είχαν αρχίσει να δημιουργούνται διάφορα δίκτυα, πάλι διαφορετικά μεταξύ τους που γέμιζαν το δίκτυο με επιπλέον κίνηση και τα IMP είχαν αρχίσει να φορτώνουν αρκετά. Φάνηκε λοιπόν ότι έπρεπε να περάσει το ARPANET σε ένα νέο μοντέλο λειτουργίας, ένα μοντέλο που σκοπό θα είχε **την διασύνδεση μεταξύ τερματικών που ανήκουν σε διαφορετικά και αυτόνομα δίκτυα** (packet switching δίκτυα εννοείται). Χρειαζόταν δηλαδή ένα **δια-δικτυακό μοντέλο λειτουργίας** που βασική οντότητα του δικτύου θα ήταν τα διάφορα **υπο-δίκτυα** και **όχι τα τερματικά.** Πλέον δηλαδή θα δρομολογούνταν δεδομένα όχι μεταξύ των IMPs αλλά μεταξύ των διαφόρων υπο-δικτύων. Επίσης προκειμένου να αυξηθεί η σταθερότητα και η ταχύτητα του δικτύου, το νέο μοντέλο έπρεπε να δίνει περισσότερες αρμοδιότητες στα τερματικά και **να αφήνει τον κορμό του δικτύου να κάνει τα απολύτως απαραίτητα**, δηλαδή ούτε error corection στα IMP, ούτε flow control -αφού ήδη την είχαν πατήσει με το NCP-, **μόνο να μεταφέρει δεδομένα απ' τον αποστολέα στον παραλήπτη**.

Με αφορμή την ιδέα για το δια-δικτυακό μοντέλο, μια ωραία μέρα ο κος. Vint Cerf , φοιτητής τότε στο UCLA, περιμένοντας να αρχίσει ένα meeting άρχισε να βάζει τις πρώτες πινελιές αυτού του μοντέλου, αργότερα με τον κο. Bob Kahn απ' την BBN συμπλήρωσαν και τις διάφορες λεπτομέρειες. Στην αρχή ονόμασαν το πρωτόκολλό τους TCP (Transmision Control Protocol) και περιελάμβανε το error detection (το οποίο γινόταν πλέον από τα τερματικά και όχι από τα IMP), το flow control, το "πακετάρισμα" των πληροφοριών, τη διευθυνσιοδότηση των πακέτων (έτσι ώστε να περιγράφεται στη διεύθυνση εκτός απ' το τερματικό, το δίκτυο στο οποίο αυτό ανήκει) και τη δρομολόγησή τους μέσα απ' τα διάφορα υπο-δίκτυα. Στη συνέχεια το κομμάτι της διευθυνσιοδότησης/δρομολόγησης κλπ το διαχώρισαν σε ένα δεύτερο πρωτόκολλο, το IP (Internet Protocol), αφού αποτελούσε το σύνολο των διαδικασιών που αφορούσαν το δια-δίκτυο (ούτε το τοπικό δίκτυο δηλαδή, ούτε τις εφαρμογές) και ήταν αυτόνομο.

<u>Διαχωρίστηκε έτσι το επίπεδο του τοπικού δικτύου (φυσικό/ενιαίο δίκτυο) απ' το επίπεδο του δια-δικτύου (το λογικό δηλαδή δίκτυο) κι έτσι τα επίπεδα έγιναν επιτέλους 4</u> όπως και στην ιστορία με το κέικ...

Τα IMP πλέον έπρεπε κι αυτά να αλλάξουν, καθότι έπρεπε πλέον να δρομολογούν δεδομένα μεταξύ διαφορετικών δικτύων και για να γίνει η διαφορά αυτή σαφής, ονομάστηκαν "δρομολογητές" (routers). Οι δρομολογητές για το διαδίκτυο αποτελούσαν τα σημεία απ' τα οποία προωθούνταν κατάλληλα τα δεδομένα, ενώ για τα τοπικά δίκτυα αποτελούσαν τα σημεία εισόδου/εξόδου στο διαδίκτυο. Σαν δομή δε οι δρομολογητές χρησιμοποιούσαν κι αυτοί το ίδιο πρωτόκολλο με τα τερματικά (το IP) και η μόνη τους διαφορά ήταν ότι αφορούσαν μόνο τα 2 πρώτα επίπεδα (του φυσικού δικτύου και του δια-δικτύου). Έτσι το TCP/IP είχε (και έχει μέχρι σήμερα) την παρακάτω δομή...

- Στο 4ο επίπεδο ή **επίπεδο εφαρμογής (application layer)** δημιουργείται από την εφαρμογή μια πληροφορία προς μεταφορά και γίνεται η επεξεργασία της κατά τη λήψη.
- Στο 3ο επίπεδο ή **επίπεδο μεταφοράς (transportation ή Host to Host layer)**, υπάρχουν οι μηχανισμοί ασφαλούς παράδοσης/λήψης της παραπάνω πληροφορίας, το flow control και η πληροφορία οργανώνεται σε τμήματα (segments) έτοιμα για μεταφορά (οι ερωτήσεις/ απαντήσεις που λέγαμε).
- Στο 2ο επίπεδο ή **επίπεδο δια-δικτύου (internetworking layer)**, γίνεται το "πακετάρισμα" των τμημάτων αυτών ώστε να μπορούν να μεταφερθούν μέσω του δια-δικτύου και η διευθυνσιοδότηση τους στο δια-δίκτυο.
- Τέλος στο 1ο επίπεδο η **επίπεδο φυσικής διασύνδεσης (physical link layer)**, τα προς παράδοση πακέτα οργανώνονται κατάλληλα, αποκτούν διεύθυνση παραλήπτη στο τοπικό φυσικό δίκτυο (αν ο παραλήπτης δεν ανήκει στο ίδιο δίκτυο αλλά σε κάποιο άλλο, τα πακέτα αποστέλλονται στον τοπικό δρομολογητή – το IMP δηλαδή που όπως είπαμε είναι το σημείο εισόδου/ εξόδου στο δια-δίκτυο-) και αποστέλλονται μέσω του φυσικού δικτύου.



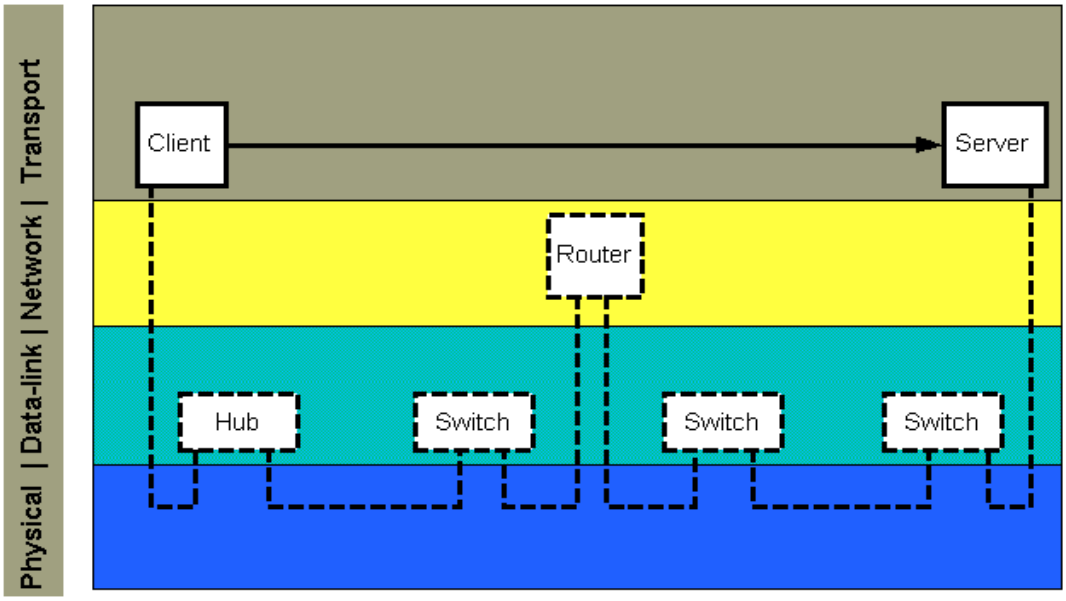
Αναφερθήκαμε εκτενώς στην κατακόρυφη επικοινωνία των επιπέδων και το πώς διαχωρίζονται οι λειτουργίες τους αλλά όχι τόσο στην οριζόντια επικοινωνία που είπαμε στην αρχή (αυτή μεταξύ υμών και του Γιάννη) και πώς αυτή παραμένει αυτόνομη. Ένας καλός τρόπος να το καταλάβουμε αυτό στην

περίπτωση του TCP/IP είναι να δούμε γιατί τελικά ο έλεγχος σφαλμάτων (error detection) δεν γίνεται στους δρομολογητές αλλά στα τερματικά.

Σκεφτείτε ότι το IP μεταφέρει πακέτα μέσα από ολόκληρα δίκτυα, φανταστείτε σε κάθε δρομολογητή να πρέπει να γίνονται έλεγχοι αν το πακέτο έφτασε σωστά απ' τον προηγούμενο δρομολογητή, αυτομάτως θα είχαμε μεγάλο φόρτο στους δρομολογητές και τεράστιο κόστος στην ταχύτητα μεταφοράς. Δείτε το σαν μια σκυταλοδρομία, ο ένας δίνει την σκυτάλη στον άλλο και ο άλλος δεν κάθεται να κοιτάξει αν η σκυτάλη είναι χτυπημένη κλπ, απλά την παίρνει και τρέχει στον επόμενο, αν καθόταν ο καθένας να κοιτάζει αν η σκυτάλη είναι καλά, τότε ο συνολικός χρόνος για να φτάσει η σκυτάλη από την μια άκρη στην άλλη θα αυξανόταν εκθετικά. Είναι δε ανούσιο να γίνεται ο ίδιος έλεγχος κάθε φορά, σε κάθε εναλλαγή της σκυτάλης. Αντιθέτως ο έλεγχος θα μπορούσε να γίνεται στα άκρα μία φορά για κάθε σκυτάλη (πακέτο δηλαδή), γεγονός που θα αποφόρτιζε το δίκτυο και θα βοηθούσε στην καλύτερη εξυπηρέτηση όλων (γιατί ας μην ξεχνάμε ότι το διαδίκτυο το χρησιμοποιούν πάρα πολλοί και δεν είναι η δουλειά του να ασχολείται με τα πακέτα του καθενός και σε τι κατάσταση βρίσκονται, απλά να τα μεταφέρει).

Για να το δούμε αυτό λίγο καλύτερα ας παρατηρήσουμε τι περίπου γίνεται σε έναν δρομολογητή. Ο δρομολογητής ουσιαστικά είναι κι αυτός τερματικό μόνο που συνδέεται με περισσότερα του ενός υπο-δίκτυα και αναλαμβάνει την προώθηση των πακέτων IP από το ένα υπο- δίκτυο στο άλλο, κάνοντας πρακτικά τον "τροχονόμο" στο δια-δίκτυο. Δουλεύει δηλαδή στο επίπεδο δια-δικτύου. Τα δεδομένα που φτάνουν στον δρομολογητή δεν περνάν στο επίπεδο μεταφοράς του, αφού δεν προορίζονται για κάποια εφαρμογή που τρέχει ο ίδιος, αντίθετα τα προωθεί κατευθείαν στο επίπεδο δια-δικτύου του επόμενου δρομολογητή ή στο επίπεδο δια-δικτύου του τερματικού. Φαίνεται δηλαδή ότι στο επίπεδο μεταφοράς των δύο τερματικών (το ανώτερο επίπεδο) η επικοινωνία μεταξύ τους γίνεται αυτόνομα, χωρίς να μεσολαβεί το επίπεδο μεταφοράς του δρομολογητή (και άρα να γίνεται error detection κλπ).

Παρόμοια "σκυταλοδρομία" γίνεται και στο φυσικό δίκτυο (όπως γινόταν πχ. με το προηγούμενο μοντέλο στα IMP -τα τότε switches του δικτύου) που πάλι δεν επηρεάζει την επικοινωνία σε επίπεδο δια-δικτύου, αφού το φυσικό δίκτυο δεν ξέρει τίποτα περί δια-δικτύου κλπ. Φαίνεται λοιπόν στο παρακάτω σχήμα ότι η "σκυταλοδρομία" που γίνεται στο φυσικό δίκτυο, με αυτή που γίνεται στο λογικό δίκτυο, καθώς και η επικοινωνία πάνω απ' αυτές στο επίπεδο μεταφοράς, γίνονται ταυτόχρονα και αυτόνομα (οριζόντια όπως είπαμε).



Πριν προχωρήσουμε σε περαιτέρω ανάλυση του TCP/IP ας δούμε πώς εξελίχθηκε η ιδέα των Layers μέχρι σήμερα, θα μας βοηθήσει περισσότερο να μπούμε στη λογική των Layers και να μπορούμε να εντάξουμε όσο το δυνατόν περισσότερα πρωτόκολλα σε αυτά.

2.5.Το OSI MODEL

Το παραπάνω μοντέλο (γνωστό και ως DoD), είχε αδυναμίες αφού αρκετά πρωτόκολλα δεν ήταν δυνατόν να καταταχθούν σύμφωνα με αυτό κι έτσι επικράτησε ένα ποιο πλήρες μοντέλο επικοινωνίας, το OSI (Open System Interconnection) Reference Model.

Τα πρωτόκολλα που τροποποιούν τα δεδομένα κρυπτογραφώντας τα, συμπιέζοντας τα κλπ (πχ. SSL) δεν ανήκουν ούτε στο επίπεδο 3 αφού δεν έχουν να κάνουν με την μεταφορά, ούτε στο επίπεδο 4 αφού δεν είναι κομμάτι της εφαρμογής. Έτσι έπρεπε να δημιουργηθεί ένα επίπεδο ανάμεσα στα 3 και 4 στο οποίο να μπαίνουν τα πρωτόκολλα αυτά, το επίπεδο αυτό ονομάστηκε **επίπεδο παρουσίασης (presentation layer)**.

Το RPC (Remote Procedure Call), το NetBIOS και γενικότερα τα πρωτόκολλα που διαχειρίζονται την επικοινωνία μεταξύ των εφαρμογών όπως έναρξη/λήξη μιας συνόδου, συγχρονισμό, flow control κλπ δεν ανήκουν ούτε στο επίπεδο 4 αφού είναι ουσιαστικά βοηθητικά πρωτόκολλα, ούτε στο επίπεδο 3 αφού δεν μεταφέρουν τίποτα. Επίσης πρωτόκολλα όπως τα RTP, RTCP κλπ που χρησιμοποιούνται για audio/video streaming και τα PPTP, L2TP που χρησιμοποιούνται για tunneling (βλ. VPN), χρησιμοποιούν το επίπεδο 3 για την μεταφορά τους και διαχειρίζονται συνόδους εφαρμογών (επίπεδο 4 δηλαδή). Επειδή τα πρωτόκολλα αυτά παρέχουν υπηρεσίες στα πρωτόκολλα του επιπέδου παρουσίασης που προαναφέραμε (το συμπιεσμένο mpeg video stream -επίπεδο παρουσίασης- μεταφέρεται πχ μέσω RTP) κατατάσσονται ανάμεσα στο επίπεδο παρουσίασης και το επίπεδο 3. Έτσι φτιάχτηκε το **επίπεδο διαχείρισης συνόδου (session layer)**.

Τέλος τα πρωτόκολλα που περιγράφουν το φυσικό δίκτυο, όπως το Ethernet, ουσιαστικά χωρίζονται σε τρία επίπεδα.

α) Αυτό που περιγράφει το φυσικό μέσο, τα χαρακτηριστικά του και τον τρόπο που μεταφέρονται δεδομένα μέσα από αυτό (όπως πχ. είναι το πρωτόκολλο 10base-T που περιγράφει την συνδεσμολογία και τον μηχανισμό μεταφοράς δεδομένων μέσω ενός καλωδίου twisted pair για 10Mbit σύνδεση).

β) Αυτό που περιγράφει τον μηχανισμό πρόσβασης στο φυσικό μέσο (όπως πχ. το CSMA/CD), το πώς μεταφέρονται οι PDU του φυσικού δικτύου (τις οποία λέμε πλαίσια -frames) και που αναλαμβάνει την διευθυνσιοδότηση στο φυσικό δίκτυο, γνωστό και ως επίπεδο MAC – Media Access Control.

γ) Αυτό που αναλαμβάνει να κάνει κάποιο βασικό error correction στο φυσικό δίκτυο και να κάνει encapsulate/decapsulate τα διάφορα πρωτόκολλα του επιπέδου 2, γνωστό και ως LLC -Logical Link Control.

Έτσι το α πλέον αποτελέι το κατώτερο επίπεδο και ονομάζεται **επίπεδο φυσικής διασύνδεσης (ή φυσικού μέσου)**, ενώ το β και το γ αποτελούν μαζί το **επίπεδο λογικής διασύνδεσης**.

Έτσι τελικά καταλήξαμε σε ένα μοντέλο 7 επιπέδων, δεν είναι τέλειο, έχει κι αυτό αδυναμίες αφού κάποια πρωτόκολλα κάνουν περισσότερες από μία λειτουργίες (όπως πχ. Το TCP που είναι ταυτόχρονα και στο επίπεδο μεταφοράς αλλά και στο επίπεδο διαχείρισης συνόδου) αλλά γενικώς μας έχει βολέψει. Το OSI λοιπόν έχει ως εξής (είναι αφισάκι-ready)...

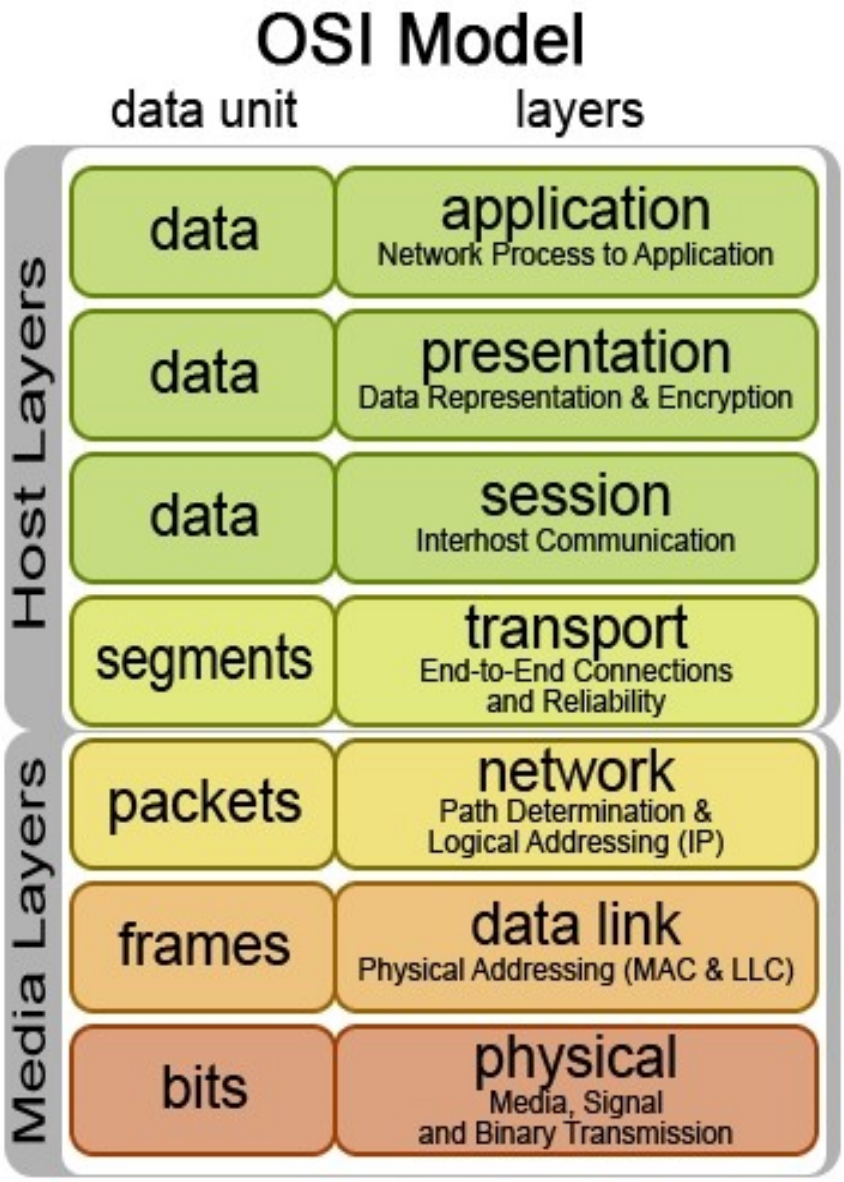
Επίπεδο εφαρμογής:
Εδώ επεξεργάζονται οι πληροφορίες που μεταφέρουμε.

Επίπεδο παρουσίασης:
Οι πληροφορίες τροποποιούνται ανάλογα με τις ανάγκες της εφαρμογής.

Επίπεδο συνόδου:
Συγχρονίζεται και συντονίζεται η μεταφορά των πληροφοριών ανάλογα με τις ανάγκες της εφαρμογής.

Επίπεδο μεταφοράς:
Οι πληροφορίες οργανώνονται σε τμήματα (segments) και μεταφέρονται ασφαλώς μέσω του δια-δικτύου.

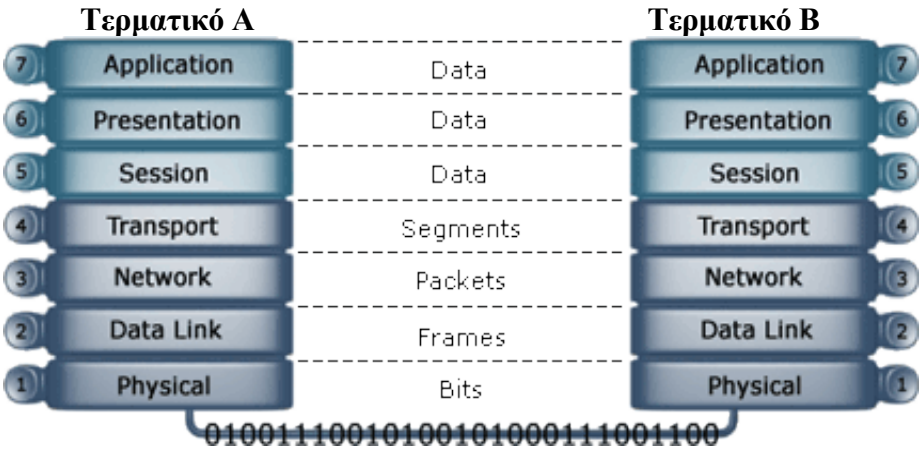
Επίπεδο δικτύου:
Τα τμήματα πληροφοριών οργανώνονται σε πακέτα (packets), αποκτούν διεύθυνση παραλήπτη κι αποστολέα στο δια-δίκτυο και προετοιμάζονται για την μεταφορά τους στο δια-δίκτυο.



Επίπεδο λογικής διασύνδεσης:
Τα πακέτα πληροφοριών προετοιμάζονται για την μεταφορά τους μέσω του φυσικού δικτύου, οργανώνονται σε πλαίσια (frames) τα οποία αποκτούν διεύθυνση παραλήπτη κι αποστολέα στο φυσικό δίκτυο και αποστέλλονται.

Επίπεδο φυσικής διασύνδεσης:
Τα πλαίσια (ή πακέτα δεδομένων αν προτιμάτε) διαδίδονται ως bits στο φυσικό μέσο, τα πρωτόκολλα σε αυτό το επίπεδο προβλέπουν την μεταφορά των δυαδικών πληροφοριών απ' το ένα άκρο του φυσικού δικτύου στο άλλο.

Κι εδώ όταν σκεφτόμαστε το τερματικό, τα επίπεδα επικοινωνούν μεταξύ τους κατακόρυφα, αφού το ένα μεταφέρει δεδομένα στο άλλο, ενώ όταν σκεφτόμαστε το δίκτυο τα επίπεδα επικοινωνούν μεταξύ τους οριζόντια κι αυτόνομα Είναι προφανές ότι αν μεταξύ δύο τερματικών σε κάποιο επίπεδο χρησιμοποιείται διαφορετικό πρωτόκολλο επικοινωνίας δεν είναι δυνατή η επικοινωνία των τερματικών.



Τώρα λοιπόν που είμαστε κομπλέ με τα επίπεδα και τις βασικές αρχές που τα διέπουν ας δούμε επιτέλους από τι αποτελείται το TCP/IP και πώς δουλεύει. Θα ξεκινήσουμε απ' το κατώτερο επίπεδο του TCP/IP που όπως είπαμε είναι το IP.

3. ΤΟ IP (Internet Protocol)

Το IP είναι το ταχυδρομείο για το δια-δίκτυο και όπως και τα δικά μας ΕΛΤΑ έχει κι αυτό αδυναμίες. Όπως τα ΕΛΤΑ καθυστερούν να στείλουν ένα γράμμα, σου στέλνουν τον λογαριασμό του ΟΤΕ αφού έχει λήξει η προθεσμία, σου στέλνουν το γράμμα τσαλακωμένο, χάνουν γράμματα στην πορεία κλπ, έτσι και το IP έχει τέτοιες αδυναμίες μιας και όπως είπαμε στο επίπεδο δια-δικτύου δεν γίνεται έλεγχος σφαλμάτων. Το IP λοιπόν, βασίζεται στην λογική της "βέλτιστης προσπάθειας"(best effort) που στην πράξη σημαίνει "στέλνουμε κάτι κι ευχόμαστε να φτάσει στον παραλήπτη του", ποιο συγκεκριμένα:

α) Δεν εγγυάται την επιτυχή αποστολή των δεδομένων, δηλαδή μπορεί να χαθούν κάποια πακέτα στην πορεία.

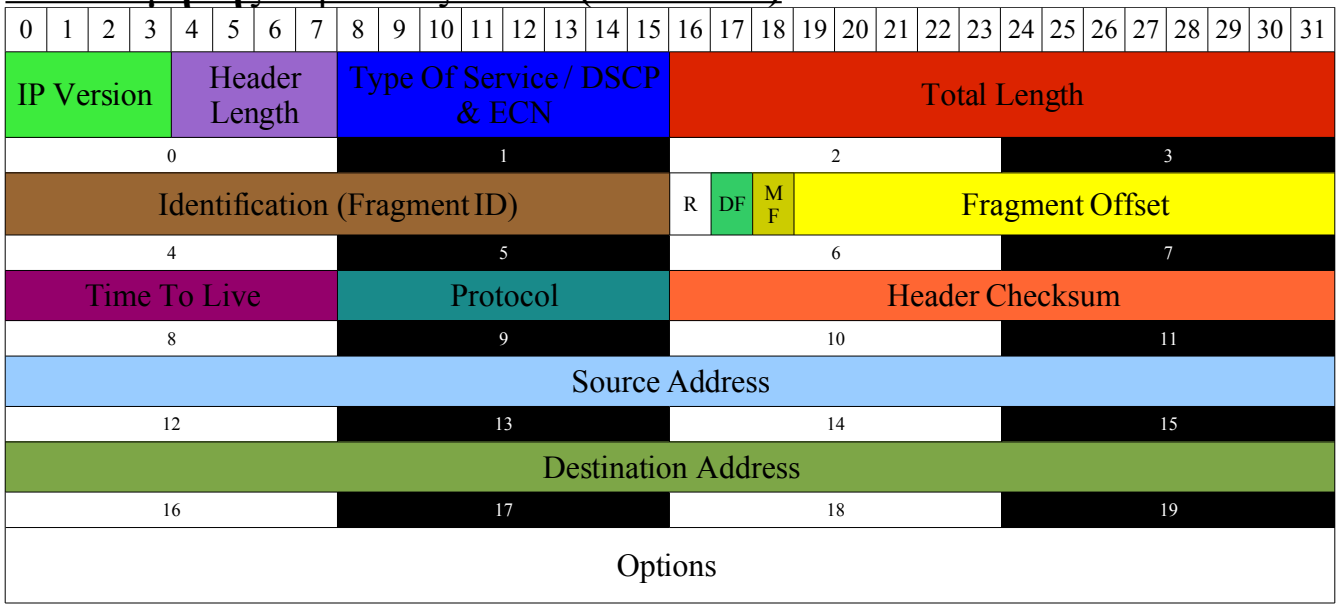
β) Δεν εγγυάται την σωστή αποστολή των δεδομένων, δηλαδή κάποια πακέτα μπορεί να έρθουν με λάθος σειρά, να έρθουν διπλά, να κάνουν τον γύρο του κόσμου κλπ

γ) Δεν εγγυάται την ασφαλή αποστολή των δεδομένων, δηλαδή μπορεί κάποια πακέτα να έρθουν κατεστραμμένα.

Οι βασικές δουλειές του IP είναι 3, ο κερματισμός (το "πακετάρισμα" που λέγαμε), η διευθυνσιοδότηση και η δρομολόγηση των πακέτων στο δια-δίκτυο. Για όλες αυτές τις διαδικασίες χρησιμοποούνται διάφορα πεδία στην κεφαλίδα της PDU του IP (το πακέτο όπως είπαμε) που μας δίνουν τις απαραίτητες πληροφορίες, οπότε καλύτερα να περάσουμε κατευθείαν στην περιγραφή και ανάλυσή της.

Επειδή από εδώ και κάτω θα χρησιμοποιούνται παραπομπές σε "RFC" καλό είναι να δείτε τι είναι τα RFC και τι περιέχουν (http://www.rfc.net/). Το RFC που περιγράφει το TCP/IP είναι το no791.

3.1.Η δομή της κεφαλίδας του IP (IP Header)



Είπαμε προηγουμένως όταν μιλάγαμε για το encapsulation ότι κάθε πρωτόκολλο χαρακτηρίζεται από την κεφαλίδα του, ας παρατηρήσουμε λοιπόν την κεφαλίδα του IP για να δούμε κάποιες γενικές έννοιες. Παρατηρείτε ότι η κεφαλίδα περιέχει κάποια πεδία, το κάθε πεδίο περιέχει μια πληροφορία για κάποια συγκεκριμένη λειτουργία του πρωτοκόλλου, επίσης η κεφαλίδα χωρίζεται σεκομμάτια των 8bit που λέγονται Octets (τα αριθμημένα μαυράσπρα τμήματα). Κάθε octet δίνει την δυνατότητα αναπαράστασης 256 αριθμών στο δεκαδικό σύστημα αρίθμησης. Προσέξτε ότι δεν λέμε bytes γιατί για κάποιους υπολογιστές το 1 byte δεν είναι 8 bit (π.χ. στους PDP-10), θυμίζω ότι το byte είναι το μικρότερο ποσό μνήμης που μπορεί να χωρέσει ένα στοιχειώδες ποσό πληροφορίας (το μικρότερο ποσό μνήμης που μπορεί να δεσμευτεί δηλαδή από τον υπολογιστή), το octet λοιπόν είναι κάτι standard που δεν έχει να κάνει με την αρχιτεκτονική του υπολογιστή, άρα μπορεί να χρησιμοποιηθεί ασφαλώς στον ορισμό του πρωτοκόλλου. Μην ξεχνάτε ότι το TCP/IP ήρθε για να συνδέσει συστήματα διαφορετικών αρχιτεκτονικών κι αυτό φαίνεται σε κάθε του κομμάτι.

Ας δούμε λοιπόν τι πληροφορία μας παρέχει το κάθε πεδίο και πώς αυτή αξιοποιείται στην συνέχεια...

Τα πρώτα 32 bit μας δίνουν γενικές πληροφορίες σχετικά με το μέγεθος του πακέτου που στέλνουμε, και πως αυτό θέλουμε να προωθηθεί στο δίκτυο. Ποιο συγκεκριμένα έχουμε τα πεδία:

IP Version (4 bits): Στο πεδίο αυτό περιέχεται ένας ακέραιος αριθμός από 0 ως 15 (τόσοι συνδυασμοί χωράνε σε 4bit όπως θα δούμε παρακάτω) που μας λέει την έκδοση του πρωτοκόλλου, είναι εκεί για λόγους συμβατότητας και πλέονέχει την τιμή 4 (IPv4), στο μέλλον όταν το IPv6 επικρατήσει θα έχει την τιμή 6 κ.ο.κ.

Header Length (4bits): Στο πεδίο αυτό περιέχεται ένας ακέραιος αριθμός από 0 ως 15 που περιγράφει το μέγεθος της κεφαλίδας σε ομάδες των 32bit. Για λόγους ευκολίας κι επειδή τα μεγαλύτερα πεδία της κεφαλίδας έχουν μέγεθος 32bit (η διεύθυνση αποστολέα/παραλήπτη που θα δούμε παρακάτω) την χωρίζουμε με τον τρόπο που φαίνεται στο σχήμα. Με την ίδια λογική δε μετράμε και το μέγεθός της σε ομάδες των 32bit (ή 32bit words όπως λέμε) όπου χωρίς το βοηθητικό πεδίο στο τέλος έχουμε συνολικά 5 τέτοιες ομάδες για μια λειτουργική κεφαλίδα.Η τιμή 5 αντιστοιχεί σε 5 x 32 = 160 /8 = 20 octets (ή 20 bytes αν προτιμάτε) που είναι και το σύνηθες μέγεθος της κεφαλίδας, το μέγιστο όριο για το μέγεθος της κεφαλίδας είναι 60octets, αφού η μέγιστη τιμή που μπορεί να πάρει το πεδίο είναι 15 (15x32 =480 /8 = 60).

3.1.1.To Type Of Service και η εξέλιξή του

Το επόμενο πεδίο **Type Of Service (8bits)** χρησιμοποιείται απ' το πρωτόκολλο για να ορίζει την προτεραιότητα που θα έχει το κάθε πακέτο, η πληροφορία αυτή χρησιμοποιείται τόσο απ' τα τερματικά (για να ανταποκριθούν ποιο γρήγορα πχ. σε κάποιο αίτημα) όσο κι απ' τους δρομολογητές (για να δρομολογήσουν γρηγορότερα το πακέτο στο δια-δίκτυο). Το όνομά του που σημαίνει "τύπος υπηρεσίας" δεν είναι τυχαίο αφού η προτεραιότητα ενός πακέτου στο δίκτυο εξαρτάται απ' το πόσο ευαίσθητη είναι η υπηρεσία που χρησιμοποιεί το πακέτο αυτό στην γρήγορη απόκριση του δικτύου. Για παράδειγμα όταν ένα πακέτο χρησιμοποιείται για την αναφορά κάποιου σφάλματος ή γενικότερα την ειδοποίηση για κάποια μεταβολή στο δίκτυο κλπ πρέπει να έχει μεγαλύτερη προτεραιότητα από την κανονική κίνηση του δικτύου.

Αυτό το πεδίο έχει ιστορία, ο λόγος ύπαρξής του πλέον είναι να παρέχει πληροφορίες χρήσιμες για την σωστή διαχείριση της κίνησης ή καλύτερα για την καλύτερη ποιότητα υπηρεσιών (QoS - Quality of Service), κατά καιρούς η χρήση των 8 αυτών bits έχει αλλάξει, αρχικά μας έλεγαν 2 πράγματα...

Τα πρώτα 3 bits χρησιμοποιούνται για να ορίσουν την προτεραιότητα του πακέτου στο δίκτυο, ενώ τα επόμενα 3 είναι περαιτέρω επιλογές για την διαχείριση του traffic.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Προτεραιότητα πακέτου			**Καθυστέρηση (delay)**	**Όγκος (throughput)**	**Απόδοση (reliability)**	**Κενό**	**Κενό**

Οι δυνατές τιμές για την προτεραιότητα σύμφωνα με το RFC791 είναι:

```
111 - Network Control
110 - Internetwork Control
101 - CRITICAL/ECP
100 - Flash Override
011 - Flash
010 - Immediate
001 - Priority
000 - Routine
```

Από κάτω προς τα πάνω λοιπόν, έχουμε κίνηση ρουτίνας (ή αλλιώς routine) που δεν χρειάζεται ιδιαίτερη μεταχείριση, κίνηση αυξημένης προτεραιότητας (priority), κίνηση άμεσης προτεραιότητας (Immediate) κλπ, τα δύο τελευταία είναι για κρίσιμη κίνηση για την λειτουργία του δικτύου (Network και Internetwork Control), το πρώτο είναι για την κίνηση μεταξύ των router (OSPF π.χ.) ενώ το δεύτερο μπορεί να χρησιμοποιηθεί μόνο για traffic του τοπικού δικτύου (όχι για δια-δικτυακό traffic) και συνήθως είναι για το keep alive του routing protocol.

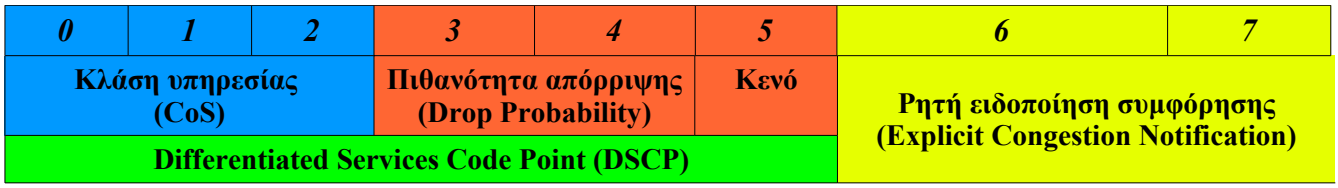
Τα επόμενα 3 πεδία είναι 1bit το καθένα και σημαίνουν τα εξής...

Καθυστέρηση: Βελτιστοποίηση για κίνηση χαμηλής καθυστέρησης (low delay)
Όγκος: Βελτιστοποίηση για κίνηση μεγάλου όγκου δεδομένων (high throughput)
Απόδοση: Βελτιστοποίηση για κίνηση υψηλής απόδοσης (high reliability)

Όταν τα πεδία αυτά φέρουν την τιμή 1 είναι ενεργά, αν όχι είναι ανενεργά και δενυπάρχει βελτιστοποίηση. Οι τρεις αυτές παράμετροι της κίνησης ουσιαστικά αναιρούν η μια την άλλη και γι' αυτό πρέπει να επιλέξουμε τι απ' τα 3 προτιμάμε. Δηλαδή δεν μπορούμε για παράδειγμα να έχουμε ταυτόχρονα και χαμηλή καθυστέρηση και μεγάλο όγκο και υψηλή απόδοση (θα είχαμε το τέλειο δίκτυο τότε), οπότε συνήθως επιλέγουμε μόνο ένα ενώ το πρωτόκολλο μας επιτρέπει μέχρι 2 επιλογές για πολύ

extreme καταστάσεις.

Πλέον ο μηχανισμός αυτός έχει αντικατασταθεί και τα 6 αυτά bits (0-5) χρησιμοποιούνται για το Differentiated Services Code Point (DSCP), όπου πρόκειται για έναν ποιο πλήρη μηχανισμό ορισμού προτεραιότητας και χαρακτηριστικών της κίνησης (βλ. RFC2474 / 2475) ο οποίος βοηθάει αρκετά στο QoS, εισάγει δε και την έννοια της κλάσης υπηρεσίας (CoS– Class of Service).



Τα πρώτα 3 bits χρησιμοποιούνται παρόμοια με το κλασικό ToS για να δηλώσουν την προτεραιότητα του πακέτου, μόνο που οι περισσότερες τιμές πλέον δεν ορίζουν ακριβώς προτεραιότητα αλλά κλάση υπηρεσίας. Ποιο συγκεκριμένα:

```
111 - Network Control
110 - Internetwork Control
101 - Express Forwarding (EF)
100 - Class 4
011 - Class 3
010 - Class 2
001 - Class 1
000 - Routine (ή καλύτερα Best Effort -BE)
```

Παρατηρείτε ότι με αυτό τον τρόπο παρέχεται συμβατότητα με το προηγούμενο πρότυπο, αφού τουλάχιστον για την κίνηση επείγουσας προτεραιότητας, η σηματοδοσία είναι η ίδια.

Τα άλλα 3 bit χρησιμοποιούνται για να ορίσουν την πιθανότητα απόρριψης του πακέτου από μηχανισμούς όπως ο RED (Random Early Detection – RFC 2309) όπου χρησιμοποιούν την απόρριψη πακέτων σε περιπτώσεις συμφόρησης του δικτύου (Congestion), τα 2 πρώτα bit χρησιμοποιούνται για να δώσουν μια τιμή από 0 (100% πιθανότητα απόρριψης σε περίπτωση συμφόρησης) ως 3 (0% πιθανότητα απόρριψης) και το τελευταίο μένει μηδέν.

ECN: Τα 2 τελευταία bits του πεδίου ToS που μένουν αχρησιμοποίητα και από το DSCP πλέον χρησιμοποιούνται για την ρητή ειδοποίηση συμφόρησης ή Explicit Congestion Notification (ECN), πρόκειται για έναν άλλο μηχανισμό που σε αντίθεση με τον RED χρησιμοποιείται κατά παραγγελία των τερματικών (άρα πρέπει και τα δύο τερματικά που επικοινωνούν μεταξύ τους να το υποστηρίζουν) και λέει στους routers ότι εφόσον διαπιστώσουν συμφόρηση στο δίκτυο να ειδοποιήσουν τα τερματικά για να ρυθμίσουν την μεταξύ τους επικοινωνία αναλόγως (αντί να κάνουν οι routers drop τα πακέτα). Δυστυχώς κάποια βλαμμένα firewall εταιριών που επιμένουν να μην τηρούν ανοιχτά πρότυπα, κόβουν (κάνουν drop) σε κάποιες περιπτώσεις τα πακέτα που χρησιμοποιούν το ECN αν και σε σωστά στημένα δίκτυα δεν έχετε να φοβηθείτε τίποτα.

3.1.2.Κερματισμός των IP πακέτων ή αλλιώς IP Fragmentation

Το "πακετάρισμα" των πληροφοριών από τα ανώτερα επίπεδα δεν είναι και τόσο απλή διαδικασία, τα τμήματα που έρχονται απ' το επίπεδο μεταφοράς είναι ολόκληρα κομμάτια πληροφοριών, αρκετά μεγάλα σε μέγεθος για να μπορούν να μεταφερθούν αυτούσια μέσο του φυσικού δικτύου. Για να πάρετε μια γεύση για το πόσο μεγάλο μπορεί να είναι ένα τέτοιο τμήμα και άρα ένα ολόκληρο πακέτο IP δείτε το επόμενο πεδίο της κεφαλίδας που μας παρέχει αυτή ακριβώς την πληροφορία.

Total Length (16bits): Σε αυτό το πεδίο περιέχεται ένας ακέραιος αριθμός από 0 ως 65535 που αναπαριστά το συνολικό μέγεθος του πακέτου IP (κεφαλίδα + δεδομένα από το επίπεδο μεταφοράς) μετρημένο σε octets. Η μικρότερη λογική τιμή που μπορεί να πάρει αυτό το πεδίο είναι 20 (20octets η κεφαλίδα + 0 octets δεδομένα) και η μεγαλύτερη 65.535octets ή αν προτιμάτε περίπου 64KBytes (η μέγιστη τιμή που μπορούν να αναπαραστήσουν τα 16bits που είναι το πεδίο). Το κάθε τερματικό πρέπει να μπορεί να διαχειριστεί πακέτα μεγέθους τουλάχιστον 576octets, με τη λογική ότι 512octets είναι τα προς μεταφορά δεδομένα και 64octets συνολικά η κεφαλίδα τους(20-60octets για την κεφαλίδα IP και ότι περισσεύει μένει ως πρόβλεψη για κεφαλίδες πρωτοκόλλων ανώτερων επιπέδων). Ωστόσο το μέγεθος των πακέτων είναι συνήθως κατά πολύ μεγαλύτερο από 576octets, ειδικά στις μοντέρνες υλοποιήσεις, μάλιστα το RFC του TCP/IP συνιστά η όποια υλοποίηση του πρωτοκόλλου να δουλεύει με πακέτα μεγαλύτερα από 576octets και τα 576octets να είναι ως το κατώτατο όριο.

Μιλώντας για το μέγεθος του κάθε IP πακέτου ας σκεφτούμε λίγο τι γίνεται σε ένα packet switched δίκτυο που τα πακέτα που μπορεί να στείλει είναι μικρότερου μεγέθους από το πακέτο IP που θέλουμε να στείλουμε. Αυτό δεν είναι κάτι σπάνιο, το Ethernet πχ. συνήθως χρησιμοποιεί μέχρι 1500octets (το MTU – Maximum Transmission Unit) για λόγους συμβατότητας (το Ethernet 10baseT χρησιμοποιούσε 1500octets max οπότε τα επόμενα πρότυπα χρησιμοποιούν το ίδιο για να είναι συμβατά μεταξύ τους), άρα πώς ένα πακέτο IP των 64KB μπορεί να μεταφερθεί μέσα απ' το Ethernet ?

Αν λάβετε υπόψη σας την εποχή που φτιάχτηκε το IP καταλαβαίνετε ότι υπήρχε πρόβλεψη για τέτοιου είδους καταστάσεις, τα τότε packet switching δίκτυα προφανώς και δεν υποστήριζαν την μεταφορά 64KB ανά πακέτο (ακόμα και τα σημερινά δίκτυα δεν το υποστηρίζουν), έτσι λοιπόν φτιάχτηκε στο IP ο μηχανισμός του κερματισμού ή αλλιώς της συναρμολόγησης/ αποσυναρμολόγησης του κάθε πακέτου. Το κάθε πακέτο σπάει σε κομμάτια ή αλλιώς fragments (που όμως μπορούν να χωρέσουν τουλάχιστον την κεφαλίδα IP και λίγα δεδομένα, γι αυτό είπαμε παραπάνω ότι το minimum που απαιτείται απ' το πρωτόκολλο είναι 576octets), τα κομμάτια αυτά μεταφέρονται μέσω του δικτύου και όταν φτάνουν στον παραλήπτη τους συναρμολογούνται πάλι στο επίπεδο δια-δικτύου και το πακέτο προωθείται αυτούσιο στα ανώτερα επίπεδα.

Τα επόμενα 32bit της κεφαλίδας IP αφορούν επίσης αυτό τον μηχανισμό, ποιο συγκεκριμένα έχουμε τα εξής πεδία:

Identification (16bits): Μια χαρακτηριστική τιμή για το κάθε κομμάτι που μεταφέρουμε, έτσι ώστε να μπορούμε να τα διαχωρίσουμε μεταξύ τους, σε 16bit χωράνε 65.535 διαφορετικές τιμές, άρα αυτός είναι θεωρητικά και ο μέγιστος αριθμός των κομματιών στα οποία μπορούμε να σπάσουμε ένα πακέτο. Λέω θεωρητικά γιατί αν ένα πακέτο των 64KB σπάσει σε 65.535 κομμάτια, το κάθε κομμάτι θα είναι μόλις 1octet οπότε δεν θα έχει κανένα νόημα. Στην ουσία δηλαδή η χαρακτηριστική τιμή που λέμε μπορεί να είναι τυχαία.

(το επόμενο bit είναι reserved και πρέπει να είναι 0)

DF (Don't Fragment) flag (1bit): Μας λέει αν το πακέτο που μεταφέρουμε επιτρέπεται να κοπεί σε κομμάτια κατά την μεταφορά του,όταν το bit είναι 0 τότε επιτρέπεται, όταν είναι 1 δεν επιτρέπεται και το πακέτο πρέπει να μεταφερθεί αυτούσιο. Αν το πακέτο δεν μπορεί να μεταφερθεί αυτούσιο από κάποιον router, τότε γίνεται drop.

MF (More fragments) flag (1bit): Αν το πακέτο δεν είναι κομμένο σε κομμάτια ή αν αυτό που πήραμε είναι το τελευταίο κομμάτι του πακέτου, αυτό το bit είναι 0, αλλιώς αν ακολουθούν κι άλλα κομμάτια είναι 1.

Fragment offset (13bits): Αυτό το πεδίο περιέχει έναν ακέραιο από 0 ως 8191 (τόσοι συνδυασμοί χωράνε σε 13bit όπως θα δούμε παρακάτω) που προσδιορίζει την θέση του κομματιού μέσα στο πακέτο (αν είναι το πρώτο κομμάτι, το δεύτερο κλπ), κάθε μονάδα σε αυτό το πεδίο αντιστοιχεί σε 8octets (64bits) με το πρώτο τμήμα να έχει την τιμή 0.

The ping of death

Σε περίπτωση που δεν το προσέξατε, κάτι δεν πάει καλά με το Fragment offset, το μέγιστο μέγεθος των δεδομένων που επιτρέπει αυτός ο μηχανισμός είναι 8191* 8octets, αν σε αυτά τα δεδομένα προσθέσετε και την κεφαλίδα IP (αφού το πακέτο που θα προκύψει στο τέλος κατά τη συναρμολόγηση θα έχει κανονικά κεφαλίδα IP, για να μπορεί να "φύγει" προς τα πάνω σαν κανονικό IP πακέτο) έχουμε από 8211octets (8191 + 20) μέχρι και 8251octets (8191 + 60). Αυτό επιτρέπει το συνολικό μέγεθος του πακέτου να φτάσει μέχρι και τα 66.008octets δεδομένων, δηλαδή περισσότερα απ' ότι προβλέπει το IP (βλ. Total Length).

Αρκεί ένα bit επιπλέον για να γίνει η "ζημιά" αφού αν το πεδίο ήταν 12bit ο μέγιστος αριθμός octets που θα μπορούσε να αναπαραστήσει θα ήταν 4095 και το μέγιστο μέγεθος του τελικού πακέτου θα ήταν 33.240bit που όμως είναι πολύ μικρότερο απ' το μέγιστο μέγεθος που επιτρέπει το IP, άρα δεν έχει νόημα, αφού ουσιαστικά αν θέλαμε να στείλουμε ένα πακέτο μεγαλύτερο των 33.240bit θα έπρεπε να το στείλουμε αυτούσιο (δεν θα δούλευε ο μηχανισμός). Τα 13bit λοιπόν μοιραία είναι μονόδρομος και πρέπει να τα ανεχτούμε.

Έχουμε έτσι έναν μηχανισμό που μας επιτρέπει να "παρανομήσουμε" και να δημιουργήσουμε (μετά τη συναρμολόγηση) στον παραλήπτη ένα πακέτο αρκετά μεγαλύτερο απ' το φυσιολογικό. Σε κάποια λειτουργικά συστήματα αυτό δεν το είχαν προβλέψει παρόλο που το RFC το αναφέρει κι έτσι με το που έβλεπαν τέτοια πακέτα κολλούσαν ή και crashάραν, το προγραμματάκι που χρησιμοποιήθηκε ως proof – of – concept ήταν το γνωστό μας ping (θα δούμε παρακάτω τι ακριβώς κάνει το ping), το οποίο με λίγο πείραγμα έστελνε στον παραλήπτη τέτοια περίεργα πακέτα. Αποτέλεσμα ήταν αμέτρητοι υπολογιστές να δεχτούν επιθέσεις και να παραλύσουν ολόκληρα δίκτυα (περισσότερα για το ping-o-death μπορείτε να βρείτε στο http://www.insecure.org/sploits/ping-o-death.html).

Για να σατιρίσει το γεγονός αυτό, την πρωταπριλιά του 2003 ο S. Bellovin έγραψε το RFC 3514 και πρότεινε το επιπλέον αυτό bit να ονομαστεί "Evil bit" και να ορίζει αν το πακέτο που το φέρει έχει κακούς σκοπούς ή όχι, έτσι θα λυνόταν μια για πάντα το θέμα της ασφάλειας στο δια-δίκτυο ;-).

3.1.3.Time To Live

Σκεφτείτε να στείλετε ένα πακέτο σε κάποιο τερματικό που είναι εκτός δικτύου ή έχει πέσει ή γενικά σε μια άκυρη διεύθυνση, το πακέτο θα περιπλανιέται στο δίκτυο από δρομολογητή σε δρομολογητή επ' άπειρον χωρίς να καταλήγει κάπου,μέχρι κάποιος να "κατεβάσει" το δίκτυο. Σε μια τέτοια κατάσταση, με πολλά τέτοια τερματικά, το δίκτυο μέσα σε μικρό χρονικό διάστημα θα είχε"μπουκώσει" και θα αδυνατούσε να ανταποκριθεί σε οποιαδήποτε λειτουργία. Έτσι χρειάζεται εξ' αρχής να ορίσουμε τον χρόνο ζωής του πακέτου για να προλάβουμε τα χειρότερα . Το επόμενο πεδίο είναι γι' αυτή τη δουλειά.

Time To Live -TTL (8bits): Σε αυτό το πεδίο περιέχεται ένας ακέραιος αριθμός από 0 ως 255 που μας λέει μέσα από πόσους δρομολογητές θα περάσει το πακέτο πριν αυτοκαταστραφεί (δηλαδή σταματήσει η διάδοσή του στο δια-δίκτυο). Ο αριθμός αυτός ξεκινάει από μια τιμή (συνήθως 64) και σε κάθε αναμετάδοση (hop) του πακέτου από κάποιο δρομολογητή η τιμή αυτή μειώνεται κατά 1, όταν το πακέτο φτάσει σε κάποιο δρομολογητή κι έχει την τιμή 0, ο δρομολογητής απορρίπτει το πακέτο, βάζοντας τέλος στο ταξίδι του και ενημερώνοντας μας (μέσω του πρωτοκόλλου ICMP που θα δούμε αργότερα) ότι ο χρόνος ζωής του πακέτου μας έληξε. Σε περίπτωση που θέλουμε να δοκιμάσουμε ξανά να φτάσουμε το απομακρυσμένο τερματικό, μπορούμε να αυξήσουμε το TTL για να επιτρέψουμε στο πακέτο να ταξιδέψει μακρύτερα.

Traceroute, μια εφαρμογή του TTL

Όλοι γνωρίζουμε το traceroute, για όσους δεν το γνωρίζουν αρκεί να πω ότι είναι το πρόγραμμα που μας δείχνει την διαδρομή που ακολουθούν τα πακέτα μας για να φτάσουν έναν συγκεκριμένο προορισμό. Το traceroute λοιπόν δουλεύει ως εξής:

Στέλνει ένα πακέτο στον προορισμό που θέλουμε με TTL 0, το πακέτο φτάνει στον πρώτο δρομολογητή, σταματάει εκεί και ο δρομολογητής μας ενημερώνει ότι ο χρόνος ζωής του πακέτου έληξε, έτσι ξέρουμε ποιος είναι ο πρώτος δρομολογητής. Μετά στέλνει ένα πακέτο με TTL 1 που λήγει στον δεύτερο δρομολογητή που με την σειρά του μας ενημερώνει και τον μαθαίνουμε, μετά στέλνει με TTL 2 για τον τρίτο δρομολογητή κ.ο.κ μέχρι να φτάσει στόχο του (ή να φτάσει την μέγιστη τιμή TTL, μετά από 255 αναμεταδόσεις δηλαδή).

Το traceroute είναι πολύ χρήσιμο εργαλείο γιατί μας δίνει πολλές πληροφορίες για το δίκτυο και την δρομολόγηση σε αυτό και καλό θα ήταν να το δουλέψετε αρκετά.

Κρύψτε τον δρομολογητή/firewall σας απ' το traceroute

Αρκετές φορές είναι χρήσιμο να κρύβουμε πληροφορίες για την δομή του δικτύου μας, συνήθως για λόγους ασφαλείας. Έτσι ένα πολύ χρήσιμο "κόλπο" που υπάρχει για να μας βοηθήσει σε αυτό είναι η αύξηση του TTL κατά 1 σε κάποιο σημείο ελέγχου στο δίκτυό μας (ένας δρομολογητής που παίζει τον ρόλο τοίχους προστασίας -firewall- για παράδειγμα, και προστατεύει το εσωτερικό μας δίκτυο από επιθέσεις) ώστε να μπερδεύει το traceroute και να μην φαίνεται ως σημείο στην διαδρομή των πακέτων μας. Αυξάνοντας το TTL κατά 1 το πακέτο θα λήξει στον αμέσως επόμενο δρομολογητή (ή τερματικό) και όχι στο σημείο ελέγχου μας το οποίο θα παραμείνει αόρατο στο traceroute.

**Στο Linux κάτι τέτοιο γίνεται πολύ εύκολα με το TTL target των iptables*

Αξίζει σε αυτό το σημείο βέβαια να αναφέρω ότι τέτοιου είδους έλεγχοι και πειράγματα γίνονται μόνο σε όρια κάποιου τοπικού δικτύου ή τέλος πάντων δικτύου που δεν έχει άλλη έξοδο στο διαδίκτυο, που δεν θα περάσουν δηλαδή μέσα από αυτό πακέτα άλλων δικτύων, γιατί τότε θα επηρεαζόταν το διαδίκτυο συνολικά. Αν σας μπέρδεψα απλά κρατήστε αυτό: ΔΕΝ κάνουμε τέτοια πειράγματα σε δρομολογητές της ραχοκοκαλιάς (Backbone) του δικτύου αλλά μόνο σε ακραίους δρομολογητές (πχ. τον DSL router μας).

Κλείνουμε με την κεφαλίδα IP με τα τελευταία 2 πεδία...

Protocol (8bits): Σε αυτό το πεδίο περιέχεται ένας ακέραιος αριθμός από 0 ως 255 που αντιστοιχεί στο πρωτόκολλο το οποίο κάνουμε encapsulate στο IP πακέτο μας (δηλαδή σε κάποιο πρωτόκολλο του επιπέδου μεταφοράς). Ο αριθμός αυτός είναι χαρακτηριστικός για το κάθε πρωτόκολλο και γίνεται από την IANA (Internet Assigned Numbers Authority - www.iana.org) την επίσημη αρχή που διατηρεί το αρχείο για τους διαφόρους αριθμούς που χρησιμοποιούνται στο διαδίκτυο, από αριθμούς πρωτοκόλλων μέχρι διευθύνσεις, ονόματα κλπ Το αρχείο με τις τελευταίες καταχωρίσεις των αριθμών πρωτοκόλλων μπορείτε να το βρείτε στη διεύθυνση http://www.iana.org/assignments/protocol-numbers .

Header Checksum (16bit): Αυτό το πεδίο μας παρέχει τον μηχανισμό ελέγχου της ακεραιότητας της κεφαλίδας, προσοχή ΟΧΙ των δεδομένων που την ακολουθούν (είπαμε ότι ο έλεγχος για τα δεδομένα γίνεται σε ανώτερα επίπεδα). Ο σκοπός είναι να έχουμε έναν απλό και γρήγορο (πάνω απ' όλα "ελαφρύ") τρόπο να βρούμε αν η κεφαλίδα που στάλθηκε απ' τον προηγούμενο έφτασε σωστά σε εμάς. Έτσι η κεφαλίδα "κόβεται" σε κομμάτια των 16bit και χρησιμοποιείται η μέθοδος της συμπλήρωσης 1's complement) -που πρακτικά πρόκειται για μια απλή μέθοδος αναπαράστασης αρνητικών αριθμών στο δυαδικό σύστημα κάνοντας NOT σε κάθε bit του αντίστοιχου θετικού αριθμού- για να αθροιστούν μεταξύ τους. Το άθροισμα που προκύπτει υπόκειται άλλο ένα NOT σε κάθε bit του και μπαίνει σε αυτό το πεδίο. Καταλαβαίνετε ότι το άθροισμα των 16bit κομματιών θα είναι 16bit αφού όλο το παιχνίδι γίνεται στο δυαδικό σύστημα και οι τιμές αλληλοαναιρούνται. Στον υπολογισμό του αθροίσματος αυτού, το πεδίο θεωρείται ότι έχει την τιμή μηδέν για να μην επηρεάζει το αποτέλεσμα. (Για να δείτε μια υλοποίηση του μηχανισμού σε C ρίξτε μια ματιά εδώ -> http://www.cs.utk.edu/~cs594np/unp/checksum.html ή εδώ -> www.netfor2.com/ipsum.htm)

Options (40 octets max): Αυτό το πεδίο είναι προαιρετικό και χρησιμοποιείται για διάφορες πληροφορίες που ξεφεύγουν απ' τους σκοπούς αυτού του Tutorial. Έχετε μόνο κατά νου ότι το μέγεθός του μπορεί να φτάσει μέχρι τα 40 octets, ενώ οι πληροφορίες αυτές χωρίζονται μεταξύ τους με ένα 0.

3.2.Διευθυνσιοδότηση

Τα επόμενα 2 πεδία των 32bit το καθένα μας παρέχουν τον βασικότερο ίσως μηχανισμό του IP που δεν είναι άλλος απ' την διευθυνσιοδότηση. Το πρώτο πρόκειται για την διεύθυνση αποστολέα ή αλλιώς source address και το δεύτερο την διεύθυνση παραλήπτη ή αλλιώς destination address.

3.2.1.Η μορφή μιας IP διεύθυνσης

Ένας **ακέραιος δεκαδικός αριθμός χωρίς πρόσημο** αναπαριστάται στο δυαδικό σύστημα με συνδυασμούς δυνάμεων του 2 ως εξής:

Κάθε bit αντιστοιχίζεται σε μια δύναμη του 2 και ο αριθμός εκφράζεται ως άθροισμα δυνάμεων του 2, για παράδειγμα ο αριθμός 51 (το nodeID μου :-)) γράφεται στο δυαδικό ως εξής...

2^0	2^1	2^2	2^3	2^4	2^5	
1	2	4	8	16	32	...>
1	1	0	0	1	1	
1+	2+	0+	0+	16+	32	= 51

Δηλαδή γράφεται 110011

Βλέπουμε ότι για να αναπαραστήσουμε το 51 μας αρκούν 6 bits, με 6 bits ο μεγαλύτερος αριθμός που μπορούμε να φτάσουμε θέτοντας όλα τα bit 1 είναι...

2^0	2^1	2^2	2^3	2^4	2^5	...>
1	2	4	8	16	32	
1	1	1	1	1	1	
1+	2+	4+	8+	16+	32	= 63

Δηλαδή 2^6-1 και γενικά με ν bits ο μεγαλύτερος αριθμός που μπορούμε να πάρουμε είναι $2^ν$-1 (αυτό για να μπορείτε από εδώ και πέρα να υπολογίζετε μόνοι σας το εύρος τιμών των πεδίων στην κεφαλίδα του IP κλπ που περιέχουν δεκαδικούς ακεραίους -χωρίς πρόσημο πάντα).

Έτσι τα 8bit ενός octet όπως είπαμε και παραπάνω μας δίνουν στο δεκαδικό έναν αριθμό από 0 ως 2^8 -1 = 255.

Για λόγους ευκολίας και οργάνωσης τα 32bit της IP διεύθυνσης "κόβονται" σε 4 τέτοια octets τα οποία τα μελετάμε ξεχωριστά. Τα octets αυτά κατόπιν συνδέονται μεταξύ τους και "φεύγουν" σαν ένας αριθμός των 32bit.

Ας το δούμε αυτό στην πράξη...

Το παρακάτω

10000000 01000000 11000000 00100000

που με μια πρώτη ματιά φαίνεται να αναπαριστά τον αριθμό (2^0+2^9+2^{16}+2^{17}+2^{26})
χωρίζεται σε 4 octets, το καθένα με μια τιμή από 0 ως 255

10000000 = 1
01000000 = 2
11000000 = 3
00100000 = 4

και γράφεται στη μορφή 1.2.3.4 γνωστή και ως dotted decimal μορφή (δεκαδική με τελίτσες).

3.2.2.Η δομή της

Η διεύθυνση του IP περιέχει δύο πληροφορίες, την ταυτότητα του δικτύου και την ταυτότητα του τερματικού, αυτό εξ' άλλου ήταν και ένας απ'τους βασικούς στόχους που είπαμε στην αρχή, να έχουμε ένα διαδικτυακό μοντέλο λειτουργίας. Σε αντίθεση με άλλα πρωτόκολλα που η διεύθυνσή τους περιγράφει μόνο το τερματικό χωρίς να δίνει πληροφορία για το δίκτυο στο οποίο ανήκει, το IP μας δίνει και τις δύο πληροφορίες κάνοντας την δρομολόγηση των πακέτων πολύ καλύτερη, λαμβάνει δηλαδή εξ' αρχής υπόψη ότι τα τερματικά ανήκουν σε διάφορα υπο-δίκτυα και όχι σε ένα μονοκόμματο δίκτυο.

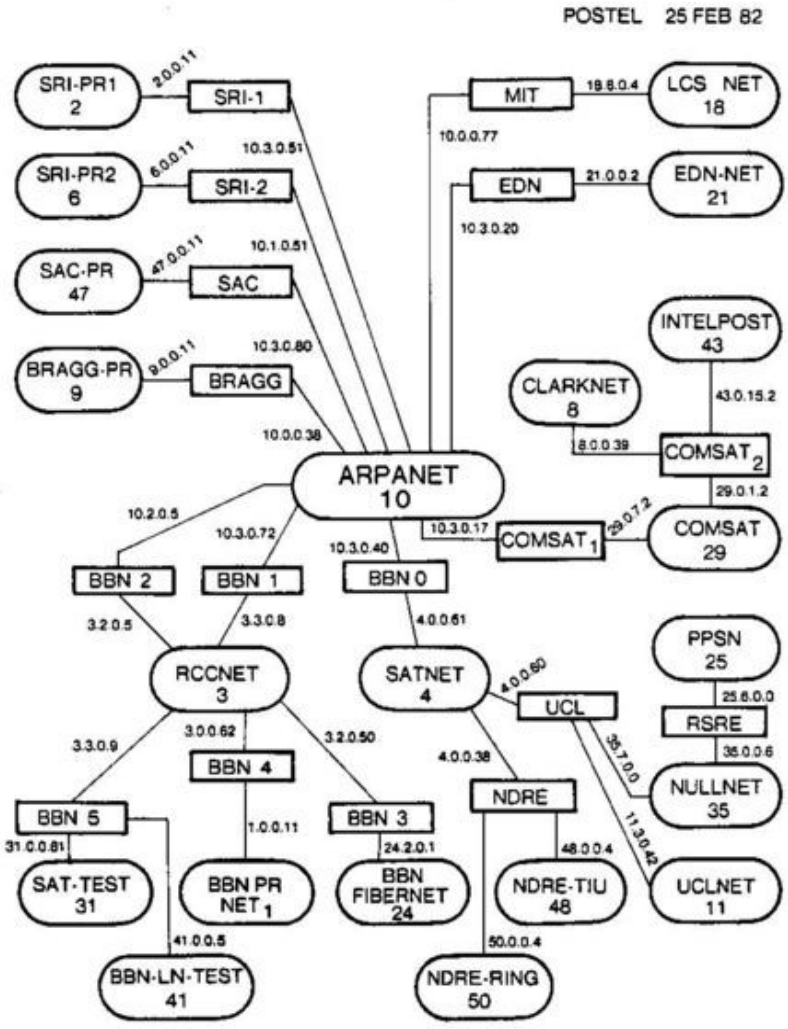
Ο τρόπος που οι πληροφορίες αυτές κωδικοποιούνταν στη διεύθυνση έχει αλλάξει κατά καιρούς γι' αυτό και θα δούμε όλη την πορεία μέχρι σήμερα.

3.2.2.1.Η αρχή στο ARPANET

Στην αρχή το πρώτο octet της διεύθυνσης (network field) περιείχε έναν χαρακτηριστικό αριθμό για το δίκτυο και τα υπόλοιπα octets χαρακτήριζαν το τερματικό (rest field), έτσι μπορούσαν με το πρωτόκολλο να υπάρξουν συνολικά 256 υπο-δίκτυα

NNN.	RRR . RRR . RRR
<network field>	<rest field>

Το 1982 αυτό δεν ήταν πρόβλημα αφού τα υπο-δίκτυα ήταν λίγα και μεγάλα. Στο παρακάτω σχεδιάγραμμα του Jon Postel (του ανθρώπου που έπαιζε τον ρόλο της IANA που προαναφέραμε, τότε) βλέπουμε τον χάρτη του Internet στις 25/2/1982,



τα παραλληλόγραμμα είναι δρομολογητές και τα οβάλ πλαίσια αναπαριστούν υπο-δίκτυα, κάθε ένα με τον χαρακτηριστικό του αριθμό να αναγράφεται. Βλέπουμε πχ. ότι τα τερματικά του ARPANET τότε είχαν διευθύνσεις της μορφής 10.x.x.x, το M.I.T. είχε διευθύνσεις της μορφής 18.x.x.x κλπ. Με την εμφάνιση των τοπικών δικτύων (LAN s) ο αριθμός των υπο-δικτύων αυξήθηκε οπότε έπρεπε να βρεθεί μια άλλη λύση, έτσι καταλήξαμε στις κλάσεις διευθύνσεων.

3.2.2.2.Οι κλάσεις διευθύνσεων

Απ' την στιγμή που χρειαζόντουσαν περισσότερα υπο-δίκτυα, το network field έπρεπε να μεγαλώσει κατά περίπτωση και να μην περιορίζεται στο πρώτο octet. Έτσι υιοθετήθηκε ένας έξυπνος μηχανισμός κατάταξης των διευθύνσεων. Για τα λίγα μεγάλα υπο-δίκτυα με πολλά τερματικά χρειαζόταν μεγάλο rest field (πολλά τερματικά) και μικρό network field (λίγα υπο-δίκτυα), τα πολλά μικρά υπο-δίκτυα χρειαζόντουσαν μεγάλο network field (πολλά υπο-δίκτυα) και μικρό rest field (λίγα τερματικά). Έτσι δημιουργήθηκαν αρχικά 3 κλάσεις, το network field της πρώτης κλάσης ήταν 1 octet, της δεύτερης 2 octets και της τρίτης 3 octets, αφήνοντας χώρο 3,2 και 1 octets αντίστοιχα για το rest field. Για να φαίνεται σε ποια κλάση άνηκε η διεύθυνση χρησιμοποιήθηκαν τα πρώτα bits του πρώτου octet, έτσι όλες οι διευθύνσεις που το πρώτο τους octet ξεκίναγε από 0 άνηκαν στην πρώτη κλάση, όσων ξεκίναγε από 10 στην δεύτερη και 110 στην τρίτη. Λογικά κάπου εδώ θα μπλεχτήκατε οπότε ας το δούμε σχηματικά...

Κλάση	Πρώτο octet	Δεύτερο octet	Τρίτο Octet	Τέταρτο Octet	Εύρος	Υπο-δίκτυα	Τερματικά
A	0nnnnnnn	rrrrrrrr	rrrrrrrr	rrrrrrrr	0.0.0.0 - 127.255.255.255	126	16,777,214
B	10nnnnnn	nnnnnnnn	rrrrrrrr	rrrrrrrr	128.0.0.0 - 191.255.255.255	16,384	65,534
C	110nnnnn	nnnnnnnn	nnnnnnnn	rrrrrrrr	192.0.0.0 - 223.255.255.255	2,097,152	254

Επίσης δημιουργήθηκε άλλη μια κλάση της οποίας οι διευθύνσεις δεν χαρακτηρίζουν τερματικά αλλά ομάδες τερματικών (θα μπορούσαμε να πούμε εικονικά υπο-δίκτυα), γνωστές και ως multicast (περισσότερα για το τι είναι δαύτες θα πούμε παρακάτω όταν ασχοληθούμε με το IGMP).

D	1110xxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	224.0.0.0 - 239.255.255.255	-	-

Τέλος όσες διευθύνσεις περίσσεψαν (δεν άνηκαν ούτε στην κλάση D) παρέμειναν reserved και μπήκαν σε μια άλλη κλάση που ονομάστηκε E.

E	1111xxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	240.0.0.0 - 255.255.255.255	-	-

Έτσι για να δούμε σε ποιο υπο-δίκτυο ανήκει ένα τερματικό, κοιτάμε την κλάση της διεύθυνσης στην οποία ανήκει, μετράμε τα octets και αγνοούμε το res field Άρα η διεύθυνση επιτελεί το έργο της αφού μας δίνει και πάλι τις ίδιες πληροφορίες.

3.2.2.3.Η κατάσταση σήμερα

Με την έλευση των ISP και την τεράστια ανάπτυξη του Internet δημιουργήθηκε η ανάγκη για έναν ποιο ευέλικτο τρόπο χωρισμού των υπο-δικτύων, έτσι οδηγηθήκαμε γύρω στο 1993 από τις κλάσεις στο CIDR ή αλλιώς Classless Inter Domain Routing (RFC1519). Το CIDR ουσιαστικά επιτρέπει να έχουμε μεταβλητού μήκους network field (ή μεταβλητού μήκους rest field αν προτιμάτε) και όχι κάποιο προκαθορισμένο μήκος, χρησιμοποιώντας την τεχνική του bitmasking, ας δούμε λίγο πώς αυτή η μέθοδος δουλεύει...

Έστω ότι έχουμε τον αριθμό 7...

2^0	2^1	2^2	...
1	1	1	...

Bitmasking σημαίνει να "πετάξουμε", ή αλλιώς να κάνουμε mask ένα κομμάτι του αριθμού κάνοντας λογικό AND τα bits του αριθμού που θέλουμε να πετάξουμε με το 0 και αυτά που θέλουμε να κρατήσουμε με το 1 (θυμίζω ότι 1 & 1 = 1, 1 & 0 = 0, 0 & 0 = 0). Έτσι για παράδειγμα για να κάνουμε mask το τελευταίο bit του αριθμού 111 = 7 θα τον κάνουμε λογικό AND με τον αριθμό 110 = 3 -τον οποίο τον ονομάζουμε bitmask-...

	2^0	2^1	2^2	...
AND	1	1	1	...
	1	1	0	...
	1	1	0	..

...και θα πάρουμε τον αριθμό 110 αφού το τελευταίο bit του αριθμού 7 έχει γίνει masked. Το ενδιαφέρον τώρα είναι πως θα πάρουμε ακριβώς το ίδιο αποτέλεσμα αν κάνουμε λογικό AND τον αριθμό 110 με τον εαυτό του...

	2^0	2^1	2^2	...
AND	1	1	0	...
	1	1	0	...
	1	1	0	..

Έτσι μπορούμε να πούμε ότι κάνοντας mask κάποιον αριθμό ουσιαστικά κρατάμε ένα κομμάτι του σταθερό (αυτό που γίνεται λογικό AND με το 1) και το υπόλοιπο το αφήνουμε να μεταβάλλεται, αγνοώντας το (αυτό που γίνεται λογικό AND με το 0). Αυτό σε μια IP διεύθυνση μας φαίνεται ιδιαίτερα χρήσιμο, αφού μπορούμε έτσι να κρατήσουμε σταθερό το network field και να αφήσουμε το rest field να μεταβάλλεται. Η μόνη διαφορά από πριν είναι ότι πλέον εκτός από την διεύθυνση IP θα χρειαστούμε και έναν αριθμό για να κάνουμε bitmask την IP για να μπορέσουμε να πάρουμε από την IP την πληροφορία που θέλουμε. Τον αριθμό αυτό τον λέμε subnet mask και ουσιαστικά πρόκειται για μια σειρά από 1 όσο μακρυά όσο το network field της συγκεκριμένης IP διεύθυνσης που μετά ακολουθούνται από μια σειρά 0 μέχρι το τέλος της διεύθυνσης, δηλαδή το rest field.

Υπάρχουν 2 τρόποι συμβολισμού της subnet mask, ο πρώτος είναι να την συμβολίσουμε ακριβώς όπως την IP διεύθυνση, πχ. 255.255.255.0 κι ο δεύτερος να δηλώσουμε δίπλα στην IP πόσα bits είναι το network field πχ. 10.47.132.0/24 (ουσιαστικά πόσους άσσους έχει η subnet mask).

Για να το δούμε όλο αυτό λίγο στην πράξη, έστω η IP 10.47.132.65...

Με το προηγούμενο σύστημα των κλάσεων η IP αυτή θα περιέγραφε το host 47.132.65 που βρίσκεται στο υπο-δίκτυο 10.0.0.0 (Class A), τώρα κάνοντας mask την IP με την subnet mask 255.255.255.0 πχ. Θα συμβεί το εξής...

	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
	10								47								132								**65**							
	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0
	255								255								255								**0**							
&	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
	10								47								132								**0**							
	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0

Το κομμάτι που έγινε masked (το τελευταίο octet δηλαδή) έχει εξαφανιστεί, ενώ το υπόλοιπο κομμάτι παρέμεινε σταθερό. Έτσι το network field είναι 24bits και το rest field είναι τα υπόλοιπα 8bits. Αυτό το υπο-δίκτυο όπως το κάναμε τώρα από Class A που ήταν, έγινε ανάλογο ενός Class C υπο-δικτύου σε μέγεθος αφού το rest field είναι το τελευταίο octet. **Η IP που προκύπτει μετά το AND (στην περίπτωσή μας η 10.47.132.0) και περιγράφει το network field, λέγεται Subnet Address και όπως καταλαβαίνετε αφού περιγράφει το δίκτυο δεν μπορεί να χρησιμοποιηθεί σε κάποιο τερματικό.**

Ας δούμε τι θα γινόταν όμως αν εμείς θέλαμε ένα μικρότερο υπό-δίκτυο, μικρότερο από Class C ή γενικά ένα υπο-δίκτυο του οποίου το network field δεν θα αποτελούνταν μόνο από "γεμάτα" octets.

Έστω λοιπόν ότι θέλουμε να φτιάξουμε ένα υπο-δίκτυο που να περιέχει μόνο 5 hosts κι όχι 254, δουλεύουμε ως εξής...

Θέλουμε το rest field να είναι αρκετά μεγάλογια να αναπαραστήσει 5 διαφορετικούς αριθμούς, όπως είπαμε παραπάνω μπορούμε να βρούμε πόσους αριθμούς μπορεί να αναπαραστήσουν ν bits από τον τύπο **πλήθος = 2^v -1** έτσι θέλουμε 5 = 2^v -1 => **6** = 2^v επειδή δεν υπάρχει ακριβώς ν για την περίπτωσή μας διαλέγουμε το ν = 3 αφού για ν=2 το πλήθος των αριθμών που μπορούμε να πάρουμε είναι μικρότερο από 6.

Γνωρίζοντας πόσο μεγάλο είναι το rest field, υπολογίζουμε το network field...
32 – 3 = 29

Άρα η subnet mask που θα χρησιμοποιήσουμε θα έχει 29 άσσους ή όπως λέμε θα έχουμε ένα /29 υπο-δίκτυο. Τώρα αφού ξέρουμε το απαραίτητο μήκος του network field μας μένει απλά να το ορίσουμε, πριν προχωρήσουμε όμως ας ρίξουμε μια ματιά στην subnet mask...

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
255								255								255								**248**							
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**

Αγνοούμε τα octets που έχουν ήδη γεμίσει και συγκεντρωνόμαστε στο τελευταίο octet ή καλύτερα στα τελευταία 5 bits της subnet mask που είναι 1.

Αν ήδη αναρωτιέστε γιατί γεμίσαμε το octet ανάποδα, η απάντηση είναι λίγο περίπλοκη και απαιτεί την κατανόηση κάποιων βασικών κανόνων σχετικά με το πώς διαβάζει/γράφει ο επεξεργαστής στην μνήμη. Γενικώς το τι είναι ίσια και τι ανάποδα όταν αναφερόμαστε σε τόσο χαμηλό επίπεδο είναι σχετικό, εξαρτάται από το πώς το βλέπει ο καθένας, κάτι που για εμάς μπορεί να φαίνεται παράλογο για τον επεξεργαστή μπορεί να είναι απόλυτα φυσιολογικό.Προς το παρόν λοιπόν κρατήστε σαν κανόνα ότι όταν φτιάχνουμε την subnet mask το κάθε octet το γεμίζουμε ανάποδα, αν ωστόσο "καίγεστε" να μάθετε τι παίζει, ανατρέξτε στο κατάλληλο παράρτημα στο τέλος. Μην πείτε ότι δεν σας προειδοποίησα...

Από αυτά τα bits καταλαβαίνουμε τις δυνατές τιμές για το network field, έτσι λοιπόν το τελευταίο κομμάτι του network field δεν θα πρέπει να περιέχει κάποιο bit set στα 3 πρώτα bits γιατί αυτό θα χαθεί κατά το bitmasking, έτσι στη συγκεκριμένη περίπτωση το network field μπορεί να τελειώνει σε 8, 16, 32, 64, 128 ή κάποιο άθροισμά μεταξύ τους, ποιο συγκεκριμένα, το τελευταίο κομμάτι του network field μπορεί να είναι (με μπλε δηλώνουμε αυτά που γίνονται and με το 0 -masked δηλαδή).

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	
0	0	0	0	0	0	0	0	**0**
0	0	0	**1**	0	0	0	0	**8**
0	0	0	0	**1**	0	0	0	**16**
0	0	0	**1**	**1**	0	0	0	**24**
0	0	0	0	0	**1**	0	0	**32**
0	0	0	**1**	0	**1**	0	0	**40**
0	0	0	0	**1**	**1**	0	0	**48**
0	0	0	**1**	**1**	**1**	0	0	**56**
0	0	0	0	0	0	**1**	0	**64**
...	...	...	...	...	...	...	...	**+8**

Φαίνεται λοιπόν ότι το τελευταίο κομμάτι του network field μπορεί να είναι οποιοδήποτε πολλαπλάσιο του 8 από το 0 μέχρι το 255. Αυτό είναι λογικό αφού οποιαδήποτε δύναμη του 2 > 8 είναι πολλαπλάσιο του 8 (και προφανώς το άθροισμα 2 ή 3ών τέτοιων αριθμών θα είναι επίσης πολλαπλάσιο του 8). Έτσι διαλέγουμε από που θέλουμε να αρχίσει το 8άρι υπο-δίκτυό μας, έστω ότι διαλέγουμε το network field να τελειώνει σε 56. Ας επαληθεύσουμε τώρα αυτά που είπαμε παραπάνω...

Θα πρέπει όποια διεύθυνση ανήκει στο υπο-δίκτυο, αν γίνει mask με την subnet mask, να μας δώσει το network field. Ας πάρουμε την 58 πχ. (πρέπει να είναι μεταξύ 56 και 63 γιατί μετά θα ανήκει στο επόμενο 8άρι υπο-δίκτυο, το 64 δηλαδή), et voila...

	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
	x								x								x								**58**							
																									0	1	0	1	1	1	0	0
	255								255								255								**248**							
&	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1
	x								x								x								**56**							
																									0	0	0	1	1	1	0	0

3.2.3.Η ευελιξία που μας δίνει η subnet mask – Subneting/Superneting

Μπορεί να φαίνεται απλό αλλά το να μεταφέρουμε την πληροφορία για το network field από την IP στην subnet mask μας έδωσε κι άλλες δυνατότητες, σκεφτείτε κάποιος να κάνει mask την ίδια IP με διαφορετική subnet mask τι θα γίνει...

Subneting

Αρχικά όπως κάναμε και παραπάνω μπορούμε να κόψουμε ένα μεγάλο δίκτυο σε πολά υπο-δίκτυα, το μόνο που χρειάζεται είναι το κάθε υπο-δίκτυο να έχει έναν δρομολογητή για να έχει είσοδο/ έξοδο στο ευρύτερο δια-δίκτυο. Παραπάνω πήραμε ένα δίκτυο /24 και το σπάσαμε σε 32 (256/8) υπο-δίκτυα /29, θα μπορούσαμε να το σπάσουμε με διαφορετικό τρόπο, πχ αντί για 32 /29 δίκτυα να είχαμε οποιοδήποτε συνδυασμό, πχ. ένα δίκτυο με 64 IPs (/26) και 24 δίκτυα των 8 IPs (/29). Οι κανόνες θα ήταν οι ίδιοι, απλά θα άλλαζε για κάθε υπο-δίκτυο η subnet mask (πχ. για ένα./26 δίκτυο -64IPs δηλαδή- η subnet mask θα τελειώνει σε .192 όπως μπορείτε να δείτε).

Να το ξαναδούμε λίγο, έστω αυτή τη φορά ότι έχουμε δίκτυα μεγαλύτερα από /24 να διαχειριστούμε, έστω λοιπόν το δίκτυο

10.47.0.0/16 (το Class B που λέγαμε πριν)

Το /16 δίκτυο (με bold φαίνεται πόσο μεγάλο είναι το network field)...

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								0								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
255								255								0								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

...σπάει σε 2 /17 δίκτυα (θυμηθείτε ότι γεμίζουμε την subnet mask ανάποδα)...

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								0								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0
255								255								128								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								128								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
255								255								128								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0

...σε 4 /18 δίκτυα...

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								0								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	0	0	0	**0**	**0**	0	0	0	0	0	0	0	0
255								255								192								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								64								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	0	0	0	**1**	**0**	0	0	0	0	0	0	0	0
255								255								192								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								128								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	0	0	0	**0**	**1**	0	0	0	0	0	0	0	0
255								255								192								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								192								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0
255								255								192								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0

...σε 8 /19 δίκτυα, σε 16 /20 δίκτυα, σε 32 /21 δίκτυα, σε 64 /22, σε 128 /23, σε 256 /24 (όταν έχει γίνει mask ολόκληρο το προτελευταίο octet), σε 512 /25 (όταν αρχίζει να γίνεται mask και το τελευταίο octet), σε 1024 /26 (των 64ων IP το καθένα), σε 2048 /27, σε 4096 /28, σε 8192 /29 (των 8 IP το καθένα, όπως και πριν), σε 16384 /30 δίκτυα (των 4ων Ips), σε 32768 /31 δίκτυα (των 2 IPs) και ?

Σε 65536 IPs ? Αφού το rest field όταν φτάσουμε στο /32 δεν θα υπάρχει (θα είναι όλη η IP το network field). Ήρθαμε λοιπόν σε ένα πρόβλημα, τι γίνεται όταν η subnet mask καλύπτει όλη την IP ? Ορίζει τερματικό ή δίκτυο ? Θα μπορούσαμε να πούμε αφού η subnet mask όπως είπαμε στην αρχή κάνει mask το network field, ότι όλη η IP πλέον είναι ένα network field και άρα όλη η IP ορίζει ένα δίκτυο (είναι μια subnet address δηλαδή). Κατ εξαίρεση λοιπόν, αφού δεν υπάρχει δίκτυο με 0 τερματικά (γιατί αφού δεν υπάρχει χόρος για το rest field, δεν υπάρχουν και τερματικά), η IP αυτή ορίζει συνήθως ένα τερματικό που επικοινωνεί κατευθείαν με τον δρομολογητή, το απομονώνει δηλαδή απ' τα υπόλοιπα τερματικά, αφού αυτά ανήκουν σε άλλο "δίκτυο" (έχουν διαφορετικό network field). Τέτοιες IPs θα δείτε συχνά σε συνδέσεις σημείου-προς-σημείο (πχ. η σύνδεσή σας με τον ISP) και γενικώς σε τερματικά που θέλουμε να απομονώσουμε. Επίσης μια τέτοια IP είναι η περίφημη loopback (η IP 127.0.0.1/32) όπου είναι ένας τρόπος για να μπορεί το τερματικό να "δει" τον εαυτό του.

Το παραπάνω μοντέλο δεν επικράτησε αμέσως, στην αρχή η πρώτη πρόταση για το subneting προέβλεπε ότι η διεύθυνση IP θα αποτελείται από 3 κομμάτια. Το network field θα αποτελούνταν μόνο απ' τα γεμάτα octets (δηλαδή στην περίπτωσή μας παραπάνω θα ήταν το 10.47) και το rest field που θα απέμενε θα έσπαγε σε άλλα 2 πεδία, το host field που θα αποτελούνταν όπως και πριν απ' το masked κομμάτι της IP και ένα νέο πεδίο, το **subnet field** που θα ήταν ότι περίσσευε, δηλαδή το unmasked κομμάτι του rest field (στο παράδειγμά μας παραπάνω, στην πρώτη πχ. περίπτωση το subnet field θα ήταν το τελευταίο bit του προτελευταίου octet -αυτό που είναι με bold, ενώ στην δεύτερη περίπτωση τα τελευταία δύο bit). Το μοντέλο αυτό δεν ήταν καθόλου ευέλικτο αφού το subnet field μπορεί να είχε διαφορετική τιμή για κάθε υπο-δίκτυο, θα είχε όμως για όλα τα υπο-δίκτυα που θα "κόβαμε" το ίδιο μήκος. Έτσι αν πχ. Είχαμε πολλά υπο-δίκτυα των 10 IPs και ένα των 100, θα έπρεπε να "κόψουμε" το subnet field ώστε να είναι αρκετά μικρό ώστενα μπορεί το host field του συγκεκριμένου υπο-δικτύου να χωρέσει 100 IPs. Επίσης το πρώτο και το τελευταίο υπο-δίκτυο που θα "κόβαμε" (αυτό που θα είχε 0 στο subnet field του -γνωστό και ως "Zero Subnet"- κι αυτό που θα είχε γεμάτο το subnet field του -γνωστό και ως "All-ones Subnet"-) δεν μπορούσαν να χρησιμοποιηθούν οπότε το πρόβλημα γινόταν ακόμα μεγαλύτερο. Καταλήξαμε λοιπόν στο να μην έχουμε το subnet field κι έτσι να μπορούμε να κόψουμε τα υπό-δίκτυά μας όπως θέλουμε, με διαφορετικό μήκος δηλαδή. Στο παρακάτω παράδειγμα εξετάζουμε αυτό ακριβώς.

Ας δούμε τι γίνεται τώρα αν θέλουμε να "κόψουμε" ένα δίκτυο σε υπο-δίκτυα διαφορετικού μεγέθους. Παραπάνω χωρίσαμε το /16 δίκτυό μας σε 2 /17, 4 /18 κλπ, η subnet mask όμως μας δίνει την ευελιξία να κόψουμε το δίκτυό μας και σε διαφορετικού μήκους υπο-δίκτυα. Ας πάρουμε ένα απ' τα 16 /20 δίκτυα που φτιάξαμε κι ας το κόψουμε σε μικρότερα υπο-δίκτυα. Έστω λοιπόν το δεύτερο /20 υπο-δίκτυο που θα έχει την παρακάτω subnet address (και αφού είναι /20 θα έχει 20 άσους στην subnet mask).

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								16								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	0	**1**	**0**	**0**	**0**	0	0	0	0	0	0	0	0
255								255								240								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0

Το συνολικό μέγεθος των διευθύνσεων που θα έχουμε είναι προφανές ότι δεν μπορεί να ξεπερνάει αυτό του /20 δικτύου μας γιατί ήδη υπάρχουν 20 άσοι στην subnet mask κι εμείς μόνο να βάλουμε άσους μπορούμε, όχι να αφαιρέσουμε. Έτσι αφού μένουν 12 bit για να "παίξουμε" μπορούμε να έχουμε ένα δίκτυο με μέγιστο αριθμό διευθύνσεων 2^12 = 4096 (συμπεριλαμβάνεται και η subnet address, γι' αυτό δεν έβαλα το -1). Έτσι έστω πχ. ότι θέλουμε ένα μεγάλο υπο-δίκτυο των 2000 τερματικών (ένα call-center πχ.), ένα μικρότερο των 1000 τερματικών (ένα κέντρο δημοσκοπήσεων), ένα των 500 τερματικών (ένας οργανισμός) κι ένα των 256 τερματικών (έστω η διοίκιση του οργανισμού), σύνολο δηλαδή 3756 τερματικά και ότι περισσεύει το βάζουμε σε ένα άλλο υπο-δίκτυο που έχουμε για όταν χρειαστεί (340 τερματικά δηλαδή). Για κάθε υπο-δίκτυο που κόβουμε είπαμε πως δεν μπορούμε να χρησιμοποιήσουμε την subnet address του σε κάποιο τερματικό, αφού αυτή χαρακτηρίζει το υπο-δίκτυο, έτσι θέλουμε:

1 υπο-δίκτυο με 2001 IPs (2000 + την subnet address)
1 υπο-δίκτυο με 1001 IPs
1 υπο-δίκτυο με 501 IPs
1 υπο-δίκτυο με 257 IPs
1 υπο-δίκτυο με ότι περισσεύει.

ξεκινάμε λοιπόν όπως και πριν:

Για το υπο-δίκτυο των 2001 IPs θέλουμε το rest field να μπορεί να χωρέσει 2001 τερματικά, η

κοντινότερη δύναμη του 2 που μας κάνει τη δουλειά είναι η 11η (2^11 = 2048), έτσι η subnet mask θα έχει 32 – 11 = 21 bits, άρα θα έχουμε ένα /21 δίκτυο. Επειδή είναι το πρώτο υπο-δίκτυο και ξεκινάει απ' το 16, θα έχει την παρακάτω subnet address. Με κόκκινο μαρκάρουμε την αλλαγή που κάναμε στην subnet mask.

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								16								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	0	**0**	**1**	**0**	**0**	**0**	0	0	0	0	0	0	0	0
255								255								**248**								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0

Το επόμενο υπο-δίκτυο θέλουμε να είναι των 1001 IPs, και πάλι εργαζόμαστε ομοίως, εδώ η δύναμη του 2 που μας κάνει τη δουλειά είναι η 10η (2^10 = 1024), οπότε η subnet mask θα έχει 32 – 10 = 22 bit και η subnet address θα ξεκινάει από εκεί που τελείωσε το προηγούμενο υπο-δίκτυο, δηλαδή το προτελευταίο octet θα έχει την τιμή 16 + 2048/256 (αφού το προτελευταίο octet αυξάνει κατά 1 κάθε φορά που γεμίζει το τελευταίο octet, δηλαδή κάθε 256 IPs) = 16 + 8 = 24. Οπότε έχουμε την παρακάτω subnet address.

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								24								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	**0**	**1**	**1**	**0**	**0**	**0**	0	0	0	0	0	0	0	0
255								255								**252**								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0

Το επόμενο υπο-δίκτυο θέλουμε να είναι των 501 IPs, οπότε η δύναμη του 2 που θέλουμε είναι η 9η (2^9 = 512) κι έτσι η subnet mask θα έχει 32 – 9 = 23bit και το προτελευταίο octet θα ξεκινάει από εκεί που τελείωσε το προηγούμενο υπο-δίκτυο δηλαδή το 28 (24 + 1024/256 = 28).

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								28								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	0	**1**	**1**	**1**	**0**	**0**	**0**	0	0	0	0	0	0	0	0
255								255								**254**								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0

Το επόμενο υπο-δίκτυο θέλουμε να είναι των 257 IPs, οπότε η δύναμη του 2 που θέλουμε είναι πάλι η 9η (2^9 = 512) αφού δεν μας φτάνουν τα 256 κι έτσι η subnet mask θα έχει πάλι 32 – 9 = 23bit και το προτελευταίο octet θα ξεκινάει από εκεί που τελείωσε το προηγούμενο υπο-δίκτυο δηλαδή το 30 (28 + 512/256 = 30).

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								30								0							
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	0	**1**	**1**	**1**	**1**	**0**	**0**	**0**	0	0	0	0	0	0	0	0
255								255								**254**								0							
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0

Είδαμε ότι για 1 τερματικό χρειάστηκε να σπαταλήσουμε 255 IPs, αυτό δυστυχώς είναι μια παράπλευρη απώλεια και δεν μπορούμε να κάνουμε κάτι, σε σχέση όμως με τα προηγούμενα μοντέλα που σπαταλούσαμε πολλές χιλιάδες IPs αυτή η μέθοδος είναι καλύτερη, το μέγιστο ποσό χαμένων IPs λοιπόν με αυτό το μοντέλο είναι 255.

Ας δούμε τώρα το τελευταίο υπο-δίκτυο που θα έχει ότι περίσσεψε. Καταρχάς ας δούμε τι περίσσεψε:

2048 το πρώτο υπο-δίκτυο
1024 το δεύτερο υπο-δίκτυο
512 το τρίτο υπο-δίκτυο
512 το τέταρτο υπο-δίκτυο

σύνολο 4096 IPs ! Είμαστε δηλαδή ακριβώς στο όριο των διευθύνσεων που επιτρέπει να χρησιμοποιήσουμε το /20 δίκτυό μας (οι subnet addresses των διαφόρων υπο-δικτύων περιλαμβάνονται).

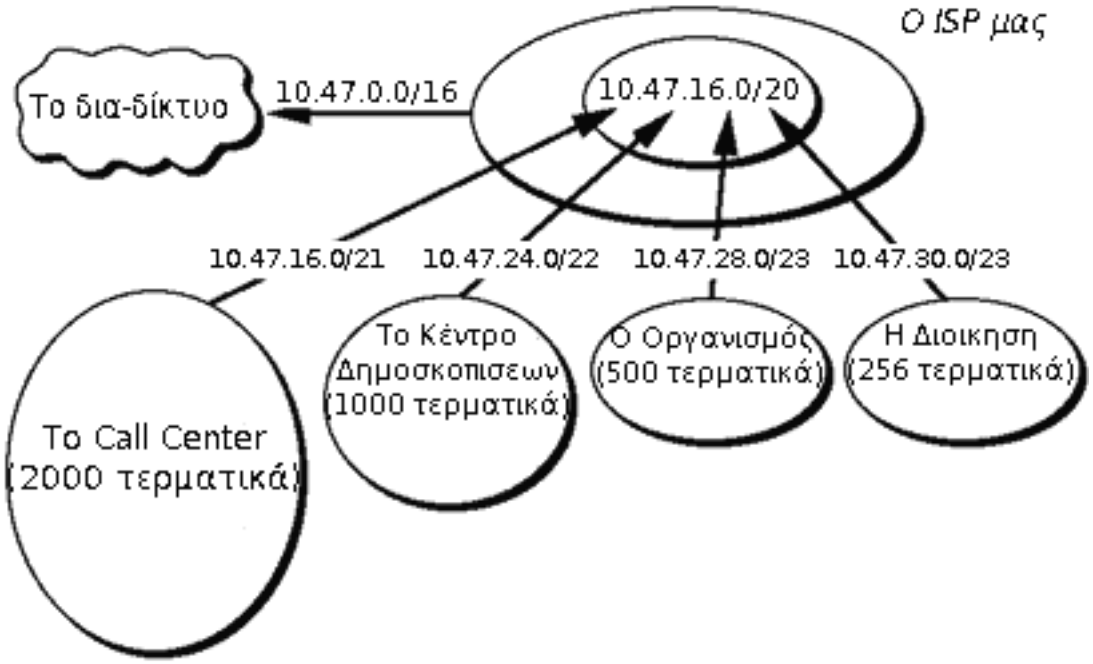
Superneting

Αφού είδαμε το πώς κόβουμε ένα μεγάλο δίκτυο σε πολλά μικρότερα ας δούμε τώρα και το ανάποδο που είναι εξίσου σημαντικό. Τώρα που κόψαμε το /20 δίκτυό μας σε πολλά μικρότερα αυτό εξαφανίστηκε ? Με μια πρώτη ματιά βλέπουμε ότι δεν υπάρχει ποια μια subnet address για να χαρακτηρίσει το /20 δίκτυό μας αλλά 4 subnet addresses (βλ. παραπάνω) που χαρακτηρίζουν 4 υπο-δίκτυα. Για να δούμε λοιπόν τι έγινε με το /20 δίκτυό μας...

Κάθε δρομολογητής στην αρχή (πριν το CIDR) κρατούσε στη μνήμη τού όλα τα υπο-δίκτυα του Internet, δηλαδή όλα τα Class A/B/C δίκτυα που υπήρχαν στον κόσμο (μιλάμε για πολλές καταχωρίσεις). Οι δρομολογητές επικοινωνούσαν μεταξύ τους και ο κάθε δρομολογητής ανακοίνωνε τα υπο-δίκτυα με τα οποία επικοινωνούσε, οι υπόλοιποι ανανέωναν την λίστα τους και ξαναυπολογίζαν τα μονοπάτια. Όσο το Internet μεγάλωνε εκτός από το πρόβλημα με τις IPs που λιγοστεύαν που είπαμε παραπάνω, υπήρχε και το πρόβλημμα ότι οι δρομολογητές είχαν αρχίσει να γεμίζουν και να έχουν πρόβλημμα με την ταχύτητα απόκρισής τους κλπ. Το δια-δίκτυο δηλαδή είχε φορτωθεί πολύ.

Επομένως το δίκτυο έπρεπε να χωριστεί σε τομείς (domains – από εκεί βγαίνει και το Inter Domain Routing στο CIRD) ή καλύτερα σε διαχειρίσιμες οντότητες, σύνολα δηλαδή από υπο-δίκτυα που υπόκεινται σε κοινή διαχείριση (έχουν τον ίδιο διαχειριστή -εταιρία/οργανισμό κλπ-). Ο κάθε δρομολογητής έτσι δεν χρειαζόταν να ξέρει την δομή ενός τέτοιου τμήματος (σε πόσα και ποια υπο-δίκτυα κόβεται) αλλά μόνο με ποιους δρομολογητές πρέπει να μιλήσει για να φτάσει σε αυτό το τμήμα κι από εκεί και πέρα οι δρομολογητές εκείνου του τμήματος θα αναλάμβαναν. Έτσι στο παράδειγμά μας παρόλο που δεν υπάρχει ενιαίο /20 δίκτυο, στο δια-δίκτυο ο δρομολογητής μας αντί να ανακοινώσει όλα τα υπο-δίκτυά του (τα 4 που φτιάξαμε), θα ανακοινώνει το /20 δίκτυο που του ανατέθηκε κι έτσι οι γύρω του δρομολογητές αντί να έχουν 4 καταχωρίσεις θα έχουν μία. Αντίστοιχα τώρα ο δρομολογητής ο οποίος διαχειρίζεται το /16 δίκτυο απ' το οποίο κόψαμε το /20 δίκτυό μας (το 10.47.0.0/16 δηλαδή), δεν θα ανακοινώνει στους γύρω του όλα τα /20 υπο-δίκτυά του αλλά το /16 δίκτυο, έτσι οι γύρω του δρομολογητές αντί να έχουν 32 καταχωρίσεις θα έχουν πάλι μία. Μειώνεται με αυτό τον τρόπο ο φόρτος στους δρομολογητές και μπαίνει μια τάξη στο δια-δίκτυο, δίνοντάς του την δυνατότητα να μεγαλώσει ακόμα περισσότερο (και πράγματι μετά και αυτή την αλλαγή έγινε η επανάσταση στο δια-δίκτυο και έγινε όπως το ξέρουμε σήμερα).

Συνοψίζοντας τα παραπάνω, το όλο σενάριο που αναφέραμε παραπάνω μπορούμε να το παρουσιάσουμε στην παρακάτω εικόνα...



Εικόνα 3: Το σενάριό μας με το "κόψιμο" του /20 δικτύου μας

3.2.4.Superneting στο Internet, τι είναι οι RIRs

Το Internet λοιπόν χωρίστηκε σε τομείς όπως είπαμε και παραπάνω, κυρίως με γεωγραφικά κριτήρια, που ονομάζονται **Regional Internet Registries (RIRs)** και καθένας τους έχει μια αρχή διαχείρισης:

Το **Asia Pacific Network Information Centre (APNIC)** είναι υπεύθυνο για τις διευθύνσεις IP στην ευρύτερη περιοχή της Ασίας και της Ωκεανίας, διαχειρίζεται μέχρι στιγμής (μέχρι τώρα που γράφεται το tutorial) τα παρακάτω υπο-δίκτυα.(περισσότερες πληροφορίες -> http://www.apnic.net/info/reports/index.html) ..

58.0.0.0/7	122.0.0.0/7	169.208.0.0/12	218.0.0.0/7
60.0.0.0/7	124.0.0.0/7	202.0.0.0/7	220.0.0.0/7
121.0.0.0/8	126.0.0.0/8	210.0.0.0/7	222.0.0.0/8

Το **American Registry for Internet Numbers (ARIN)** είναι υπεύθυνο για τις διευθύνσεις IP στην Βόρεια Αμερική (Η.Π.Α./ Καναδά), τα νησιά της Καραϊβικής και τον Βόρειο Ατλαντικό. Μέχρι στιγμής διαχειρίζεται τα παρακάτω υπο-δίκτυα (περισσότερες πληροφορίες -> http://www.arin.net/statistics/index.html)

24.0.0.0/8	69.0.0.0/8	76.0.0.0/8	205.0.0.0/8
63.0.0.0/8	70.0.0.0/8	96.0.0.0/8	206.0.0.0/8
64.0.0.0/8	71.0.0.0/8	97.0.0.0/8	207.0.0.0/8
65.0.0.0/8	72.0.0.0/8	98.0.0.0/8	208.0.0.0/8
66.0.0.0/8	73.0.0.0/8	99.0.0.0/8	209.0.0.0/8
67.0.0.0/8	74.0.0.0/8	199.0.0.0/8	216.0.0.0/8
68.0.0.0/8	75.0.0.0/8	204.0.0.0/8	

Το **Africa Network Information Centre (AfriNIC)** είναι υπεύθυνο για τις διευθύνσεις IP στην Αφρική και κομμάτια του Ινδικού ωκεανού, μέχρι στιγμής διαχειρίζεται το 41.0.0.0/8 και διάφορα άλλα υπο-δίκτυα μικρότερα από /8 (περισσότερεςπληροφορίες -> http://www.afrinic.net/statistics/index.htm).

Το **Latin American and Caribbean Internet Addresses Registry (LACNIC)** είναι υπεύθυνο για τις διευθύνσεις IP στην Λατινική Αμερική και την Καραϊβική και μέχρι στιγμής διαχειρίζεται τα παρακάτω υπο-δίκτυα (περισσότερες πληροφορίες -> http://lacnic.net/en/sobre-lacnic/reporte/index.html)

189.0.0.0/8	190.0.0.0/8	200.0.0.0/8	201.0.0.0/8

Τέλος το δικό μας, το **Réseaux IP Européens Network Coordination Center (RIPE NCC)** είναι υπεύθυνο για τις διευθύνσεις IP στην Ευρώπη, την Μέση Ανατολή, την Κεντρική Ασία και το τμήμα της Αφρικής βόρεια του Ισημερινού. Μέχρι στιγμής διαχειρίζεται τα παρακάτω υπο-δίκτυα (περισσότερες πληροφορίες -> https://www.ripe.net/ripe/docs/ripe-ncc-managed-address-space.html)

62.0.0.0/8	82.0.0.0/8	90.0.0.0/8	196.200.0.0/13
77.0.0.0/8	83.0.0.0/6	91.0.0.0/8	212.0.0.0/7
78.0.0.0/7	84.0.0.0/7	193.0.0.0/8	217.0.0.0/8
80.0.0.0/8	88.0.0.0/8	194.0.0.0/7	

Τις διευθύνσεις στους RIRs τις δίνει η IANA (Internet Assigned Numbers Authority) που είναι η αρχή που ορίζει όλους τους αριθμούς που χρησιμοποιούνται στα διάφορα πρωτόκολλα του διαδικτύου (τις διευθύνσεις IPv4 θα τις βρείτε εδώ -> http://www.iana.org/assignments/ipv4-address-space). Η IANA στην αρχή ήταν ένας άνθρωπος, ο "πατέρας του Internet" J. Postel που ήταν και αυτός που οργάνωνε τα RFCs (ο πρώτος RFC-editor).

Έτσι τώρα αν έχουμε μια IP στο 80.0.0.0/8, έστω την 80.1.2.3 που ανήκει στο 80.1.2.0/24, και θέλουμε να στείλουμε κάτι στην IP 88.1.2.4 που ανήκει στο 88.1.2.0/24 το μόνο που χρειάζεται να ξέρουν οι γύρω δρομολογητές είναι πώς απ' το 80.1.2.0/24 θα πάμε στο 80.0.0.0/8, από εκεί και πέρα αναλαμβάνουν οι δρομολογητές που ανήκουν στο 88.0.0.0/8 να συνεχίσουν.

3.2.5.Διευθύνσεις Broadcast

Είπαμε στην αρχή όταν μιλάγαμε με τον Γιάννη στο τηλέφωνο ότι θα μπορούσαμε να μιλήσουμε με όλη του την οικογένεια αν θέλαμε με ανοιχτή ακρόαση, θα μπορούσαμε επίσης να μιλήσουμε μόνο με 2 μέλη της οικογένειάς του κλπ. Στο δια-δίκτυο υπάρχουν ανάλογες ανάγκες επικοινωνίας και ήδη πριν αναφερθήκαμε σιωπηλά σε μία. Είπαμε λοιπόν ότι οι διάφοροι δρομολογητές στο δια-δίκτυο "ανακοινώνουν" ο καθένας στους υπόλοιπους τα υπο-δίκτυα τα οποία διαχειρίζονται, μέχρι τώρα όμως έχουμε μιλήσει μόνο για επικοινωνία μεταξύ 2 άκρων (unicast), τι γίνεται όμως όταν κάποιος θέλει να "ανακοινώσει" κάτι σε ομάδα τερματικών (και οι δρομολογητές είπαμε ότι είναι σαν τερματικά, απλά δεν έχουν επίπεδο μεταφοράς κλπ) ? Όταν θέλει να "ανακοινώσει" κάτι σε ένα ολόκληρο δίκτυο ?

Ας ξεκινήσουμε απ' το δεύτερο που είναι εύκολο. Οι περισσότεροι από εσάς θα έχετε παρατηρήσει πως όταν συνδέεστε σε ένα "δίκτυο Windows (εδώ μπαίνει το trademark)" κατευθείαν (υποτίθεται) βρίσκετε τους υπόλοιπους υπολογιστές του δικτύου και μπορείτε να ανταλλάξετε αρχεία κλπ μεταξύ σας. Στην ουσία ο υπολογιστής σας ανακοινώνει στους υπόλοιπους υπολογιστές του δικτύου ένα μύνημα "ποιος είναι εδώ ?" κι αυτοί ανταποκρίνονται. Θα μπορούσαμε να πούμε ότι αυτό μπορεί να γίνει στέλνοντας από ένα πακέτο IP στον καθένα με διεύθυνση αποστολέα την δικιά μας και διεύθυνση παραλήπτη κάθε φορά την κατάλληλη, όταν μπαίνουμε όμως στο δίκτυο δεν ξέρουμε ποιοι άλλοι είναι μέσα για να τους στείλουμε "συστημένο" πακέτο. Έτσι υπάρχει μια ειδικής χρήσης διεύθυνση IP γι' αυτή τη δουλειά που μόλις στείλουμε κάτι εκεί, πάει σε όλους τους υπολογιστές του εκάστοτε υπο-δικτύου, αυτή η IP λέγεται **"Broadcast"** και είναι πάντα η τελευταία IP του υπο-δικτύου. Αν δηλαδή στο παραπάνω σενάριο ήμασταν στο Call Center με τα 2000 τερματικά και θέλαμε να πούμε σε όλα τους "κάντε διάλειμμα" πχ. θα στέλναμε ένα πακέτο με το εν λόγο μήνυμα στην τελευταία διεύθυνση του υπο-δικτύου, δηλαδή την 10.47.23.255.

Ένας εύκολος τρόπος για να υπολογίζουμε την διεύθυνση Broadcast σε ένα δίκτυο είναι να κάνουμε λογικό OR (θυμίζω 1 OR 0 = 1, 1 OR 1 = 1, 0 OR 0 =0) την subnet address με την inverted subnet mask (την subnet mask που οι άσσοι έχουν γίνει 0 και το ανάποδο -που της έχει γίνει NOT-). Έτσι για να βρούμε την Broadcast address του υπο-δικτύου 10.47.16.0/21 κάνουμε τα παρακάτω...

	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
	10								47								16								0							
	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
|	0								0								7								255							
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1
	10								**47**								**23**								**255**							
	0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Είναι προφανές ότι η Broadcast όπως και η subnet address δεν μπορεί ούτε κι αυτή να χρησιμοποιηθεί σε τερματικό. Έτσι μετά το /32 δίκτυο που δεν περιέχει τερματικά (είπαμε παραπάνω τι είναι), γίνεται σαφές πως ούτε το /31 δίκτυο μπορεί να έχει τερματικά αφού έχει μόνο 2 διευθύνσεις IPs και οι δύο πλέον χρησιμοποιούνται, η πρώτη για subnet address και η δεύτερη ως Broadcast. Οπότε το /31 δίκτυο δεν υπάρχει και δεν θα δείτε ποτέ υπό φυσιολογικές συνθήκες subnet mask με 31 άσσους (κάποιοι παραβλέπουν το πρότυπο και βάζουν κανονικά την Broadcast σε τερματικό, εσείς γυρίστε τους την πλάτη και κάντε το σωστό, όπως λέει το πρωτόκολλο). Άρα όλα τα υπο-δίκτυα που θα κόβετε από εδώ και πέρα θα πρέπει να ξεκινούν απο /30 τουλάχιστον (4 IPs σύνολο = 2 τερματικα + subnet address + broadcast).

Εκτός απ' την διεύθυνση Broadcast κάποιου υπο-δικτύου, υπάρχει και άλλη μια διεύθυνση που λέγεται **limited Broadcast** και είναι για να στέλνουμε ανακοινώσεις στους υπολογιστές του **τοπικού** μας δικτύου, η 255.255.255.255 (δεν είναι subnet mask αυτή τη φορά αλλά κανονική διεύθυνση IP). Ένα παράδειγμα χρήσης της limited Broadcast είναι όταν θέλουμενα βρούμε κάποιον printer στο τοπικό μας δίκτυο ή ακόμα καλύτερα όταν συνδεόμαστε στο τοπικό μας δίκτυο και ζητάμε απ' τον DHCP Server να μας δώσει μια διεύθυνση IP (ειδικά στο δεύτερο που δεν ξέρουμε καν την διεύθυνση του υπο-δικτύου στο οποίο βρισκόμαστε, η linited broadcast είναι μονόδρομος). Όπως έχουμε την limited broadcast που αναφέρεται σε "αυτό το δίκτυο" έχουμε και την αντίστοιχη subnet address, την 0.0.0.0/0 (δεν έχει κανέναν άσσο στην subnet mask).

Γενικώς η subnet address ενός υπο-δικτύου είναι για να χαρακτηρίζει ότι έρχεται απ' το δίκτυο αυτό, ενώ η Broadcast ότι απευθύνεται σε αυτό. Γι' αυτό και τη subnet address θα τη βλέπετε στο πεδίο source address ενώ την broadcast στο πεδίο destination address.

3.2.6.Διευθύνσεις Multicast

Πάμε τώρα στην πρώτη περίπτωση που θέλουμε να μιλήσουμε όχι με όλους αλλά με μια ομάδα τερματικών (ή δρομολογητών), στο τοπικό μας δίκτυο ή και στο δια-δίκτυο. Για να το πετύχουμε αυτό δεσμεύεται για κάθε ομάδα τερματικών μια ειδική διεύθυνση IP (που ανήκει στο παλιό Class D, δηλαδή το 224.0.0.0/4) η οποία μπορεί να χρησιμοποιηθεί σαν destination address για να στείλουμε ένα μήνυμα σε όλα τα τερματικά της ομάδας.

Την διεύθυνση αυτή τη λέμε Multicast και ακολουθεί κι αυτή κάποιους κανόνες (επειδή το θέμα αυτό αγγίζει το κομμάτι της δρομολόγησης, δεν θα το αναλύσω περαιτέρω σε αυτή τη φάση)...

α) Local Network Control Block (224.0.0.0/24)
Οι διευθύνσεις σε αυτή την περιοχή (πλέον δεν μιλάμε για υπο-δίκτυα, κάθε διεύθυνση είναι ένα group τερματικών, οπότε εδώ αναφερόμαστε στο πλήθος των group, το εύρος δηλαδή των διευθύνσεων) χρησιμοποιούνται για ομάδες τερματικών που βρίσκονται στο ίδιο υπο-δίκτυο. Όλα τα πακέτα που έχουν ως destination address μία από αυτές τις διευθύνσεις δεν δρομολογούνται απ' τους δρομολογητές σε άλλα δίκτυα ότι TTL κι αν έχουν.

Για παράδειγμα μπορεί σε αυτό το εύρος διευθύνσεων να έχουμε τις παρακάτω διευθύνσεις - ομάδες:

224.0.0.1 -> Όλα τα τερματικά του υπο-δικτύου (αντίστοιχη δηλαδή της limited broadcast)
224.0.0.2 -> Όλοι οι δρομολογητές που υπάρχουν (συνορεύουν) σε αυτό το υπο-δίκτυο
224.0.0.5-6 -> Χρησιμοποιείται για τις ανακοινώσεις των δρομολογητών που χρησιμοποιούν το OSPF (ένα πρωτόκολλο μεταξύ των δρομολογητών για να συντονίζονται καλύτερα μεταξύ τους στις μεταβολές του δικτύου)
224.0.0.9 -> Χρησιμοποιείται για τις ανακοινώσεις των δρομολογητών που χρησιμοποιούν RIP (ένα άλλο πρωτόκολλο για τον ίδιο σκοπό -που όμως δουλεύει διαφορετικά)

β) Internetwork Control Block (224.0.1.0/24)
Οι διευθύνσεις σε αυτό το εύρος χρησιμοποιούνται για ομάδες τερματικών που ξεφεύγουν απ' τα όρια ενός υπο-δικτύου, για παράδειγμα η ομάδα με διεύθυνση 224.0.1.1 χρησιμοποιείται απ' το SNTP (Simple Network Time Protocol) για να ανακοινώνει την ώρα σε όλα τα μέλη της ομάδας, ενώ η ομάδα με διεύθυνση 224.0.1.41 χρησιμοποιείται για τις ανακοινώσεις των H323 Gateways (βλ. VoIP).

Οι παρακάτω ομάδες έχουν άλλες λειτουργίες που δεν μας αφορούν σε αυτή τη φάση αλλά τις αναφέρω εδώ να υπάρχουν.

γ) AD-HOC Block (224.0.2.0/24 – 224.0.255.0/24)
Οι διευθύνσεις σε αυτό το εύρος χρησιμοποιούνται από εφαρμογές που δεν ανήκουν σε κάποια απ' τις παραπάνω κατηγορίες και συνήθως αναφέρονται σε μικρού μεγέθους ομάδες (μικρότερες από ένα /24 δίκτυο).

δ) SDP/SAP Block (224.2.0.0/16)
Οι διευθύνσεις σε αυτό το εύρος χρησιμοποιούνται από εφαρμογές που παίρνουν διεύθυνση IP χρησιμοποιώντας το Session Announcement Protocol, καθώς και για το MBONE (βλ παρακάτω στο IGMP). Για να μην μπερδεύεστε χρησιμοποιούνται για εφαρμογές multimedia όπως IPTV κλπ (δείτε κι εδώ -> http://www.mythtv.org/wiki/index.php/IPTV).

ε) Source Specific Multicast Block (232.0.0.0/8)
Οι διευθύνσεις σε αυτό το εύρος χρησιμοποιούνται για ομάδες τερματικών που μπορούν να επιλέξουν ποια από τα μηνύματα που στέλνονται σε όλα τα μέλη της ομάδας θέλουν να λαμβάνουν και ποια όχι (δηλαδή σε αυτές τις ομάδες υπάρχουν διάφορα τερματικά που στέλνουν πακέτα γενικώς και τα υπόλοιπα μέλη διαλέγουν ποια από τα πακέτα θα κρατήσουν και ποια όχι, ξεφεύγει δηλαδή απ' την προηγούμενη λογική της "ανακοίνωσης")

στ) GLOP Block (233.0.0.0/8)
Αυτό παραείναι περίπλοκο για να το εξηγήσω σε αυτή τη φάση οπότε το αφήνω, googlίστε το :P

ζ) Administratively Scoped Address Block (239.0.0.0/8)
Οι διευθύνσεις σε αυτό το εύρος χρησιμοποιούνται για ομάδες που θέλει να φτιάξει ο κάθε administrator στον τομέα (domain) που διαχειρίζεται, μπορεί δηλαδή να περιλαμβάνουν τερματικά από όλα τα υπο-δίκτυα που διαχειριζόμαστε και γι' αυτό τον λόγο δεν υπάρχουν standard ομάδες εδω πέρα, μας δίνεται η ελευθερία να κάνουμε ότι θέλουμε.

Καλά μέχρι εδώ, πώς δημιουργούνται όμως οι ομάδες ? Πώς δηλαδή μπορούμε να δημιουργήσουμε μια ομάδα τερματικών και πώς μπορούμε κατόπιν να συμμετάσχουμε σε μια ήδη υπάρχουσα ? Το θέμα αυτό επειδή είναι λίγο πολύπλοκο και χρειάζεται να μιλήσουμε για ένα βοηθητικό πρωτόκολλο του επιπέδου δια-δικτύου που δουλεύει παράλληλα με το IP, το IGMP που θα αναλύσουμε στη συνέχεια.

3.2.7.Άλλες διευθύνσεις ειδικής χρήσης

Είδαμε ήδη 3 τύπους διευθύνσεων IP που δεν μπορούν να μπουν σε τερματικό γιατί έχουν άλλη λειτουργία στο δίκτυο, είπαμε για τις Broadcast, τις Multicast και την subnet address. Η IANA όμως (βλ. 3.2.4) έχει δεσμεύσει και άλλα εύρη διευθύνσεων IPs που έχουν ειδική χρήση στο διαδίκτυο.

α) Οι private διευθύνσεις, είναι εύρη διευθύνσεων που χρησιμοποιούνται μόνο σε τοπικά δίκτυα και δεδομένα από και προς αυτές τις διευθύνσεις δεν δρομολογούνται στο διαδίκτυο (θα τις δείτε και ως non-routable IPs). Έχουμε λοιπόν τα παρακάτω "prive" υπο-δίκτυα., κατανεμημένα κατά σειρά μεγέθους.

10.0.0.0/8 (αυτό χρησιμοποιούμε στο AWMN)
172.16.0.0/12
192.168.0.0/16 (αυτό χρησιμοποιείται στα περισσότερα τοπικά δίκτυα και θα το δείτε σχεδόν παντού)

β) Η διεύθυνση loopback που αναφέραμε παραπάνω (127.0.0.1/32) και γενικώς όλο το 127.0.0.0/8, διευθύνσεις σε αυτό το εύρος δεν μπορούν να μπουν σε τερματικά.

γ) Το εύρος DHCP Fail (169.254.0.0/16) είναι μια ομάδα διευθύνσεων IP απ' τις οποίες διαλέγεται τυχαία μια σε περίπτωση που ο DHCP server δεν ανταποκρίθηκε στο αίτημά μας. Αν πχ. Πέσει ο DHCP Server του δικτύου σας, τότε όλα τα τερματικά σας θα διαλέξουν αυτόματα μια IP από αυτό το εύρος.

δ) Το εύρος TEST-NET (192.0.2.0/24) χρησιμοποιείται στο documentation και για να μην υπάρχουν μπερδέματα δεν μπορεί να χρησιμοποιηθεί σε τερματικά.

ε) Το εύρος 192.88.99.0/24 χρησιμοποιείται από τους δρομολογητές που υπάρχουν για την μετάβαση στο Ipv6 (6to4 relay routers)

στ) Το εύρος 198.18.0.0/15 χρησιμοποιείται από μηχανήματα δοκιμών για να κάνουν benchmarks στο δίκτυο

ζ) Το εύρος 14.0.0.0/8 τέλος είναι reserved

Επίσης είναι reserved και όλες οι IPs που δεν έχουν διατεθεί ακόμα από την IANA σε κάποιο RIR τις οποίες και λέμε μαζί με όλες τις IPs που δεν επιτρέπεται να βγούν στο δια-δίκτυο "Bogons". Λίστες με Bogons μπορείτε να βρείτε εδώ -> http://www.completewhois.com/bogons/data/

4.ΤΑ ΒΟΗΘΗΤΙΚΑ ΠΡΩΤΟΚΟΛΛΑ

Κάποιες από τις λειτουργίες που περιγράψαμε ως τώρα, όπως πχ. η ειδοποίηση ενός τερματικού ότι το TTL έληξε που είπαμε όταν μιλάγαμε για το TTL και το Traceroute ή το πώς φτιάχνονται κλπ οι ομάδες multicast, ενώ αφορούν έναν μηχανισμό του IP δεν γίνονται από το ίδιο το IP αλλά από κάποια βοηθητικά πρωτόκολλα που συνυπάρχουν στο επίπεδο δια-δικτύου (το επίπεδο 3 σύμφωνα με το OSI).

Έτσι για την αναφορά σφαλμάτων και γενικότερα για τον έλεγχο της επικοινωνίας σε επίπεδο δια-δικτύου έχουμε το Internet Control Message Protocol (ICMP), για την δημιουργία και την διαχείριση των ομάδων multicast υπάρχει το Internet Group Management Protocol (IGMP) και τέλος για το ταίριασμα των διευθύνσεων IP σε διευθύνσεις του τοπικού φυσικού δικτύου, έχουμε το Address Resolution Protocol (ARP).

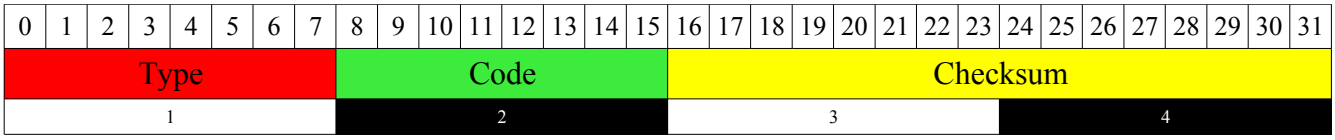
Ας τα δούμε λοιπόν με τη σειρά...

4.1.To Internet Control Message Protocol (ICMP)

Το ICMP χρησιμοποιείται για να στέλνουμε διάφορα μηνύματα που αφορούν αναφορά λαθών, ενδείξεις για την κατάσταση του δικτύου, καθώς και κάποιες υποδείξεις για την δρομολόγηση των δεδομένων. Αν και η μονάδα πληραφορίας του (PDU) γίνεται encapsulate στο IP με αριθμό πρωτοκόλλου 1 (protocol = 1), το ίδιο δεν αποτελεί πρωτόκολλο του επιπέδου μεταφοράς αφού δεν χρησιμοποιείται απευθείας από τις εφαρμογές. Η μόνη εφαρμογή που χρησιμοποιεί απευθείας το ICMP (που δημιουργεί δηλαδή κατευθείαν μηνύματα ICMP χωρίς την μεσολάβηση του επιπέδου μεταφοράς) είναι το γνωστό μας "ping" που χρησιμοποιείται για να δούμε αν ένα τερματικό υπάρχει και πόσο καλά αποκρίνεται. Τα μηνύματα που λαμβάνονται, επεξεργάζονται στο επίπεδο 3 κι αν κάποιο χρειάζεται να πάει στην εφαρμογή, παρακάμτεται και πάλι το επίπεδο μεταφοράς και η PDU πάει κατευθείαν στο επίπεδο εφαρμογής (έτσι γίνεται με το Traceroute το οποίο δουλεύει λαμβάνοντας συγκεκριμένα ICMP μηνύματα όταν λήγει το TTL -θα δούμε παρακάτω τι μηνύματα). Αφού το ICMP δεν ανήκει στο επίπεδο μεταφοράς, δεν εγγυάται ούτε αυτό την ασφαλή μετάδοση των πληροφοριών και όπως και το IP, τα διάφορα μηνύματα ICMP μπορεί να χαθούν στο δρόμο (έτσι καταλαβαίνουμε και το packet loss όταν κάνουμε ping -βλ. παρακάτω).

4.1.1.Η μονάδα πληροφορίας (PDU) και τα μηνύματα του ICMP

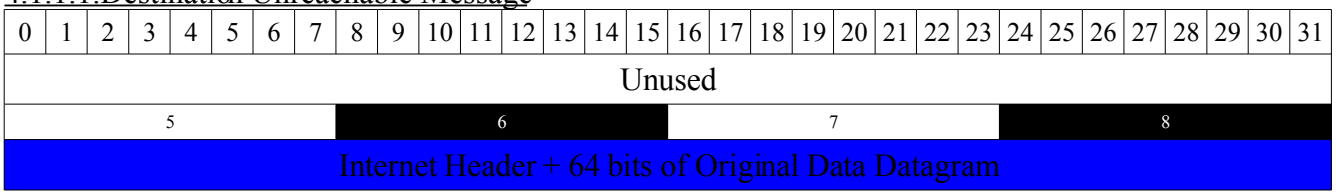
Όλα τα μηνύματα ICMP έχουν την παρακάτω κεφαλίδα...



...και κατόπιν ακολουθεί μια δομή που ορίζεται απ' το πεδίο Type που χαρακτηρίζει τον τύπο του μηνύματος. Το Code ορίζει το περιεχόμενο του μηνύματος και το Checksum είναι το αντίστοιχο του Header Checksum στο IP.

Έτσι έχουμε τους εξής τύπους μηνυμάτων:

4.1.1.1.Destination Unreachable Message



Τα μηνύματα αυτά χρησιμοποιούνται απ' τους δρομολογητές για να δηλώσουν την αδυναμία παράδοσης του πακέτου στον παραλήπτη του και χρησιμοποιούν το Type 3.

Υπάρχουν τα εξής μηνύματα (Codes):

0 = net unreachable

Το συγκεκριμένο υπο-δίκτυο δεν υπάρχει στον πίνακα του δρομολογητή ή έχει βγει εκτός λειτουργίας (έχει άπειρο metric όπως λέμε -το metric χοντρικά είναι μια τιμή πουχαρακτηρίζει την απόσταση/κατάσταση σύνδεσης κλπ για το κάθε υπο-δίκτυο που "βλέπει" ο δρομολογητής, όσο ποιό μικρό είναι τόσο περισσότερο "προτιμάται" τα πακέτα να περάσουν από αυτό το υπο-δίκτυο).

1 = host unreachable

Ο δρομολογητής του υπο-δικτύου στο οποίο υποτίθεται ότι ανήκει ο παραλήπτης δεν μπορεί να του παραδώσει το πακέτο, δεν μπορεί δηλαδή να τον βρει στο υπο-δίκτυο.

2 = protocol unreachable

Το πακέτο παραδόθηκε αλλά το πρωτόκολλο το οποίο κάνει encapsulate δεν είναι διαθέσιμο στο συγκεκριμένο αποστολέα.

3 = port unreachable

Το πακέτο παραδόθηκε αλλά η συγκεκριμένη port (θα δούμε τι είναι αυτό όταν μιλήσουμε για το TCP και το UDP, ας πούμε ότι είναι το νούμερο που χαρακτηρίζει την κάθε εφαρμογή που χρησιμοποιεί αυτά τα δύο πρωτόκολλα-ένας τρόπος να δηλώσουμεγια ποια εφαρμογή προορίζεται το πακέτο δηλαδή) δεν είναι διαθέσιμη.

4 = fragmentation neededand DF set

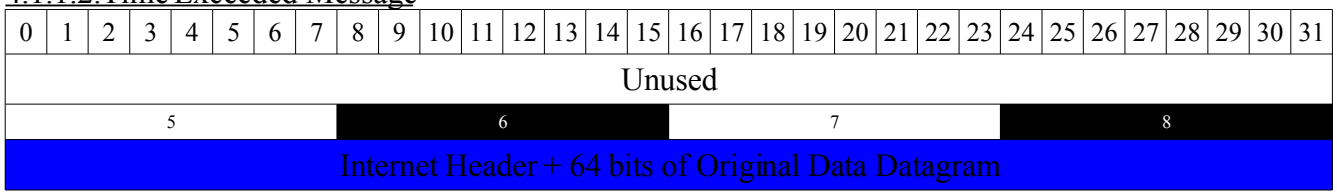
Ενώ έχουμε ορίσει στο πακέτο να μεταφερθεί αυτούσιο χωρίς να σπάσει σε fragments (έχουμε θέσει δηλαδή το Don't Fragment -DF- flag), χρειάζεται από εδώ και πέρα ο κερματισμός του για να μεταφερθεί.

5 = source route failed

Κάποιες επιλογές που μπαίνουν στο πεδίο Options του IP (συγκεκριμένα οι επιλογές που αφορούν το strict/loose source routing) επιτρέπουν στον αποστολέα να ορίσει στους δρομολογητές την διαδρομή (από ποιους δρομολογητές δηλαδή περνάει το πακέτο) που θέλει να ακολουθήσει το πακέτο για να φτάσει στον παραλήπτη. Αν δεν είναι δυνατόν να ακολουθηθεί αυτή η διαδρομή τότε αποστέλλεται αυτό το μήνυμα.

Τα μηνύματα 0,1,4 και 5 παραδίδονται σε δρομολογητές (και κατόπιν στον αποστολέα), ενώ τα μηνύματα 2 και 3 στον αποστολέα. Μετά απ' την κεφαλίδα ακολουθούν 32bit κενά (0 δηλαδή) και κατόπιν η κεφαλίδα IP και τα 64 πρώτα bits του πακέτου το οποίο ήταν η αιτία.

4.1.1.2.Time Exceeded Message



Τα μηνύματα αυτά έχουν Type 11 και δηλώνουν τα παρακάτω (Codes):

0 = time to live exceeded in transit

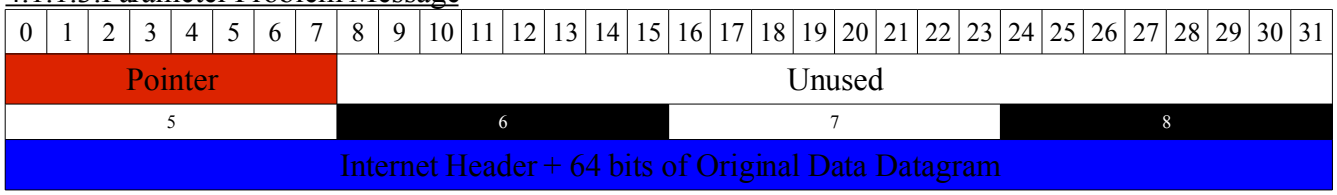
Το TTL έληξε (έγινε 0) κατά την μεταφορά του πακέτου και άρα το πακέτο απορρίφθηκε απ' τον δρομολογητή (αυτό χρησιμοπαιείται απ' το Traceroute)

1 = fragment reassembly time exceeded

Ο παραλήπτης δεν συναρμολόγησε το πακέτο εγκαίρως (δεν μάζεψε δηλαδή όλα τα fragments ή άργισε να τα βάλει στη σειρά κλπ).

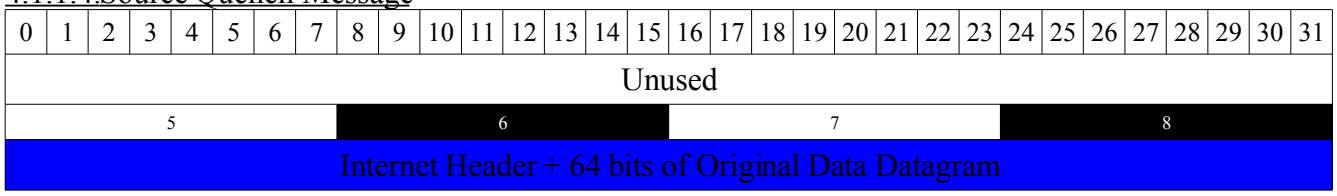
Κι εδώ μετά απ' την κεφαλίδα ακολουθούν 32bit κενά (0 δηλαδή) και κατόπιν η κεφαλίδα IP και τα 64 πρώτα bits του πακέτου το οποίο ήταν η αιτία. Το Code 0 στέλνεται σε δρομολογητή (και κατόπιν στον αποστολέα) ενώ το Code 1 στέλνεται κατευθείαν στον αποστολέα.

4.1.1.3.Parameter Problem Message



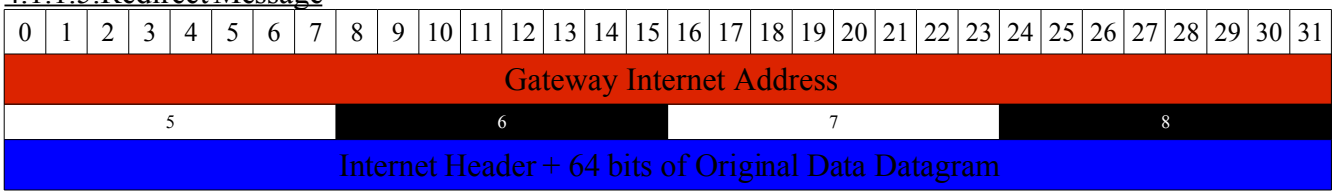
Αυτό είναι ένα μήνυμα που στέλνεται απ' τους δρομολογητές σε περίπτωση που βρουν κάποιο λάθος στην κεφαλίδα του πακέτου που παραλαμβάνουν. Έχει Type 12 και Code 0 και στο επόμενο octet μετά την κεφαλίδα (πεδίο Pointer) περιλαμβάνεται ο αριθμός του octet της κεφαλίδας του αρχικού πακέτου (βλ. τα μαυρόασπρα στην κεφαλίδα IP) στο οποίο εντοπίστηκε το λάθος. Όπως και παραπάνω μεσολαβούν 24bit (32 – 8 που είναι το octet με τον αριθμό του προβληματικού octet της κεφαλίδας) και κατόπιν η κεφαλίδα IP και τα 64 πρώτα bits του πακέτου το οποίο ήταν η αιτία.

4.1.1.4.Source Quench Message



Το μήνυμα αυτό έχει Type 4 και Code 0 και δηλώνει ότι κατά την παραλαβή του πακέτου ήταν αδύνατη η επεξεργασία του απ' το τερματικό ή τον δρομολογητή (στέλνεται δηλαδή και από δρομολογητές και από τερματικά). Στην περίπτωση του δρομολογητή σημαίνει ότι η "ουρά" των πακέτων που έχει προς αποστολή έχει γεμίσει και στην περίπτωση του τερματικού ότι τα πακέτα έρχονται με μεγάλη ταχύτητα και δεν μπορεί να τα επεξεργαστεί. Το μήνυμα αυτό σε αντίθεση με τα παραπάνω δεν σημαίνει παράλληλα ότι το πακέτο αυτό δεν έφτασε, μπορεί να σταλεί και ως προειδοποίηση.

4.1.1.5.Redirect Message



Στην περίπτωση που στο υπο-δίκτυο στο οποίο βρισκόμαστε υπάρχουν περισσότεροι από ένας δρομολογητές και το πακέτο το οποίο στέλνουμε μπορεί να φτάσει γρηγορότερα από κάποιον άλλο απ' τους δρομολογητές αυτούς, τότε ο δρομολογητής στον οποίο στείλαμε το πακέτο, μας στέλνει ένα τέτοιο μήνυμα με Type 5 για να μας πει να στέλνουμε τα πακέτα μας από εδώ και πέρα στον δρομολογητή του υπο-δικτύου μας που έχει καλύτερη ταχύτητα και κατόπιν προωθεί το πακέτο μας. Αυτός ο μηχανισμός ενώ φαίνεται πολύ λογικός αποτελεί ένα πρώτης τάξεως κενό ασφαλείας, αφού μπορεί κάποιος να προσποιηθεί τον δρομολογητή στο υπο-δίκτυό μας, να μας στείλει "πειραγμένα" τέτοια μηνύματα κι έτσι όλη η κίνησή μας να περνάει μέσα από αυτόν, επιτρέποντάς του να υποκλέψει τα δεδομένα μας. Για τον λόγο αυτό συνιστάται να απορρίπτονται τέτοια μηνύματα στο τερματικό σας, εκτός αν ξέρετε καλά τι κάνετε.

Τα μηνύματα αυτά μπορείνα σημαίνουν τα παρακάτω (Codes)

0 = Redirect datagrams for the Network.
Λέει στον αποστολλέα να στέλνει στον συγκεκριμένο δρομολογητή όλα τα πακέτα που προορίζονται για το συγκεκριμένο υπο-δίκτυο.

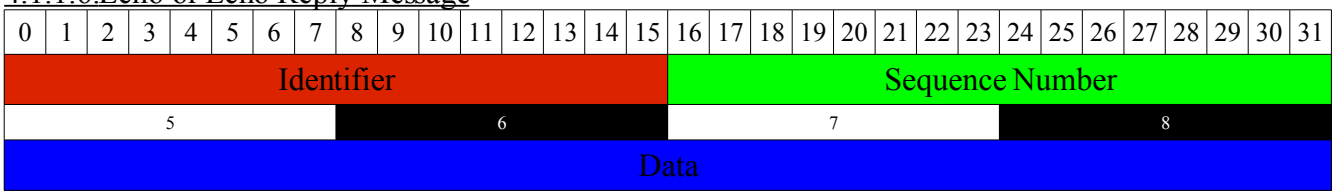
1 = Redirect datagrams for the Host.
Λέει στον αποστολέα να στέλνει στον συγκεκριμένο δρομολογητή όλα τα πακέτα που προορίζονται για το συγκεκριμένο παραλήπτη.

2 = Redirect datagrams for the Type of Service and Network.
Λέει στον αποστολέα να στέλνει στον συγκεκριμένο δρομολογητή όλα τα πακέτα που προορίζονται για το συγκεκριμένο υπο-δίκτυο και έχουν το συγκεκριμένο ToS.

3 = Redirect datagrams for the Type of Service and Host.
Λέει στον αποστολέα να στέλνει στον συγκεκριμένο δρομολογητή όλα τα πακέτα που προορίζονται για το συγκεκριμένο αποστολέα και έχουν το συγκεκριμένο ToS.

Μετά την κεφαλίδα ακολουθείη διεύθυνση του δρομολογητή στον οποίο παραπέμπεται ο αποστολέας (32bit) και κατόπιν η κεφαλίδα IP και τα 64 πρώτα bits του πακέτου που ήταν η αιτία.

4.1.1.6.Echo or Echo Reply Message

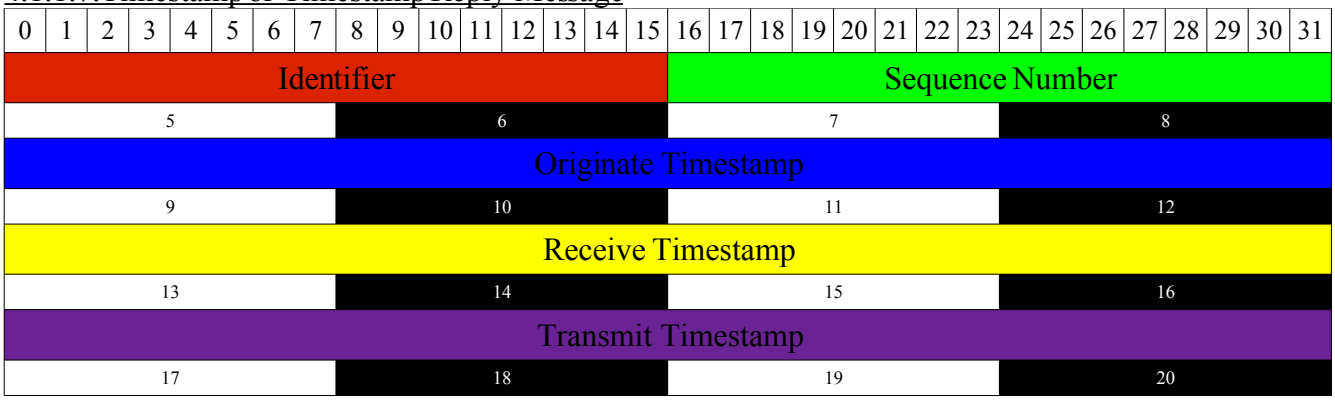


Σκοπός αυτού του τύπου μηνύματος είναι να στέλνουμε δεδομένα σε ένα τερματικό και να μας τα επιστρέφει πίσω, να κάνει δηλαδή echo (αντίλαλο). Το κάθε μήνυμα που στέλνουμε έχει έναν χαρακτηριστικό αριθμό έτσι ώστε να μπορούμε να ταιριάξουμε τα μηνύματα που στέλνουμε με τις απαντήσεις που παίρνουμε που τον λέμε Identifier και είναι 16bit, καθώς και έναν αύξων αριθμό για να

ξέρουμε ποιο μήνυμα προηγείται και ποιο έπεται. Το μήνυμα που στέλνουμε έχει type 8 (echo request) και το μήνυμα που παίρνουμε έχει type 0 (echo reply) και έχει και στις δύο περιπτώσεις code 0. Η δομή της PDU είναι η κεφαλίδα ICMP + 16bits ο Identifier + 16bits ο αύξων αριθμός (sequence number) + τα δεδομένα που θέλουμε να γίνουν echo, το μήνυμα δηλαδή.

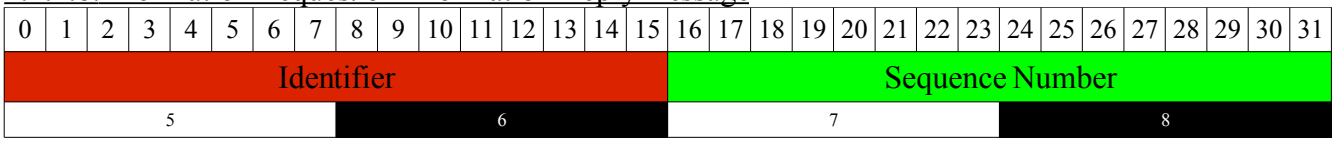
Το γνωστό μας ping στέλνει λοιπόν echo request και περιμένει να λάβει echo reply, στο σώμα του μηνύματος βάζει random δεδομένα όσου μεγέθους του ζητήσουμε (βλ. "ping of death"). Με αυτό τον τρόπο μπορούμε να δούμε αν ένα τερματικό ανταποκρίνεται (είναι UPδηλαδή) και τον χρόνο απόκρισης του δικτύου (συναρτήσει μάλιστα και του μεγέθους των PDU). Από το sequence number τέλος καταλαβαίνουμε πόσα πακέτα χάθηκαν στο δρόμο κι έτσι μετράμε το packet loss.

4.1.1.7. Timestamp or Timestamp Reply Message



Αυτό το μήνυμα είναι σαν το echo μόνο που αντί να βάζουμε ένα οποιοδήποτε μήνυμα μετά το Sequence number, έχουμε 3 standard πεδία των 32bit. Ένα πεδίο στο οποίο ο αποστολέας βάζει την χρονική στιγμή που έστειλε το μήνυμα (το πεδίο Originate Timestamp), ένα πεδίο στο οποίο ο παραλήπτης βάζει την χρονική στιγμή που παρέλαβε το μήνυμα (Receive Timestamp) κι ένα πεδίο που ο παραλήπτης βάζει την χρονική στιγμή που έστειλε πίσω την απάντηση (Transmit Timestamp). Εδώ το Timestamp Request έχει type 13 και το Timestamp Reply έχει type 14.

4.1.1.8. Information Request or Information Reply Message



Κι αυτό το μήνυμα είναι σαν το echo μόνο που εδώ δεν έχουμε καθόλου data μετά το Sequence Number. Ο στόχος αυτού του μηνύματος είναι να τοκάνουμε broadcast, οπότε και θα πάρουμε μια απάντηση από κάποιο τερματικό του δικτύου μας επιτρέποντάς μας να καταλάβουμε σε ποιο υπο-δίκτυο βρισκόμαστε. Το Information Request έχει type 15 και το Information Reply 16.

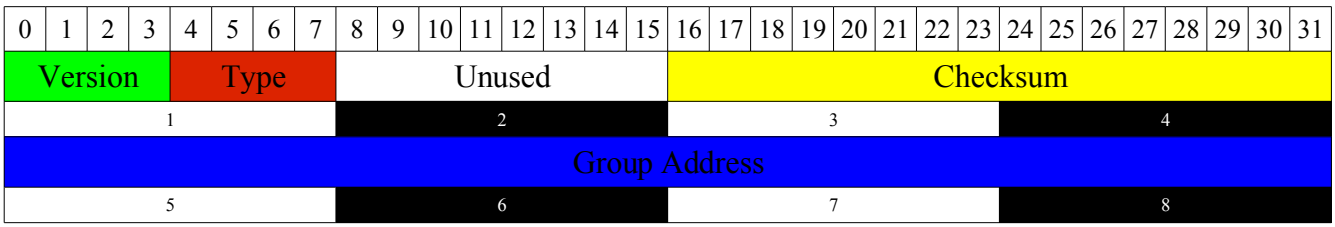
4.2. Internet Group Management Protocol

Το IGMP όπως είπαμε και παραπάνω αλλά και όταν αναφερόμασταν στις διευθύνσεις multicast είναι το πρωτόκολλο για τη διαχείριση των ομάδων multicast. Υπάρχουν 3 εκδόσεις του πρωτοκόλλου (η έκδοση 0 που αναφέρεται στο RFC988 του 1986 δεν βγήκε ποτέ προς τα έξω γι' αυτό δεν θα πούμε τίποτα γι' αυτή, ξεκινάμε απ' την έκδοση 1 που αναφέρεται στο RFC1112 του 1989), οι εκδόσεις 1 και 2 (RFC2236) χρησιμοποιούνται κυρίως ενώ η έκδοση 3 (RFC3376) δεν έχει ακόμα επεκταθεί. Το IGMP όπως και το ICMP γίνεται κατευθείαν encapsulate στο IP με αριθμό πρωτοκόλλου 2 (protocol = 2) και πρόκειται για πρωτόκολλο επικοινωνίας μεταξύ των τερματικών και του τοπικού δρομολογητή που πρέπει να υποστηρίζει multicasting (multicast router ή mrouter) γι' αυτό και τα πακέτα IGMP έχουν TTL = 1 (δεν βγαίνουν δηλαδή έξω απ' το τοπικό υπο-δίκτυο). Το multicasting γενικά δεν είναι κάτι διαδεδομένο στο Internet και δεν είναι πολλοί οι δρομολογητές που το υποστηρίζουν (αυξάνονται με τον καιρό). Για να συνδεθούν μεταξύ τους τα διάφορα υπο-δίκτυα που υποστηρίζουν multicast δημιουργήθηκε το MBONE ή Multicast BackBone, ένα εικονικό δίκτυο το οποίο χρησιμοποιεί το υπο-δίκτυο 224.2.0.0/16 και με τον καιρό, καθώς περισσότεροι δρομολογητές υποστηρίζουν multicast στο internet, μάλλον θα καταργηθεί. Το multicasting χρησιμοποιείται κυρίως για streaming media, όταν δηλαδή θέλουμε να δούμε live ένα video stream πχ. και να γλυτώσουμε όσο το δυνατόν περισσότερο bandwidth. Με το multicast το τερματικό που κάνει streaming δεν χρειάζεται να στέλνει στον καθένα ξεχωριστά τα δεδομένα (unicast) ή να τα στέλνει σε ολόκληρα υπο-δίκτυα το ίδιο (broadcast), κάτι που θα είχε ως αποτέλεσμα το link με το οποίο συνδέεται στο δια-δίκτυο να "μπουκώσει". Αντίθετα την δουλειά αυτή την αναλαμβάνουν οι δρομολογητές multicast που δρουν ως αναμεταδότες (relays) κι έτσι γλιτώνουμε την "κίνηση" (bottleneck) στο δίκτυο και έχουμε καλύτερη real-time αποστολή.

Υπάρχουν 2 ειδών ομάδες, οι μόνιμες και οι προσωρινές, οι μόνιμες έχουν μια standard διεύθυνση και μπορούν ανά πάσα στιγμή να έχουν οποιοδήποτε αριθμό μελών, ακόμα και μηδέν, ενώ οι προσωρινές παίρνουν δυναμικά μια διεύθυνση την ώρα που δημιουργούνται και διαγράφονται όταν σταματήσουν να έχουν μέλη (όταν περάσει κάποιο χρονικό όριο χωρίς να απαντήσουν μέλη).

4.2.1. Η μονάδα πληροφορίας (PDU) και τα μηνύματα του IGMPv1

Ας αρχίσουμε απ' την κεφαλίδα του IGMPv1 που μοιάζει αρκετά με την κεφαλίδα του ICMP, κι εδώ βλέπουμε τα πεδία Type και Checksum που έχουν τον ίδιο ρόλο.



Το πεδίο Version περιέχει την έκδοση του πρωτοκόλλου και το πεδίο Group Address την IP της ομάδας multicast.

Έχουμε τα παρακάτω μηνύματα...

4.2.1.1. Host Membership Query

Αυτό το μήνυμα έχει Type = 1 και στέλνεται απ' τον multicast router στην διεύθυνση 224.0.0.1 (γνωστή και ως "all-hosts group") με TTL όπως είπαμε 1, για να ανακαλύψει ποια τερματικά στο καθένα απ' τα υπο-δίκτυα με τα οποία επικοινωνεί είναι μέλη σε ομάδες multicast. Όταν στέλνεται αυτό το μήνυμα το πεδίο Group Address είναι μηδέν και αγνοείται. Σε κάθε υπο-δίκτυο πρέπει να υπάρχει μόνο ένας

δρομολογητής που να στέλνει Queries (και λέγεται "Querier"), έτσι αν κάποιος δρομολογητής λάβει Query από κάποιο απ' τα υπο-δίκτυά του δεν θα πρέπει να στείλει Query ο ίδιος προς το υπο-δίκτυο αυτό.

4.2.1.2.Host Membership Report

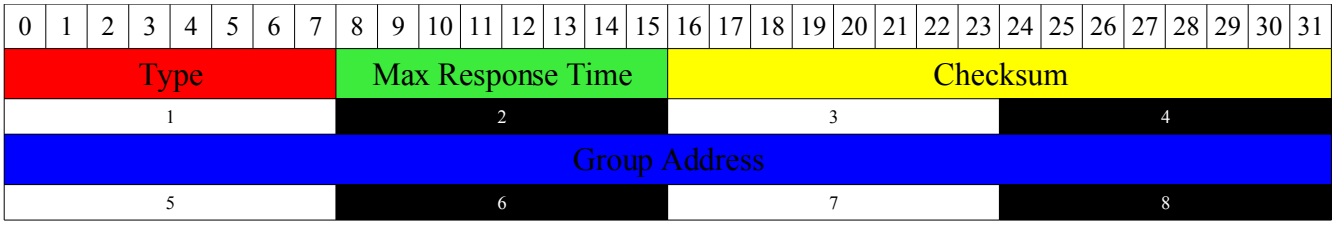
Αυτό το μήνυμα έχει Type = 2 και στέλνεται απ' τα τερματικά στην διεύθυνση multicast της ομάδας τους για να δηλώσουν τη συμμετοχή τους σε αυτή την ομάδα (μπορούν να στείλουν παραπάνω από ένα report αν ανήκουν σε παραπάνω από μία ομάδες). Ο multicast router είναι αυτόματα μέλος σε όλες τις ομάδες άρα λαμβάνει κι ο ίδιος το μήνυμα. Για να είναι έγκυρο το μήνυμα, πρέπει το πεδίο Group Address να περιέχει την ίδια διεύθυνση με αυτή που στάλθηκε το μήνυμα (την διεύθυνση της ομάδας multicast δηλαδή).

Όταν ένα τερματικό θέλει να γίνει μέλος σε μία ομάδα ή να δημιουργήσει μια ομάδα, στέλνει κατευθείαν ένα μήνυμα Membership Report (ακόμα και χωρίς να έχει προηγηθεί Query γι' αυτή την ομάδα). Ο δρομολογητής στέλνει σε τυχαία χρονικά διαστήματα Queries, στην αρχή σχετικά συχνά για να φτιάξει τη λίστα των ομάδων και αργότερα με συχνότητα το πολύ ένα Query ανά λεπτό. Εδώ το πράγμα περιπλέκεται λίγο, **τον δρομολογητή δεν τον ενδιαφέρει ποια μέλη έχει η κάθε ομάδα, αυτό που θέλει να δει με τα Queries είναι ποιες ομάδες έχουν ακόμα μέλη -και άρα είναι ενεργές- σε καθένα απ' τα υπο-δίκτυά του**. Έτσι του αρκεί να απαντήσει για κάθε υπο-δίκτυο ένα μέλος από κάθε ομάδα κάθε φορά για να είναι εντάξει και να διατηρήσει την ομάδα στη μνήμη του. Για το σκοπό αυτό και για να μην γεμίζει το δίκτυο με reports, κάθε τερματικό έχει για κάθε ομάδα στην οποία είναι μέλος έναχρονόμετρο που ξεκινάει από μια τυχαία τιμή μεταξύ 1 και 10 δευτερόλεπτα όταν λαμβάνει το Query απ' τον δρομολογητή, και λήγει στα 10 δευτερόλεπτα. Με αυτό τον τρόπο υπό φυσιολογικές συνθήκες για κάθε ομάδα δεν θα έχουμε ταυτόχρονα reports αλλά θα προηγείται το report του τερματικού που θα λήξει ποιο γρήγορα το χρονόμετρο του ή το report κάποιου τερματικού που μόλις μπήκε στην ομάδα (κι άρα δεν έχει ακόμα χρονόμετρο). Όταν τα υπόλοιπα μέλη της ομάδας λάβουν το report μηδενίζουν τα χρονόμετρά τους κι έτσι δεν στέλνουν ξανά report.

Φαίνεται έτσι πώς ένα τερματικό δημιουργεί μια ομάδα ή γίνεται μέλος σε μια ήδη υπάρχουσα, καθώς και το πώς διατηρούνται οι ομάδες σε καθένα από τα υπο-δίκτυα. Αν στο υπο-δίκτυο δεν υπάρχει δρομολογητής, τα τερματικά μπορούν να δημιουργήσουν ομάδες multicast στέλνοντας απλά reports, ο δρομολογητής είναι εκεί για να κάνει τις ομάδες multicast δια-δικτυακές.

Δεν γίνεται όμως σαφές πώς ένα τερματικό αποχωρεί απ' την ομάδα. Αυτό γίνεται μηδενίζοντας το χρονόμετρό του και μη απαντώντας πλέον σε Queries. Για να φύγει όμως απ' την ομάδα πρέπει να περάσει κάποιος χρόνος στον οποίο σπαταλούνταν πολύτιμο bandwidth. Αυτό ήταν ένα βασικό πρόβλημα για το πρωτόκολλο κι έπρεπε να βρεθεί ένας σωστότερος μηχανισμός για να δηλώσει ένα τερματικό την αποχώρησή του. Έτσι ξεκίνησε η δεύτερη έκδοση του πρωτοκόλλου που επικρατεί σήμερα.

4.2.2.Η μονάδα πληροφορίας (PDU) και τα μηνύματα του IGMPv2



Το πεδίο Type έχει το ίδιο νόημα με το IGMPv1 μόνο που εδώ μεγάλωσε (παρέχεται η συμβατότητα όμως με το IGMPv1 όπως θα δούμε) ενώ τα bits 8-15 πλέον χρησιμοποιούνται για το Max Response

Time που αντικαθιστά την τιμή των 10 δευτερολέπτων που είπαμε παραπάνω και πλέον με κάθε Querry δηλώνεται ο μέγιστος χρόνος απόκρισης (ο χρόνος στον οποίο λήγει το χρονόμετρο) σε δέκατα του δευτερολέπτου.

Έχουμε τα παρακάτω μηνύματα...

4.2.2.1.Membership Query

Αυτό το μήνυμα έχει type 0x11 (17) και είναι παρόμοιο με το Host Membership Query στο IGMPv1 με τη διαφορά ότι εδώ έχουμε τη δυνατότητα έχοντας την Group Address κενή(μηδέν) να κάνουμε ένα γενικό query για όλα τα groups σε κάποιο subnet, ενώ όταν η Group Address αντιστοιχείστην διεύθυνση κάποιας ομάδας Multicast, τότε το query αφορά μόνο αυτή την ομάδα.

4.2.2.2.Version 2 Membership Report

Το αντίστοιχο του Membership Reportστο IGMPv1 μόνο που εδώ έχει type 0x16 (22) + ότι ο χρόνος πλέον εξαρτάται κι από το Max Response Time.

4.2.2.3.Leave Group

Η όλη "μαγκιά" του IGMPv2, αυτό το μήνυμα με type 0x17 (23) δηλώνει στον δρομολογητή την αποχώρηση ενός τερματικού απ' την ομάδα Multicast. Αν το τερματικό δεν ήταν το τελευταίο που έστειλε Membership Report τότε δεν χρειάζεται να στείλει αυτό το μήνυμα (optimization για το traffic μιας και όπως είπαμε παραπάνω τον δρομολογητή τον ενδιαφέρει αν υπάρχουν ή όχι μέλη στο subnet).

Υπάρχουν 2 καταστάσεις για τον δρομολογητή Multicast σε ένα Subnet, μπορεί να είναι Querier, δηλαδή να κάνει Queries στο συγκεκριμένο Subnet και Non-Querier δηλαδή να μην κάνει Queries στο συγκεκριμένο Subnet. Ο προκαθορισμένος ρόλος ενός δρομολογητή Multicast είναι να κάνει τον Querier εκτός αν λάβει στο Subnet κάποιο Querry από κάποιον άλλο δρομολογητή με IP μικρότερη από αυτού. Τότε σταματάει να είναι Querierκαι αναλαμβάνει ο άλλος δρομολογητής αυτό τον ρόλο στο Subnet. Όπως και στο IGMPv1 ο δρομολογητής στέλνει αρκετά Queries για να κάνει την "χαρτογράφηση" των ομάδων multicast στο Subnet και κατόπιν δουλεύει ο ίδιος μηχανισμός με timers μόνο που εδώ αντί για 10 δευτερόλεπτα έχουμε το Max Response Time. Όταν ένα τερματικό τώρα θέλει να αποχωρίσει απ' την ομάδα Multicast σε αντίθεση με το IGMPv1 που έπρεπε να λήξει ο timer, αρκεί να στείλει ένα μήνυμα Leave Group στη διεύθυνση της ομάδας Multicast των δρομολογητών (all-routers group με IP 224.0.0.2).

Επειδή γενικώς το IGMP είναι λίγο περίεργο στους μηχανισμούς του καλό θα ήταν να διαβάσετε και τα RFCs 2236 για το IGMPv2 και 3376 για το τελευταίο (draft) IGMPv3 που περιλαμβάνει και ένα μηχανισμό "επιλεκτικής λήψης" στα πλαίσια μιας ομάδας, πχ. στην περίπτωση που ένα τερματικό είναι μέλος σε μια ομάδα αλλά θέλει να δέχεται πακέτα από συγκεκριμένα μέλη της ομάδας (Source filtering).

4.3. Address Resolution Protocol

Είπαμε στην αρχή όταν μιλήσαμε για διευθυνσιοδότηση ότι υπάρχει η διεύθυνση του φυσικού δικτύου (ο αριθμός του τηλεφώνου στο παράδειγμά μας) καθώς και η διεύθυνση του λογικού δικτύου (το όνομα του ατόμου με τον οποίο μιλάμε) αλλά δεν είπαμε πώς αυτές οι δύο διευθύνσεις αντιστοιχίζονται, δεν είπαμε δηλαδή πώς εμείς μάθαμε το τηλέφωνο του Γιάννη κι αυτός το δικό μας, μια διαδικασία που ορίσαμε ότι γίνεται στο επίπεδο δια-δικτύου (3).

Το ARP λοιπόν αναλαμβάνει να αντιστοιχίσει την διεύθυνση του λογικού δικτύου σε μία διεύθυνση στο φυσικό δίκτυο, θα μπορούσαμε να το παρομοιάσουμε ως το πρωτόκολλο που απαντάει στην ερώτηση "με ποιο τερματικό πρέπει να μιλήσω για να φτάσω στη IP διεύθυνση X ?"

Είναι προφανές ότι τα αιτήματα ARP (τα πακέτα δηλαδή με τις ερωτήσεις) πρέπει να είναι ορατά σε όλα τα τερματικά του φυσικού δικτύου, να γίνονται δηλαδή broadcast στο φυσικό δίκτυο, κι αυτό γιατί δεν ξέρουμε με ποιο απ' όλα τα τερματικά θέλουμε να μιλήσουμε (αυτό ψάχνουμε με το ARP όπως είπαμε). Παρόλο που το φυσικό δίκτυο (Ethernet, 802.11, ATM, FDDI, Tokenring, whatever) δεν εξετάζεται σε αυτό το tutorial αρκεί να ξέρετε ότι όπως υπάρχουν διευθύνσεις broadcast και multicast στο λογικό δίκτυο, υπάρχουν και αντίστοιχες στο φυσικό δίκτυο. Για παράδειγμα στο Ethernet η MAC FF:FF:FF:FF:FF:FF είναι η διεύθυνση broadcast, αντίστοιχη της IP 255.255.255.255 στο λογικό δίκτυο, οπότε όποιο πλαίσιο (η PDU όπως είπαμε του φυσικού δικτύου) αποστέλλεται σε αυτή τη διεύθυνση παραλαμβάνεται από όλα τα τερματικά του φυσικού δικτύου. Άρα για να κλείσουμε τα πακέτα του ARP γίνονται encapsulate σε πλαίσια που αποστέλλονται σε όλα τα τερματικά του φυσικού δικτύου, γίνονται δηλαδή broadcast στο φυσικό δίκτυο κι έτσι τα βλέπουν όλοι.

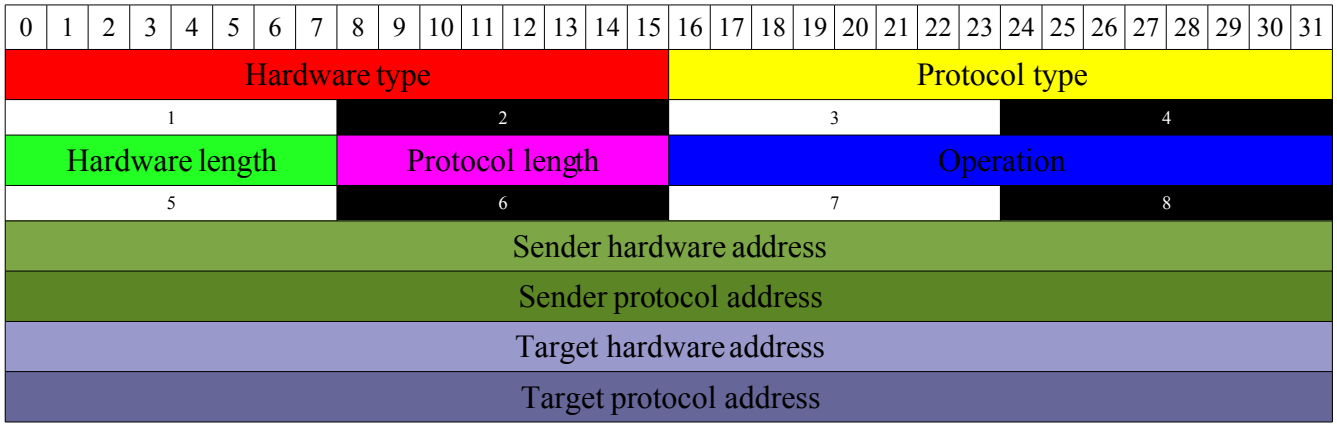
ΠΡΟΣΟΧΗ! Το ARP σε αντίθεση με το ICMP και το IGMP **δεν γίνεται encapsulate στο IP**, είναι ένα ανεξάρτητο πρωτόκολλο που λειτουργεί παράλληλα με το IP γι' αυτό και αρκετές φορές θα δείτε να το κατατάσουν όχι στο επίπεδο 3 (δια-δικτύου) αλλά στο επίπεδο "2.5" με την έννοια ότι παρέχει μια υπηρεσία στο επίπεδο 3 αλλά δεν ανήκει στο επίπεδο 2 (φυσικού δικτύου).

Με τη χρήση του ARP, το κάθε τερματικό στο φυσικό δίκτυο δημιουργεί μια λίστα με τις αντιστοιχίσεις διευθύνσεων φυσικού και λογικού δικτύου (σαν ένα τηλεφωνικό κατάλογο), για να μη χρειάζεται κάθε φορά να ζητάει την ίδια πληροφορία (αφού δεν αναμένεται να αλλάζει αυτή η αντιστοιχία στο δίκτυο σε τακτά χρονικά διαστήματα). Τη λίστα αυτή τη λέμε ARP Cache και απ' τη στιγμή που δημιουργηθεί μια καταχώρηση παραμένει στη λίστα για 2 λεπτά. Για κάθε αίτημα που χρησιμοποιεί την καταχώρηση αυτή (για κάθε αναζήτηση στον τηλεφωνικό κατάλογο του ίδιου ονόματος) η καταχώριση ανανεώνεται (χωρίς να σταλεί νέο μήνυμα ARP) για άλλα 2 λεπτά κι αν αυτό συνεχιστεί επιτυχώς για 10 λεπτά η καταχώριση ανανεώνεται υποχρεωτικά με την αποστολή ενός νέου μηνύματος ARP.

Είναι δυνατόν αν θέλουμε να βάλουμε μια στατική καταχώριση στην ARP Cache κάποιου τερματικού χωρίς να χρησιμοποιηθεί το ARP σε περίπτωση που πχ. Θέλουμε να δώσουμε "με το ζόρι" μια IP σε ένα άλλο τερματικό που δεν έχει πάρει ακόμα IP ή που δεν ξέρουμε τι IP έχει πάρει (κλασικό παράδειγμα όταν έχουμε ξεχάσει την default IP ενός router/managed switch κλπ). Τότε αν γνωρίζουμε τη MAC του τερματικού με το οποίο θέλουμε να μιλήσουμε μπορούμε να αντιστοιχίσουμε σε αυτό μια IP της αρεσκείας μας και να επικοινωνούμε μαζί του με αυτό τον τρόπο (βέβαια μετά θα πρέπει να το ρυθμίσουμε γιατί δεν είναι και ο ποιο φυσιολογικός τρόπος όπως καταλαβαίνετε, αφού η καταχώριση αυτή δεν θα είναι εμφανής και στο υπόλοιπο δίκτυο).

Να το δούμε λίγο ποιο αναλυτικά...

4.3.1. Η μονάδα πληροφορίας και τα μηνύματα του ARP



Το πεδίο Hardware type περιέχει έναν ακέραιο που χαρακτηρίζει τον τύπο του φυσικού δικτύου, για Ethernet πχ. η τιμή του είναι 1.

Το πεδίο Protocol type περιέχει έναν αριθμό που χαρακτηρίζει το πρωτόκολλο που χρησιμοποιείται στο λογικό δίκτυο, για το IPv4 που μελετάμε εμείς πχ. η τιμή του είναι 0x0800.

Τα πεδία Hardware length και Protocol length περιέχουν έναν ακέραιο που περιγράφει πόσα octets είναι η διεύθυνση του φυσικού και του λογικού δικτύου αντίστοιχα στα οποία αναφερόμαστε. Για το Ethernet η διεύθυνση είναι 6octets (48bits) ενώ για το IP 4octets.

Το πεδίο Operation περιέχει έναν ακέραιο και ορίζει αν το πακέτο που αποστέλλουμε είναι ερώτηση (1) ή απάντηση (2).

Όταν γίνεται ένα αίτημα ARP (Operation = 1) το πακέτο ARP γίνεται broadcast όπως είπαμε στο φυσικό δίκτυο. Περιέχει στο πεδίο Sender hardware address τη διεύθυνση στο φυσικό δίκτυο του αποστολέα, στο πεδίο Sender protocol address την διεύθυνση στο λογικό δίκτυο του αποστολέα, και τέλος στο πεδίο Target protocol address τη διεύθυνση λογικού δικτύου του παραλήπτη. Προφανώς η διεύθυνση φυσικού δικτύου του παραλήπτη είναι άγνωστη άρα 0.

Το τερματικό που έχει την Target protocol address παραλαμβάνει όπως και όλα τα υπόλοιπα τερματικά του φυσικού δικτύου το πακέτο ARP και απαντάει, αυτή τη φορά όχι με broadcast αλλά συγκεκριμένα στον αποστολέα (αφού ξέρει την διεύθυνσή του στο φυσικό δίκτυο), επιστρέφοντας ένα πλήρες αυτή τη φορά πακέτο ARP (με Operation = 2), αντιστρέφοντας τις διευθύνσεις αποστολέα/παραλήπτη (αφού αυτή τη φορά άλλος είναι ο αποστολέας και άλλος ο παραλήπτης).

ARP Anouncements (Gratuitous ARP)

Σε περίπτωση που θέλουμε να δημιουργήσουμε μια καταχώρηση στην ARP Cache όλων των τερματικών του φυσικού δικτύου (πχ. για να τους ενημερώσουμε για την είσοδο ενός νέου τερματικού στο φυσικό δίκτυο), υπάρχει η δυνατότητα να στείλουμε ένα αίτημα ARP στον εαυτό μας (δηλαδή ένα πακέτο ARP όπου Sender protocol address = Target protocol address). Το πακέτο αυτό θα γίνει broadcast στο φυσικό δίκτυο κι έτσι όλοι θα ξέρουν την Sender hardware address κι άρα θα συμπληρώσουν ανάλογα την ARP Cache τους.

ΠΑΡΑΡΤΗΜΑ

Α) Endianness ή γιατί γράφουμε ανάποδα...

Όπως σας είπα και παραπάνω το τι είναι ίσια και τι ανάποδα όταν μιλάμε για την σειρά των bits και των bytes είναι κάτι σχετικό. Ας ξεκινήσουμε από την παραδοχή ότι κάθε αριθμός έχει δύο άκρα, ας ξεχάσουμε λίγο την έννοια της αρχής και του τέλους (ας πούμε ότι ο αριθμός είναι σαν το λουκάνικο που έχει 2) κι ας ξεχωρίσουμε την τιμή και την ουσία του αριθμού από την αναπαράστασή του, έτσι η αναπαράστασή του αριθμού 1234

1234

έχει τα άκρα 1 και 4 με το πρώτο άκρο (1) να συμβολίζει τις χιλιάδες και το τελευταίο (4) τις μονάδες. Σκεφτείτε τι θα γινόταν αν το πρώτο άκρο ενός αριθμού συμβόλιζε τις μονάδες και το τελευταίο τις χιλιάδες, τότε ο αριθμός

1234

θα γραφόταν το ίδιο αλλά θα είχε διαφορετική τιμή...

Χιλιάδες	***Εκατοντάδες***	***Δεκάδες***	***Μονάδες***	
1	2	3	4	
1000+	200+	30+	4 =	1234

Μονάδες	***Δεκάδες***	***Εκατοντάδες***	***Χιλιάδες***	
1	2	3	4	
1+	20+	300+	4000 =	4321

Θα μπορούσαμε να παρομοιάσουμε τώρα την τάξη του κάθε ψηφίου της αναπαράστασης του αριθμού (μονάδες,χιλιάδες κλπ) με τις θέσεις (στοιχειώδη κομμάτια -βλ. όταν μιλάγαμε για τα octets) μνήμης. Έτσι έστω ότι ο αριθμός 1234 έμπαινε στη μνήμη στην θέση 100, αν ακολουθούσε την πρώτη κατάταξη θα ήταν...

Θέση	***100***	***101***	***102***	***103***
Τιμή	1	2	3	4

ενώ αν ακολουθούσε την δεύτερη κατάταξη θα ήταν...

Θέση	***100***	***101***	***102***	***103***
Τιμή	4	3	2	1

Στην πρώτη περίπτωση δηλαδή η θέση μνήμης 100 αντιστοιχεί στις χιλιάδες , ενώ στην δεύτερη περίπτωση στις μονάδες. Όταν ξεκινάμε απ' το "μεγάλο" άκρο λοιπόν λέμεότι γράφουμε με την μέθοδο Big Endian (συντομογραφία για το big end in first -note: στα αγγλικά η λέξη end σημαίνει και "τέλος" οπότε μην το μλέκετε, στην περίπτωσή μας είπαμε σημαίνει "άκρο"-)ενώ όταν ξεκινάμε απ' το "μικρό" τέλος λέμε ότι γράφουμε με την μέθοδο Little Endian.

Οι x86 επεξεργαστές χρησιμοποιούν την μέθοδο Little Endian για να γράφουν/διαβάζουν δεδομένα, όπως γράφουμε δηλαδή και στο tutorial, ενώ κάποιοι άλλοι την μέθοδο Big Endian ή και σε κάποιες περιπτώσεις διάφορες παραλλαγές (mixed endian κλπ). Αν λοιπόν τα δεδομένα που θέλουμε να επεξεργαστούμε είναι γραμμένα με άλλη μέθοδο από αυτή που χρησιμοποιεί ο επεξεργαστής, πρέπει να τα μετατρέψουμε κατάλληλα, ακόμα καλύτερα θα πρέπει να ορίσουμε από πριν με ποια απ' τις δύο μεθόδους έχουν γραφτεί τα δεδομένα για να ξέρουμε πώς θα τα διαχειριστούμε ανάλογα με το σύστημα που διαθέτουμε.

Ας δούμε τώρα πώς αυτό επηρεάζει την IP και την subnet mask...

Η PDU του IP λοιπόν όπως ορίζει το RFC791 είναι γραμμένη με την μέθοδο Big Endian, έτσι η διεύθυνση 10.47.132.65 στην πραγματικότητα γράφεται...

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
10								47								132								**65**							
0	0	0	1	0	0	1	0	0	0	1	0	1	1	1	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1

Δηλαδή το bit που αντιστοιχεί στην μεγαλύτερη δύναμη του 2 ($2^7 = 128$) έρχεται πρώτο, ενώ στην μνήμη ενός x86 γράφεται με την μικρότερη δύναμη του 2 ($2^0 = 1$) πρώτα. Επέλεξα αυτό τον τρόπο γιατί θεωρώ ότι ποιο πολύ "περίεργο" θα φαινόταν να ξεκινάμε απ' την μεγαλύτερη δύναμη του 2 παρά με την μικρότερη, με την λογική ότι διαβάζουμε στην γλώσσα μας απ' τα αριστερά προς τα δεξιά. Επίσης το διάλεξα για να σας κεντρίσω το ενδιαφέρον ;-).

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
10								47								132								**65**							
0	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0

Έτσι το παράδειγμα με την subnet mask γίνεται...

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
255								255								255								**248**							
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**

..και λύνεται το μυστήριο.

Αν θέλετε να το ψάξετε περισσότερο (επειδή δεν είναι και τόσο απλό όσο φαίνεται) κοιτάξτε και τα παρακάτω URL...

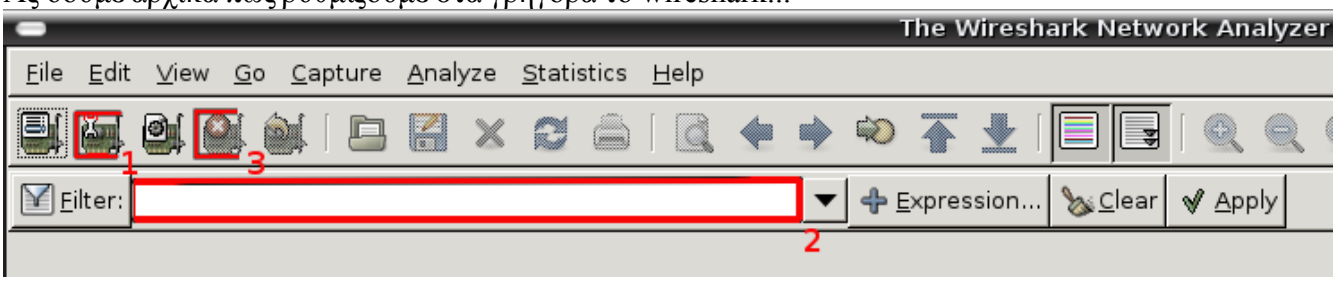
http://en.wikipedia.org/wiki/Little-endian
http://www.noveltheory.com/TechPapers/endian.asp

B) Από τη θεωρία στη πράξη, network data analysis και Wireshark...

Σε αυτό το παράρτημα θα δούμε λεπτομερώς μερικές τυπικές καθημερινές διαδικασίες στο δίκτυο χρησιμοποιώντας ένα απ' τα καλύτερα εργαλεία ανάλυσης της κίνησης στο δίκτυο (φυσικά είναι ελεύθερο λογισμικό), το πασίγνωστο Ethereal που πλέον είναι γνωστό ως Wireshark.

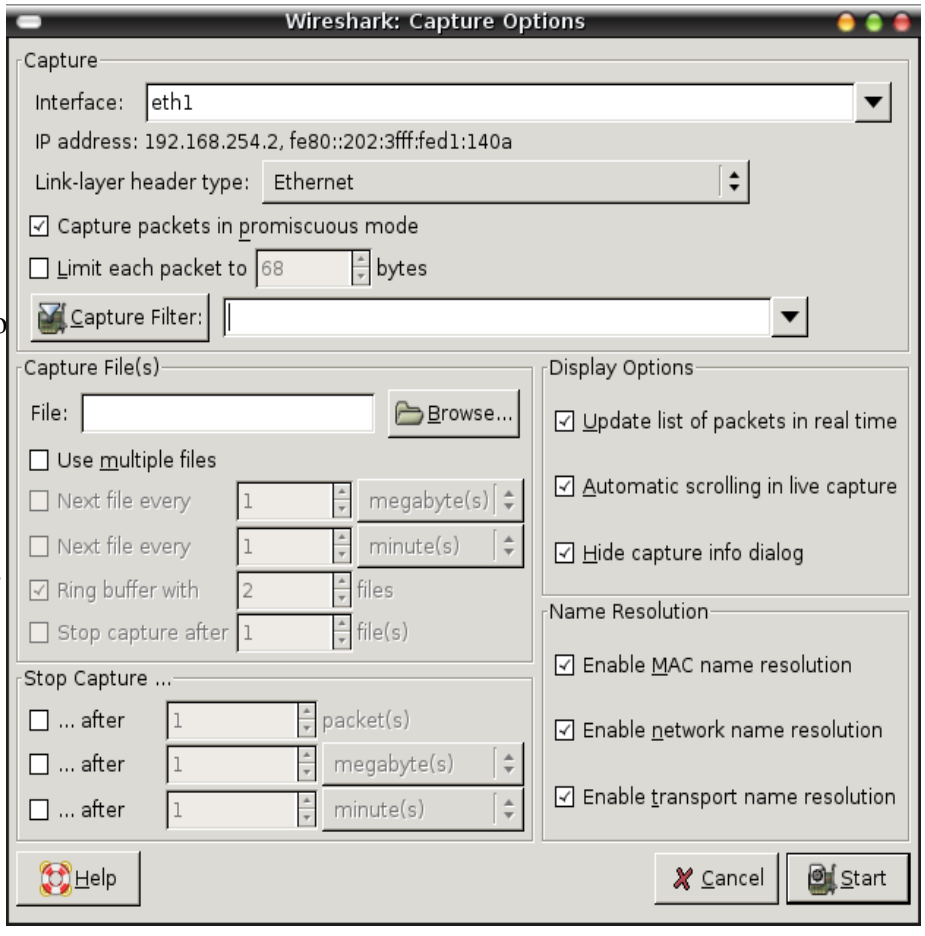
Το Wireshark είναι ένα πρόγραμμα που καταγράφει κάθε πλαίσιο που φτάνει στην κάρτα δικτύου μας (επίπεδο 2 δηλαδή) και το αναλύει ώστε να φαίνεται τι κάνει encapsulate μέχρι και το επίπεδο εφαρμογής. Ξεδιπλώνει δηλαδή τους "φακέλους" και μας δείχνει σε κάθε επίπεδο τι γίνεται encapsulate σε πραγματικό χρόνο, χωρίς να παρεμβαίνει στην ροή των δεδομένων (η κάρτα μας μπορεί να λειτουργεί κανονικά). Το Wireshark υποστηρίζει μερικές εκατοντάδες πρωτόκολλα κι έτσι είναι πολύ εύκολο να καταλάβετε τι γίνεται στο δίκτυό σας, ακόμα κι αν δεν γνωρίζετε την δομή του κάθε πρωτοκόλλου.

Ας δούμε αρχικά πώς ρυθμίζουμε στα γρήγορα το wireshark...



Θα αγγίξουμε για τις ανάγκες αυτού του tutorial μόνο δύο σημεία του Wireshark, όπως φαίνεται και στο σήμα. Το πρώτο είναι οι ρυθμίσεις που αφορούν στην κάρτα (1) και το δεύτερο τα φίλτρα που μπορούμε να εφαρμόσουμε κατά την καταγραφή δεδομένων (2).

Ξεκινάμε λοιπόν πατώντας στο εικονίδιο 1 για να ρυθμίσουμε τις παραμέτρους της καταγραφής...



Στη καρτέλα Capture βλέπουμε την κάρτα δικτύου μας και τις παραμέτρους της. Το ενδιαφέρον εδώ είναι το chekbox "Capture packets in promiscuous mode", που ρυθμίζει την κάρτα δικτύου μας κατά τέτοιο τρόπο ώστε να μας δίνει και τα πλαίσια τα οποία δεν προορίζονται για την ίδια (δεν έχουν ως παραλήπτη την δικιά μας διεύθυνση φυσικού δικτύου). Αυτό βέβαια έχει μόνο νόημα αν η κάρτα μας είναι συνδεδεμένη σε κάποιο hub ή στην θύρα "uplink" του switch μας γιατί κανονικά ένα switch στέλνει τα δεδομένα που προορίζονται για κάποια διεύθυνση φυσικού δικτύου, μόνο στο ανάλογο μπριζάκι (έχει κι αυτό κάτι σαν ARP Cache). Αν σας ενδιαφέρει να το ψάξετε περισσότερο, googlίστε τα "switch",

“promiscuous mode” και αν σας ενδιαφέρουν και οι μέθοδοι παράκαμψης ψάξτε και για “ARP poisoning”. Το πεδίο Capture filter είναι παρόμοιο με το (2) αλλά αντί το φίλτρο να εφαρμάζεται εκ' των υστέρων στα δεδομένα που έχουμε καταγράψει (θα το δούμε παρακάτω), εφαρμόζεται στα δεδομένα πριν καταγραφούν απ' το πρόγραμμα.

Στην καρτέλα “Display Options” και “Name Resolution” τα τσεκάρουμε όλα, οι επιλογές σημαίνουν:

α) “Update list of packets in real time” -> εμφάνιση των πακέτων στη λίστα σε πραγματικό χρόνο (αυτή η επιλογή υπάρχει γιατί ενδεχομένως να υπάρχει μεγάλη κίνηση οπότε κάνουμε disable το συγκεκριμένο και τα δεδομένα φαίνονται στη λίστα αφού σταματήσει η καταγραφή).

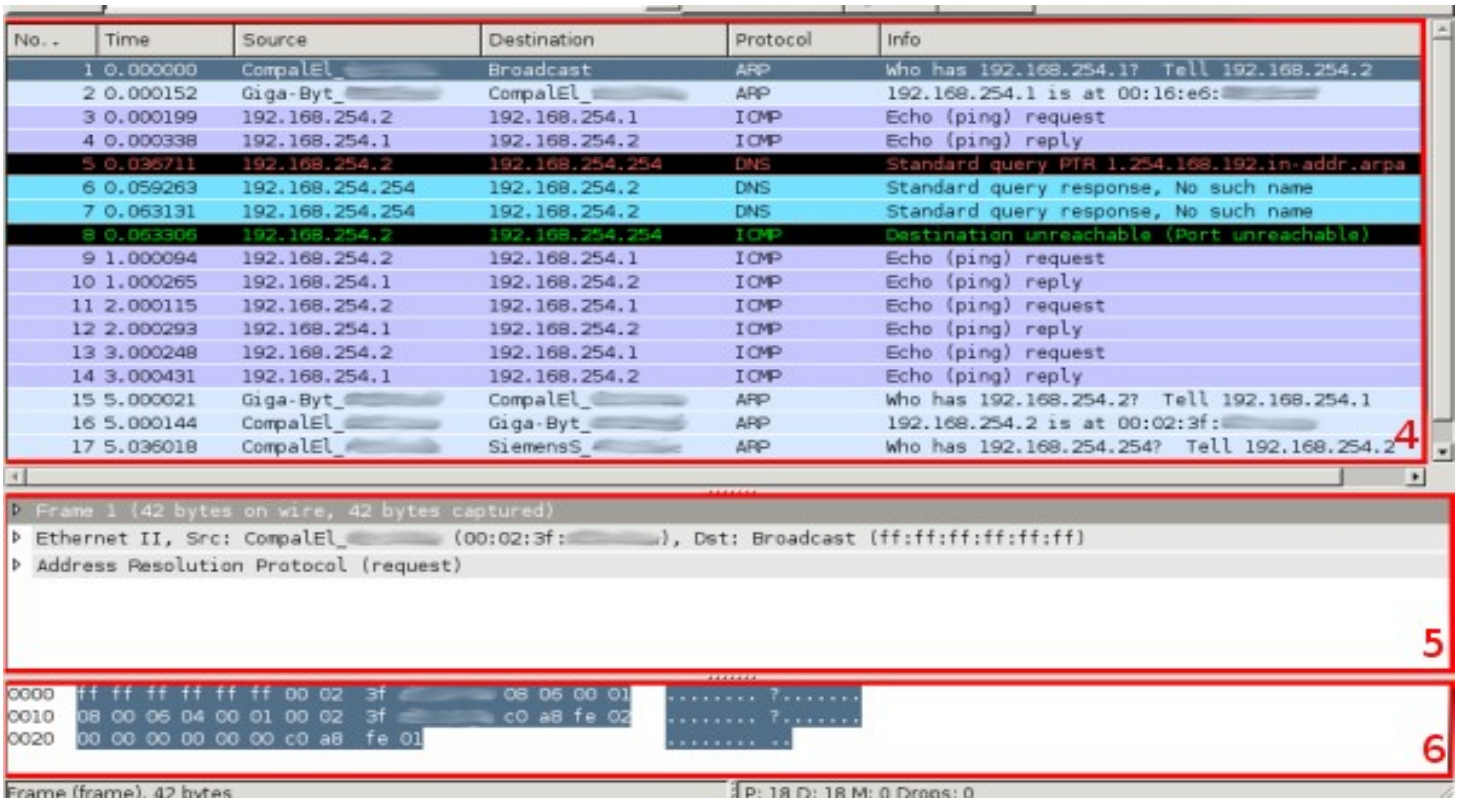
β) “Automatic scrolling in live capture” -> αυτόματη κύλιση κατά την καταγραφή (αυτή η επιλογή εμφανίζεται εφόσον έχετε επιλέξει το α και “κατεβαίνει” αυτόματα τη λίστα κάθε φορά που καταγράφεται ένα πακέτο).

γ) “Hide capture info dialog” -> απόκρυψη του παραθύρου με τα στατιστικά της καταγραφής (αυτό το παράθυρο δείχνει την επί τοις εκατό κατανομή των πακέτων ανάλογα με το πρωτόκολλό το οποίο ακολουθούν, το κάνουμε hide για να μη το έχουμε μπροστά μας και να μπορούμενα δούμε τη λίστα κατευθείαν).

δ) “Enable MAC name resolution” -> αντιστοίχιση των διευθύνσεων φυσικού δικτύου σε ονόματα κατασκευαστών κλπ (ένα κομάτι της διεύθυνσης φυσικού δικτύου είναι χαρακτηριστικό για κάθε κατασκευαστή, με αυτή την επιλογή το πρόγραμμα μας δείχνει από ποιόν -θα το δούμε στη λίστα αργότερα-).

ε) “Enable transport name resolution” -> αντιστοίχιση των διαφόρων χαρακτηριστικών του επιπέδου μεταφοράς σε ονόματα (μεταφράζει πχ. τα port numbers σε πρωτόκολα, επειδή δεν έχουμε αναφερθεί καθόλου σε πρωτόκολλα μεταφοράς σε αυτή τη φάση το αφήνω εδώ και ψάξτε το).

Έχοντας λοιπόν ρυθμίσει τις παραμέτρους καταγραφής πατάμε το κουμπί start και κάνουμε για αρχή ένα ping από τον υπολογιστή μας (192.168.254.2) σε έναν άλλο υπολογιστή του δικτύου (192.168.254.1) και σταματάμε την καταγραφή πατώντας το αντίστοιχο κουμπί (3). Θα έχουμε έτσι ένα αποτέλεσμα σαν το παρακάτω...



Επειδή η διεύθυνση φυσικού δικτύου του υπολογιστή στον οποίο κάνουμε ping δεν υπάρχει στην ARP Cache, το πρώτο πράγμα που πρέπει να γίνει είναι ένα ARP Query στο δίκτυο. Πράγματι οι 2 πρώτες γραμμές στη λίστα με τα πακέτα (4) μας δείχνουν αυτό ακριβώς...

	Time	Source	Destination	Protocol	Info
1	0.000000	CompalEl_	Broadcast	ARP	Who has 192.168.254.1? Tell 192.168.254.2
2	0.000152	Giga-Byt_	CompalEl_	ARP	192.168.254.1 is at 00:16:e6:
3	0.000199	192.168.254.2	192.168.254.1	ICMP	Echo (ping) request
4	0.000338	192.168.254.1	192.168.254.2	ICMP	Echo (ping) reply

Βλέπουμε λοιπόν αρχικά ένα ARP Query από την δικιά μας διεύθυνση φυσικού δικτύου (η MAC addres μας που τα πρώτα τμήματά της "00:02:3F" έχουν αντιστοιχηθεί στον κατασκευαστή "CompalEL" -> Compal Electronics -τα υπόλοιπα τα έχω σβήσει για προφανείς λόγους) προς όλα τα τερματικά του φυσικού δικτύου (Destination -> Broadcast)...

Αν επιλέξουμε το συγκεκριμένο πακέτο, τότε στο πεδίο 5 αν αναπτύξουμε το tree βλέπουμε μια καλύτερη ανάλυση...

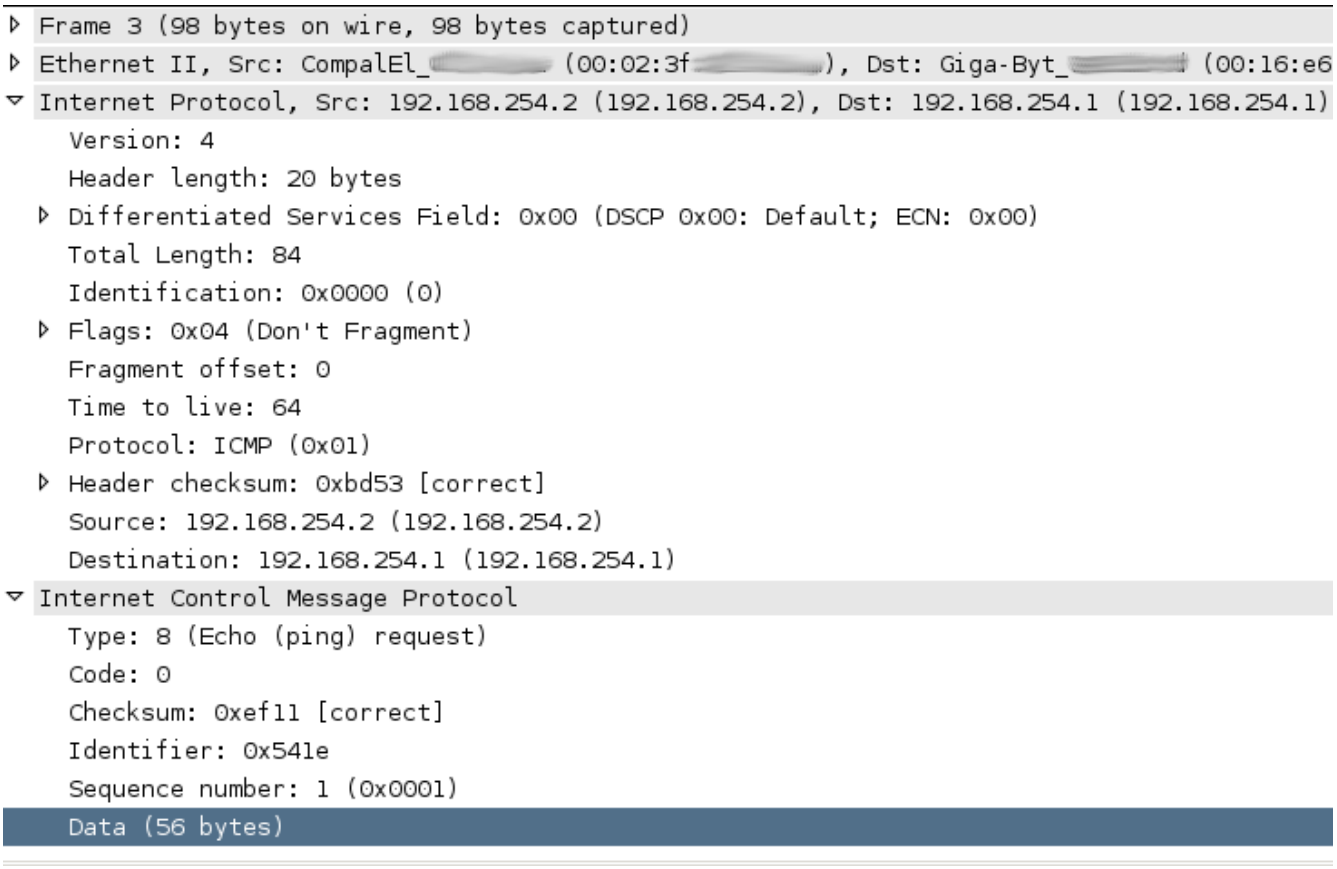
```
▸ Frame 1 (42 bytes on wire, 42 bytes captured)
▸ Ethernet II, Src: CompalEl_ (00:02:3f:), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: CompalEl_ (00:02:3f:)
    Sender IP address: 192.168.254.2 (192.168.254.2)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.254.1 (192.168.254.1)
```

...Βλέπουμε λοιπόν ότι το πρώτο frame (πλαίσιο όπως είπαμε, η PDU του επιπέδου 2) που καταγράφηκε ήταν ένα Ethernet frame (2η γραμμή) που είχε εμάς ως αποστολέα εμάς και ως παραλήπτη όλο το φυσικό δίκτυο (διεύθυνση Broadcast στο Ethernet όπως είπαμε είναι η FF:FF:FF:FF:FF:FF) -δεν χρειάζεται να το αναλύσουμε περισσότερο γιατί εδώ δεν μας ενδιαφέρει το Ethernet- και έκανε encapsulate ένα πακέτο ARP που όπως βλέπουμε και στη δομή του είναι ένα τυπικό ARP request (παρατηρήστε ότι όπως είπαμε η Target MAC Address είναι μηδέν και το opcoode είναι 1). Να δούμε και το reply...

```
▸ Frame 2 (60 bytes on wire, 60 bytes captured)
▸ Ethernet II, Src: Giga-Byt_ (00:16:e6:), Dst: CompalEl_ (00:02:3f:)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    Sender MAC address: Giga-Byt_ (00:16:e6:)
    Sender IP address: 192.168.254.1 (192.168.254.1)
    Target MAC address: CompalEl_ (00:02:3f:)
    Target IP address: 192.168.254.2 (192.168.254.2)
```

...που όπως βλέπουμε έχει opcode 2 και η Source MAC Address είναι συμπληρωμένη (είναι η Source γιατί ο παραλήπτης και ο αποστολέας εναλλάσσονται).

Βλέπετε πώς όλα αυτά που είπαμε παραπάνω για το ARP φαίνονται στην πράξη, ας συνεχίσουμε τώρα με το ping που κάναμε και συγκεκριμένα το ICMP echo request...



...Φαίνονται όλα τα στοιχεία της κεφαλίδας IP που εξηγήσαμε παραπάνω, η διεύθυνση αποστολέα και παραλήπτη στο δια-δίκτυο, τα χαρακτηριστικά του κερματισμού, το Time to live και γενικώς όλη η κεφαλίδα IP. Επίσης βλέπουμε ότι το πακέτο IP γίνεται encapsulate σε ένα Ethernet frame με αποστολέα εμάς (CompalEl) και παραλήπτη το τερματικό που τώρα ξέρουμε μετά το ARP request την διεύθυνσή του στο φυσικό δίκτυο (Giga-Byt -> GigaByte ο κατασκευαστής της κάρτας δικτύου του).

Βλέπουμε επίσης και το ICMP πακέτο που γίνεται encapsulate στο παραπάνω IP πακέτο και πρόκειται για ένα echo request (Type ->8) με sequence number 1 (γιατί είναι το πρώτο πακέτο που φεύγει), identifier όπως είπαμε μια ψευδοτυχαία αλλά μοναδική τιμή και κατόπιν τα τυχαία δεδομένα που ακολουθούν.

Στο πεδίο (6) τώρα μπορείτε να δείτε και τα χύμα δεδομένα όπως παραλήφθηκαν απ' την κάρτα (επίπεδο φυσικού δικτύου δηλαδή, 0 και 1 αλλά στο 16δικό σύστημα αρίθμησης) για να παρατηρήσετε τα δεδομένα που ακολουθούν την κεφαλίδα ICMP και γενικώς το encapsulation των PDUs. Επιλέγοντας στο πεδίο (5) οποιοδήποτε στοιχείο του πλαισίου, αυτό γίνεται αυτόματα highlighted στο πεδίο (6) με τα χύμα δεδομένα.

Και το reply...

```
▷ Frame 4 (98 bytes on wire, 98 bytes captured)
▷ Ethernet II, Src: Giga-Byt_[redacted] (00:16:e6:[redacted]), Dst: CompalEl_[redacted] (00:02:3f
▽ Internet Protocol, Src: 192.168.254.1 (192.168.254.1), Dst: 192.168.254.2 (192.168.254.2)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 84
    Identification: 0x6f2d (28461)
  ▷ Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (0x01)
  ▷ Header checksum: 0x8e26 [correct]
    Source: 192.168.254.1 (192.168.254.1)
    Destination: 192.168.254.2 (192.168.254.2)
▽ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xf711 [correct]
    Identifier: 0x541e
    Sequence number: 1 (0x0001)
    Data (56 bytes)
```

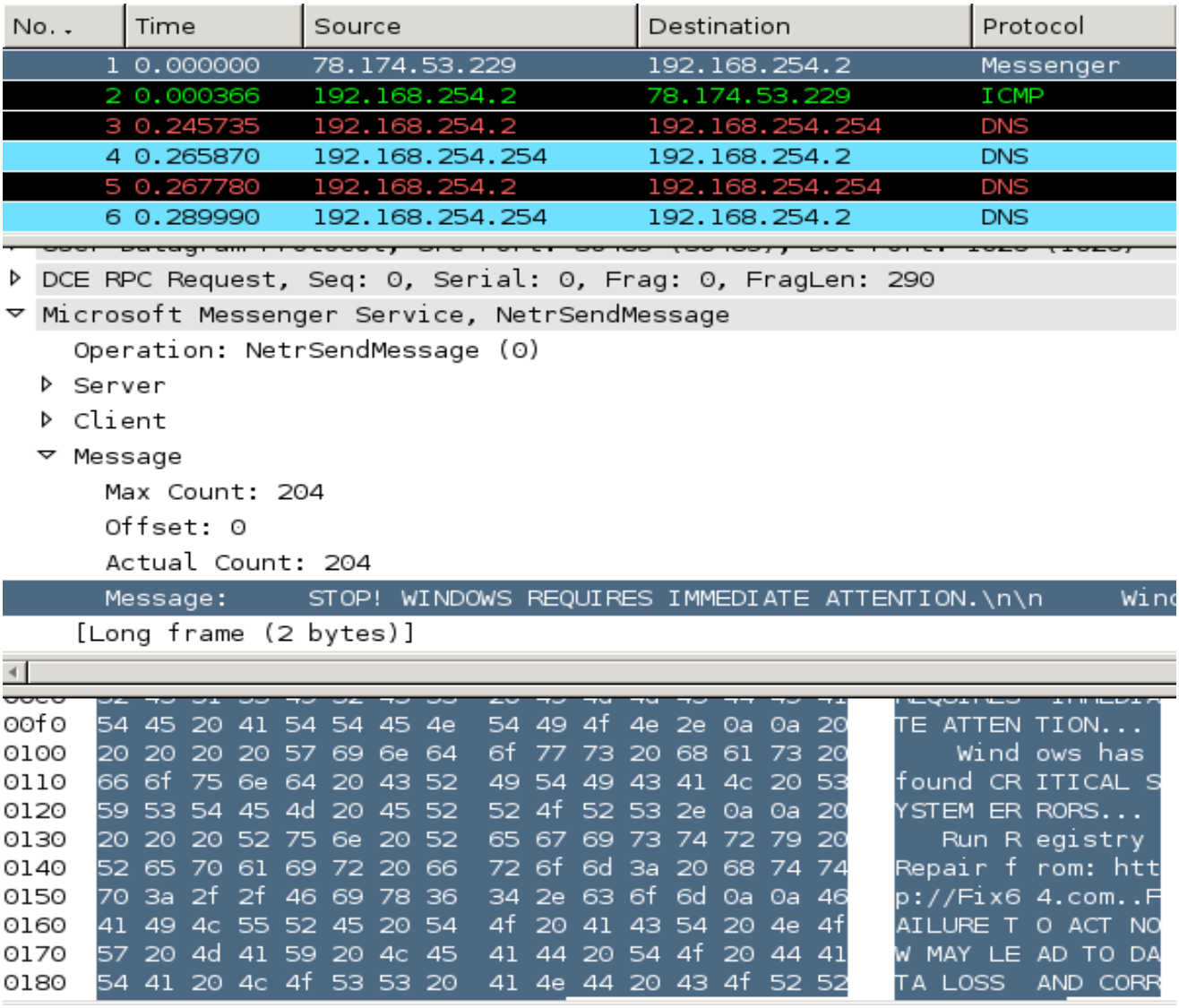
...όπου βλέπουμε τις διευθύνσεις αποστολέα και παραλήπτη να έχουν εναλλαγεί τόσο στο φυσικό δίκτυο όσο και στο δια-δίκτυο, το identifier να είναι το ίδιο, το sequence number είναι επίσης το ίδιο (για να μπορούμε να αντιστοιχίσουμε το request με το reply) και το Type είναι 0.

Ας δούμε τώρα πώς φαίνεται ένα ping με κερματισμό, θα στείλουμε δηλαδή πακέτα μεγαλύτερου μεγέθους απ' το MTU που μπορεί να στείλει το Ethernet (1500octets). Για να το τραβήξουμε στα άκρα κάνουμε ένα ping με το μεγαλύτερο δυνατό μέγεθος για να έχουμε πολλά fragments (η εντολή είναι "ping 192.168.254.1 -s 65507").

```
▷ Frame 45 (429 bytes on wire, 429 bytes captured)
▷ Ethernet II, Src: CompalEl_d1:14:0a (00:02:3f:d1:14:0a), Dst: Giga-Byt_83:3b:6d (00:16:e6
▽ Internet Protocol, Src: 192.168.254.2 (192.168.254.2), Dst: 192.168.254.1 (192.168.254.1)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 415
    Identification: 0x6502 (25858)
  ▷ Flags: 0x00
    Fragment offset: 65120
    Time to live: 64
    Protocol: ICMP (0x01)
  ▷ Header checksum: 0x773a [correct]
    Source: 192.168.254.2 (192.168.254.2)
    Destination: 192.168.254.1 (192.168.254.1)
  ▷ [IP Fragments (65515 bytes): #1(1480), #2(1480), #3(1480), #4(1480), #5(1480), #6(1480)
▷ Internet Control Message Protocol
```

Το ενδιαφέρον είναι να δείτε την λίστα των πακέτων (4) όπου θα φαίνονται όλα τα fragments αλλά και το πρώτο πακέτο να δείτε παραπάνω φαίνεται ότι είναι fragmented, τα fragments που έχουν ακολουθήσει και το offset τους. Βλέπουμε επίσης το Total Length που όπως είπαμε είναι το πλήθος των octets ολόκληρου του πακέτου μετά τη συναρμολόγησή του καθώς και τα διάφορα fragments που το καθένα έχει μέγεθος 1480octets (+20 octets η κεφαλίδα του IP = 1500 octets). Μη σας μπερδεύει το Fragmentation Offset, αν δείτε ένα ένα τα fragments θα δείτε κανονικά αυτό το πεδίο να ορίζεται στη κεφαλίδα κάθε fragment, ανάλογα με τη θέση του fragment στο πακέτο.

Το Wireshark γενικώς είναι για να περάσετε αρκετές ώρες να χαζεύετε το δίκτυο και τα διάφορα πακέτα, θα μάθετε πολλά για το πώς δουλεύουν τα διάφορα πρωτόκολλα και θα έχετε και μια πολύ καλή εικόνα του δικτύου σας σε όλα τα επίπεδα. Γνωρίζοντας το τι πηγαινοέρχεται στο δίκτυό σας μπορείτε μετά να εφαρμόσετε διάφορες πολιτικές, τόσο σχεδιασμού, δρομολόγησης, όσο και ασφάλειας. Αν δε είστε τυχεροί, μπορεί να πέσετε και σε κάτι τέτοιο...



Κλείνοντας να δούμε και λίγο το θέμα των φίλτρων που είπαμε στην αρχή. Αν πάτε στο πεδίο (2) και πληκτρολογήσετε το όνομα ενός πρωτοκόλλου ή μια διεύθυνση IP μπορείτε να φιλτράρετε τη λίστα με τα πακέτα που κατεγράφησαν, τα φίλτρα όμως στο Wireshark είναι ένα μεγάλο κεφάλαιο. Μπορείτε να βάλετε διάφορα κριτήρια φιλτραρίσματος, μπορείτε να χρησιμοποιήσετε κάποιο ήδη καθορισμένο απ τη λίστα (πατήστε στο κουμπί δίπλα απ' το πεδίο (2)) ή και να γράψετε κάποιο δικό σας (βλ. http://www.wireshark.org/docs/).

Κάπου εδώ η μικρή εισαγωγή μας στο TCP/IP τελειώνει, εύχομαι καλές αναζητήσεις και πολύ ψάξιμο και σας ευχαριστώ που κατεβάσατε και διαβάσατε αυτό το Tutorial, ελπίζω να σας άρεσε όσο άρεσε και σε εμένα. Ελπίζω να βρω κάποια στιγμή χρόνο να γράψω και για τα ανώτερα επίπεδα του TCP/IP, το TCP και το UDP καθώς και λίγο περισσότερα πράγματα για δρομολόγηση, CoS και QoS και άλλα καλούδια.

http://www.awmn.net

http://www.hellug.gr

http://www.digitalrights.gr

http://www.gentoo.org

http://www.madwifi.org