Lecture 1: Introduction: Models for routing and faults

30th July, 5th and 6th August, 2008

1.1 Introduction: A simple network model

A computer network is a collection of computers interconnected through a medium. In this class we will model computer networks using graphs. The set of computers is represented by the vertex set V and the medium is represented by the edge set E. Implicit in this model is the understanding that only certain pairs of computers may communicate "directly". Those pairs of computers that are not directly connected need to rely on other computers to relay information between them. This restriction gives rise to the problem of routing, i.e., what is the "best" way to get information from one computer to another in a computer network.

Before we proceed we will have to model the network and the capabilities and constraints of its various elements. We do not attempt to describe in detail all the different kinds of computer networks that exist or all the varyingly nuanced models that exist for them. Instead here we introduce a simple network model that allows us to illustrate certain fundamental aspects of the routing problem.

The model. A network is modelled by a graph $G = (V, E)$. Each communication in this network takes the form of a discrete object known as a *packet*. Each packet p is a tuple of the form $\langle s_p, t_p \rangle$ where $s_p, t_p \in V$. We assume that each node $u \in V$ has the following capabilities:

- It can **generate** a packet of the form $\langle u, v \rangle$ for all $v \in V \setminus \{u\}$.
- It can **consume** a packet of the form $\langle v, u \rangle$ for all $v \in V \setminus \{u\}$.
- It can **receive** a packet from all $v \in \Gamma(u) = \{(u, v) : u \in V, (u, v) \in E\}$.
- It can **buffer** packets of the form $\langle v, w \rangle$ for $v, u \in V \setminus \{u\}$.
- It can **forward** a packet of the form $\langle v, w \rangle$ according to a *routing algorithm* which is a function $f : V \times V \to \Gamma(u)$. If more than one packet is waiting in the buffer to cross a particular edge, the algorithm must also incorporate a *tie-breaking rule*.

Further we assume that the network is *synchronous* i.e. all nodes operate with a common clock. We take time to move in discrete steps and we assume that each edge can carry only one packet in one each direction in a given time step. With such a restriction its clear that at one time there may be multiple packets waiting at a node to cross an edge. So we allow for buffering at each node. Typically buffers cannot be of arbitrary size but for now we ignore this aspect and allow each node to buffer as many packets as it needs to.

In this model an *routing problem* is a set of packets with the times at which they are injected into the system. An important criterion in such a model is the number of time steps taken to successfully route every packet to its destination. Let us now study this model for a simple network.

1.2 Routing: A simple example

We consider a ring network: n nodes numbered 0 to $n-1$ with edges of the form $(i, (i+1) \mod n)$. We study a routing problem in which m packets are injected into this network at time 0. Any packet has only two possible routes to its destination so we consider the family of algorithms wherein the packets take the shorter of the two paths to their destination. We refer to this as a family since there are a number of possible tie-breaking rules that can be employed to determine which packet is taken out of the buffer and forwarded. We will analyze the algorithm in which the packet with more distance left to cover is given priority. In case more than one packet has the same distance left to cover then tie-breaking is done on the basis of packet id i.e. the packet with the higher id number gets to go first. We call this algorithm *Furthest-to-go*.

Theorem 1.1 *Given m packets injected into the ring network on n nodes at the same time,* Furthest-to-go *requires at most $m + \lfloor n/2 \rfloor - 1$ steps to send all packets to their destinations.*

Proof. Let $P = \{p_1, p_2, \ldots, p_m\}$ be the set of packets. For packet p_i at node v we define a *rank*, $r_v(p_i)$ as the number of edges left for the packet to travel plus $i/(m+1)$. Note that no two packets can have the same rank at a particular node.

With this definition we can say that *Furthest-to-go* is the algorithm that picks the packet with the highest rank from the buffer and forwards it.

We analyze the number of steps taken by *Furthest-to-go* by arguing backwards from the last packet to be delivered. Let this packet be $q_1 \in P$. Let us call it's destination v_0. We will now build two sequences, $q_1, q_2, \ldots q_s$ of packets and $v_0, v_1, \ldots, v_s$ of nodes as follows:

- First, we follow q_1 back in time till we reach the first node (moving backwards) where q_1 was delayed because another packet had to be sent along the edge q_1 wanted to travel on.
- We call this node v_1 and we denote the packet that was sent in preference to q_1 as q_2. Note that q_1 might have had to wait for several time steps at v_1, so it is not clear which of the various packets that were sent before it should be labelled q_2. We specify that the first packet that was sent out after q_1 reached v_1 should be called q_2.
- In general, we trace a packet q_i backwards from node v_{i-1} till the first vertex where it was delayed. This vertex is denoted v_i and the packet which delayed q_i is denoted q_{i+1}.
- If, at some point we trace back a packet q_j to it's source i.e. we trace it back and it is never delayed all the way back to it's source, then we set $s = j$, name its source v_s and terminate the process of building the sequences.

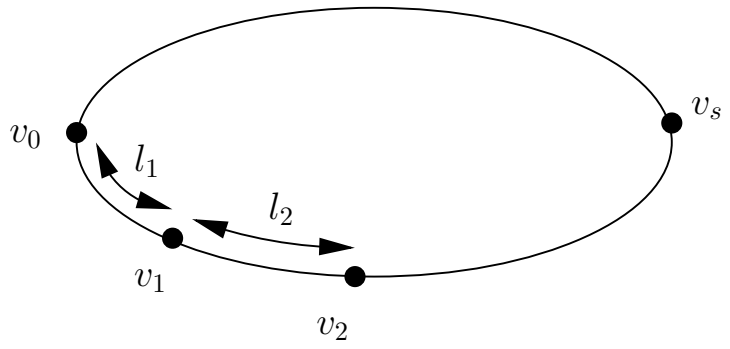

Figure 1: Constructing the two sequences.

Figure 1 illustrates the structure that is formed. It also shows another sequence $l_1, l_2, \ldots, l_s$ which is defined as follows: l_i is the number of edges between v_{i-1} and v_i, for all $1 \leq i \leq s$.

Let us note three simple facts about these sequences:

1. $r_{v_0}(q_1) > 0$.
2. $r_{v_i}(q_{i+1}) > r_{v_i}(q_i)$.
3. $r_{v_i}(q_i) = r_{v_{i-1}}(q_i) + l_i$.

All three of these facts follow from the way the sequences are defined. Putting these three facts together we are able to show

Lemma 1.1

$$\sum_{i=1}^{s} l_i \leq \left\lfloor \frac{n}{2} \right\rfloor .$$

Proof. We add all the instances of (2) and (3) along with (1) to get that

$$r_{v_s}(q_s) \geq \sum_{i=1}^{s} l_i,$$

and then rely on the fact that $r_{v_s}(q_s) < \lfloor n/2 \rfloor + 1$ since all path lengths are at most $\lfloor n/2 \rfloor$. Further the l_is are all integers so we can dispense with the additional 1. ■

Note that although the construction does not specify that the packets in the sequence should all be unique this is in fact the case. This follows from the fact that the ranks are *strictly* increasing as we go up the sequence. If we assume that packet q_i and q_j are the same packet for some values $j < i$ then we get the following contradiction using facts (2) and (3)

$$r_{v_i}(q_i) \geq r_{v_{i-1}}(q_i) > r_{v_{i-1}}(q_{i-1}) \geq r_{v_{i-2}}(q_{i-1}) > \cdots > r_{v_j}(q_j).$$

Hence, we can claim that the number of packets in the sequence is less than the total number of packets injected i.e. $s \leq m$. Since the total time the last packet takes to get delivered is the sum of the l_i values, plus one unit of delay at each of $v_1, v_2, \ldots, v_{s-1}$, the total time taken for the last packet to be delivered is $s - 1 + \sum_{i=1}^{s} l_i$. Hence the result follows. ■

Note that in the worst-case *Furthest-to-go* is an optimal strategy since we can easily set up an instance of the problem where at least $m + \lfloor n/2 \rfloor - 1$ steps are required. This instance occurs when all m packets are injected into the same node and all their destinations are also the same: the node at distance precisely $\lfloor n/2 \rfloor$ from the source.

1.3 Modeling faults

A system is said to be faulty if it does not function as per its specification. An important part of the process of rectifying a fault is localizing it to a specific component of the system and characterizing the faulty behaviour of the component to the extent possible. For computer networks as we have modelled them, faults could occur at nodes or at edges. Modelling faults is an intricate process in general. The extent of the faulty behaviour of the component must be taken into account, as must be the effect of that faulty behaviour on the system's performance both in terms of the correctness of the output as well the efficiency of producing that output. In this class we will consider the effect of faultiness on the ability of a computer network to efficiently route packets between computers. We will take various parameters into account in the course of the class. For now we just look at one important and fundamental aspect of the routing problem:

> What pairs of nodes in the network are unable to communicate when faults occur.

We model faults simply: if a set $U \subseteq V$ of nodes is faulty we consider the faulty network to be transformed into the subgraph of G induced by the node set $V \setminus U$ i.e. we remove all the faulty nodes and all the edges incident on the faulty nodes. See Figure 2 for an example. In Figure 2, when the set

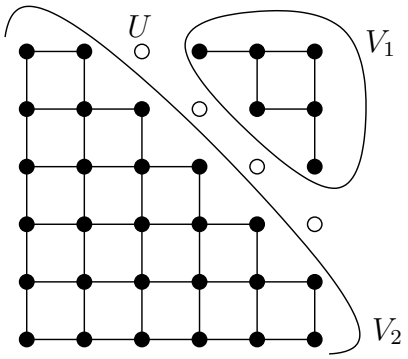

Figure 2: V_1 and V_2 are disconnected when U becomes faulty.

U becomes faulty it divides the remaining graph into two components which cannot communicate with each other. Clearly in a $\sqrt{n} \times \sqrt{n}$ mesh we can

achieve such a separation with as few as 2 faulty nodes although we can find sets of nodes which are $\theta(n)$ in size whose removal would leave the remaining graph in a single connected component. In a line network on n nodes, if one of the end points gets faulty the other $n-1$ nodes can communicate freely among themselves although if the central node gets faulty then $\theta(n^2)$ pairs of vertices get disconnected. The size of the components of a faulty network which has become disconnected is an important parameter in many real-world scenarios. It may be that if the network has a large enough component then it can fulfil its function without having to repair the faults. In other scenarios it could be that service is affected only for small components while large components get satisfactory service. We will study the usefulness of faulty networks which retain large connected components in greater detail through the course of this class.

Even in networks that don't get disconnected faults can effect the efficiency of routing. For example in the ring network of Section 1.2 the bounds on the number of steps would go up to $m+n-2$ if even a single node should get faulty.

In order to be able to analyze the effects of faults we must have a model for the occurence of faults. In this class we will consider two broad classes of fault models. These two classes do not cover all possible fault models by any means but form an important beginning in studying more general models.

1. **Adversarial faults**. Here we assume that there is a malicious entity, known as the *adversary*, with a bounded amount of power k. The adversary knows what the functionality of the network is and is allowed to make up to k nodes faulty in order to degrade this functionality.

2. **Random faults**. This model has a parameter p such that $0 \leq p \leq 1$. Faultiness is modelled by a stochastic process here i.e. a collection of random variables. Each node in the network gets faulty with probability p. The process at each node is independent of the process at all other nodes.

We consider the latter model to be more representative of real-world problem (although the independence assumption may not be particularly realistic) and we will concentrate on it for a large part of this class.

1.4 Random faults: The existence of large components

In this section we will consider two networks with infinite vertex sets and try and determine how the random fault model affects it. For ease of argument we will assume that each *edge* is faulty with a certain probability independent of all other edges rather than each node. In this section we begin to study the size of the components of such a network and how the parameter p affects this size.

The structure of the networks we study is a graph that we call the *d-dimensional lattice*, denoted $\mathbb{L}^d$. Each vertex in this graph is a d-dimensional vector of integers i.e. the vertex set is $\mathbb{Z}^d$. We denote the edge set by $\mathbb{E}^d$ and it is defined as follows:

$$\mathbb{E}^d = \{(u, v) : u, v \in \mathbb{E}^d, \sum_{i=1}^{d} |u_i - v_i| = 1\},$$

where u_i is the ith component of the vertex u. In this lecture we will look at the cases $d = 1$ and $d = 2$ i.e. the infinite line and the two dimensional mesh.

Faults which occur independently at each edge of such graphs with the same probability are studied in Percolation theory. In what is known as the bond percolation setting each edge $e \in \mathbb{E}^d$ has two states, *open* (corresponding to a non-faulty state) and *closed* (corresponding to a faulty state). Each edge is in an open state with probability p for some $0 \leq p \leq 1$ and closed with probability $q = 1 - p$. Formally speaking we take a probability space in which the sample space is $\Omega = \prod_{e \in \mathbb{E}^d} \{0, 1\}$, points of which are denoted by $\omega = (\omega(e) : e \in \mathbb{E}^d)$ and are referred to as *configurations*. The value $\omega(e) = 1$ corresponds to the edge e being open and $\omega(e) = 0$ corresponds to it being closed. The σ-field of this probability space, $\mathcal{F}$, is the subsets of Ω generated by the finite dimensional cylinders i.e. each element of $\mathcal{F}$ corresponds to a subset of the configurations in Ω in which the state of the edges in some finite subset of $\mathbb{E}^d$ is fixed. The measure defined on this space is a product measure

$$\mathrm{P}_p = \prod_{e \in E_d} \mu_e,$$

where $\mu(e)$ is a Bernoulli measure on $\{0, 1\}$ for each e, given by

$$\mu_e(\omega(e) = 1) = p \text{ and } \mu_e(\omega(e) = 0) = 1 - p.$$

For example, the probability of a configuration having $e_1 = 1\,, e_2 = 0\,, e_3 = 1$ is $\mu_{e_1}(e_1 = 1) \cdot \mu_{e_2}(e_2 = 0) \cdot \mu_{e_1}(e_1 = 1) = p^2(1-p)$.

We denote by C the maximal component connected by open edges containing the origin on $\mathbb{L}^d$. The *percolation probability*, denoted $\theta(p)$ is a key figure in the study of percolation. It is defined as

$$\theta(p) = \mathrm{P}_p(|C| = \infty)$$

A fundamental fact of percolation theory is that there is a critical value $p_c(d)$ (often written just p_c when the dimension is understood) such that

$$\theta(p) \begin{cases} 0 & \text{if } p < p_c \\ > 0 & \text{if } p > p_c \end{cases}$$

This value, $p_c(d)$ is called the critical probability and is defined as

$$p_c(d) = \sup\{p : \theta(p) = 0\}.$$

Let us consider the case of $d = 1$. Here we claim that p_c has the trivial value 1. In order to do that we first introduce a result from probability theory.

Given a sequence of events $\{A_n\}$ we note that the $\limsup A_n$ is also referred to a A_n *infinitely often* (or A_n i. o.) since it can be thought of as that set of outcomes whose occurence makes an infinitely large subset of the events in the sequence happen. The Borel-Cantelli Lemma helps us characterize the probability of A_n i.o.

Lemma 1.2 (Borel-Cantelli Lemma 1) *Given a probability space* $(\Omega, \mathcal{F}, P)$ *and a sequence* $\{A_n\}$ *such that* $\forall i : A_i \in \mathcal{F}$, *if* $\sum_{i=1}^{\infty} P(A_i) < \infty$ *then* $P(A_n \ i.o.) = 0$.

To put it simply, the Borel-Cantelli Lemma says that if the sum of probabiilities of the events in the sequence is finite then an infinite subset of the events occur simultaneously with probability 0. With this result in hand we now proceed.

Claim 1.2

$$p_c(1) = 1.$$

Proof. For $i \geq 0$ define A_i to be the event that all edges starting from node 2^k to $2^{k+1} - 1$ are open *or* the edges from $-2^{k+1} + 1$ to -2^k are open. Note that

$$\mathrm{P}(A_k) = 1 - (1 - p^{2^k})^2 \leq 2 \cdot p^{2 \cdot 2^k}.$$

Hence, as long as $p < 1$. $\sum_{k=1}^{\infty} \mathrm{P}(A_k) < \infty$. And so, by the Borel-Cantelli lemma we get that $\mathrm{P}(A_k \text{ i.o.}) = 0$ i.e. there must be some k for which A_k is not true with probability 1. And this implies that as long as $p < 1$, the probability of having an infinite component is 0. ■.

However for $d \geq 2$ we get non-trivial critical probabilites. The rest of this lecture is devoted to establishing this. We begin by claiming that the critical probability is a non-increasing function of the dimension.

Claim 1.3 *For $d \geq 1$,*

$$p_c(d+1) \leq p_c(d).$$

This claim follows from observing that at any value of p, if you have a infinite component in $\mathbb{L}^d$ then you will also have an infinite component in $\mathbb{L}^{d+1}$. We claim, without the proof that a strict version of this inequality also holds.

We now show that $p_c(2)$ can be bounded away from 0 and 1.

Theorem 1.4

$$\frac{1}{3} \leq p_c(2) \leq \frac{2}{3}.$$

Proof. Let us first show that $p_c(2) > 1/3$. Denote by $\sigma(n)$ the number of paths of length n in $\mathbb{L}^2$ beginning at the origin. We define a random variable $N(n)$ to be the number of these paths that are open. Now, if the origin is a part of an infinite cluster then there exist paths of all possible lengths starting at the origin. Restating this in terms of $N(n)$ we can that the event that the origin is part of an infinite cluster implies the event that $N(n)$ is at least 1 for each value of n i.e. for all n

$$\begin{aligned}
\theta(p) &\leq \mathrm{P}_p(N(n) \geq 1) && (1)\\
&= \sum_{i=1}^{\infty} \mathrm{P}_p(N(n) = i) && (2)\\
&\leq \sum_{i=1}^{\infty} i \cdot \mathrm{P}_p(N(n) = i) && (3)\\
&= \mathrm{E}_p(N(n)) && (4)
\end{aligned}$$

Now, the probability of a given path of length n being open is p^n and there are $\sigma(n)$ such paths, so

$$\mathrm{E}_p(N(n)) \;=\; p^n\sigma(n). \tag{5}$$

Let us now try to bound $\sigma(n)$. Every path starts at the origin so there are four choices for the first edge. After that at each vertex on the path we have at most 3 choices, eliminating the obviously wrong choice of going back to the vertex we just came from. Hence

$$\sigma(n) \leq 4 \cdot 3^{n-1}.$$

Using this and (5) in (1-4) we get

$$\theta(p) \leq \frac{4}{3}(3p)^n.$$

Hence, if $3p < 1$, $\theta(p) \to 0$ as $n \to \infty$, proving that for $p < 1/3$, the origin cannot be part of an infinite cluster, hence p_c must be at least $1/3$.

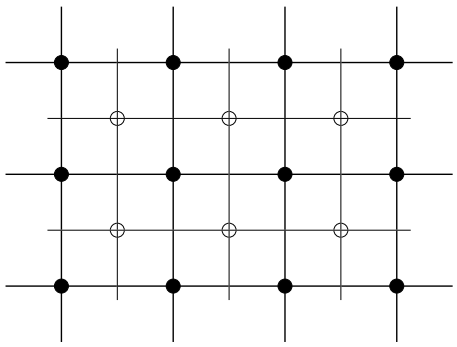

Figure 3: The dual of $\mathbb{L}^2$.

In order to prove the second inequality i.e $p_c \leq 2/3$, we will introduce the *dual* of the lattice $\mathbb{L}^2$ which we will denote $\mathbb{L}^2_d$. This is defined in the way that the dual of a planar graph is usually defined in graph theory, by placing a vertex in every face and by putting an edge between two vertices corresponding to two faces that share an edge (see Figure 3.) Note that there is a natural one-to-one correspondence between the edges of the dual and the edges of $\mathbb{L}^2$. We define a bond percolation process on the dual using

this correspondence by declaring an edge of the dual to be closed if it crosses a closed edge of $\mathbb{L}^2$ and open if the edge of $\mathbb{L}^2$ it crosses is open.

By convention we denote the vertices of the dual as $(x+\frac{1}{2}, y+\frac{1}{2})$ where x and y are integers.

Now, to see the direction our proof will take, let us note that if the origin was contained in a finite cluster that would mean there is a set of closed edges surrounding this cluster. This set of edges corresponds to a circuit of closed edges surrounding the origin in the dual. With this in mind we can proceed.

First, let us count the number of circuits of length n surrounding the origin in the dual. We denote this quantity $\gamma(n)$. Note that any such circuit must touch on a point of the form $(k+\frac{1}{2}, \frac{1}{2})$ for some $0 \leq k \leq n$. We can now consider each circuit as a walk of length $n-1$ beginning from such a vertex and returning to some neighbour of $(k+\frac{1}{2}, \frac{1}{2})$. Hence, given the number of choices for k, and recalling the definition of $\sigma(n)$ from the earlier part of the proof, we have

$$\gamma(n) \leq n\sigma(n-1) \leq \frac{4}{3} \cdot n \cdot 3^{n-1}.$$

Let us denote by $M(n)$ the number of these circuits of the dual of length n surrounding the origin which are closed. Since a circuit of length n is closed with probability $(1-p)^n$ we get that

$$\mathrm{E}_p(M(n)) \leq \frac{4}{3} \cdot n \cdot (3(1-p))^{n-1}. \tag{6}$$

Now, define two events G_m and F_m as follows

- G_m : All edges in $B(m)$ are open.
- F_m : There is a closed circuit in $\mathbb{L}^2_d$ containing $B(m)$ in its interior.

Note that if G_m occurs and F_m *does not* occur, then there is a path of length n starting at the origin for all values of n. Hence the event $G_m \wedge \bar{F_m}$ implies the event that the origin is part of an infinite component i.e.

$$\theta(p) \geq \mathrm{P}_p(G_m \wedge \bar{F_m}) = \mathrm{P}_p(G_m) \cdot \mathrm{P}_p(\bar{F_m}). \tag{7}$$

The latter part following from the fact that G_m and F_m are events depending on disjoint sets of edges.

Let us now try to upper bound $\mathrm{P}_p(F_m)$ so that we can complete the lower bound in (7).

We know that a circuit surrounding $B(m)$ must have length at least as much as the perimeter of $B(m)$ i.e. $8m$. So, for F_m to be true there must be a circuit of length at least $8m$ surrounding the origin that is closed. Note that all circuits of this length may not contain $B(m)$ completely, and hence we get the following inequality

$$\mathrm{P}_p(F_m) \leq \mathrm{P}_p\left(\sum_{n=8m}^{\infty} M(n) \geq 1\right)$$

Now, since $\mathrm{P}_p(M(n) \geq 1) \leq \mathrm{E}_p(M(n))$, using (6), we have

$$\mathrm{P}_p(F_m) \leq \sum_{n=8m}^{\infty} \frac{4}{3} \cdot n \cdot (3(1-p))^{n-1}.$$

If $3(1-p) < 1$ this sum converges, and it is possible to find a large enough value of m^* of m such that $\mathrm{P}_p(F_{m^*}) \leq \frac{1}{2}$. Using this in (7), and noting the fact that since m is a finite value $\mathrm{P}_p(G_m) > 0$ for any $p > 0$, we get

$$\theta(p) \geq \frac{1}{2} \cdot \mathrm{P}_p(G_{m^*}) > 0.$$

Hence, if $p > 2/3$ we are definitely above the critical probability. And this completes the proof of the second inequality. ■

Notes

The example of routing on the ring in Section 1.2 is taken from Christian Scheideler's lecture notes from his class *Theory of Network Communication*, Fall 2002, Lecture 1.[1] The material in Section 1.4 is a simplified version of a well-known proof taken from Grossglauser and Thiran's lecture notes for their class *Models and Methods for Random Networks*, 2006-07.[2]

[1] Available from http://www.cs.jhu.edu/~scheideler/.

[2] http://icawww1.epfl.ch/class-nooc/