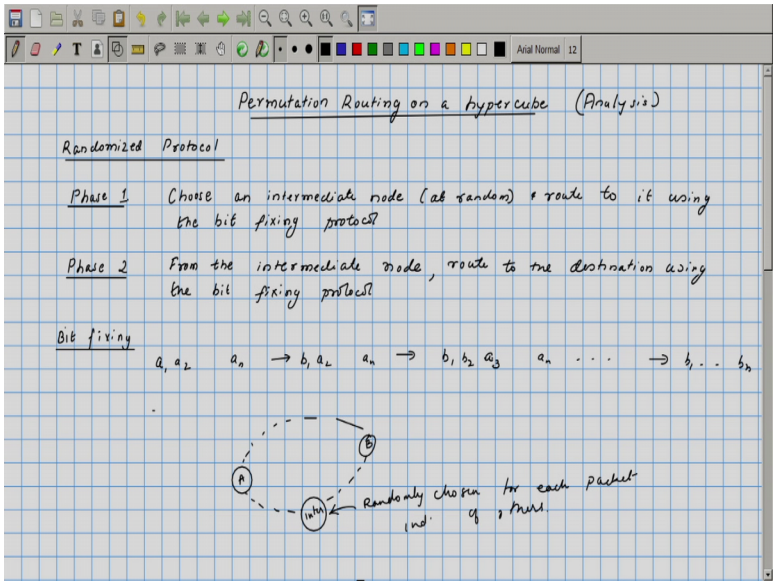
Randomized Algorithm
Prof. Benny George Kenkireth
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 12
Permutation Routing on a Hypercube (Analysis)

In this lecture we will see the Analysis of Permutation Routing on a Hypercube.

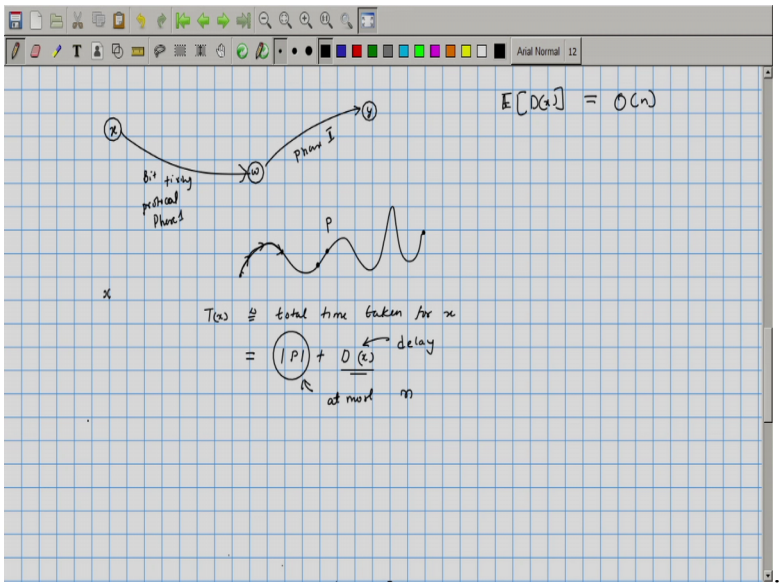
(Refer Slide Time: 00:37)



So, we have randomized protocol which basically chooses an intermediary note at random and the packet from any node is being first sent to the intermediary node using the bit fixing protocol and then, the next phase from the intermediary note, the packet is sent to the destination again using the bit fixing protocol. So, bit fixing protocol basically fixes 1 beta time, you had a packet at node a 1 a 2 a n which is need to be sent to b 1 b 2 b n. It is first sent to the node b 1 a 2 a n followed by b 1 b 2 a 3 a n so on. So, each bit at a time if fix and set.

We will show that if we do this kind of randomization, it will improve the performance. We will see that there do not be any bad instances ok. So, what we need to do is look at the maximum possible delay that can happen in an expected sense.

(Refer Slide Time: 01:43)



So, if you had a packet at node x which we will call as packet x and this needs to be sent to some node y, we are doing it via an intermediary node let us call it as w. So, this transmission which involves lot of intermediary vertices in between is done entirely using the bit fixing protocol and that is your phase 1. And again same thing is done and that is the phase 2 where in the packet sent from the randomly chosen intermediary note w to y.

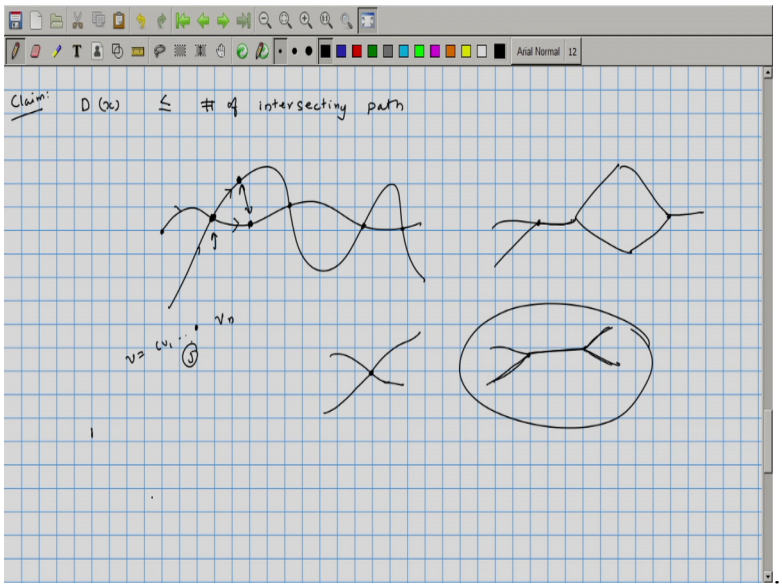
We will just analyse the phase 1, phase 2 analysis exactly same we can think of restarting phase 1 once again, but for this we will have to assume that we will start phase 2 only after all the packets have reached their intermediary node which we can find of assume, ok. So, whatever is the bounds that we can obtain for the transmission time for phase 1 multiplied into 2 and you will get the bound for phase 2. Let us just look at the phase 1 more carefully.

So, if you have a packet x let us say that it goes from its source to destination along a bit fixing path, ok. So, let us call this is the bit fixing path. What happens is when it is making these transitions from one node to other, it may incur delays its certain nodes. It may have to wait at certain intermediary nodes. So, the time it spends travelling, we will call that is travel time and the time with waits, we will call it as delay time. So, total time taken for any particular packet to go from its source to destination as going to be if you call it as T x. T x denotes the total time taken for x, then x is equal to length of this path

plus the delay. So, we will delay; we will denote by D x. This is the time it spends in waiting, ok.

And this quantity length of path this is at most n. If we were routing on n hypercube, then any bit fixing path cannot have length more than n. So, this at most n so, total time taken is n plus delay. So, we will show that the expected delay is going to be less than O of n or it is we will bound it O of n. So, how do we show this?

(Refer Slide Time: 04:51)



What we will show is that delay, these are claim; delay of any node is less than number of intersecting paths. Why does a packet incur delays? Because it is intersecting with some particular some other path if there were no intersection of course delay would not, there would not be any delay, but we can say that the number of intersecting paths there is an upper bound on the delay. How do we show this? And we need to look at the structure these in intersecting paths more carefully.

So, let us say this is 1 bit fixing path, this is another bit fixing path. Can it intersect in this particular fashion? It s intersecting at let us say three different places. Each of this is the vertex maybe this is another form of intersection. They come together at a vertex, stay it together for sometime and then their paths diverge and again they come together, they stay together, ok. These are how paths can usually intersect, but or these possible? We will show the only kind of intersections possible are of this kind. You can either meet

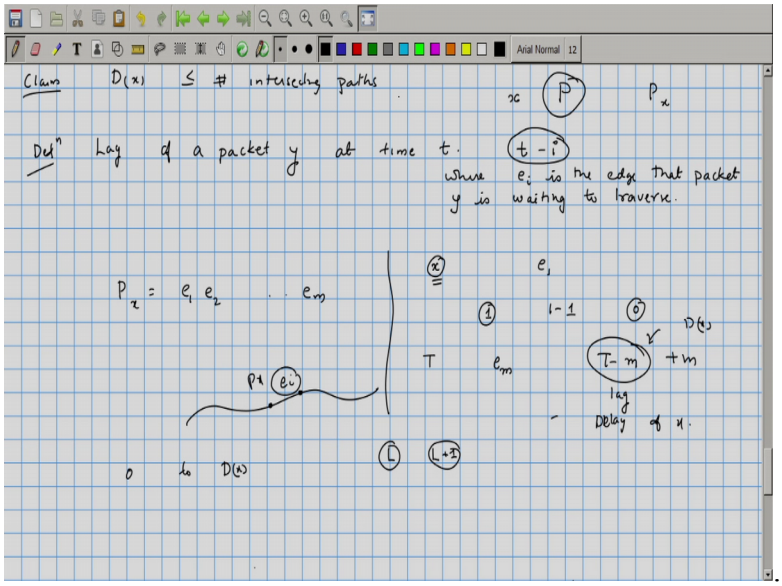
at a vertex and diverges or we can meet at a vertex stay together for a while and then diverges, straight forward because all these paths are bit fixing paths.

Suppose let us say they diverges, that means there is at least 1 node here which will agree on the initial coordinates with the final destination and these two by virtue of the paths diverging these have different coordinates, if you look at the initial path. So, let us say this was a vertex v is equal to v 1 v n, and let us say for both these things up to some when all the vertices up to jth position were fixed.

Now, the fact that they next vertex means that one more vertex got fixed and that vertex is one more coordinate got fixed. If we look at let us say the bitwise representation of the destination, one more vertex, I mean one more coordinate got fixed and they are different and therefore, they cannot come back together. So, that means that if they diverges, then they diverge forever so, this is one possibility. Of course they can come together and they can stay together for a while and then, they diverge, they diverge, they cannot again come to other again.

So, this is a fact about the intersecting path that we will use later on. They intersect at one place in one contiguous block and after that they do not intersect, they just diverge, ok. So, that is the kind of paths that are there in bit fixing protocol, but what we want to claim is D x is less than the number of intersecting paths.

(Refer Slide Time: 08:07)



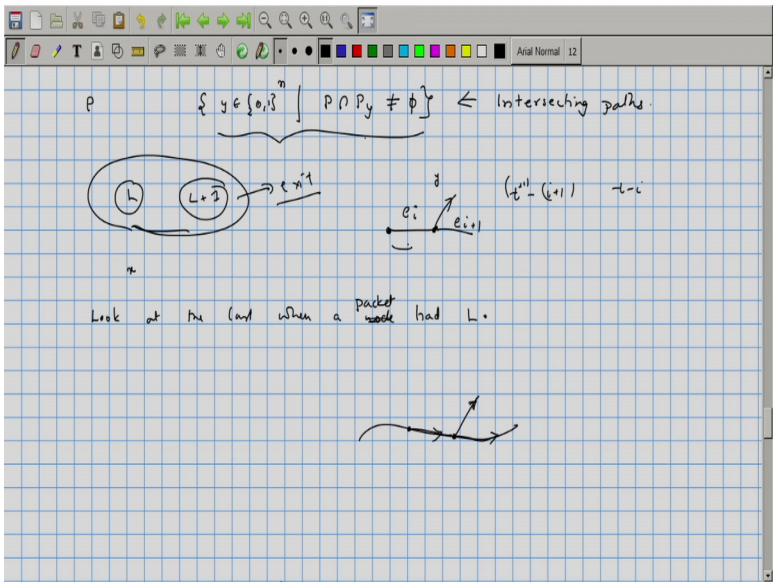
For proving that we will introduce the term called lag of a packet say y at time t, ok. So, we are interested in packets that share the path with x.

So, if you think of P as the path for the packet x or P x is the path for packet x, we are looking at all the other packets which intersect with this and for each of them we will introduce a term call lag at time t, ok. So, this is just nothing, but this define to be t minus i where e i is the edge that packet y is waiting to traverse, ok. So, if you think of this path P x as say e 1 e 2 e m that is this path Px and if you look at some particular edge e i and suppose another packet is waiting to travel this edge e i. Whatever be the for every packet that intersects the path of x, it may either keep on travelling or it might wait for some particular edge to be available for traversal, ok. So, if that edge is e i, then t minus i is called as the lag of this packet y.

Let us understand the definition of lag little more carefully. If you look at the packet x itself at the very start it is waiting to traverse edge e 1 and time is t equals 1. So, 1 minus 0 is its lag whereas, at the very end let us say it takes T time to complete its traversal. At time T just before it finishes what happens it is waiting to traverse the edge e m, ok. So, its lag is t minus m, ok. So, this is the lag, this is nothing, but the delay that x is incurred.

Because t minus m plus m is a total time that it has taken. This is what we had defined as delay, ok. So if you look at the packet x, its delay increase from 0 to D x. This is true for any particular packet. Its delay will increase from 0 to D x, what we will show is each of these delay. So, if you think about this lag, the lag is slowly increasing from 0 to D x, we will attribute each of that increase let us say it increases from L to L plus 1, ok. We will find the unique packet to which we can attribute this increase and that unique packet will belong to the collection of intersecting paths, ok.

(Refer Slide Time: 11:59)



So, suppose P is any path and let us say that we have the set of all y belonging to 0 1 to the power n such that P intersection P y is not empty, ok. So, this is the intersecting paths, ok. So, for each element in this mean for every increase from L to L plus 1, we will find one element inside this to which we can attribute this increase in delay. And therefore, the total increase in delay is going to be at most the size of the intersecting path and that will basically help us prove this claim that delay is less than the number of intersecting paths, ok.

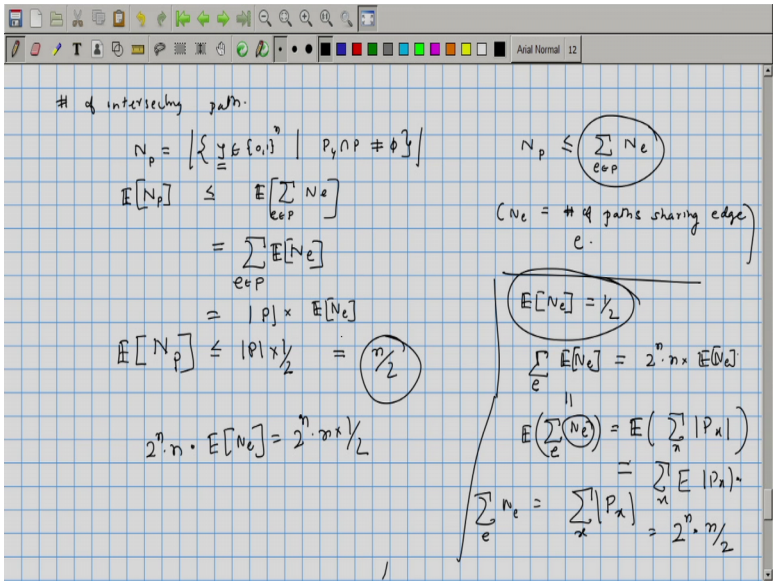
Now, let us see how the lag propagates through the system. If the lag of the vertex x or the packet x is increasing from L to L plus 1, that means there is some other packet which is traverse in this particular edge, ok. Let us say that was some edge e i. So, e i being traversed, it was being used by some other packet and the packet the x is waiting on that, ok. When the weight is complete, the lag goes from L to L plus 1 whereas, a lag for x increases, means the lag for the packet which traversed will remain the same because let us say this is packet y which is traversing its lag is t minus say the next edge if it is taking the edge i plus 1. So, t plus 1 minus i plus 1 which is again t minus i. So, its lag is not increasing of course these are the possibility that after this it just exceeds, ok. It does not wait for e i plus 1. It exit just exceeds in which case its lag is I mean we can say it is undefined, ok.

But if it is there in this collection, then either the lag remains the same or it increases by 1, this is the only possibility ok. So, we can look at I mean if the log of x was increasing from L to L plus 1, we know that there are other packets whose lag is remaining at L, ok. So, look at the last time. So, look at the last time when a node had or when a packet had lag L, ok.

That means at the next instance it is exiting the path. We had this path and this was a packet which moved across this and then, at the next instance it is exiting, ok. Well if it remains in the system, of course if it remains along this path, it means the lag is not changing. There are other nodes whose lag is L. So, we know that there should be a last time when there is some packet whose lag is L, ok. To that packet we can attribute the increase from L to L plus 1 and since, each path intersects either of this kind or the kind where it stays together and diverges, we know that the exciting happens only once, ok.

So, since we attribute this change to an exit, we can guarantee that all the increases are essentially belonging to one unique element inside this intersecting paths. Therefore, the total number of intersecting paths is a bound on the total delay, ok. So, as the first part now we have to bound the number of intersecting paths, ok.

(Refer Slide Time: 16:25)



So, let us look at any path and we will see how many paths can intersect with this. We will introduce some variables, let say N p is equal to. So, look at all elements belonging

to 0 1 to the power N, such that P y intersection P is non empty ok. The size of this is what we will call as N p, ok.

So, for each path this could be a different and p is chosen randomly and p is a random quantity and we want to compute the expected size of N p ok. Note that N p can be expressed as a sum of i i d random variables, ok. For each y belonging to 0 1 to the power n, the path either intersects with p or it does not intersect ok. If you look at a packet starting at y that might either intersect or not intersect with p and if you set this as indicator random variables, the sum of them will be N p. We can say that N p surely is less than sum over all edges on the path N e. So, e belongs to p, if you sum up over the edges where N e is the number of paths sharing.

So, these paths are all bit fixing paths edge e, ok. Look at all the paths which are the edge e. Their sum is what is I mean, the sum can be said to be always greater than N p. Now, we can just apply linearity of expectation. So, this is going to be less than expectation of summation of N e, e belonging to p and this is going to be equal to summation N e expectation N, ok. So, this is going to be size of p multiplied by expectation of N e where expectation of N e is the expected number of path sharing a particular edge.

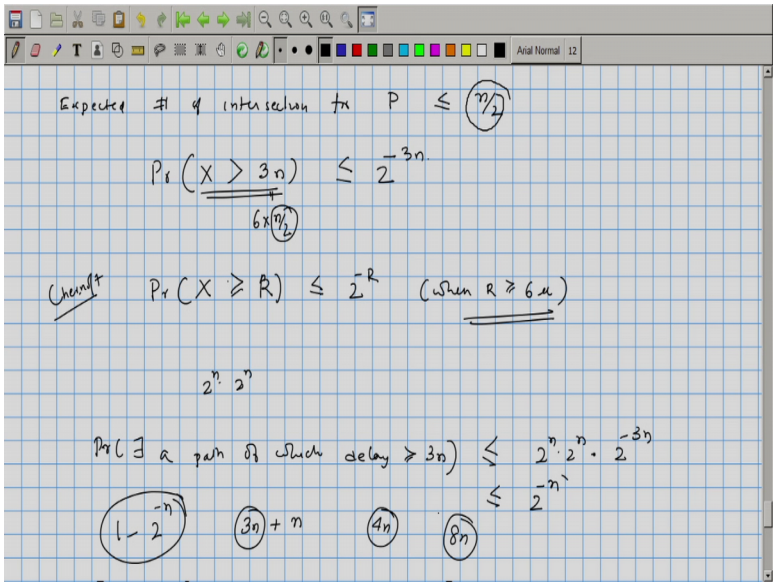
We will show that expected number of edges. Sorry expected number of paths through an edge e is half, ok. This is an easy proof that it is double counting. If you look at all these paths by symmetry, the number of bit fixing paths which share the edge e should be equal for everything for all the possible edges. So, sum overall the edges of the hyper graph expectation of N e should be equal to number of edges that is 2 to the power n times n into expectation of N e.

But this is also equal to expectation of summation over all edges N e, and the inner term summation over all edges N e this we will count in two different ways. Look at all paths which share the edge e and add them or just look at all the vertices. So, summation e N e is equal to summation of overall vertices length of the path, ok. If you look at all bit fixing paths and look at how many I mean edges are there in each path summed up over all the bit fixing path starting at the various vertices that will be equal to N e. So, this is equal to. So, because of this we can write this is equal to expectation of summation over x p x and this is equal to expectation of p x times and a sum over all x, ok.

And the total number of x is again 2 to the power n and the expected length of a path each path you can say that since the bit fixing path is formed by flipping a coin, the bit might agree or might not agree in which case the path expectation of the path length will be n by 2. So, here we get that 2 raised to n times n into expectation of N e is equal to 2 raised to n into n into half ok. So, expectation of N e is basically half. In other words, we can conclude that any edge is expected to be there on half a path, half a bit fixing path, ok.

So, we can say that the expected number of intersections for any path p is going to be less than p into half length of p into half and length of p is at most n. So, this is n by 2, ok.

(Refer Slide Time: 22:35)



So, what we have proved is the expected number of intersections for P is going to be less than n by 2, ok. Now, we can apply. So, if we think of x is a random variable which denotes the intersection the number of intersection we can apply Chernoff's bound. So, the probability that x is greater than say 3 times n, where 3 times n is you can think of it as 6 times n by 2, where n by 2 is the quantity which is surely greater than the expectation.

So, we will apply this form of Chernoff's bound x greater than say R is going to be less than 2 raise to minus R when R is greater than 6 times the expectation, ok. So, that equation that requirement is satisfied. So, this is going to be less than 2 raise to minus 3

n, ok. So, in a nutshell if you take one particular path, the expected number of intersections at motion by 2 and therefore, the probability that the number of intersections is greater than 3 n that is going to be less than 2 raised to minus 3 n, ok.

Now, we can take a bound or so, this is for one particular path that can be at most I mean any path is uniquely specified by start and an end because it is a bit fixing path. So, there at most 2 raised to n times. 2 raised to n paths take union bound or all of them you will get the probability that there exist a path on which delay is greater than 3 n that will be less than 2 raised to n times 2 raised to n. So, we just applying a crude union bound times 2 raised to minus 3 n. So, this is going to be less than 2 raised to minus n, ok.

So, we can look at the contrary positive that and say that with 1 minus 2 raise to minus n probability every path will basically be traversed with delay at most 3 times n and therefore, the total time taken would be 3 raised to n plus the path length which is at most n. So, in 4 n time you can say with high probability phase 1 would be complete. So, if we can finish phase 1 with high probability in 4 n time, then phase 2 also can be finished in same time.

So, in let us say 8 n time the entire phase 1 and phase 2 can be complete, ok. So, that concludes the analysis of randomised bit fixing, randomised fermentation routing on a hypercube.