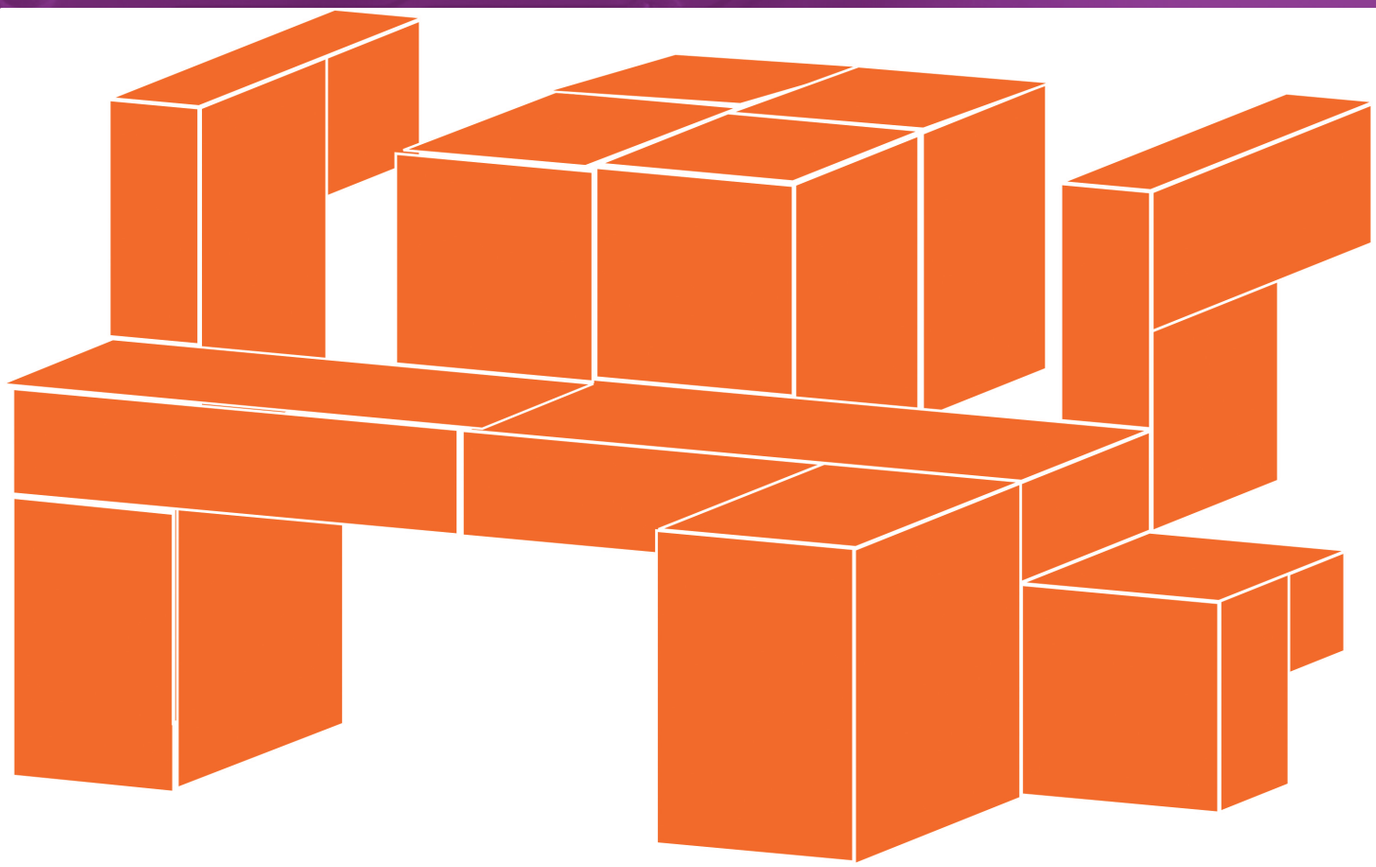
TUTORIAL_TRAFFIC 25-08-08 14:38 Σελίδα 76



ΑΠΑΡΑΙΤΗΤΑ ΕΡΓΑΛΕΙΑ

1. iptables (CONFIG_NETFILTER_XT_TARGET_MARK, CONFIG_NETFILTER_XT_TARGET_CLASSIFY, κ.λπ.)
2. tc, μέρος του πακέτου iproute ή iproute2
3. Πυρήνας της σειράς 2.6 με τα απαραίτητα modules (CONFIG_NET_SCHED, CONFIG_NET_SCH_FIFO, CONFIG_NET_SCH_HTB, CONFIG_NET_SCH_SFQ, CONFIG_NET_SCH_TBF κ.λπ.)

ΕΠΙΜΕΛΕΙΑ: ΓΙΩΡΓΟΣ ΚΑΡΓΙΩΤΑΚΗΣ, ΑΠΟΛΛΩΝ ΟΙΚΟΝΟΜΟΠΟΥΛΟΣ

Ο Γιώργος Καργιωτάκης είναι (wannabe) μαθηματικός, εργάζεται στο R&D τμήμα μίας εταιρείας σχετικά με VoIP και ως freelance system administrator. Είναι μέλος της ομάδας ILUG και διατηρεί blog στη διεύθυνση http://void.gr/kargig/. Ο Απόλλων Οικονομόπουλος είναι Μηχανολόγος Μηχανικός, υποψήφιος διδάκτορας στο Εθνικό Μετσόβιο Πολυτεχνείο και μέλος της Κοινότητας Ελεύθερου Λογισμικού του ΕΜΠ.

ΟΔΗΓΟΙ ΓΙΑ ΠΡΟΧΩΡΗΜΕΝΟΥΣ

Traffic Shaping με Linux

Εχετε ADSL; Ακολουθήστε τις οδηγίες μας για να εκμεταλλευτείτε τη σύνδεσή σας στο έπακρο!

Με την έλευση των ευρυζωνικών συνδέσεων, οι δυνατότητες του οικιακού χρήστη πολλαπλασιάστηκαν. Ομως, η αύξηση του διαθέσιμου εύρους ζώνης (bandwidth) συνοδεύτηκε από την παράλληλη αύξηση των απαιτήσεων των δικτυακών εφαρμογών. Η μετάδοση φωνής και εικόνας σε πραγματικό χρόνο, τα γρήγορα "κατεβάσματα", οι διαδραστικές Web εφαρμογές και τα on-line παιχνίδια έχουν τις δικές τους απαιτήσεις ταχυτήτων και αποκρισιμότητας και πολλές φορές αντιμάχονται το ένα το άλλο για τη χρήση της οικιακής γραμμής μας.

Μία νέα πρόκληση γεννιέται, λοιπόν, για τον οικιακό χρήστη: πώς μπορεί να αξιοποιήσει στο βέλτιστο το εύρος ζώνης της σύνδεσής του, εξασφαλίζοντας ότι αυτό θα καταμερίζεται λογικά ανάμεσα στις εφαρμογές. Η απάντηση ακούει στον όρο Quality of Service (Ποιότητα Υπηρεσίας), ο οποίος περιλαμβάνει τις έννοιες Traffic Shaping (Διαμόρφωση Κίνησης) και Queueing (Ελεγχος Ουρών).

Ταχύτητα και συμφορήσεις στο TCP/IP

Πριν προχωρήσουμε στο τεχνικό σκέλος του άρθρου, κρίνεται σκόπιμη μία απλή εισαγωγή στον τρόπο με τον οποίο ρυθμίζεται η ταχύτητα μετάδοσης δεδομένων σε ανομοιόμορφα δίκτυα, όπως είναι το Internet.

Ας δούμε ένα απλό παράδειγμα. Εστω ότι έχουμε 2 υπολογιστές (Α και Β), οι οποίοι είναι συνδεδεμένοι με το Διαδίκτυο. Ο υπολογιστής Α βρίσκεται σε μία γραμμή υψηλών ταχυτήτων (100Mbps), ενώ ο υπολογιστής Β χρησιμοποιεί μία γραμμή DSL πολύ χαμηλότερης ταχύτητας για τη σύνδεση με το Διαδίκτυο. Σε μία ενδεχόμενη επικοινωνία μεταξύ τους, λοιπόν, ο Α θα αρχίσει να στέλνει δεδομένα με ταχύτερο ρυθμό απ' ό,τι μπορεί να τα λάβει ο Β. Τα δεδομένα αυτά θα ταξιδέψουν μέχρι τον πάροχο του Β, όπου θα συναντήσουν τη "στένωση" της DSL. Στο σημείο εκείνο, τα δεδομένα θα αρχίσουν να σχηματίζουν μία ουρά (queue), η οποία θα μεγαλώνει όσο ο ρυθμός με τον οποίο ο Α στέλνει δεδομένα είναι μεγαλύτερος από την ταχύτητα της γραμμής του Β. Επειδή η ουρά που δημιουργείται, παραμένει στη μνήμη (RAM) κάποιου δρομολογητή, δεν μπορεί να μεγαλώνει απεριόριστα, οπότε το μέγεθός της έχει ένα όριο. Μόλις φτάσει αυτό το οριακό μέγεθος, ο δρομολογητής (router) του παρόχου απορρίπτει τα νέα πακέτα που έρχονται με προορισμό τον Β, δείχνοντας έτσι ότι υπάρχει συμφόρηση (congestion) σε ένα κομμάτι του δικτύου. Με τη σειρά του, ο Α θα αρχίσει να μειώνει το ρυθμό αποστολής δεδομένων, μέχρι να σταματήσουν να χάνονται δεδομένα.

Αντίστοιχες ουρές δημιουργούνται και όταν ο Β επιχειρήσει να στείλει δεδομένα στον Α, αυτή τη φορά, όμως, στον aDSL router του Β, ο οποίος βρίσκεται ανάμεσα στο οικιακό LAN του Β και στην "αργή" aDSL γραμμή. Με άλλα λόγια, ουρές αναμονής δημιουργούνται πάντα αμέσως πριν από τη στένωση, κατά τη φορά μετάδοσης των δεδομένων. Επομένως, οι ουρές που μπορούν αποτελεσματικά να ελεγχθούν, είναι αυτές που ανήκουν στο upstream μίας γραμμής, αφού οι ουρές του downstream βρίσκονται σε άλλο τμήμα του δικτύου. Συμπτωματικά δε, το upstream των οικιακών συνδέσεων aDSL είναι

TUTORIAL_TRAFFIC 25-08-08 14:38 Σελίδα 77

σαφώς πιο περιορισμένο από το downstream και η εξάντλησή του οδηγεί σε πτώση των ταχυτήτων στην άλλη κατεύθυνση της γραμμής.

Η "ζημιά" που προκαλείται, επομένως, σε τέτοιες περιπτώσεις είναι διπλή: αφενός εξαντλείται με μη ελεγχόμενο τρόπο το εύρος ζώνης της γραμμής (αφού μία σύνδεση μπορεί να "μονοπωλήσει" την ουρά), αφετέρου όλα τα πακέτα (ακόμη και τα διαδραστικά) πρέπει να περιμένουν στην ουρά, αυξάνοντας έτσι το χρόνο που απαιτείται μέχρι να φτάσουν στον προορισμό τους. Στην ιδανική περίπτωση που το bandwidth θα ήταν πολύ μεγαλύτερο από το απαιτούμενο, δεν θα είχαμε ανάγκη το traffic shaping. Στον πραγματικό κόσμο, όμως, όπου το εύρος ζώνης είναι περιορισμένο, δύο είναι τα μέτρα που μπορούμε να λάβουμε: να το κατανείμουμε σύμφωνα με τους δικούς μας κανόνες, ακόμη και σε οριακές καταστάσεις, και να δώσουμε προτεραιότητα στη διαδραστική κίνηση ώστε να παρακάμπτει τις ουρές, όταν αυτές σχηματίζονται.

Queueing vs Shaping

Στο πρόβλημα αυτό έρχεται να μας βοηθήσει το Linux. Με τα προηγμένα χαρακτηριστικά δρομολόγησης και διαχείρισης κίνησης που διαθέτει, μας δίνει τη δυνατότητα να ελέγξουμε τον τρόπο με τον οποίο καταμερίζεται το εύρος ζώνης στην ανοδική κατεύθυνση μίας δικτυακής γραμμής και τον τρόπο με τον οποίο λειτουργεί η ουρά αναμονής.

Στο σημείο αυτό, πρέπει να τονίσουμε ότι, για να είναι επιτυχής η διαμόρφωση κίνησης σε μία σύνδεση aDSL, πρέπει ο router να τρέχει Linux ή διαφορετικά να μεταφέρουμε την ουρά αναμονής από τον router σε έναν H/Y με Linux. Στην περίπτωση μίας aDSL σύνδεσης με USB modem, ο H/Y στον οποίο συνδέεται το modem, μπορεί να διαχειριστεί απευθείας την ουρά αναμονής της γραμμής. Στην περίπτωση ενός Ethernet router, ο έλεγχος της γραμμής θα πρέπει να μεταφερθεί σε έναν H/Y με Linux (ο οποίος και θα αναλάβει χρέη δρομολογητή) μέσω του πρωτοκόλλου PPPoE, διαφορετικά θα πρέπει στον Ethernet router να συνδέεται μόνο ένας H/Y με Linux, για να έχει νόημα οποιαδήποτε ρύθμιση της κίνησης του δικτύου.

Σύμφωνα με όσα είπαμε παραπάνω, υπάρχουν 2 παράμετροι που μπορούν να ρυθμιστούν, προκειμένου να βελτιστοποιηθεί η απόδοση μίας γραμμής:

■ *το εύρος ζώνης που κατανέμεται ανά εφαρμογή ή ανά τύπο εφαρμογής*

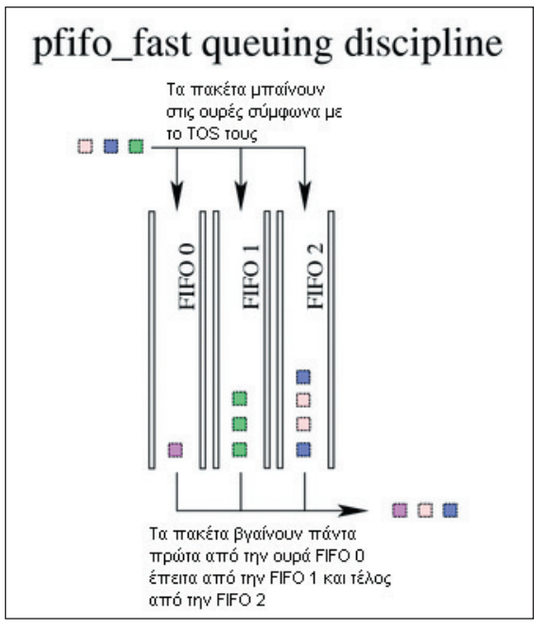
Ο μηχανισμός ρύθμισης του εύρους ζώνης παραμένει ως άνω, ωστόσο, μας δίνεται η δυνατότητα να τον εφαρμόσουμε ξεχωριστά για διάφορες κατηγορίες κίνησης. Ουσιαστικά, μπορούμε να χωρίσουμε κάθε δικτυακή διεπαφή σε περισσότερες εικονικές διεπαφές, καθεμία με δικό της εύρος ζώνης, και να χρησιμοποιήσουμε διαφορετική εικονική διεπαφή για κάθε είδος κίνησης που μας ενδιαφέρει.

■ *οι κανόνες που ισχύουν για τα πακέτα που εισέρχονται στις ουρές*

Στην πρώτη περίπτωση, μιλάμε για διαμόρφωση της κίνησης (traffic shaping), ενώ στη δεύτερη για διαχείριση της ουράς (queueing). Και τα δύο ελέγχονται μέσω διαφορετικών αλγορίθμων διαχείρισης ουράς (queueing algorithms ή queueing disciplines - qdiscs).

Πώς λειτουργούν το shaping και το queueing

Κάθε δικτυακή διεπαφή (network interface) στο Linux διαθέτει τη δική της ουρά αναμονής, η οποία είναι τύπου pfifo_fast (Packet-limited First In - First Out) και διέπεται από τους δικούς της κανόνες (qdiscs). Η ουρά αυτή αποτελείται από τρεις υπο-ουρές με διαφορετικές προτεραιότητες, στις οποίες κατατάσσεται η κίνηση ανάλογα με το πεδίο TOS (Type Of Service) του IP Header. Κάθε υπο-ουρά είναι τύπου FIFO, δηλαδή, τα πακέτα βγαίνουν από την ουρά με τη σειρά με την οποία μπήκαν. Ωστόσο, δεν φεύγουν πακέτα και από τις τρεις υπο-ουρές ταυτόχρονα, αλλά προτιμάται κάθε φορά η υπο-ουρά με το



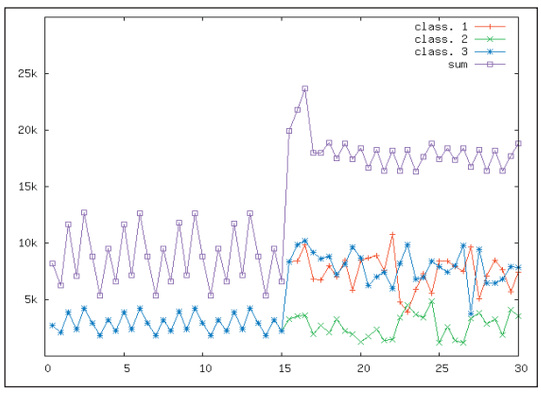
Η ουρά αναμονής pfifo_fast.

χαμηλότερο αριθμό προτεραιότητας, που έχει πακέτα. Ο αλγόριθμος αυτός δεν κάνει μορφοποίηση (shaping) της κίνησης, παρά μόνο επιτρέπει στα πακέτα με μεγαλύτερη προτεραιότητα να μπουν μπροστά από τα πακέτα χαμηλότερης προτεραιότητας. Ωστόσο, πέρα από τον pfifo_fast, το Linux διαθέτει ένα σύνολο από διαφορετικούς αλγορίθμους, με ή χωρίς δυνατότητα μορφοποίησης της κίνησης και καθορισμού προτεραιοτήτων:

■ Το **HTB (Hierarchical Token Bucket)** είναι ένας αλγόριθμος διαχείρισης ουράς, ο οποίος δημιουργεί, πάνω σε μία φυσική διεπαφή δικτύου, έναν αριθμό από εικονικές διεπαφές (κλάσεις), καθεμία με ένα όριο ταχύτητας και με το δικό της αλγόριθμο διαχείρισης ουράς αναμονής. Κάθε κλάση του HTB έχει μία κλάση-γονέα και μπορεί να έχει μία ή περισσότερες κλάσεις-παιδιά, δημιουργώντας έτσι μία ιεραρχική δομή. Τα "παιδιά" μίας κλάσης μπορούν να μοιραστούν το περίσσευμα εύρους ζώνης του "γονέα".

■ Ο **SFQ (Stochastic Fairness Queueing)** είναι ένας αλγόριθμος διαχείρισης ουράς, ο οποίος επιχειρεί να δίνει ίσες πιθανότητες στις διάφορες εφαρμογές να στείλουν τα πακέτα τους, σε αντίθεση με τον αλγόριθμο FIFO, στον οποίο τα πακέτα φεύγουν με τη σειρά που έρχονται, οπότε μία εφαρμογή μπορεί να "γεμίσει" όλη την ουρά με δικά της πακέτα.

■ Ο **TBF (Token Bucket Filter),** σε αντίθεση με το HTB, είναι ένας classless αλγόριθμος (δείτε "Κατηγορίες αλγορίθμων"), ο οποίος είναι ικανός να θέσει ένα ανώτατο όριο στην κίνηση. Τα πακέτα μαζεύονται σε έναν "κουβά" (bucket) και αυτός αδειάζει με μία συγκεκριμένη συχνότητα. Αν, για παράδειγμα, έχουμε μία εφαρμογή που θέλουμε να



Προσομοίωση κίνησης χωρίς traffic shaping.

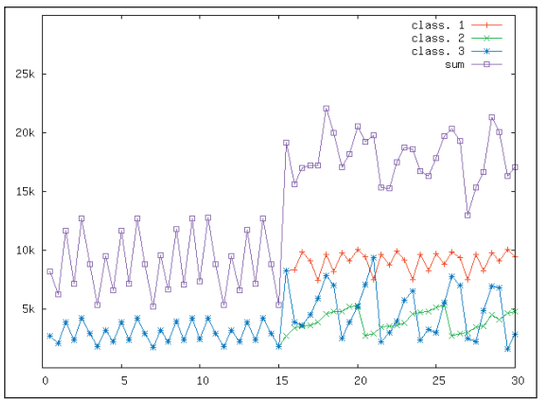
ΚΑΤΗΓΟΡΙΕΣ ΑΛΓΟΡΙΘΜΩΝ

Οι αλγόριθμοι διαχείρισης ουράς (qdiscs) που παρέχει το Linux, χωρίζονται σε δύο μεγάλες κατηγορίες, τους classful και τους classless. Οι classless αλγόριθμοι είναι σχετικά απλοί και χειρίζονται όλη την κίνηση που τους παραδίδεται με τον ίδιο τρόπο. Αντίθετα, οι classful αλγόριθμοι έχουν τη δυνατότητα διαχωρισμού της κίνησης που τους παραδίδεται σε κατηγορίες (κλάσεις). Κάθε κλάση έχει το δικό της queueing discipline, το οποίο με τη σειρά του μπορεί να είναι πάλι classful ή classless. Οι κλάσεις απλώς ομαδοποιούν την κίνηση και θέτουν συνολικά όρια ταχυτήτων ανά κατηγορία, ενώ τα qdiscs μορφοποιούν τις ουρές.

TUTORIAL_TRAFFIC 25-08-08 14:38 Σελίδα 78

CLASSIFY ΑΝΤΙ ΤΟΥ MARK

Ενας εναλλακτικός τρόπος για να πετύχουμε κατηγοριοποίηση της κίνησης, είναι να χρησιμοποιήσουμε το CLASSIFY target έναντι του MARK. Η διαφορά των δύο αυτών targets είναι πως, όταν ένα πακέτο κάνει match με τον πρώτο κανόνα CLASSIFY, τότε μπαίνει κατευθείαν στην κλάση που έχουμε ορίσει και σταματάει να διατρέχει τις αλυσίδες κανόνων του netfilter, ενώ όταν κάνει match με ένα κανόνα MARK, τότε συνεχίζει να διατρέχει την αλυσίδα των κανόνων. Ο τελευταίος κανόνας με τον οποίο κάνει match, είναι αυτός που θα του δώσει το τελικό MARK. Σε κάθε περίπτωση, η σειρά των κανόνων παίζει πολύ σημαντικό ρόλο, διαφορετικά το πακέτο μπορεί να καταλήξει σε λάθος κλάση.



Προσομοίωση κίνησης με traffic shaping.

έχει πάντα ένα ανώτατο όριο αποστολής δεδομένων μπορούμε να χρησιμοποιήσουμε το TBF για να το πετύχουμε.

Ο συνδυασμός των αλγόριθμων HTB και SFQ, στον οποίο θα επικεντρωθούμε στη συνέχεια του άρθρου, επιφέρει βέλτιστα αποτελέσματα για μία πληθώρα εφαρμογών, ενώ ταυτόχρονα είναι αρκετά εύκολη η ρύθμισή τους. Τα βήματά μας θα είναι:

1 Κατάστρωση του "χάρτη" των κατηγοριών

Για να λειτουργήσει αποτελεσματικά το shaping των πακέτων, είναι απαραίτητο να καθορίσουμε αναλυτικά τι είδους κίνηση έχουμε συνήθως στη γραμμή μας, να κατηγοριοποιήσουμε την κίνηση και να θέσουμε κάποιες προτεραιότητες σε αυτήν. Τι είδους κίνηση προτιμούμε να "φεύγει" πιο γρήγορα, ποια πακέτα δεν μας ενδιαφέρουν πολύ, θέλουμε να θέσουμε όρια στην κίνηση και, αν ναι, ποια; Η διαδικασία είναι αρκετά επίπονη και δεν υπάρχει κάποια λύση που να δουλεύει για κάθε είδους κίνηση, χρειάζεται πάντα αρκετή δουλειά. Οσο καλύτερη κατηγοριοποίηση και μεγαλύτερη εξειδίκευση γίνει τόσο πιο καλό θα είναι το τελικό αποτέλεσμα του shaping.

2 Δημιουργία των αντίστοιχων δομών στον πυρήνα

Στη συνέχεια, με τη βοήθεια του εργαλείου tc, θα μετατρέψουμε το "χάρτη" της κίνησης σε κλάσεις και ουρές στον πυρήνα. Κάθε κλάση θα αποκτήσει μία μοναδική ταυτότητα, η οποία θα χρησιμοποιείται για την ανακατεύθυνση της κίνησης προς την κλάση (δείτε πλαίσιο "Κλάσεις").

3 "Μαρκάρισμα" των πακέτων και αντιστοίχισή τους στις κατηγορίες

Το επόμενο βήμα μετά την κατάστρωση του σχεδίου για το shaping, είναι να αρχίσουμε να μαρκάρουμε την κίνηση με κάποια tags μέσω του iptables. Για κάθε κατηγορία βάζουμε πάντα ένα ξεχωριστό tag. Ετσι, μπορούμε έπειτα με το εργαλείο tc να αντιστοιχίσουμε αυτά τα tags σε κλάσεις, ώστε να αρχίσει να λειτουργεί το shaping.

Παράδειγμα χρήσης

Υποθέτουμε πως έχουμε μία "τυπική" aDSL γραμμή με 1Mbit upload και θέλουμε να την εκμεταλλευτούμε όσο καλύτερα γίνεται. Θέλουμε να έχουμε πάντα πολύ γρήγορη απόκριση στις συνεδρίες ssh, να έχουμε χαμηλά ping times και οι P2P συνδέσεις μας να μην επιβαρύνουν την τυπική πλοήγηση στο Διαδίκτυο (Web, FTP και mail). Επιλέγουμε, λοιπόν, να χρησιμοποιήσουμε το HTB, για να χωρίσουμε την εξερχόμενη κίνηση σε 3 κατηγορίες:

1 Κίνηση πραγματικού χρόνου. Στην κατηγορία αυτή θα μπαίνουν όλα τα πακέτα τύπου ICMP, UDP, καθώς και τα πακέτα που αφορούν σε συνδέσεις SSH. Εδώ θα μπουν επίσης τα πακέτα τύπου TCP SYN και ACK, ώστε να υπάρχει όσο το δυνατόν μικρότερη καθυστέρηση στα κρίσιμα αυτά πακέτα.

2 Κίνηση HTTP/FTP. Η κατηγορία αυτή αφορά στα πακέτα που έχουν να κάνουν κυρίως με το browsing μας. Μια και θέλουμε αυτό να μην επηρεάζεται από τα υπόλοιπα downloads, θα πρέπει να τα ξεχωρίζουμε κάπως.

3 Λοιπά (P2P και μη-κατηγοριοποιημένη κίνηση).

Θα δεσμεύσουμε 100, 200 και 400kbps αντίστοιχα από το 1Mbps upload της aDSL γραμμής, αλλά θα επιτρέψουμε σε κάθε κλάση να χρησιμοποιήσει τυχόν περίσσευμα εύρους ζώνης. Για το χειρισμό των συνδέσεων κάθε κλάσης, θα χρησιμοποιήσουμε SFQ, ώστε να γίνεται πιο "δίκαιη" μοιρασιά.

Δημιουργία κλάσεων

Για να δημιουργήσουμε τις κλάσεις μας, πρέπει πρώτα να προσαρτήσουμε ένα HTB qdisc ως το root qdisc της διεπαφής eth0:

```
tc qdisc add dev eth0 root handle 1: htb default 30
```

Το 1: είναι η "ταυτότητα" που θα χρησιμοποιήσουμε για τις υπο-κλάσεις που θα δημιουργήσουμε, ενώ το default 30 σημαίνει ότι τυχόν πακέτα που δεν εμπίπτουν σε κάποια από τις υπο-κλάσεις, θα πηγαίνουν στην κλάση 1:30.

Δημιουργούμε τις υπο-κλάσεις μας, πάλι με χρήση HTB, και με τα αντίστοιχα όρια ταχυτήτων:

```
tc class add dev eth0 parent 1: classid 1:10 htb rate
100kbit ceil 900kbit prio 0
tc class add dev eth0 parent 1: classid 1:20 htb rate
200kbit ceil 700kbit prio 1
tc class add dev eth0 parent 1: classid 1:30 htb rate
400kbit ceil 700kbit prio 2
```

Σε κάθε κλάση ορίζουμε ένα ελάχιστο εγγυημένο bandwidth (rate) και ένα ανώτατο όριο (ceil) που μπορεί να φτάσει η κλάση δανειζόμενη bandwith από άλλες, εφόσον υπάρχει περίσσευμα. Η ρύθμιση prio καθορίζει την προτεραιότητα μίας κλάσης. Αν δύο κλάσεις ζητούν περισσότερο bandwidth (αφού έχουν φτάσει το rate τους), τότε αυτή

CLASSFUL ΑΛΓΟΡΙΘΜΟΙ

Οπως είπαμε, κάθε interface έχει το δικό του root qdisc, το οποίο είναι τύπου pfifo_fast. Το root qdisc μπορεί να αντικατασταθεί με τη χρήση της εντολής

```
tc qdisc add dev <interface name> root handle
<ταυτότητα qdisc> <προδιαγραφή qdisc>
```

<ταυτότητα qdisc> είναι ένας αριθμός ακολουθούμενος από άνω-κάτω τελεία.

Σε ένα classful qdisc μπορούμε να προσθέσουμε υπο-κλάσεις, χρησιμοποιώντας την εντολή:

```
tc class add dev <interface name> parent
<ταυτότητα γονεϊκού qdisc ή γονεϊκής κλάσης>
classid <major:minor> <προδιαγραφή κλάσης>
```

Αντίστοιχα, κάθε κλάση μπορεί να έχει υπο-κλάσεις με τον ίδιο τρόπο. Το <parentid> είναι η ταυτότητα του qdisc ή της κλάσης στην οποία προσαρτάται η νέα υπο-κλάση. Το classid γράφεται πάντα ως δύο αριθμοί major:minor. Το major αντιστοιχεί στο qdisc στο οποίο δημιουργείται η κλάση ή η υπο-κλάση, ενώ το minor είναι ένας αριθμός μοναδικός για κάθε κλάση. Προσοχή θέλει το ότι όλες οι κλάσεις που βρίσκονται κάτω από το ίδιο classful qdisc έχουν το ίδιο major number, ασχέτως των μεταξύ τους σχέσεων.

Τέλος, στις ακραίες κλάσεις (leaf classes), προστίθεται πάντα ένα classless qdisc με το δικό του, μοναδικό handle

```
tc qdisc add dev <interface name> parent
<ταυτότητα γονεϊκής κλάσης> handle <handleid:>
```

Παράδειγμα:

```
# root qdisc
tc qdisc add dev ppp0 root handle 1: htb default 20
# Πρώτο επίπεδο κλάσεων
tc class add dev ppp0 parent 1: classid 1:1 htb rate
10kbps prio 0
tc class add dev ppp0 parent 1: classid 1:2 htb rate
30kbps prio 1
# Δεύτερο επίπεδο κλάσεων
tc class add dev ppp0 parent 1:1 classid 1:10 htb
rate 5kbps prio 0
tc class add dev ppp0 parent 1:1 classid 1:20 htb
rate 5kbps prio 1
# Ακραία qdiscs
tc qdisc add dev ppp0 parent 1:10 handle 2: sfq
perturb 10
tc qdisc add dev ppp0 parent 1:20 handle 3: pfifo
tc qdisc add dev ppp0 parent 1:2 handle 4: pfifo_fast
```

TUTORIAL_TRAFFIC 25-08-08 14:38 Σελίδα 79

με το μικρότερο prio θα πάρει το περισσότερο bandwidth. Οι ρυθμίσεις του HTB δεν σταματούν εκεί, όμως. Για να μην προκληθεί σύγχυση, δεν θα αναφέρουμε όλες τις ρυθμίσεις, αλλά όσοι θέλουν να το ψάξουν λίγο παραπάνω, μπορούν να ασχοληθούν και με τις ρυθμίσεις burst και cburst του HTB.

Οι υπο-κλάσεις μας χρειάζονται το δικό τους αλγόριθμο διαχείρισης, ο οποίος, όπως είπαμε, θα είναι το SFQ, χάριν ισότητας:

```
tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
tc qdisc add dev eth0 parent 1:30 handle 30: sfq perturb 10
```

Η τιμή perturb καθορίζει ανά πόσα δευτερόλεπτα θα γίνεται επανεξέταση της κίνησης σε μία κλάση και δεν πρέπει να είναι πολύ μικρή (προτείνεται η τιμή 10).

Κατηγοριοποίηση

Αφού δημιουργήσαμε τις κλάσεις μας, πρέπει να "σημαδέψουμε" το είδος της κίνησης που θα καταλήξει σε κάθε κλάση. Για να το πετύχουμε αυτό, χρησιμοποιούμε το MARK target του netfilter, με τη βοήθεια του εργαλείου iptables.

```
iptables -t mangle -A OUTPUT -p icmp -j MARK —set-mark 0x10
iptables -t mangle -A OUTPUT -p udp -j MARK —set-mark 0x10
iptables -t mangle -A OUTPUT -p tcp —dport 22 -j MARK —set-mark 0x10
iptables -t mangle -A OUTPUT -p tcp —syn -j MARK —set-mark 0x10
iptables -t mangle -A OUTPUT -p tcp —tcp-flags URG URG -j MARK —set-mark 0x10
iptables -t mangle -A OUTPUT -p tcp -m multiport —dports 80,110,143,443,993,995,20,21,25 -j MARK —set-mark 0x20
```

Στο παραπάνω παράδειγμα μαρκάρουμε όλα τα ICMP και UDP πακέτα με την τιμή 0x10. Επίσης, μαρκάρουμε τα TCP πακέτα με destination port το 22 (SSH), τα tcp syn και όσα έχουν tcp flags URG με 0x10. Τέλος, όλα τα πακέτα προς τα ports 80, 110, 143, 443, 995, 20, 21 (HTTP, POP3, IMAP, HTTPS, IMAPS, POP3S, FTP και SMTP) παίρνουν την τιμή 0x20.

Αφού γίνει το απαραίτητο μαρκάρισμα από το iptables, χρησιμοποιούμε και πάλι το tc, ώστε να στείλουμε τα μαρκαρισμένα πλέον πακέτα σε κλάσεις.

```
tc filter add dev eth0 protocol ip handle 0x10 fw flowid 1:10
tc filter add dev eth0 protocol ip handle 0x20 fw flowid 1:20
```

Οπως βλέπουμε παραπάνω, τα πακέτα που έχουν μαρκαριστεί με 0x10, θα τα στείλουμε στην κλάση 1:10, ενώ αυτά με 0x20 θα τα στείλουμε στην κλάση 1:20. Παρατηρήστε ότι δεν έχουμε προσθέσει κανόνες για μαρκάρισμα πακέτων για την κλάση 1:30, διότι, όταν δημιουργούσαμε το root qdisc, ορίσαμε την ορίσαμε ως την κλάση (default 30) που επιλέγεται αυτόματα για τα πακέτα που δεν εμπίπτουν ρητά σε καμία άλλη κλάση.

Αλλος ένας τρόπος να πετύχουμε κατηγοριοποίηση, είναι μέσω του εργαλείου u32 του tc. Το u32 είναι ένα πολύ δυνατό φίλτρο του εργαλείου tc, μέσα από το οποίο μπορούν να φτιαχτούν κανόνες για την κατηγοριοποίηση πακέτων, βασισμένοι στο destination/source IP, στο destination/source TCP/UDP port, στο TOS byte και στο πρωτόκολλο. Ετσι, αποφεύγουμε τη χρήση του iptables. Δυστυχώς, η σύνταξή του είναι αρκετά πιο περίπλοκη από αυτή του iptables, γι' αυτό συνήθως προτιμάται για την κατηγοριοποίηση ο τρόπος που παρουσιάσαμε παραπάνω. Το παρακάτω παράδειγμα επιλέγει τα ACK σε πακέτα μικρότερα των 64 bytes:

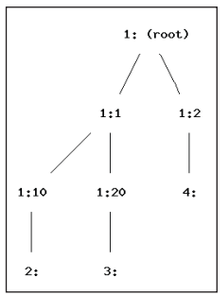
```
tc filter add dev eth0 parent 1:0 protocol ip prio 10 u32 \
 match ip protocol 6 0xff \
 match u8 0x05 0x0f at 0 \
 match u16 0x0000 0xffc0 at 2 \
 match u8 0x10 0xff at 33 \
 flowid 1:3
```

Επίλογος

Στο παρόν άρθρο παρουσιάσαμε μία πρώτη προσέγγιση στο ζήτημα της διαχείρισης δικτυακής κίνησης με το Linux. Οι δυνατότητες που παρέχει ο πυρήνας του Linux, είναι εντυπωσιακές, η αξιοποίησή τους, όμως, προϋποθέτει εξοικείωση με τον τρόπο λειτουργίας ενός δικτύου και αρκετό διάβασμα.

Τους αναγνώστες που επιθυμούν να επεκταθούν, τους παραπέμπουμε στις manpages του tc και των διαφόρων qdiscs (tc, tc-tbf, tc-htb, tc-sfq, tc-pfifo, tc-pfifo_fast, tc-cbq), καθώς και στις ιστοσελίδες του LARTC (http://lartc.org) και του HTB (http://luxik.cdi.cz/~devik/qos/htb/). Αρκετά καλό κείμενο είναι επίσης το Traffic Control HOWTO (http://tldp.org/HOWTO/Traffic-Control-HOWTO/). Καλή ανάγνωση!

■



Ιεραρχικό διάγραμμα κλάσεων.

ΤΟ SCRIPT ΠΟΥ ΤΑ ΚΑΝΕΙ ΟΛΑ

Ενα μικρό script για τον αυτοματισμό των παραπάνω εντολών:

```
DEV_EXT=eth0
TC=/sbin/tc
IPTABLES=/sbin/iptables
if [ "$1" = "status" ]
then
 echo "— — External Interface"
 $TC -s qdisc ls dev $DEV_EXT
 $TC -s class ls dev $DEV_EXT
 exit
fi
tc qdisc del dev $DEV_EXT root 2> /dev/null > /dev/null
if [ "$1" = "stop" ]
then
 exit
fi
$TC qdisc add dev $DEV_EXT root handle 1: htb default 30
$TC class add dev eth0 parent 1: classid 1:10 htb rate 100kbit ceil 900kbit prio 0
$TC class add dev eth0 parent 1: classid 1:20 htb rate 200kbit ceil 700kbit prio 1
$TC class add dev eth0 parent 1: classid 1:30 htb rate 400kbit ceil 700kbit prio 2
$TC qdisc add dev $DEV_EXT parent 1:10 handle 10: sfq perturb 10
$TC qdisc add dev $DEV_EXT parent 1:20 handle 20: sfq perturb 10
$TC qdisc add dev $DEV_EXT parent 1:30 handle 30: sfq perturb 10
$IPTABLES -t mangle -A OUTPUT -p icmp -j MARK —set-mark 0x10
$IPTABLES -t mangle -A OUTPUT -p udp -j MARK —set-mark 0x10
$IPTABLES -t mangle -A OUTPUT -p tcp —dport 22 -j MARK —set-mark 0x10
$IPTABLES -t mangle -A OUTPUT -p tcp —syn -j MARK —set-mark 0x10
$IPTABLES -t mangle -A OUTPUT -p tcp —tcp-flags URG URG -j MARK —set-mark 0x10
$IPTABLES -t mangle -A OUTPUT -p tcp -m multiport —dports 80,110,143,443,993,995,20,21,25 -j MARK —set-mark 0x20
$TC filter add dev eth0 protocol ip handle 0x10 fw flowid 1:10
$TC filter add dev eth0 protocol ip handle 0x20 fw flowid 1:20
$TC filter add dev eth0 parent 1:0 protocol ip prio 10 u32 \
 match ip protocol 6 0xff \
 match u8 0x05 0x0f at 0 \
 match u16 0x0000 0xffc0 at 2 \
 match u8 0x10 0xff at 33 \
 flowid 1:3
```