Dynamic Analysis

Static (batch) processes:

- Input is fixed at the beginning of the computation.
- Goal: Minimize the time till termination.

Dynamic Processes:

- Input is continuously injected to the system.
- Goal: Characterize the long term (steady state) performance of the process.

Dynamic analysis is particularly useful for P2P networks, since the structure of the network is constantly changing.

Static vs. Dynamic Packet Routing

Static Routing:

A set of communication requests are injected at time 0.

Goal: Minimize the time to satisfy all requests.

Benchmark: Time to route an arbitrary permutation.

Dynamic Routing:

New Packets are continuously injected to the system.

Benchmark: steady state analysis.

Packet Routing on a Ring

N-node one-directional ring.

In each step, each node generates a new packet with probability λ. We assume this because it is the simplest assumption possible within the context of the problem.

Packets have random destinations (chosen uniformly at random from the N nodes).

An edge can transmit one packet per step.

Stability

Stability is an important benchmark of a network - the performance of network applications depends heavily on the amount of delay a packet encounters on average.

$W_{avg}(t) =$ average waiting time of packets that entered the system at time t.

$L(t) =$ the number of packets in queues at the end of step t.

A system is stable if both $E[W_{avg}(t)]$ and $E[L(t)]$ are bounded with respect to t.

Two expectations are necessary because one expectation measures with regards to time and the other with regards to number of packets.

Upper bound on λ

Theorem 1. *The system cannot be stable for* $\lambda > \frac{2}{N}$.

Proof. The average route of a packet is $N/2$.

There are no more than N packet transitions per step.

To keep the system stable a process can insert new packet every $N/2$ steps on the average. Basically we say that packets cannot be injected faster than the average service rate, which is $\frac{N}{2}$. Or, put in another way, a node shouldn't generate more than one packet per $\frac{N}{2}$ steps. $\square$

Lower bound on λ

Theorem 2. *A routing algorithm with farthest-first priority is stable for any $\lambda < 2/N$.*

Definition 1. *Packet r was delayed t steps in crossing edge e if*

1. r traversed edge e at step $t + k$.

2. e is the kth edge in the route of r.

Lemma 1. *If a packet r was delayed t steps in crossing edge e, then there is an interval of t steps such that some packet crossed edge e in each step of this interval.*

Proof. Because of the farthest-first priority scheme any packet that delayed r must also cross edge e.

We say that packet s delayed packet r at time τ if s moves at that step, r does not move, and all the packets between s and r do not move.

A packet can delay r only once.

All the packets that delayed r are moving in front of r, thus crossing e in the interval $[k, \ldots, t-1]$.

We require the farthest-first priority scheme because otherwise there may be a packet s that is routed before r, but does not cross e. $\square$

The wide-channel model

In the wide-channel model packets are never delayed.

A packet crosses edge e at time k if e is the kth edge on its route.

Lemma 2. *There is a constant ϵ such that for a sufficiently large t_0 the probability that a given packet is delayed by at least t_0 steps is bounded by $e^{-\epsilon t_0}$.*

Proof. If a packet was delayed t steps then there is an edge e such that the packet was delayed t steps in crossing that edge.

(The definition of delayed is important here. By saying that the packet was delayed t steps in crossing an edge we don't mean the packet waited for t steps in the queue for that edge, we simply mean that the packet that without delay would have crossed the edge at k instead crosses that edge at $t + k$.)

This implies that there is an interval of t steps such that at least t packets crossed edge e in that interval in the wide-channel model.

The expected number of packets that cross e in the wide-channel at a given step is bounded by

$$\sum_{i=1}^{N} \frac{\lambda(N-i)}{N} < \frac{2}{N}\frac{N(N-1)}{2N} = \gamma < 1$$

Use Chernoff's bound to find the bound on the probability that the number of packets crossing an edge at a particular time is greater than t_0. We can use Chernoff's bound because in wide channel mode there are no queues and our random variables are independent.

$P_r(X > t_0) = P_r(X > (1+\delta)\mu) = P_r(X > (1+\frac{1}{\gamma}-1)\gamma t_0)$

Since γ is a little less than 1, γ^{-1} is a little greater than 1, and $\delta = \gamma^{-1} - 1 < 1$.

Use the generalized form of Chernoff's Bound,

$$P_r(X > t_0) < e^{\frac{-\gamma t_0(\gamma^{-1}-1)^2}{3}}$$

We can simplify the numerator since γ, t_0 are constants.

$$P_r(X > t_0) < e^{-\epsilon t_0} \quad \square$$

Theorem 3. *For any $\lambda < 2/N$ the system is stable and the expected time a packet is delivered is $O(N)$.*

Proof. The route length is bounded by N.

Let r.v. X denote the delay.

Then

$$E[X] = \sum_{t \geq 1} \Pr(X \geq t)$$

$$= t_0 + \sum_{t > t_0} e^{-\epsilon t} = O(1)$$

$\square$

Theorem 4. *With probability* $1-1/N$ *a given packet is not delayed more than* $O(\log N)$ *steps.*

Proof. The probability that a packet is delayed by t_0 steps or less is $1 - e^{-\epsilon t_0}$.

Calculate the probability that a packet is delayed by more than t_0 steps.

$$P = e^{-\epsilon t_0}$$

$$lnP = lne^{-\epsilon t_0}$$

$$lnP = -\epsilon t_0 lne$$

Plug in $t_0 = c \log N$, with $c = \epsilon^{-1}$. Take $\log N$ with base e.

$$lnP = -\epsilon \epsilon^{-1} lnN lne$$

$$lnP = -1 lnN$$

$$lnP = lnN^{-1}$$

$$P = N^{-1}$$

The probability that the delay is $O(\log N)$ is $1 - \frac{1}{N}$. $\square$

Theorem 5. *With probability* $1 - 1/N$ *a given queue at a given time step has no more than 3 elements.*

Proof. The size of the queue is bounded by the number of new elements created by the node feeding the edge. More than 2 new elements are generated in the interval $C \log N$ with probability of choosing two intervals inside $C \log N$ to generate packets. This is $(\frac{C \log N}{2})\lambda^2$.

$$(\frac{C \log N}{2})\lambda^2 < (\frac{eC \log N}{2})^2 \frac{2}{N}^2.$$

Let $C = e^{-1}$.

$$\frac{eC \log N}{2}^2 \frac{2}{N}^2 = \frac{\log N}{2}^2 \frac{2}{N}^2 \leq \frac{1}{N}. \quad \square$$

Dynamic Circuit Switching

2N-input/output $\log N$-dimensional butterfly network. This is a butterfly network where there are 2 inputs on each of the nodoes in the first level and 2 outputs on each of the nodes in the last level.

Each input has N queues for N possible destinations.

Important question: How many simultaneous connections can I support?

Each input generates in each step a new request with probability p_0. Request have random output destinations.

Algorithm

In each step each input tries to establish a path to its (random) destination.

In order to route a message from input i to output j, the unique greedy path from i to j is reserved.

If two paths try to use the same edge - one path is canceled (request returns to queue).

Lemma 3. *If input v tries to establish a path at a given iteration, the probability that the path is established is at least $1/\log N$.*

Proof.

Consider the case in which all inputs try to establish paths, each to a random destination.

Let p_i be the probability that an edge e at level i is used by a path. Then,

$$p_{i+1} = \frac{p_i}{2} + \frac{p_i}{2} - \frac{p_i^2}{4}$$

This is the probability that both of the p_i edges leading to p_{i+1} are picked, which is the probability of picking the first p_i plus the probability of picking the second p_i minus the union probability that both p_i's are picked. We can represent the final term as a multiplication because the probabilities are independent, since both edges are fed from different inputs.

$$p_{i+1} = 2\frac{p_i}{2} - \frac{p_i^2}{4}$$

$$p_i \sim \frac{4}{i+4}$$

$$p_{\log N+1} \sim \frac{4}{\log N}$$

The probability that a packet is delivered successfully is $p_{\log N+1}$. We know by linearity of expectation that the probability that $2N$ packets are delivered successfully is $2N * p_{\log N+1}$ or $\frac{8N}{\log N}$.

Thus the expected number of packets that will reach their destinations is at least $N/\log N$. Thus $1/\log N$ of the paths are established. By symmetry a given request is established with probability at least $1/\log N$. $\square$

Theorem 6. *The system is stable if arrival rate to the inputs is less than* $1/\log N$.

Proof. The rate at which packets are injected cannot be greater than the rate at which we expect packets to be serviced. □

Theorem 7. *The expected wait in the queue is* $O(N \log N)$.

Proof. Each request in the queue has to wait for all the requests before it to be serviced. Requests are delayed with high probability by no more than $O(\log N)$, so the last request in the queue can expect to wait at $O(N \log N)$. □