Πανεπιστήμιο Πατρών

Τμήμα Μηχανικών ΗΥ & Πληροφορικής

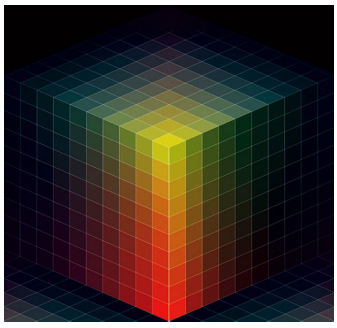
Παράλληλοι Αλγόριθμοι

Fast Algorithms for Bit-Serial Routing on a Hypercube

Επιμέλεια:
Ιωάννης Δουράτσος
Κυριάκος Ισπόγλου
Ιωάννα Τζανέτου

Επίβλεψη:
Εύη Παπαϊωάννου

Ιούλιος, 2014



Contents

1 Introduction

Στο paper το οποίο μελετάμε περιγράφεται ένας αλγόριθμος που βασίζεται στη τυχαιότητα για τη δρομολόγηση πακέτων σε ένα υπερκύβο μέσα σε O(logN) βήματα δρομολόγησης bit. Ο αλγόριθμος αυτός βελτιώνει τον καλύτερο προηγούμενο αλγόριθμο κατά μια λογαριθμική τιμή. Ταυτόχρονα ο ίδιος αλγόριθμος λύνει το πρόβλημα της online circuit switching δρομολόγησης σε ένα O(1) dilated υπερκύβο.
Στον υπερκύβο είναι δύσκολη γενικά η υλοποίηση γρήγορων bit-serial αλγορίθμων, κάτι που οφείλεται στο ότι δεδομένου ενός πακέτου μήκους logN bits, το κεφάλι του πακέτου θα βρίσκεται κοντά στον προορισμού πριν καν η ουρά του ξεκινήσει από την αφετηρία, ενώ κατά την διάρκεια της διαδρομής του μέσα στο κανάλι θα χρησιμοποιεί έως και logN καλώδια αποτρέποντας τα άλλα πακέτα από το να τα χρησιμοποιήσουν για τη δρομολόγηση τους.

Ο αλγόριθμος ο οποίος παρουσιάζεται είναι βέλτιστος για την δρομολόγηση πακέτων στον υπερκύβο, αφού δρομολογεί ένα πακέτο M bits μέσα σε O(M+logN) βήματα δρομολόγησης bits δηλαδή απαιτεί μόλις ένα βήμα για κάθε bit του πακέτου, αφού μαζί με τα bit που περιέχουν τη πληροφορία για το κόμβο προορισμού του πακέτου, το πακέτο αποτελείται από M+logN bits.

Το αποτέλεσμα του αλγορίθμου μπορεί να χρησιμοποιηθεί για την on-line δρομολόγηση σε ένα υπερκύβο. Αυτό βελτιώνει τους μέχρι τώρα υπάρχοντες αλγορίθμους, που βασίζονταν στην αντικατάσταση κάθε ακμής του υπερκύβου με τέσσερις κατευθυνόμενες ακμές, οι οποίοι όλοι χρειάζονταν $\Omega(log^3 N)$ για να υπολογίσουν τα μονοπάτια που πρέπει να ακολουθήσει κάθε πακέτο. Με τον αλγόριθμο που θα παρουσιαστεί, μπορούν να υπολογιστούν τα κατάλληλα μονοπάτια σε O(logN) βήματα δρομολόγησης bit αν υπάρχουν O(1) αντίγραφα των ακμών του υπερκύβου προς κάθε κατεύθυνση.

2 Σχετικές Εργασίες

Όσον αφορα το πρώτο μέρος της εργασίας, δηλαδή την εύρεση ενός αποδοτικού αλγορίθμου δρομολόγησης σε υπερκύβους έχουν γίνει αρκετές προσπάθεις. Η καλύτερη προσπάθεια για έναν ντετερμινιστικό αλγόριθμο δρομολόγησης, έγινε από τους Cypher και Plaxton στην εργασία «Deterministic Sorting in nearly logarithmic time on the hypercube and related computers», με συνολική πολυπλοκότητα:

$$O(logN * log^2 logn)$$

Ωστόσο ένας πιθανότικός αλγόριθμος δρομόλογησης θα έδινε καλύτερους χρόνους εκτέλεσης. Οι Leighton, Maggs και Rao στην εργασία τους «Universal packet routing algorithms» προτείναν ένα τυχαιοποιημένο αλγόριθμο που με μεγάλη πιθανότητα έχει πολυπλοκότητα:

$$O(c + D + logN)$$

όπου c είναι η μέγιστη συμφόρηση, D είναι η μέγιστη απόσταση και N ο αριθμός των πακέτων,

Τώρα για το δεύτερο σκέλος της εργασίας, δηλαδή την προσπάθεια εύρεσης ενός καλύτερου κάτω φράγματος για πλήθος των βημάτων που απαιτούνται, η μέχρι τώρα καλύτερη τιμή ήταν:

$$\Omega(\frac{\sqrt{n}}{d^{3/2}})$$

Η οποία δημοσιεύτηκε από τους Borodin και Hopcroft στην εργασία «Routing, mergin, and sorting on parallel models of computation», ενώ αργότερα οι Kaklamanis, Krizanc και Tsantilas με την εργασία «Tight bounds for oblivious routing in the hypercube» δημοσίευσαν ένα καλύτερο φράγμα για την δρομολόγηση οποιασδήποτε μετάθεσης με κάποιον oblivious αλγόριθμο δρομολόγησης:

$$\Omega(\frac{\sqrt{n}}{d})$$

Όπου d είναι ο μέγιστος βαθμός του δικτύου. Ακόμα για τον υπερκύβο των N κόμβων, υπολογίζεται ένα ακόμα άνω φράγμα, με την εύρεση ενός ντετερμινιστικού oblivious αλγορίθμου ο οποιός δρομολογεί οποιαδήποτε μετάθεση αριθμών σε $\Theta(\frac{\sqrt{n}}{logN})$ βήματα, ενώ το μέχρι στιγμής καλύτερο φράγμα ήταν $\Theta(\sqrt{n})$.

Όπως θα δούμε, τα αποτελέσματα που παρουσιάζονται εδώ είναι αρκετά καλύτερα από τα προηγούμενα.

3 Oblivious routing

Πολύ σημαντικό για την κατανόηση του περιεχομένου του paper είναι η κατανόηση της έννοιας ενός oblivious αλγορίθμου δρομολόγησης. Κάθε πακέτο περιέχει εκτός από τα data και πληροφορία την οποία χρησιμοποιεί για να επιτευχθεί σωστά η δρομολόγηση του (ticket). Η πληροφορία αυτή αρχικά αποτελείται μόνο από το όνομα του πακέτου και τη διεύθυνση του κόμβου στον οποίο κατευθύνεται, και κατά τη διάρκεια του ταξιδιού του πακέτου μέσα στο δίκτυο η πληροφορία αυτή μπορεί να αλλάξει. Η δρομολόγηση εξαρτάται από τα περιεχόμενα του ticket. Οι oblivious αλγόριθμοι δρομολόγησης είναι αυτοί στους οποίους η αλλαγή δεδομένων μέσα στο ticket ενός πακέτου δεν εξαρτώνται από την ύπαρξη άλλων πακέτων ή από τα tickets των άλλων πακέτων που βρίσκονται στο δίκτυο με κανένα τρόπο.

Σε αντίθεση με τους περισσότερους αλγορίθμους δρομολόγησης σε υπερκύβο που βασίζονται στην τυχαιότητα, ο αλγόριθμος που προτείνεται δεν είναι oblivious. Αποδεικνύεται επίσης ότι κάθε oblivious αλγόριθμος δρομολόγησης που βασίζεται στην τυχαιότητα χρειάζεται $\Omega((M + \frac{logN}{logd}) * \frac{logN}{logd+loglogN})$ βήματα δρομολόγησης bit με μεγάλη πιθανότητα όπου M το μέγεθος πακέτου χωρίς τα bit που χρησιμοποιούνται για τη διευθυνσιοδότηση και d ο μέγιστος βαθμός των κόμβων του γραφήματος. Όταν αυτό το όριο το εφαρμόσουμε στον υπερκύβο, τότε το κάτω όριο ορίζεται ως $\Omega(\frac{log^2 N}{loglogN})$ βήματα δρομολόγησης bit.

Για την παρουσίαση του αλγορίθμου χρησιμοποιείται το δίκτυο b-dilated πεταλούδας, το οποίο προκύπτει από το δίκτυο πεταλούδας όταν αντικαταστήσουμε κάθε ακμή του δικτύου με ένα bundle από b ακμές. Με βάση το δίκτυο αυτό παρουσιάζονται δύο αλγόριθμοι, ο πρώτος από τους οποίους είναι απλούστερος αλλά απαιτεί από κάθε switch του δικτύου να μπορεί να ανταπεξέλθει σε logN νέα πακέτα ταυτόχρονα, ενώ ο δεύτερος αν και πιο περίπλοκος στην εκτέλεση του, απαιτεί από κάθε switch τον έλεγχο δύο των πολύ πακέτων σε κάθε βήμα. Αφού οι αλγόριθμοι περιγραφούν στο δίκτυο πεταλούδας που είναι ευκολότερο στην κατανόηση, δείχνεται πως μπορούν να χρησιμοποιηθούν στον υπερκύβο κάνοντας κατάλληλη αντιστοίχηση ανάμεσα στις ακμές και τους κόμβους του υπερκύβου και της b-dilated πεταλούδας.

Ας περιγράψουμε τον αλγόριθμο στην b-dilated πεταλούδα. Αρχικά, σε ένα δίκτυο πεταλούδας κάθε κόμβος μπορεί να αναφερθεί από μια δυάδα (r,l) όπου r η γραμμή του κόμβου και l το επίπεδο. Σε ένα δίκτυο πεταλούδας με n επίπεδα, η γραμμή είναι ένας δυαδικός αριθμών των n bits και το επίπεδο είναι ένας ακέραιος από 0 έως n.

Για κάθε επίπεδο $l < n$, κάθε κόμβος (r,l) έχει ακμή προς τον $(r, l + 1)$ και τον $(r^{l+1}, l + 1)$, όπου r^{l+1}το r όταν το bit $l + 1$αντιστραφεί. Αν υποθέσουμε ότι οι κόμβοι εισόδου και οι κόμβοι εξόδου αναγνωρίζονται από το ίδιο label τότε σε ένα δίκτυο πεταλούδας n επιπέδων έχουμε $N = n * 2^n$ κόμβους. Για την ευκολότερη κατανόηση θα υποθέσουμε επίσης ότι οι κόμβοι είναι κατευθυνόμενοι από αριστερά προς τα δεξιά (καθώς δηλαδή αυξάνει το επίπεδο), οπότε θα καλούμε ακμές εισόδου τις ακμές που κατευθύνονται προς ένα switch ενώ ακμές εξόδου αυτές που φεύγουν από ένα switch.

Εφόσον σε κάθε κανονικό δίκτυο πεταλούδας κάθε switch έχει δύο ακμές εισόδου και δύο ακμές εξόδου, τότε είναι πολύ εύκολο να δούμε ότι σε ένα b-dilated δίκτυο όπου θα έχουμε αντικαταστήσει κάθε ακμή με ένα bundle από b ακμές, κάθε switch θα έχει ως είσοδο δύο bundles από b ακμές εισόδου το κάθε ένα και ως έξοδο άλλα δύο bundles από b ακμές εξόδου.

4 Μοντέλα switches

Τα switches είναι αυτά που θα καθορίσουν και τη πολυπλοκότητα του αλγορίθμου δρομολόγησης που θα χρησιμοποιήσουμε. Υπάρχουν δύο μοντέλα switches, ανάλογα με το πόσες ακμές εισόδου μπορεί να ελέγχει ένα switch σε κάθε βήμα. Αν θεωρήσουμε ότι κάθε switch μπορεί να ελέγξει όλες τις ακμές που προσπίπτουν σε αυτό μέσα σε ένα βήμα, τότε έχουμε μοντέλο δυνατού switch ενώ αν μπορεί να εξετάσει μόνο μία ακμή από κάθε bundle τότε έχουμε το μοντέλο αδύναμου switch. Σε κάθε περίπτωση, κάθε switch μπορεί να ενώσει μια ακμή εισόδου με μία ακμή εξόδου έτσι ώστε στο μέλλον κάθε bit που θα έρθει στην ακμή εισόδου να μεταφερθεί στην ακμή εξόδου ένα βήμα αργότερα χωρίς να χρειαστεί να το εξετάσει το switch. Η διαφορά έγκειται στο ότι στο μοντέλο δυνατού switch το switch μπορεί να εκτελέσει ένα ταίριασμα ανάμεσα σε οσαδήποτε ζευγάρια ακμών εισόδου-εξόδου, ενώ στο μοντέλο αδύναμου switch περιορίζεται σε ένα ζευγάρι το πολύ. Σε κάθε περίπτωση, κάθε switch μπορεί να δώσει στην έξοδο του το πολύ ένα bit σε κάθε βήμα. Προσπαθώντας να δρομολογήσουμε μεταθέσεις σε ένα δίκτυο πεταλούδα μπορούμε εύκολα να αναγάγουμε το πρόβλημα στην περίπτωση που κάθε είσοδος είναι η αφετηρία n πακέτων και κάθε έξοδος αποτελεί τον προορισμό n πακέτων. Η δρομολόγηση των πακέτων γίνεται σε δύο φάσεις:

1. Στην πρώτη φάση κάθε πακέτο στέλνεται σε ένα τυχαίο κόμβο εξόδου του επιπέδου n
2. Στη δεύτερη φάση κάθε πακέτο δρομολογείται από τον κόμβο αυτό στο πραγματικό κόμβο προορισμού στο επίπεδο n

Με αυτό το τρόπο, εξασφαλίζουμε ότι με μεγάλη πιθανότητα το πολύ O(logN) πακέτα θα περάσουν από κάθε switch κατά τη διάρκεια κάθε φάσης. Σε ένα δίκτυο που είναι dilated κατά O(logN) είναι προφανές ότι το πολύ ένα πακέτο θα χρησιμοποιεί κάθε φορά οποιαδήποτε ακμή του δικτύου, κάτι που σημαίνει οτι ο αλγόριθμος αυτός είναι κατάλληλος και για circuit switching. Ο τρόπος με τον οποίο τα πακέτα δρομολογούνται μέσα στο δίκτυο, είναι συγκεκριμένος: στην άκρη κάθε ακμής βρίσκεται μια ουρά η οποία μπορεί να κρατήσει ένα μικρό αριθμό από bits. Συνεπώς, εφόσον κάθε πακέτο θα περιέχει τουλάχιστον n bits (λόγω της πληροφορία που περιέχει για τον προορισμό του) , δεν μπορεί να αποθηκευτεί σε μια ακμή, αλλά πρέπει να απλωθεί σε πολλές. Το πακέτο μεταβαίνει από ακμή σε ακμή μόνο όταν το κεφάλι μετακινηθεί σε επόμενη ακμή και αφήσει άδειο χώρο στην ουρά στην οποία μέχρι τότε ήταν αποθηκευμένο. Τότε το επόμενο bit μεταβαίνει στη θέση του κεφαλιού κ.ο.κ ώστε όλο το πακέτο συνολικά να κινηθεί μέσα στο δίκτυο. Στη κορυφή κάθε πακέτου βρίσκεται ένα κομμάτι πληροφορίας που σχετίζεται με τον προορισμό του πακέτου, το οποίο αποτελείται από 2n bits, το οποίο αναλύεται σε n bits πληροφορίας για τον ενδιάμεσο προορισμό του πακέτου και n bits πληροφορίας για το τελικό προορισμό του πακέτου. Κάθε φορά που το πακέτο περνάει από ένα switch, το πιο σημαντικό bit αφαιρείται και χρησιμοποιείται ώστε να αποφασιστεί σε πιο bundle από ακμές εξόδου θα αποσταλεί το πακέτο.

5 Δρομολόγηση στο μοντέλο ισχυρών switches

Στο μοντέλο αυτό όπως είπαμε κάθε switch μπορεί να ελέγξει όλες τις ακμές κάθε bundle ταυτόχρονα και μπορεί να ενώσει οσεσδήποτε ακμές εισόδου με ακμές εξόδου μέσα σε ένα βήμα. Για αυτό ακριβώς το λόγο, κανένα πακέτο δεν θα καθυστερήσει, κάτι που κάνει τον αλγόριθμο δρομολόγησης πολύ απλό. Σε κάθε βήμα, κάθε switch ελέγχει τις ακμές εισόδου του για νέα πακέτα. Μόλις βρεί ένα νέο πακέτο, αφαιρεί το αρχικό bit από τον header του πακέτου και ενώνει την ακμή από την οποία προήρθε το πακέτο με μια ακμή εξόδου που δεν χρησιμοποιείται η οποία ανήκει στο bundle ακμών που προσδιορίζεται από το bit που μόλις διάβασε. Λόγω του ότι δουλεύουμε σε δίκτυο πεταλούδας που είναι logN-dilated, ξέρουμε ότι πάντα θα μπορούμε να βρούμε μια τέτοια ακμή σε κάθε bundle και συνεπώς ο αλγόριθμος δρομολόγησης δουλεύει σε κάθε περίπτωση.

6 Δρομολόγηση στο μοντέλο αδύναμων switches

Στη περίπτωση των αδύναμων switches ο αλγόριθμος δρομολόγησης γίνεται πολύ πιο περίπλοκος, αφού εμφανίζονται προβλήματα όταν πολλαπλά πακέτα φτάσουν στο ίδιο bundle ακμών εισόδου την ίδια χρονική στιγμή. Όπως αναφέραμε ήδη, ένα switch τότε μπορεί να ελέγξει μόνο μια από αυτές, το οποίο σημαίνει και ότι μπορεί να συνδέσει μόνο μια με κάποια κατάλληλη ακμή εξόδου, δηλαδή τα υπόλοιπα πακέτα θα καθυστερήσουν. Ο αλγόριθμος δρομολόγησης θα πρέπει να εξασφαλίζει ότι η συνολική καθυστέρηση που υφίσταται κάθε πακέτο είναι το πολύ O(log N) βήματα bit.

Ο αλγόριθμος δρομολόγησης που προτείνεται εδώ, είναι μια παραλλαγή του αλγορίθμου του Ranade για τη δρομολόγηση πακέτων σε δίκτυα πεταλούδας. Σε αυτόν τον αλγόριθμο, ανατείθεται αρχικά σε κάθε πακέτο μια τυχαία βαθμολογία (κάποιο rank), η οποία αποθηκεύεται μπροστά από την επικεφαλίδα προορισμού. Το σημαντικότερο -που διατηρείται αναλλοίωτο από τον αλγόριθμο- είναι ότι η ροή των πακέτων που φεύγουν από κάθε switch είναι διατεταγμένη σε μια μη-φθίνουσα διάταξη. Προκειμένου να διατηρηθεί αυτή η ροή πακέτων ταξινομημένη, ένα switch δεν στέλνει ποτέ κάποιο πακέτο μέχρι να βεβαιωθεί ότι κανένα άλλο πακέτο με μικρότερη βαθμολογία δεν θα φτάσει αργότερα.

Το πρόβλημα με την εφαρμογή του αλγορίθμου του Ranade στο bit μοντέλο είναι ότι παίρνει πολύ χρόνο για να συγκρίνει τις βαθμολογίες δύο πακέτων στο ίδιο switch. Στον αλγόριθμο που παρουσιάζεται εδώ, ανατείθονται τυχαία οι βαθμολογίες των πακέτων ενώ διατηρείται αναλλοίωτο το γεγονός ότι τα πακέτα που φεύγουν από το switch είναι ταξινομημένα ως προς την βαθμολογία τους.Η αλλαγή που κάνουμε στο σχήμα είναι ότι τα πακέτα δεν περιέχουν την βαθμολογία τους. Αντί αυτού εισάγεται ένας νέος τύπος πακέτων που ονομάζεται token. Τα tokens χρησιμοποιούνται για να αντιπροσωπεύουν τις βαθμολογίες των πακέτων: από κάθε bundle περνάνε ακριβώς r tokens (από διαφορετικές ακμές) πριν από ένα πακέτο με βαθμολογία r. Ένα token μπορεί να θεωρηθεί ότι έχει και αυτό μια βαθμολογία. Η βαθμολογία του

token ισούται με τον αριθμό των token που έχουν περάσει μέσα από το bundle πριν από αυτό (με άλλα λόγια το i-οστό token που περνάει από το bundle θα έχει βαθμολογία i). Ένα token δεν φέρει καμία άλλη πληροφορία εκτός από το είδος του και για αυτό το λόγο μπορεί να αναπαρασταθεί με O (1) bits.

Οι μηχανισμοί του αλγορίθμου είναι αρκετά απλοί. Ένα switch διατηρεί έναν δείκτη σε μια ακμή για κάθε ένα τα τα 2 bundle εισόδου και τα 2 bundle εξόδου. Αυτές οι ακμές καλούνται ως η ενεργή είσοδος και η ενεργή έξοδος αντίστοιχα.

- Αν και οι δύο ενεργες ακμές εισόδου έχουν token στις ουρές τους, τότε το switch στέλνει ένα token και στις δύο ενεργές ακμές εξόδου, και ενημερώνει και τους 4 δείκτες ακμών. Αν ένα από τα ενεργά άκρα εισόδου έχει ένα token στην ουρά του και το άλλο ένα πακέτο, τότε το switch αφαιρεί από το πρώτο bit της επικεφαλίδας του πακέτου, ενώνει το πακέτο στην τρέχουσα ακμή του bundle εξόδου που καθορίζεται από αυτό το bit, και ενημερώνει τους δείκτες για τα bundles που χρησιμοποιήθηκαν από το πακέτο.
- Αν και οι δύο από τις ενεργές ακμές εισόδου έχουν πακέτα σε ουρές τους, τότε το switch προωθεί το ένα από αυτά, και ενημερώνει τους δείκτες για τα bundles που χρησιμοποιήθηκαν από αυτό.
- Εάν οποιαδήποτε από τις ενεργές ακμές εισόδου έχει μια άδεια ουρά, τότε το switch δεν κάνει τίποτα.

Για την εκκίνηση του αλγορίθμου, ανατείθεται σε κάθε πακέτο p μια τυχαία βαθμολογία rank(p) από το σύνολο [0,. . . , w-1], όπου το w=O(logN). Τα switches στο επίπεδο 0 στέλνουν μια ροή πακέτων αναμιγμένη με tokens, έτσι ώστε να προηγούνται r tokens πριν από κάθε πακέτο με βαθμολογία r. Συνολικά αποστέλλονται, w tokens σε κάθε δέσμη. Τα tokens με βαθμό w–1 ακολουθούν όλα τα υπόλοιπα πακέτα και tokens στο δίκτυο. Η φάση δρομολόγησης ολοκληρώνεται όταν κάθε έξοδος έχει λάβει ένα token με βαθμολογία w-1 και από τα 2 bundle εισόδου του. Για ακόμη ευκολότερη υλοποίηση από πλευράς του δικτύου, τα tokens με βαθμολογία w-1 μπορεί να είναι ειδικά μαρκαρισμένα έτσι ώστε οι έξοδοι να μην χρειάζεται να μετρούν τον αριθμό των token που έχουν λάβει.

Κατά τη διάρκεια του αλγορίθμου, κάθε ακμή χρησιμοποιείται είναι από ένα πακέτο, ή ένα token, ή και καθόλου. Κατά συνέπεια, η πεταλούδα πρέπει να επεκταθεί περαιτέρω με την προσθήκη w ακμών σε κάθε bundle. Δεδομένου ότι w = O (logN), η πεταλούδα παραμένει επεκτεταμένη κατά O (logN) .

Η απόδειξη ότι τα heads των πακέτων καταφθάνουν γρήγορα στους προορισισμούς τους χρησιμοποιεί ένα επιχείρημα που βασίζεται στις ακολουθίες καθυστέρησης. Μία k-delay ακολουθία καθυστέρησης αποτελείται από 4 συνιστώσες:

1. Μια διαδρομή προς τα πίσω από μια έξοδο σε μια εισόδου,
2. Μια ακολουθία (όχι απαραίτητα διακριτή) από switches που εμφανίζονται στην διαδρομή με τη σειρά $s_1, s_2, ...s_k$,
3. Μια ακολουθία $p_1, p_2, ...p_k$διαφορετικών πακέτων, έτσι ώστε το s_iνα διέρχεται μέσα από p_i
4. Μια αλληλουχία $r_1, r_2, ...r_k$ από μη αύξοντες βαθμολογίες

Μια k-delay ακολουθία προκύπτει, εάν $rank(p_i) = r_i$ για $1 <= i <= k$. Το αν θα συμβεί ή όχι μια καθυστέρηση εξαρτάται αποκλειστικά από τις βαθμολογίες των τυχαίων αριθμών που εκχωρούνται στα πακέτα. Ο χώρος πιθανοτήτων αποτελείται από w^N δυνατές αναθέσεις από βαθμολογιών, ενώ μια k-delay ακολουθία καθυστέρησης παρουσιάζεται σε $w^(-k)$ από αυτές. Συνεπώς, αυτό συμβαίνει με πιθανότητα $\frac{1}{w^k}$.

Λήμμα 1: Για κάθε d>w, αν κάποιο πακέτο φτάσει στον ενδιάμεσο προορισμό του στο στάδιο n + d, τότε πρέπει να έχει συμβεί μια (d-W)-delay ακολουθία καθυστέρησης.

Απόδειξη: Θα κατασκευάσουμε μια ακολουθία καθυστέρησης χαλώντας πόντους καθυστέρησης. Η καθυστέρηση ενός πακέτου στο επίπεδο l τη χρονική στιγμή t είναι t - l. Εάν κάποιο πακέτο φτάσει στον προορισμό του στο βήμα n + d, τότε έχει καθυστέρηση d. Χρησιμοποιούμε αυτά τα d σημεία για την κατασκευή της ακολουθίας. Για κάθε σημείο που ξοδεύεται είτε εισάγουμε ένα πακέτο στην ακολουθία, είτε μειώνουμε τη βαθμολογία του επόμενου πακέτου στην ακολουθία. Μιας και μπορούν να γίνουν το πολύ w μειώσεις στην βαθμολογία, η ακολουθία θα πρέπει να περιέχει τουλάχιστον, d-w πακέτα.

Η κατασκευή της ακολουθίας καθυστέρησης ξεκινάει με το τελευταίο πακέτο που πρέπει να φθάσει στον προορισμό του. Ας υποθέσουμε ότι ένα πακέτο p_1 φτάνει στον προορισμό του s_1, σε χρόνο n + d. Πρώτον,

εισάγουμε τα p_1, s_1, και $r_1 = rank(p_1)$ στην ακολουθία καθυστέρησης. Στη συνέχεια ακολουθούμε το pl πίσω στο χρόνο μέχρι να γίνει το τελευταίο που θα καθυστερήσει. Η καθυστέρηση παραμένει d σε κάθε βήμα στο μονοπάτι, μέχρι το βήμα στο οποίο το p_1 απέτυχε να προωθηθεί, όπου η καθυστέρηση πέφτει στο d-1.

Αν το p_1 είχε καθυστερήσει στο switch s_2 επειδή κάποιο άλλο πακέτο p_2 είχε σταλεί αντ 'αυτού, τότε $rank(p_2) = rank(p_1)$, και εισάγονται τα p_2, s_2 και $r_2 = rank(p_2)$ στην ακολουθία, και ακολουθούμε το p_2 back in time μέχρι να γίνει το τελευταίο που θα καθυστερήσει. Σε αυτήν την περίπτωση έχει εισαχθεί ένα πακέτο με κόστος ενός πόντου καθυστέρησης. Στο δίκτυο πεταλούδας, αν οι διαδρομές των δύο πακέτων αρχίζουν να αποκλίνουν, δεν συναντηθούν ποτέ στο μέλλον. Ετσι, αν το p_2 καθυστερήσει το p_1, τότε το p_2 θα παραμένει μπροστά από p_1 όσο διαδρομές τους τέμνονται. Κατά συνέπεια, κανένα πακέτο δεν εισάγεται εντός της ακολουθίας περισσότερο από μία φορά. Διαφορετικά, υποθέτουμε ότι το p_1 καθυστέρησε επειδή ένα token t_1 στάλθηκε στην θέση του. Στην περίπτωση αυτή είναι $rank(t_1) = rank(p_1) - 1$. Δεν εισάγουμε τίποτε στην ακολουθία, αλλά αντίθετα ακολουθούμε το t_1 πίσω στο χρόνο μέχρι να γίνει το τελευταίο που θα καθυστερήσει. Σε αυτή την περίπτωση, έχουμε χάσει ένα πόντο καθυστέρησης, αλλά έχουμε μειωσεί επίσης τον βαθμό του επόμενου πακέτου στην ακολουθία κατά τουλάχιστον ένα.

Η τρίτη περίπτωση είναι το p_1 να καθυστέρησε επειδή η τρέχουσα ακμή στην άλλη είσοδο του bundle, c, δεν είχε κάτι να στείλει. Σε αυτή την περίπτωση δεν χρειάζεται να εισαχθεί τίποτε στην ακολουθία, αλλά να πάμε πίσω μέσω του c σε ένα switch s στο προηγούμενο επίπεδο στο προηγούμενο χρονικό. Στο προηγούμενο βήμα, το switch s δεν πρέπει να είχε στείλει τίποτα από το bundle c. Αυτό μπορεί να συμβεί για έναν από δύο λόγους: Είτε το s έστειλε ένα πακέτο p_2 σε άλλο bundle, ή ένα από τα bundles εισόδου c' στο s δεν είχε τίποτα να στείλει. Στην πρώτη περίπτωση, τοποθετούμε τα p_2, $s2 = s$, και $r2 = rank(p_2) = rank(p_1)$ στην ακολουθία, και ακολουθούμε το p_2 πίσω στο χρόνο μέχρι να γίνει το τελευταίο που θα καθυστερήσει. Στην τελευταία περίπτωση, πάμε πίσω μέσω του c' σε ένα switch s' στο προηγούμενο επίπεδο κατά το προηγούμενο βήμα και να συνεχίζουμε. Συνεχίζοντας με αυτόν τον τρόπο, οφείλουμε τελικά να φτάσουμε σε ένα switch που έστειλε ένα πακέτο.

Τώρα που έχουμε αναλύσει τους τρόπους με τους οποίους ένα πακέτο μπορεί να καθυστερήσει, πρέπει να εξετάσουμε τους τρόπους με τους οποίους μπορεί να καθυστερήσει ένα token. Ας υποθέσουμε ότι ακολουθούμε ένα token t_1 πίσω στο χρόνο μέχρι να γίνει το τελευταίο που θα καθυστερήσει σε ένα switch s.

Αν το t1 καθυστερήσει επειδή το s έστειλε ένα πακέτο p αντί για αυτό, τότε εισάγουμε τα $p_2 = p$, $s_2 = s$, και $r_s = rank(p) = rank(t_1)$ στην ακολουθία. Διαφορετικά, αν το t_1 καθυστερήσει επειδή το s έστειλε ένα άλλο token t_1' αντί γιαυτό, τότε $rank(t_1') = rank(t_1) - 1$, και ακολουθούμε το t1' πίσω στο χρόνο μέχρι να γίνει το τελευταίο που θα καθυστερήσει. Η τελευταία πιθανότητα είναι ότι το t_1 καθυστέρησε εξαιτίας ενός bundle εισόδου c στο s, που δεν είχε τίποτα να στείλει. Σε αυτήν την περίπτωση, πηγαίνουμε πίσω μέσω του c ακρισβώς σαν να είχε καθυστερήσει ένα πακέτο για τον ίδιο λόγο.

Η παραπάνω ανάλυση των περιπτώσεων δείχνει ότι για κάθε πόντο καθυστέρησης που σπαταλήθηκε, είτε ένα πακέτο εισάγεται εντός της ακολουθίας καθυστέρησης, ή η βαθμολογία του επόμενου πακέτου στην ακολουθία μειώνεται, συνεπώς η ακολουθία πρέπει να περιέχει τουλάχιστον d - w πακέτα.

Θεώρημα 2: Για κάθε σταθερά k_1, υπάρχει μια σταθερά k_2 τέτοια ώστε η πιθανότητα κάποιο πακέτο να μην μπορεί να φτάσει στον ενδιάμεσο προορισμό του σε $k_2 * logN$ βήματα είναι το πολύ $\frac{1}{k_1}$.

Απόδειξη: Προκειμένου να βρούμε ένα φράγμα της πιθανότητας ένα πακέτο να καθυστερήσει πολύ, αρκεί μόνο να βρούμε ένα φράγμα της πιθανότητας να εμφανίστεί κάποια ακολουθία καθυστέρησης. Για να γίνει αυτό, απλά καταγράφουμε τον αριθμό των διαφορετικών ακολουθιών καθυστέρησης, και τον πολλαπλασιάζουμε με την πιθανότητα που έχουν ώστε οποιαδήποτε απο αυτές συμβεί.

Η ανάλυση για τη δεύτερη φάση της δρομολόγησης είναι παρόμοια. Αν απαιτείται η πρώτη φάση να ολοκληρωθεί πριν από την έναρξη της δεύτερης φάσης, τότε κάθε μια από αυτές χρειάζεται O(log N) bit βήματα, και άρα ο συνολικός χρόνος είναι O(log N) bit βήματα. Στην πραγματικότητα, είναι δυνατόν να γίνει κάποιο pipelining στις δύο φάσεις, έτσι ώστε ορισμένα πακέτα να ξεκινήσουν την δεύτερη φάση πριν τελειώσει η πρώτη. Το αλγόριθμος που προκύπτει είναι απλούστερος και εξακολουθεί να απαιτεί μόνο O (log N) bit βήματα. Η ανάλυση είναι λίγο πιο περίπλοκη επειδή ένα πακέτο p_1 μπορεί να προκαλέσει την καθυστέρηση ενός άλλου πακέτου p_2 στη πρώτη φάση, και το p_2 να καθυστερήσει το p_1 στην δεύτερη φάση. Μόλις το head ενός πακέτου φτάσει στον πραγματικό προορισμό του, τα M bits δεδομένων που ακολουθούν

φθάνουν σταδιακά στον προορισμό σε O(M) bit βήματα. Δεδομένου ότι, με μεγάλη πιθανότητα, όλα τα heads θα φθάσουν στον προορισμό τους σε O (log N) bit βήματα, ο συνολικός χρόνος θα είναι, με μεγάλη πιθανότητα, επίσης O(M + log N) .

7 Αντιστοίχιση σε υπερκύβο

Θυμίζουμε ότι παρόλο που οι αλγόριθμοι που περιγράφονται παραπάνω υλοποιούνται σε δίκτυο πεταλούδας, αρχικό έναυσμα για την ανάπτυξη τους ήταν η ανάγκη για γρήγορη δρομολόγηση σε υπερκύβο. Για αυτό το λόγο, σε αυτή την ενότητα δείχνεται πως ένας υπερκύβος με N κόμβους μπορεί να προσομοιώσει τον αλγόριθμο δρομολόγησης που εκτελείται από μια επεκτεταμένη κατά O(logN) πεταλούδα N κόμβων με σταθερό slow-down. Η ιδέα είναι να χωρίστεί η πεταλούδα στην μέση και να ενσωματωθεί κάθε μισό σε υπερκύβος με σταθερό φορτίο, συμφόρηση, και διαστολή. Τα δύο μισά είναι συνδεδεμένα μεταξύ τους με μια σειρά από μονοπάτών με σταθερή συμφόρησης και διαστολή $O(logN)^2$. Παρά το γεγονός ότι ένα πακέτο μπορεί να καθυστερήσει για O (log N)-bit βήματα, ενώ ταξιδεύει από το ένα μισό στο άλλο, ο συνολικός χρόνος παραμένει O (log N).

Η ενσωμάτωσή αυτή θα βασιστεί κατά το μεγαλύτερο μέρος στην διάσπαση της τοπολογίας αστέρα. Αυτό με τη σειρά του θα βασιστεί τους Κωδικες Διόρθωσης 1 Λαθους με τη χρήση κωδίκων που κατασκευάστηκαν για πρώτη φορά από το Shannon για συμβολοσειρές μήκους $2^r - r - 1$. Πιο συγκεκριμένα, για κάθε. λέξη w μήκους $2^r - r - 1$ υπάρχει μια κωδική λέξη, c(w), μήκους $2^r - 1$, έτσι ώστε η κάθε συμβολοσειρά μήκους $2^r - 1$ να έχει απόσταση Hamming το πολύ 1 από ακριβή κωδική λέξη.

Όπως γνωρίζουμε, ένας υπερκύβος είναι ένα δίκτυο, όπου ο αριθμός των κόμβων -έστω N- είναι μια δύναμη του δύο, δηλαδή $N = 2^s$ όπου ο εκθέτης s ονομάζεται "διάσταση του κύβου". Κάθε κόμβος έχει μια διεύθυνση των s bit και κάθε μη κατευθυνόμενη ακμή συνδέει κόμβους των οποίων η διεύθυνση διαφέρει ακριβώς κατά ένα bit.

Έστω ένας υπερκύβος με $s = 2^r - 1$ διαστάσεις. Ένας κόμβος, u είναι ένα κεντρικός αστέρας, εάν το u είναι κωδική λέξη: $u\epsilon\{c(w)|w\epsilon(0,1)^{2^r-r-1}\}$. Ένας αστέρας ορίζεται ως ο κεντρικός αστέρας μαζί με όλους τους γείτονές του. Ο κόμβος $c(w)^i$ διαφέρει από το c(w) στο i-οστό bit και ονομάζεται i-οστό άκρο του αστέρα c(w). Μιας και το c είναι ένας τέλειος κώδικας διόρθωσης 1 λάθους, κάθε κόμβος στον υπερκύβο είναι ακριβώς ένας αστέρας, δηλαδή, οι αστέρες διαχωρίζουν τους κόμβους του υπερκύβου. Μια ακμή ενός υπερκύβου είναι είτε μεταξύ ενός κεντρικού αστέρα και ενός άκρου του ή μεταξύ των άκρων διαφορετικών αστέρων.

Έστω $n = 2^k - 1$. Για λόγους απλότητας γίνεται η υπόθεση η πεταλούδα είναι $2n - k + 1$ από 2^{2n-k} και ότι κάθε νέα ακμή αποτελείται στην πραγματικότητα από n ακμές. Θα δείξουμε πως να την ενσωματώσουμε σε ένα υπερκύβο διάστασης 2n (σημειώνεται ότι ο υπερκύβος έχει περίπου τον διπλάσιο αριθμό κόμβων). Σαν πρώτο βήμα, θα ενσωματωθούν τα επίπεδα 0 έως n της πεταλούδας. Για να γίνει αυτό θα πρέπει πρώτα να δώθεί η ενσωμάτωση των κόμβων της πεταλούδας σε αυτά τα επίπεδα και στη συνέχεια να δωθεί μια αντιστοίχηση των ακμών της πεταλούδας.

Ένας κόμβος στα πρώτα $n+1$ επίπεδα της πεταλούδας μπορεί να περιγραφεί από το ζεύγος $(m \circ l, i)$, όπου $m\epsilon(0,1)^{n-k}$ και αποτελείται από τα MSB της γραμμής στην οποία ανήκει ο κόμβος, $l\epsilon(0,1)^n$ και αποτελείται από τα LSbits, και $0 <= i <= n$ είναι το επίπεδο στο οποίο ανήκει ο κόμβος. Δεδομένου ότι οποιαδήποτε διαδρομή στην πεταλούδα διέρχεται μέσω κόμβων με την ίδια τιμή του m για τα πρώτα n+1 επίπεδα, όλοι οι κόμβοι με την ίδια τιμή του m θα θεωρούνται ως υπο-πεταλούδαB_m. Η αντιστοίχηση των κόμβων από τις υπο-πεταλούδες σε κόμβους του υπερκύβου ορίζεται ως εξής:
$(mol, i) \longmapsto c(m)^i \circ l$, για $1 <= i <= n$. *Αντιστοίχισε το* $(m \circ l, 0)$ *(καθώς επίσης και* $(m \circ l, n)$ *από πριν) στο* $c(m)^i \circ l$.

Σημειώνεται ότι, αφού το m αποτελείται από n-k bits, το c(m) θα αποτελείται από n bits και συνεπώς ο υπερκύβος έχει διάσταση 2n. Έστω τότε C_l ένας υποκύβος, όπου τα LSbits όλων των κόμβων έχουν τιμή l. Μπορούμε εύκολα να παρατηρήσουμε ότι όλοι οι κόμβοι στη γραμμή l της κάθε υπο-πεταλούδας αντιστοιχίζονται σε έναν αστέρα του C_l. Στην πραγματικότητα, το C_l διαμερίζεται από τους αστέρες που σχηματίζονται από τη l-οστή γραμμή κάθε υπο-πεταλούδας. Έστω δύο γραμμές της Bm, που διαφέρουν κατά ένα bit, ας πούμε l και l^i. Η μια αντιστοιχίζεται στον αστέρα c(m) του C_l και η άλλη στον αστέρα c(m) του C_l^i, έτσι ώστε τα αντίστοιχα άκρα κάθε αστέρα να διαφέρουν στο i-οστό bit. Η αντιστοίχηση δείχνει ότι μπορούμε να χρησιμοποιήσουμε την i-οστή διάσταση του υπερκύβου για να δρομολογήσουμε τη i-οστή

διάσταση (το i-οστό επίπεδο δηλαδή) της πεταλούδας.

Τώρα αρκεί να δειχθεί το πώς μπορεί να γίνει η αντιστοίχηση των ακμών της πεταλούδας στις ακμές του υπερκύβου. Έστω η νέα ακμή μεταξύ του (l, i-1) και (l^i, i) στην υπο-πεταλούδα B_m για $2 <= i <= n$ (Θα χειριστούμε το $(l, 0) \longmapsto (l^1, 1)$ ξεχωριστά). Ο πρώτος κόμβος της ακμής αυτής είναι το i-οστό πρώτο άκρο του αστέρα c(m) στο C_l και το άλλο άκρο της ακμής, είναι το i-οστό άκρο του αστέρα c(m) στο C_l^i. Θα πρέπει οι νέες ακμές στο $_m$ να μοιραστούν μεταξύ των n ακμων μεταξύ των άκρων c(m) των αστέρων στους 2 υπο-κύβους. Έστω $(m \circ l, i-l) \longmapsto j(m \circ l^i, i)$ να είναι η j-οστή από τις νέες ακμές μεταξύ των (m ο l, i-l) και $(m \circ l^i, i)$ στην πεταλούδα. Αυτή η ακμή αντιστοιχίζεται στην ακόλουθη διαδρομή στον υπερκύβο. Ξεκινώντας με το $c(m)^{i-1}$ στο C_l θα πρέπει πρώτα να γίνει η μετάβαση στη θέση $c(m)^j$ μέσω $c(m)^{i-1,j}$. Στη συνέχεια επιλέγεται το άκρο το οποίο οδηγεί στο $c(m)^j$ από το C_l. Τέλος, επιλέγεται το $c(m)^i$ μέσω του $c(m)^{i,j}$ από το C_l^i. Πιο συνοπτικά:

$$c(m)^{i-1} \circ l \longrightarrow c(m)^j \circ l \longrightarrow c(m)^j \circ l^i \longrightarrow c(m)^{i,j} \circ l^i \longrightarrow c(m)^i \circ l^i$$

Στην συνέχεια πραγματοποιείται η ακόλουθη αλλαγή στην ακμή $(l, 0) \longrightarrow j(l^1, 1)$:

$$c(m)^n \circ l \longrightarrow c(m)^{n,j} \circ l \longrightarrow c(m)^j \circ l \longrightarrow c(m)^j \circ l^1 \longrightarrow c(m)^{1,j} \circ l^1 \longrightarrow c(m)^1 \circ l^1$$

Μια ακμή στο δίκτυο της πεταλούδας $(m \circ l, i-1) \longrightarrow j(m \circ l, i)$, $2 <= i <= n$, αντιστοιχίζεται με παρόμοιο τρόπο εκτός από το ότι δεν χρειάζεται να αλλάξουμε το i-οστό bit του l:

$c(m)^{i-1} \circ l \longrightarrow c(m)^{i-1,j} \circ l \longrightarrow c(m)^j \circ l \longrightarrow c(m)^{i,j} \circ l \longrightarrow c(m)^i \circ l$ για $2 <= i <= n$. Χρειάζεται και εδώ η προφανής αλλαγή για i=1:

$$c(m)^n \circ l \longrightarrow c(m)^{n,j} \circ l \longrightarrow c(m)^j \circ l \longrightarrow c(m)^{1,j} \circ l \longrightarrow c(m)^1 \circ l$$

Ισχυρισμός 3: *Το πολύ 16 ακμές από την πεταλούδες αντιστοιχίζονται μέσω οποιασδήποτε ακμής του υπερκύβου από την προηγούμενη ενσωμάτωση.*

Απόδειξη: Εκ κατασκευής, γνωρίζουμε ότι οι ακμές της πεταλούδας δρομολογούνται μόνο μέσω ακμών που συνδέουν άκρα από διαφορετικούς αστέρες. Υπάρχουν δύο περιπτώσεις που πρέπει να μελετηθούν, ανάλογα με το αν είναι ή όχι οι αστέρες σε διαφορετικούς υποκύβους

Περίπτωση 1: Υπερκύβοι της μορφής:$(c(\underline{m})^{\underline{j}} \circ \underline{l}, c(\underline{m})^{\underline{j}} \circ \underline{l^i})$: Μια τέτοια ακμή πρέπει να είναι η τρίτη ακμή του υπερκύβου που χρησιμοποιείται ως μια ακμή διέλευσης στην πεταλούδα. Επιπλέον, η ακμή της πεταλούδας πρέπει να είναι της μορφής $(m \circ 1, i-1) \longrightarrow j(m \circ l^i, i)$ όταν $i = \underline{i}, l = \underline{l}$ ή $l = \underline{l^i}, j = \underline{j}$ και $m = \underline{m}$. Ως εκ τούτου, αυτή η ακμή του υπερκύβου χρησιμοποιείται από το πολύ δύο ακμές της επεκτεταμένης πεταλούδας.

Περίπτωση 2: Υπερκύβοι της μορφής: $(c(\underline{m})^{\underline{i}} \circ l, c(\underline{m})^{\underline{(i,j)}} \circ \underline{l})$:
Μια τέτοια ακμή μπορεί να είναι η πρώτη, η δεύτερη, η τέταρτη, ή η πέμπτη ακμή για ένα μονοπάτι από $(m \circ l, i-l) \longrightarrow j(m \circ l^i, i)$, ή οποιαδήποτε άλλη από τις τέσσερις ακμές σε ένα μονοπάτι για το $(m \circ l, i-1) \longrightarrow j(m \circ l, i)$. Σε κάθε περίπτωση, υπάρχουν δύο πιθανές επιλογές για τις τιμές των m, l, i, και j, ανάλογα με την κατεύθυνση στην οποία χρησιμοποιείται η ακμή. Ως εκ τούτου, κάθε τέτοια ακμή χρησιμοποιείται το πολύ 16 φορές.

Τα υπόλοιπα επίπεδα της επεκτεταμένης πεταλούδας μπορούν να αντιστοιχίζονται στον υπερκύβο κατ' ανάλογο τρόπο. Ειδικότερα, μπορεί να αντιστοιχηθεί το τελευταίο $n+1$ επίπεδο, της πεταλούδας στον υπερκύβου χρησιμοποιώντας τον ίδιο αλγόριθμο όπως πριν, εκτός από το ότι ο κόμβος της πεταλούδας $(m' \circ l', i')$ αντιστοιχίζεται στο κόμβο $c(l')^(i' - n + k)$ ο m του υπερκύβου όπου $m' \epsilon \{0,1\}^n, l' \epsilon \{0,1\}^{n-k}$ και $n - k <= i' <= 2n - k + 1$. (Σημειώνεται ότι απλά αλλάξαμε τους ρόλους των MSbits και των LSbits). Οι ακμές της πεταλούδας αντιστοιχίζονται με ανάλογο τρόπο.

Δυστυχώς, η μέθοδος ενσωμάτωσης που μόλις περιγράφηκε αντιστοιχίζει κάθε κόμβο στα μεσαία $k+1$ επίπεδα της πεταλούδας σε δύο διαφορετικά μέρη. Μπορούμε να επιλύσουμε αυτό το πρόβλημα χρησιμοποιώντας μόνο τη θέση που προβλέπεται από την πρώτη ενσωμάττωση των κόμβους από τα επίπεδα $n-k$ έως n.

Μέχρι στιγμής έχει μείνει το πρόβλημα της ενσωμάτωσης των ακμών από το επίπεδο n στο επίπεδο

$n+1$, δεδομένου ότι οι κόμβοι στο επίπεδο n έχουν ενσωματωθεί στα πρώτα $n+1$ επίπεδα της πεταλούδας, καθώς και οι κόμβοι στο επίπεδο $n+1$ έχουν ενσωματωθεί στα τελευταία $n+1$ επίπεδα της πεταλούδας. Όμως, δεν είναι γνωστό πως να ενσωματώθούν αυτές τις ακμές με σταθερή διαστολή, αλλά μπορούν να ενσωματώθούν με σταθερή συμφόρησης και O (logN) διαστολή ως εξής: Υπενθυμίζεται ότι οι κόμβοι στο επίπεδο n της πεταλούδας αντιστοιχίζονται ένας-προς-έναν στα n-οστά άκρα των αστέρων, ενώ οι κόμβοι του επιπέδου $n+1$ αντιστοιχίζονται ένας-προς-ένας με τα $k+1$ άκρα των άστέρων. Η διαδρομή για την j-οστή ακμή ξεκινά με μια διαδρομή στο j-οστό άκρο του αστέρα. Αυτό μπορεί να γίνει με O(1) συμφόρηση και διαστολή 2, τόσο για τους κόμβους του επιπέδου n όσο και του επιπέδου $n+1$ χρησιμοποιώντας τα μονοπάτια

$$c(m)^n \circ l \longrightarrow c(m)^{n,j} \circ l \longrightarrow c(m)^j \circ l$$

ή

$$c(m)^{k+1} \circ l \longrightarrow c(m)^{k+1,j} \circ l \longrightarrow c(m)^j \circ l$$

Μιας και κάθε ένα από αυτά τα άκρα των αστέρων είναι διαφορετικό, οι διαδρομές μπορεί να ολοκληρωθούν σε 0(1) συμφόρηση και O(log N) διαστολή χρησιμοποιώντας την δρομολόγηση του Benes. Ο υπερκύβος μπορεί τώρα εύκολα να προσομοιώσει τους αλγόριθμους δρομολόγησης της ενότητας 2 για δίκτυα πεταλούδας. Κάθε κόμβος του υπερκύβου προσομοιώνει το πολύ 2 κόμβους της πεταλούδας. Επίσης, δεδομένου ότι τα πρώτα n επίπεδα των ακμών της πεταλούδας είναι ενσωματομένα με σταθερή συμφόρηση και διαστολή, ο υπερκύβος μπορεί να προσομοιώσει την μετάδοση σε αυτές τις ακμές με σταθερή καθυστέρηση. Το ίδιο ισχύει και για τα τελευταία n-k-1 επίπεδα των ακμών της πεταλούδας. Μόνο οι ακμές από το επίπεδο n στο n + 1 μπορούν να προκαλέσουν προβλήματα. Δεδομένου ότι αυτές οι ακμές έχουν ενσωματωθεί με διαστολή O(logN), απαιτούνται O (logN) bit βήματα για να προσωμειωθεί η μετάδοση σε αυτές. Ευτυχώς, η αλγόριθμοι δρομολόγησης χρησιμοποιούν τις ακμές της πεταλούδας σε αύξουσα διάσταση από 0 σε 2n - k - 1, οπότε η καθυστέρηση O(log N) βημάτων που δημιουργείται κατά μήκος των ακμών από το επίπεδο n στο n + 1 είναι προσθετική.

8 Κάτω φράγμα για oblivious routing

Σε αυτή τη φάση θα δειχθεί ότι κάθε oblivious αλγόριθμος δρομολόγησης χρειάζεται $\Omega(\frac{(M+\frac{logN}{logd}logN}{(logd+loglogN})$ bit βήματα με μεγάλη πιθανότητα για σχεδόν όλες τις μεταθέσεις, όπου M είναι το ελάχιστο μέγεθος του πακέτου, χωρίς σε αυτή να συμπεριλαμβάνονται οι πληροφορίες για την δρομολόγηση του πακέτου.

Το αποτέλεσμα επεκτείνει το κλασσικό κάτω φράγμα των Borodin-Hopcroft με δύο τρόπους. Πρώτον, το κάτω φράγμα ισχύει για σχεδόν όλες τις μεταθέσεις, και το δεύτερο, εφαρμόζεται τόσο σε ντετερμινιστικούς όσο και σε βασισμένους στην τυχαιότητα oblivious αλγόριθμους. Η μοναδική υπόθεσή που γίνεται είναι ότι κάθε πακέτο έχει τουλάχιστον Δ bit πληροφοριών διευθυνσιοδότησης, εάν βρίσκεται σε μια απόσταση Δ μακριά από τον προορισμό του. Χωρίς αυτή την υπόθεση, το κάτω φράγμα είναι $(MlogN/(logd + log1logN))$ bit βήματα.

Με την πρώτη ματιά, αυτό το κατώτερο όριο φαίνεται να είναι σε αντίθεση με αυτό του αλγορίθμου των O (log N) bit βημάτων για την O (log N)-dilated πεταλούδα που περιγράφηκε στην ενότητα 2. Αν και ο αλγόριθμος στην ενότητα 2 είναι node-oblivious, δεν είναι όμως edge-oblivious. Με άλλα λόγια, η επιλογή των ακμών που χρησιμοποιούνται σε μια διαδρομή ενός πακέτου εξαρτάται από τις επιλογές γίνονται από άλλα πακέτα, ακόμη και αν οι κόμβοι στην διαδρομή επιλέγονται με oblivious τρόπο. Όσον αφορά τη δρομολόγηση στον υπερκύβο, δεν υπάρχει διαφορά μεταξύ του edge-oblivious και του node-oblivious αλγόριθμου επειδή δεν υπάρχουν πολλαπλές ακμές. Δεδομένου ότι $d = logN$ για το υπερκύβο και το M είναι τυπικά Ω(logN), θα διαπιστωθεί ότι κάθε oblivious αλγόριθμος στον υπερκύβο N-κόμβων χρειάζεται $\Omega(log^2 N/loglogN)$ bit βήματα με μεγάλη πιθανότητα για τις περισσότερες μεταθέσεις.

Για την απόδειξη του φράγματος, κάθε σύνδεση σε ένα δίκτυο με N κόμβους αναπαριστάται με δύο κατευθυνόμενες ακμές στον υποκείμενο γράφο. Ένα άνω όριο για τον αριθμό των ακμών είναι επομένως Nd. Έστω p μια ακολουθία ακμών οι οποίες σχηματίζουν ένα μονοπάτι (όχι απαραίτητα απλό) στο γράφημα, και έστω $P_{u,v}$ το σύνολο όλων των μονοπατιών από το u στο v. Για έναν δεδομένο αλγόριθμο δρομολόγηση που βασίζεται στην τυχαιότητα, έστω $P_{u,v}$ η τυχαία μεταβλητή που δηλώνει τη διαδρομή μεταξύ u και v, που επιλέγεται από τον αλγόριθμο. Χρησιμοποιώντας αυτόν τον συμβολισμό, για κάθε κόμβο ή ακμή x και για όλους τους κόμβους u και v ισχύει ότι:

$$\sum_{p \epsilon P_{u,v}} Pr[P_{u,v} = p] = 1,$$

και

$$Pr[x \epsilon P_{u,v}] = \sum_{p: x \epsilon p} Pr[P_{u,v} = p] = 1$$

Ένας αλγόριθμος δρομολόγησης που βασίζεται στη τυχαιότητα είναι oblivious αν όλες οι διακριτες τυχαίες μεταβλητές $P_{u,v}$ είναι ανεξάρτητες.

Για κάθε κόμβο v, οποιοδήποτε σύνολο κόμβων T, και οποιοδήποτε κόμβο ή ακμή x, το βάρος (weight) του x σε σχέση με u και το T, ορίζεται ως:

$$W_x(v, T) = \sum_{t \epsilon T} Pr[x \epsilon P_{v,t}].$$

Η τιμή του $W_x(v, T)$ μπορεί να ερμηνευτεί με πολλούς τρόπους, όμως η ερμηνεία που θα φανεί χτήσιμη σε εμάς είναι ότι εάν επιλέξουμε ένα τυχαίο προορισμό $t \epsilon T$ και ένα τυχαίο μονοπάτι από το v στο t σύμφωνα με την κατανομή $P_{v,t}$, τότε το x θα πρέπει να περιέχεται στη διαδρομή με πιθανότητα $\frac{W_x(v,T)}{|T|}$. Ομοίως, αν έχει επιλεγεί μια τυχαία διαδρομή από v σε έναν τυχαία επιλεγμένο προορισμό στο V, τότε το x θα πρέπει να ανήκει στο μονοπάτι με πιθανότητα τουλάχιστον $\frac{W_x(v,T)}{N}$.

Ο στόχος εδώ είναι να δείχθεί ότι υπάρχουν πολλές ακμές e που έχουν πολλούς κόμβους v, για τους οποίους το $W_e(v, T)$ είναι μεγάλο για κάποιο T. Τότε θα είμαστε σε θέση να υποστηρίξουμε ότι πολλές ακμές έχουν τουλάχιστον κάποια πιθανότητα να έχουν πολλά πακέτα που περνούν μέσα από αυτές.Στη συνέχεια, θα δείξουμε επίσης ότι τουλάχιστον μια από αυτές τις ακμές έχει υψηλή συμφόρηση με πολύ υψηλή πιθανότητα.

Αρχικά αποδεικνύεται ότι για κάθε κόμβο v και κάθε μεγάλο σύνολο T, υπάρχουν πολλά u για τα οποία η τιμή του $W_u(u, T_{u,v})$ είναι μεγάλη, όπου $T_{u,v}$ είναι υποσύνολο του T και ο u είναι πολύ μακριά από το $T_{u,v}$. Στο εξής, θα χρησιμοποιείται ο συμβολισμός dist(x, T) για την ελάχιστη απόσταση μεταξύ του x και κάθε στοιχείου του T στο γράφημα. Χωρίς απώλεια της γενικότητας, θα γίνει επίσης η υπόθεση ότι $d = N^{o(1)}$.

Λήμμα 4:Για κάθε σύνολο T, με τουλάχιστον $\frac{}{2}$κόμβους, και οποιοδήποτε κόμβο u, υπάρχουν τουλάχιστον $\frac{\sqrt{N}}{2\sqrt{d}}$ κόμβοι u για τους οποίους $W_u(v, T_{u,v}) >= \frac{\sqrt{N}}{4\sqrt{d}}$ και $dist(u, T_{u,v}) >= \Omega(\frac{logN}{logd})$ για κάποιο $T_{u,v}$ υποσύνολο του T.

Στην συνέχεια αποδεικνύεται ότι υπάρχει μια ακμή e, για την οποία το $W_e(u, T_{e,v})$ είναι μεγάλο για πολλλα v, όπου το $T_{e,v} \subseteq T$ είναι πολύ μακριά από την e.

Λήμμα 5: Για οποιαδήποτε δύο σύνολα S και T με τουλάχιστον N/2 κόμβους το καθένα, υπάρχει μια ακμή e, και τουλάχιστον $\frac{\sqrt{N}}{4d^{3/2}}$ κόμβοι $v \epsilon S$ τέτοιοι ώστε:

$W_e(v, T_{e,v}) \geq \frac{\sqrt{N}}{4d^{3/2}}$

όπου $T_{e,v} \subseteq T$ και $dist(e, T_{e,v}) \geq \Omega(logN/logd)$

Πλέον όλα είναι έτοιμα για την απόδειξη του για την απόδειξη του κάτω φράγματος των oblivious αλγορίθμων.

Θεώρημα 6: Κάθε oblivious αλγόριθμος δρομολόγησης χρειάζεται $\Omega(\frac{(M+\frac{logN}{logd} logN}{(logd+loglogN)}$ bit βήματα με πιθανότητα τουλάχιστον ίση με $1 - e^{-N^{1/4}}$ για να δρομολογήσει όλα , αλλά ένα $e^{-N^{1/4}}$ μέρων των N! πιθανοί μεταθέσεων.

Απόδειξη: Θα δείξουμε ότι για μια τυχαία μετάθεση, η πιθανότητα κάθε ακμή να έχει λιγότερα από c*logN / (log d + log log N) πακέτα που διέρχονται από αυτήν είναι Ω (log N / log d) βήματα μακριά από τον προορισμό τους είναι το πολύ $e^{-2N^{1/4}}$ για μια αρκετά μικρή σταθερά c. Στην αυτή την περίπτωση, η πιθανότητα λαμβάνεται από την την επιλογή των μεταθέσεων, καθώς και την επιλογή των μονοπατιών. Αυτό απευθείας συνεπάγεται το θεώρημα εάν περισσότερα από ένα $e^{-1/4}$ μέρη των πιθανών μεταθέσεων τελειώσουν σε λιγότερα από O((M + (log N / log d)) c logN / (log d + log log N)) βήματα bit με πιθανότητα που υπερβαίνει την$e^{-N^{1/4}}$, τότε θα έχει περισσότερες από $e^{-2N^{1/4}}$ ευκαιρίες να ολοκληρώσει σε λιγότερα βήματα με μια τυχαία μετάθεση πράγμα το οποίο δεν είναι δυνατόν.

Εδώ κατασκευάζεται μια τυχαία μετάθεση προσδιορίζοντας μία διαδρομή κάθε φορά. Κάθε φορά επιλέγεται τυχαία ένα μονοπάτι για κάθε κόμβο v σε μια καθορισμένη ακολουθία, επιλέγοντας μονοπάτι από τις κατανομές$P_{u,v}$, όπου το u επιλέχτηκε τυχαία από τους κόμβους που δεν έχουν ήδη οριστεί ώστε να λάβει κάποιο πακέτο. (Ο περιορισμός, της κατασκευής μιας τυχαίας μετάθεσης και όχι ένός τυχαίου προορισμού προϋποθέτει ότι το πολύ ένα πακέτο θα παραδοθεί σε κάποιον κόμβο. Το επιχείρημα για τους τυχαία προορισμούς είναι κάπως πιο εύκολο).

Η ανάλυση ξεκινάει επιλέγοντας μια ακμή e και $m = \sqrt{N}/4d^3/2$ κόμβους $u1, u2, ...um \epsilon S$ όπου S =T=V, τέτοιο ώστε για $1 <= i <= m$,

$W_e(v_i, T_{e,v_i}) \geq \frac{\sqrt{N}}{4d^{3/2}}$ όπου $T_{e,v_i} \subseteq T$ και $dist(e, T_{e,v_i}) \geq \Omega(logN/logd)$

Σύμφωνα με το Λήμμα 5, μια τέτοια ακμή και ένα τέτοιο σύνολο των κόμβων θα υπάρχει πάντα. Πρώρα επιλέγεται ένας τυχαίος προορισμός t_1 από το T και μια τυχαία διαδρομή p_1 από το Pu_1, t_1 για τον u1. Δεδομένου ότι τα $W_e(u1, T) >= W_e(u1, Te, u1)$, γνωρίζουμε ότι η διαδρομή $p1$ περιέχει την e και συνεχίζει αργότερα για τουλάχιστον $\Omega(logN/logd)$ βήματα με πιθανότητα τουλάχιστον $1/4d^{3/2}\sqrt{N}$.

Στην συνέχεια επιλέγουμε ένα τυχαίο προορισμό t_2 από το T - t_1 και μια τυχαία διαδρομή p_2 από το Pu_2, t_2για τον u_2. Αφού:

$W_e(v_2, T - t_1) \geq W_e(v_2, T_{e,v_2} - t_1) \geq W_e(v_2, T_{e,v_2}) - 1 \geq \frac{\sqrt{N}}{4d^{3/2}} - 1 \geq \frac{\sqrt{N}}{8d^{3/2}}$

γνωρίζουμε ότι η p_2 χρησιμοποιεί την e και συνεχίζει αργότερα για τουλάχιστον $\Omega(logN/logd)$ βήματα μέτρα με πιθανότητα τουλάχιστον$1/8d^{3/2}\sqrt{N}$. Επιπλέον, γνωρίζουμε ότι αυτή η πιθανότητα είναι ανεξάρτητη της πιθανότητας το p_1 να χρησιμοποιεί την e και να συνεχίζει αργότερα για $\Omega(logN/logd)$ βήματα.

Με παρόμοιο τρόπο, μπορεί να δείχθεί ότι τα $p_1, ...p_{m/2}$χρησιμοποιούν την e και στη συνέχεια για τουλάχιστον $\Omega(logN/logd)$ βήματα με πιθανότητα $1/8d^{3/2}\sqrt{N}$, και ότι αυτές οι πιθανότητες είναι ανεξάρτητες. Αφού υπάρχουν $\sqrt{N}/8d^{3/2}$ μονοπάτια, καθένα από τα οποία χρησιμοποιεί την e και συνεχίζουν μετά γα $(logN/logd)$ βήματα με πιθανότητα $1/8d^{3/2}\sqrt{N}$ η πιθανότητα ότι λιγότεροι από το r = clogN / (log d + log log N) διαδρομές να χρησιμοποιούν την e και να συνεχίζουν για $\Omega(logN/logd)$ βήματα είναι το πολύ:

$1 - (\frac{1}{64d^3 r})^r = 1 - 2^{-O(c*logc*logN)}$

Επιλέγοντας το c να είναι μια αρκετά μικρή σταθερά, η πιθανότητα αυτή μπορεί να γίνει το πολύ $1–N^{-1/4}$. Με άλλα λόγια, τουλάχιστον $\Omega((+logN/logd)logN/(logd + loglogN))$ bits διέρχονται μέσα από την e από τα πακέτα που ξεκινούν από τους κόμβους $u_1, u_2, ...u_{m/2}$ με πιθανότητα τουλάχιστον $N^{-1/4}$.

Μπορούμε τώρα να αρχίσουμε ξανά από την αρχή, θέτοντας $S = V - \{u_1, .., u_{m/2}\}$ και $= V - t_1, ..., t_{m/2}$. Αφου $m = \sqrt{N}/4d^{3/2}, |S| = |T| >= N/2$ και μπορούμε να χρησιμοοπιήσουμε το Λήμμα 5 για να βρούμε μια ακμή e' και κόμβους $u'_1, ...u'_{m/2} \epsilon S$ τέτοιο ώστε για 1<=i<=m

$W_{e'}(u'_i, T_{e'}, u_{i'}) \geq \frac{\sqrt{n}}{4d^{3/2}}$ όπου $_{e',u'} \subseteq T$ και $dist(e', _{e',u'_I}) >= \Omega(logN/logd)$. Υποστηρίζοντας όπως και πριν (εκτός από το ότι τώρα το ti είναι επιλέγεται ομοιόμορφα από την $T - t'_1, ..., t'_{i-1} = V - t_1, ..., t_{m/2}, t'_1, ...t'_{i-1}$, μπορούμε να αποδείξουμε ότι τουλάχιστον, $\Omega((+logN/logd)logN/(logd + loglogN))$ bits έχουν περάσει από την e' από τα πακέτα που ξεκινούν από τους κόμβους$u'_1, .., u'_{m/2}$ με πιθανότητα τουλάχιστον $^-1/4$.

Τώρα, όλη αυτή τη διαδικασία επαναλάμβανεται $(N/2)/(m/2) = 4d^{3/2}\sqrt{N}$ φορές, επιλέγοντας κάθε φορά τους προορισμούς και τα μονοπάτια για m/2 περισσότερους κόμβους. Κάθε φορά γίνεται σοβαρή συμφόρηση σε μια ακμή με πιθανότητα τουλάχιστον $N^{-1/4}$. Αφού αυτές οι πιθανότητες είναι όλες ανεξάρτητες, η πιθανότητα ότι καμία από τις ακμές δεν θα υποστεί σοβαρή συμφόρηση είναι το πολύ

$(1 - N{-}1/4)^{4d^{3/2}\sqrt{N}} \leq e^{-4d^{3/2}N^{1/4}} \leq e^{-2N^{1/4}}$

Ως εκ τούτου, για μια τυχαία μετάθεση και μια τυχαία επιλογή διαδρομής, ο χρόνος θα είναι $\Omega((M + logN/logd)logN/(logd + loglogN))$ με πιθανότητα $1 - e^{-2N^{1/4}}$.