A packet is **active** at a node v_{i-1} if it reaches v_{i-1} and has the possibility of traversing e_i.

A packet is active if it is active at some node of P.

Since traversing e_i "fixes" the j-th bit, a packet can cross that edge only in its j-th transition.

Assume that e_i connects $v_{i-1} = (a_1, ..., a_{j-1}, a_j, ..., a_n)$ to $v_i = (a_1, .., \bar{a}_j, ..., a_n)$.

Only packets that started in address

$$(*, ..., *, a_j, ..., a_n)$$

can reach vertex v_{i-1}, before the jth bit is fixed and only if their destination addresses are

$$(a_1, ..., a_{j-1}, *, *, ..., *)$$

.

There are 2^{j-1} possible packets, each has probability $2^{-(j-1)}$ to reach v_{i-1}.

Thus expected number of active packets per vertex is 1.

Let H_k, $= 1 \ldots N$, be the indicator r.v. for the packet starting from node k to be active.

$H = \sum_{i=1}^{N} H_k$.

$E[H] \leq m \leq n$.

$\Pr(H \geq 6n \geq 6E[H]) \leq 2^{-6n}$

$\Pr(T(P) \geq 30n) \leq \Pr(H \geq 6n) + \Pr(T(P) \geq 30n | H < 6n)$

$\leq 2^{-6n} + \Pr(T(P) \geq 30n | H < 6n)$

Lemma 1. *If a packet leaves a path (of another packet) it cannot return to that path in the same phase.*

Proof. Leaving a path at the i-th transition implies different i-th bit, this bit cannot be changed again in that phase. $\square$

Lemma 2. *The number of transitions that a packet takes on a given path is distributed $G(\frac{1}{2})$.*

Proof. The packet has probability $1/2$ of leaving the path in each transition. $\square$

The Geometric Distribution

Assume that an experiment has probability p for success $1-p$ for failure. How many trials we need till the first success.

$$Pr(X = i) = (1-p)^{i-1}p.$$

X has a Geometric distribution with parameter p $X \sim G(p)$.

$$\sum_{i=1}^{\infty} Pr(X = i) = \sum_{i=1}^{\infty}(1-p)^{i-1}p =$$

$$p\frac{1}{1-(1-p)} = 1.$$

Assume that X get values in $\mathcal{N}$.

$$E[X] = \sum_{i \geq 0} iPr(X = i) = \sum_{i \geq 1} Pr(X \geq i)$$

Let $X \sim G(p)$,

$$E[X] = \sum_{i \geq 1} Prob(X \geq i) = \sum_{i \geq 1} (1-p)^{i-1}$$

$$= \frac{1}{1-(1-p)} = \frac{1}{p}.$$

The Geometric distribution is **memoryless**:

For any $k > r$,

$$Pr(X > k \mid X > r) = \frac{(1-p)^k}{(1-p)^r}$$

$$= (1-p)^{(k-r)} = Pr(X > (k-r))$$

Conditioning on $H \leq 6n$, if $Z = \sum_{i=1}^{n} Z_i > 30n$, then we had less than $6n$ successes in $36n$ trails with probability $1/2$ for success.

$$Pr(Z = \sum_{i=1}^{n} Z_i > 30n \mid H \leq 6n)$$
$$\leq e^{-18n\frac{1}{2}(\frac{2}{3})^2} \leq 2^{-3n-1}.$$

For a given packet the probability that it spent more than $30n$ steps in phase 1 is bounded by

$$\leq 2^{-6n} + \Pr(T(P) \geq 30n | H < 6n) \leq 2^{-3n}.$$

Since there are at most 2^{2n} possible packets in the hypercube, the probability that there is any packet path with $T(P) \geq 30n$ is at most $2^{2n}2^{-3n} = O(1/N)$.

The proof of phase 2 is by symmetry:

The proof of phase 1 argued about the number of packets crossing a given path, no "timing" considerations.

The path from "one packet per node" to random locations is similar to random locations to "one packet per node" in reverse order.

Thus, the distribution of the number of packets that crosses a path of a given packet is the same.

The total number of packet traversals across the edges of any packet path during Phase 1 and 2 together is bounded by $60n$ with probability $O(1/N)$.

Multi-Commodity Flow

Consider a **directed** network with s sources $x_1, ..., x_s$ and s sinks $y_1, ..., y_s$.

Node x_i is the source of commodity i, y_i is the destination of commodity i.

For each edge e there is a capacity bound T on the amount of flow through that edge.

Assume that we earn r_i for each unit of flow of commodity i.

Let F^i be the amount of flow of commodity i, $0 \leq F^i \leq 1$.

We want to maximize the gain function

$$G = \sum_{i=1}^{s} r_i F^i.$$

Let $IN(v)$ and $OUT(v)$ denote the sets of edges leading into and out of vertex v.

Let F_e^i denote the amount of commodity i flowing through e.

We can formulate this problem as a *linear programming* problem:

Maximize $G = \sum_{i=1}^{s} r_i F^i$,

such that:

For any source x_i, $\sum_{e \in OUT(x_i)} F_e^i = F^i$,

For any sink y_i, $\sum_{e \in IN(y_i)} F_e^i = F^i$,

For any edge e, $\sum_{i=1}^{s} F_e^i \leq T$,

for any internal node v, and commodity i, $\sum_{e \in IN(v)} F_e^i = \sum_{e \in OUT(v)} F_e^i$.

For any edge and commodity $F_e^i \geq 0$.

Linear Programming

Let $\mathcal{A}$ be an $k \times n$ matrix, $\bar{x}, \bar{c}, \bar{b} \in R^n$.

Maximize

$$G(\bar{x}) = \bar{c}^T \bar{x}$$

Subject to

$$\mathcal{A}\bar{x} \leq \bar{b}.$$

Each constraint

$$\sum_{j=1}^{n} a_{i,j} x_j \leq b_i$$

defines a **half space** of R^n.

Each

$$\sum_{j=1}^{n} a_{i,j} x_j = b_i$$

is a **hyperplane**.

A finite intersection of hyperplanes defines a **polyhedron**.

We are looking for a point in the polyhedron defined by

$$\mathcal{A}\bar{x} \leq \bar{b}$$

that maximizes

$$G(\bar{x}) = \bar{c}^T \bar{x}$$

Linear programming problem can be solved efficiently (in polynomial time).

Integer linear programming (optimal solution with integer values) is NP-hard.