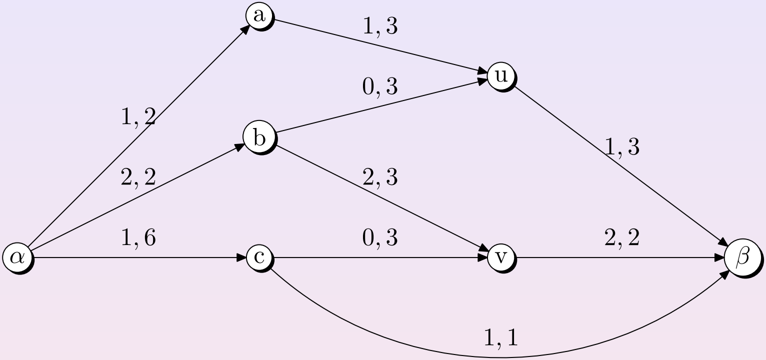
Modelling data networks



Richard G. Clegg (richard@richardclegg.org)— 8th November 2007

Available online at http://www.richardclegg.org/lectures accompanying printed notes provide full bibliography.

(Prepared using LaTeX and beamer.)

Difficulties in modelling the Internet

- See [Floyd & Paxson 2001].
- The internet is big (and growing).
- The internet is heterogenous to a large degree.
- No central maps exist of the internet.
- The internet is not always easy to measure.
- The internet is rapidly changing.
- It is extremely important to be able to model the internet.

The internet cannot possibly be modelled, yet we must model the internet. How can this be resolved?

Aim of this lecture

- A general approach to such modelling problems.
  - What is known in literature?
  - What should be inputs to model.
  - What should be outputs from model.
- Introduce mathematical techniques for network modelling.
  - Markov chains.
  - Queuing theory.
  - Graph theory.
- An approach to "sanity checking" your modelling.
  - Model validation.
  - Sensitivity analysis.

Steps to modelling

- How you model the network depends critically on the problem you are solving.
- What are you trying to show with your model?
- Metrics: what are we trying to measure?
  1. Throughput?
  2. Goodput?
  3. System efficiency?
- Validation: what real data can be used to check the model?
- Sensitivity: what happens if your assumptions change?
  1. What if the demand on the system is slightly different?
  2. What happens if delays and bandwidths are changed?
  3. What happens if users stay longer or download more?

Important questsions for modelling

1. How much of the network do we model?
   - Whole internet (then we can't even model every computer – every AS?)
   - A few typical nodes?
   - A sub net?
   - A single queue and buffer?
2. What level of modelling is appropriate?
   - Mathematical – solution "instant" (or quick)
   - Detailed simulation
   - Combined approach (equations abstract away some details with approximations)
3. How far down the network stack need we go?

Model example one – peer-to-peer network

Modelling Task

Test the possible improvements expected if we try a locality aware peer selection policy on a global bittorrent network.

What must our model include?

1. The distribution of nodes (peers) on the overlay network (not the whole network).
2. The delay and throughput between these peers (must depend on distance to some extent).
3. How users arrive and depart.
4. What users choose to download.

Note that this might already be a vast modelling task with hundreds of thousands or even millions of nodes.

Approach to model one – peer-to-peer network

- Research existing P2P models, do any fit? Don't reinvent the wheel.
- Real data: What real-life measurements exist to validate against?
- If we are modelling a new peer selection we must be sure our model covers existing peer selection well.
- Metrics: what must we measure in our model?
  1. Overall throughput/goodput?
  2. Distribution of time taken for peers to make their download?
  3. Total resources used in system?
- Validation: Instrumented P2P clients exist – how do they compare to our simulation.
- Sensitivity: Different distribution of users? Different delays and throughputs?

Model example two – Buffer overflow model

Modelling task

Given a router with a buffer, how does the buffer size in packets affect the probability of packet loss?

What must our model include?

1. A model of the incoming packets to the buffer.
2. The rate at which packets leave the buffer.
3. Possibly distribution of packet lengths in bytes.
4. Possibly the feedback (TCP) between packet loss and arrival rate.

Approach to model two – Buffer overflow model

- Research: what is known about the statistics of internet traffic?
- What is the distribution of inter-arrival times and packet lengths?
- Metrics:
  1. Packet loss.
  2. Packet delay.
- Sensitivity: What if we change the following parameters:
  1. The total arrival rate.
  2. The bandwidth of the outgoing link.
- Validation: Real traffic traces (CAIDA has a collection).

Model example three – TCP protocol model

Modelling Task

Test a possible improvement to the TCP model which aims to improve fairness and throughput when flows share a link.

What must our model include?

1. Individual packet model with existing TCP protocol as accurately as possible.
2. A reasonable estimate of how long each connection lasts and the rate at which new connections.
3. A model of the probability of round trip time for the parts of the connection not on the link being modelled.
4. A model of the probability of packet loss on the link (due to buffer overflow?)

Approach to model three – TCP protocol model

- Can existing network models help (ns-2 could be an obvious choice)?
- What if the existing protocol shares a link with flows using the old protocol.
- Metrics:
  1. Throughput and goodput.
  2. Fairness between flows.
- Sensitivity, what if we change these parameters:
  1. Number of flows using existing and new protocol.
  2. Bandwidth of link.
  3. Round trip time of flows.
  4. Probability of packet loss.
- Validation: Does our model agree with real measurements?

Areas of modelling interest(1)

Now let us focus on several specific areas of interest to modellers.

1. Topology modelling — how are the nodes in the internet connected to each other?
   - See the internet as nodes and edges (graph theory).
   - Consider numbers of hops between nodes.
2. User/flow arrival modelling — how does traffic arrive on the internet?
   - See arrivals as a stochastic process (probability/statistics)
   - How long do connections last?
3. Application level protocols — what traffic do applications place on the internet?
   - For example peer-to-peer networks use an overlay (graph theory again?)
   - A web page might make connections to many different places.

Areas of modelling interest(2)

1. Traffic statistics — what does the traffic along a link look like in statistical terms?
    - See internet traffic as a stochastic process (queuing theory).
    - How does TCP congestion control alter this?
2. Transport/network protocols — how do TCP/IP protocols affect the traffic?
    - See internet traffic as a feedback process (control theory).
    - How do these protocols interact with the rest of the network?
3. Other things to model:
    - Reliability modelling — what happens when links or nodes fail?
    - Overlay networks — P2P increasingly important.

Internet topology

- Two levels of topology are usually considered "router level" and "autonomous system" (AS) level.
- Router level topology is still the least well-known — often ISPs take trouble to protect this information for security reasons.
- Topology metrics — these quantities are all rigorously defined and can be found in the literature:
  1. Graph diameter (longest possible "shortest path" between nodes).
  2. Node degree distribution (what proportion of nodes have k neighbours).
  3. Assortivity/disassortivity (do well-connected nodes connect with each other?) – sometimes called "rich club".
  4. Clustering (triangle count) – are the neighbours of a node also neighbours of each other.
  5. Clique size – largest group where everyone is everyone's neighbour (a clique in graph theory).

AS level topology

Power law networks

The node-degree distribution in AS networks is particularly well-studied. Let $P(k)$ be the proportion of nodes with degree k (having k neigbours). To a good approximation

$$P(k) \sim k^{-\alpha},$$

where α is a constant.

- Power law topology of the AS graph shown by [Faloutsos x3].
- This graph has some interesting properties — some extremely highly connected nodes, what happens if they fail?
- Same type of graph as:
  1. Links on websites, wikipedia and many other similar online systems.
  2. Academic citations in papers.
  3. Human sexual contacts.

Mathematics to generate AS topology

Albert–Barabasi [Barabasi 99] "Preferential attachment" model

Constructive Start with a small "core" network. When a new node arrives, attach it to an old node with the following probability

$$\mathbb{P}\left[\text{Attaching to node } i\right] = \frac{d(i)}{\sum_{j \in \text{all nodes}} d(j)},$$

where $d(i)$ is the degree of node i.

- This model "grows" a network with a powerlaw.
- Many similar models have been created which are more general.
- Current best model may be [Zhou 2004] Positive Feedback Preference which adds a small "faster than exactly proportional" term.

User/flow arrival modelling

- As a first approximation the arrival of users can be modelled as a Poisson process.
- You might want to consider periodic effects:
  1. Daily – with people's sleep cycles.
  2. Weekly – weekends different.
  3. Yearly – year-on-year growth in traffic.
- Perhaps simpler just to simulate some peak hour and some estimate of growth?

Application level protocols

- If you are modelling a specific application there will be details associated with this.
- Common applications (www, ftp, p2p) will have existing research — read what is done before setting out on your own.
- If no studies are done what could you compare your application to?
- Could your application be viewed as:
  1. A series of ftp-like transfers of data.
  2. UDP bursts at a given rate for given periods of time
  3. A p2p application which might use existing p2p research methods.
- An important thing to simulate is the length of transfers and for many applications this is heavy-tailed.

What is a Heavy-Tailed distribution?

Heavy-Tailed distribution

A variable X has a heavy-tailed distribution if

$$\mathbb{P}[X > x] \sim x^{-\beta},$$

where $\beta \in (0, 2)$ and $\sim$ again means asymptotically proportional to as $x \to \infty$.

- Obviously an example of a power law.
- A distribution where *extreme values* are still quite common.
- Examples: Heights of trees, frequency of words, populations of towns.
- Best known example, Pareto distribution $\mathbb{P}[X > x] = (x/x_m)^{-\beta}$ where $x_m > 0$ is the smallest value X can have.

Heavy tails and the internet

- The following internet distributions have heavy tails:
  1. Files on any particular computer.
  2. Files transferred via ftp.
  3. Bytes transferred by single TCP connections.
  4. Files downloaded by the WWW.
- This is more than just a statistical curiousity.
- Consider what this distribution would do to queuing performance (no longer Poisson).
- Non mathematicians are starting to take an interest in heavy tails (reference to “the long tail”.

Long-Range Dependence (LRD) and the Internet

- In 1993 LRD was found in a time series of bytes/unit time measured on an Ethernet LAN [Leland et al '93].
- This finding has been repeated a number of times by a large number of authors (however recent evidence suggests this may not happen in the core).
- A higher Hurst parameter often increases delays in a network. Packet loss also suffers.
- If buffer provisioning is done using the assumption of Poisson traffic then the network will probably be underspecifed.
- The Hurst parameter is "a dominant characteristic for a number of packet traffic engineering problems".

Long-Range Dependence (LRD)

Let $\{X_1, X_2, X_3, \ldots\}$ be a weakly stationary time series.

The Autocorrelation Function (ACF)

$$\rho(k) = \frac{\mathrm{E}\left[(X_t - \mu)(X_{t+k} - \mu)\right]}{\sigma^2},$$

where μ is the mean and σ^2 is the variance.

The ACF measures the correlation between X_t and X_{t+k} and is normalised so $\rho(k) \in [-1, 1]$. Note symmetry $\rho(k) = \rho(-k)$.
A process exhibits LRD if $\sum_{k=0}^{\infty} \rho(k)$ diverges (is not finite).

Definition of Hurst Parameter

The following functional form for the ACF is often assumed

$$\rho(k) \sim |k|^{-2(1-H)},$$

where $\sim$ means asymptotically proportional to and $H \in (1/2, 1)$ is the Hurst Parameter.

More about LRD

- Think of LRD as meaning that data from the distant past continue to effect the present.
- LRD was first spotted by a hydrologist (Hurst) looking at the flooding of the Nile river.
- For this reason Mandelbrot called it "the Joseph effect".
- Stock prices (once normalised) also show LRD.
- LRD can also be seen in the temperature of the earth (once the trend is removed).
- Models include Markov chains, Fractional Brownian Motion (variant on Brownian motion), Chaotic maps and many others.

Transport and network level protocols

- It might be important if we are considering a packet level model to model specific details of the TCP/IP protocols.
- Usually this will involve simulating the window size (additive increase multiplicative decrease) of the TCP protocol.
- Remember that a detailed simulation to this level will extremely limit the number of nodes which can be simulated.
- A mathematical model will be demonstrated in the next section.
- In addition, the ns-2 model will be shown which is a packet level simulation of TCP/IP.

Other things to model

- Of course depending on the nature of your modelling, there may well be other aspects of the network to be modelled.
- Some examples might be:
    1. Reliability of nodes and links.
    2. An overlay network.
    3. Possible hostile attacks to the network.
- In all cases, an important starting point is to find out what research already exists in the area.
- Are any real-life data sets available which could inform your modelling? Could you gather such data?

Mathematical modelling

- To create a simulation model we need to be able to write down equations for the system.
- The more work we can do “on paper” the easier the computational burden.
- This will be illustrated with two mathematical models related to networks.
- The first model is a buffer model using Markov chains.
- The second model is a model of TCP/IP to estimate throughput.
- These models can be used as a basis for computer simulation.

Queuing analysis of the leaky bucket model

- A "leaky bucket" is a mechanism for managing buffers and to smooth downstream flow.
- What is described here is sometimes known as a "token bucket".
- A queue holds a stock of "permit" generated at a rate r (one permit every $1/r$ seconds) up to a maximum of W.
- A packet cannot leave the queue without a permit – each packet takes one permit.
- The idea is that a short burst of traffic can be accommodated but a longer burst is smoothed to ensure that downstream can cope.
- Assume that packets arrive as a Poisson process at rate λ.
- A Markov model will be used [Bertsekas and Gallager page 515].
- A long digression is now necessary to explain the concept of the Markov chain.

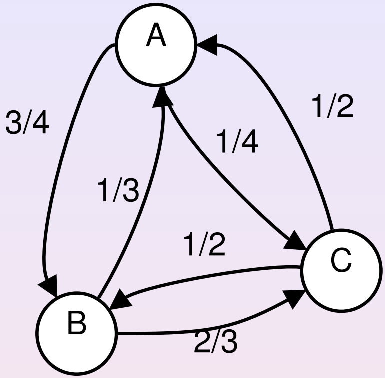
Introducing Markov chains

Markov Chains

Markov chains are an elegant and useful mathematical tool used in many applied areas of mathematics and engineer but particularly in queuing theory.

- Useful when a system can be in a countable number of "states" (e.g. number of people in a queue, number of packets in a buffer and so on).
- Useful when transitions between "states" can be considered as a probabilistic process.
- Helps us analyse queues.

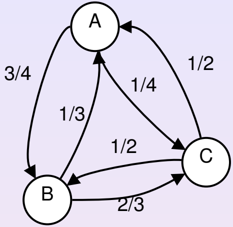
Introducing Markov chains – the hippy hitchhiker



- The hippy hiker moves between A-town, B-town and C-town.
- He moves once and only once per day.
- He moves with probabilities as shown on the diagram.

The hippy hitcher (2)

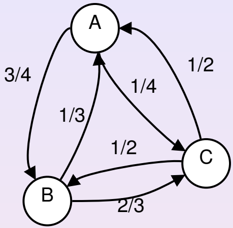
- Want to answer questions such as:
- What is probability he is in A-town on day n?
- Where is he most likely to "end up"?



- First step – make system formal. Numbered states for towns 0, 1 2 for A, B, C.
- Let p_{ij} be the probability of moving from town i to j on a day ($p_{ii} = 0$).
- Let $\lambda_{i,j}$ be the probability he is in town j on day i.
- Let $\boldsymbol{\lambda}_i = (\lambda_{i,0}, \lambda_{i,1}, \lambda_{i,2})$ be the vector of probabilities for day i.
- For example $\boldsymbol{\lambda}_0 = (1, 0, 0)$ means definitely in A town (0) on day 0.

The hippy hitcher (3)

- Define the probability transition matrix P.
- Write down the equation for day n in terms of day $n+1$.
- We have: $\lambda_{j,n} = \sum_i \lambda_{i,n-1} p_{ij}$.



Transition matrix

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{bmatrix}.$$

Matrix equation is $\boldsymbol{\lambda}_i^T = \boldsymbol{\lambda}_{i-1}^T \mathbf{P}$.

Equilibrium probabilities

- The matrix equation lets us calculate probabilities on a given day but where does hippy "end up".
- Define "equilibrium probabilities" for states $\pi_i = \lim_{n\to\infty} \lambda_{n,i}$.
- Think of this as probability hippy is in town i as time goes on.
- Define equilibrium vector $\boldsymbol{\pi} = (\pi_0, \pi_1, \pi_2)$.
- Can be shown that for a finite connected aperiodic chain this vector exists is unique and does not depend on start position $\boldsymbol{\lambda}_0$.
- From $\boldsymbol{\lambda}_i^T = \boldsymbol{\lambda}_{i-1}^T \mathbf{P}$ then $\boldsymbol{\pi}_i^T = \boldsymbol{\lambda}_{i-1}^T \boldsymbol{\pi}$.
- This vector and the requirement that probabilities sum to one uniquely defines π_i for all i.

Equilibrium probabilities – balance equations

- The matrix equation for $\boldsymbol{\pi}$ can also be thought of as "balance equations".
- That is in equilbrium, at every state the flow in a state is the sum of the flow going into it.
- $\pi_j = \sum_i p_{ij}\pi_i$.
- This and $\sum_i \pi_i = 1$ are enough to solve the equations for π_i.

$$
\begin{aligned}
\pi_0 + \pi_1 + \pi_2 &= 1 && \text{probabilities sum to one} \\
\pi_1 p_{10} + \pi_2 p_{20} &= \pi_0 && \text{balance for city 0} \\
\pi_1 p_{01} + \pi_2 p_{21} &= \pi_1 && \text{balance for city 1} \\
\pi_1 p_{02} + \pi_2 p_{12} &= \pi_2 && \text{balance for city 2}
\end{aligned}
$$

Solves as $\pi_0 = 16/55$, $\pi_1 = 21/55$ and $\pi_2 = 18/55$ for hippy.

Markov chain summary

- A Markov chain is defined by a set of states and the probability of moving between them.
- This type of Markov chain is a discrete time homogeneous markov chain.
- Continuous time Markov chains allow transitions at any time not just once per "day".
- Heterogenous Markov chains allow the transition probabilities to vary as time changes.
- Markov chains can be used in many types of problem solving, particularly queues.

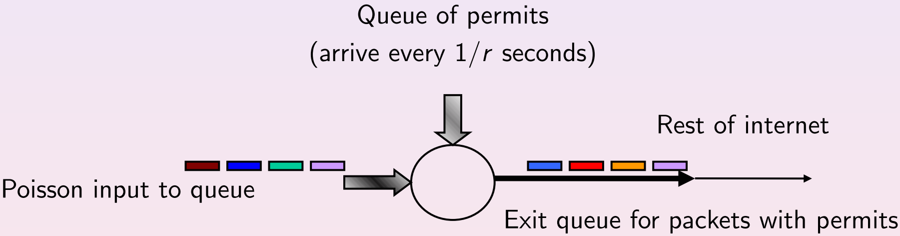
The Birth-Death process

- A "birth-death" process is a basic type of "queue".
- "birth" means an arrival at a queue and a "death" a departure.
- Model as Markov chain – state is number of people (or packets) in queue.
- $p_{n,n+1}$ is probability of arrival in state n (one person joins queue of n people).
- $p_{n,n-1}$ is probability of departure from state n (one person leaves queue of n people).
- π_n is the probability that there are n people in the queue after it reaches "equilibrium".

Modelling the leaky bucket

Use a discrete time Markov chain where we stay in each state for time $1/r$ seconds (the time taken to generate one permit). Let a_k be the probability that k packets arrive in one time period. Since arrivals are Poisson,

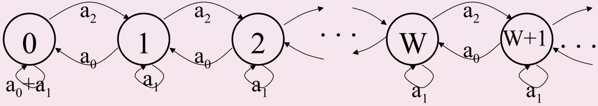
$$a_k = \frac{e^{-\lambda/r}(\lambda/r)^k}{k!}.$$



A Markov chain model of the situation

- In one time period (length $1/r$ secs) one token is generated (unless W exist) and some may be used sending packets.
- States $i \in \{0, 1, \ldots, W\}$ represent no packets waiting and $W - i$ permits available. States $i \in \{W+1, W+2, \ldots\}$ represent 0 tokens and $i - W$ packets waiting.
- If k packets arrive we move from state i to state $i + k - 1$ (except from state 0).
- Transition probabilities from i to j, $p_{i,j}$ given by

$$p_{i,j} = \begin{cases} a_0 + a_1 & i = j = 0 \\ a_{j-i+1} & j \geq i - 1 \\ 0 & \text{otherwise} \end{cases}$$



Solving the Markov model

Let π_i be the equilibrium probability of state i. Now, we can calculate the probability flows in and out of each state.
For state one

$$\pi_0 = a_0\pi_1 + (a_0 + a_1)\pi_0$$
$$\pi_1 = (1 - a_0 - a_1)\pi_0/a_0.$$

For state $i > 0$ then $\pi_i = \sum_{j=0}^{i+1} a_{i-j+1}\pi_j$. Therefore,

$$\pi_1 = a_2\pi_0 + a_1\pi_1 + a_0\pi_2$$
$$\pi_2 = \frac{\pi_0}{a_0}\left(\frac{(1 - a_0 - a_1)(1 - a_1)}{a_0} - a_2\right).$$

In a similar way, we can get π_i in terms of $\pi_0, \pi_1, \ldots, \pi_{i-1}$.

Solving the Markov model (part 2)

- We could use $\sum_{i=0}^{\infty} \pi_i = 1$ to get result but this is difficult.
- Note that permits are generated every step except in state 0 when no packets arrived (W permits exist and none used up).
- This means permits arrive at rate $(1 - \pi_0 a_0)r$.
- Rate of tokens arriving must equal λ unless the queue grows forever (each packet gets a permit).
- Therefore $\pi_0 = (r - \lambda)/(ra_0)$.
- Given this we can then get π_1, π_2 and so on.

Completing the model

- Want to calculate T average delay of a packet.
- If we are in states $\{0, 1, \ldots, W\}$ packet exits immediately with no delay.
- If we are in states $i \in \{W+1, W+2, \ldots\}$ then we must wait for $i - W$ tokens $(i - W)/r$ seconds to get a token.
- The proportion of the time spent in state i is π_i.
- The final expression for the delay is

$$T = \frac{1}{r} \sum_{j=W+1}^{\infty} \pi_j (j - W).$$

- For more analysis of this model see Bertsekas and Gallagher page 515.

Graph Theory

- Graph theory is another incredibly useful branch of mathematics for modelling networks.
- Graph theory is a formal way to analyse networks as mathematical objects.
- Routers and connections are abstracted to notions of "nodes" and "arcs".
- Graph theory gives us useful proofs and algorithms – for example how to find a shortest path.
- There is only time for a brief flavour of graph theory in this lecture.

Graph Theory – Basic Definitions

A graph $G = (\mathcal{N}, \mathcal{A})$ is a finite set of $\mathcal{N}$ nodes and a set $\mathcal{A}$ of unordered pairs (i, j) where $i, j \in \mathcal{N} : i \neq j$ (known as arcs).

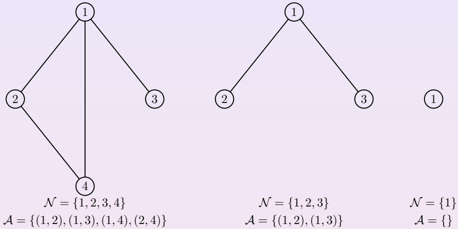


Figure: Example graphs

A *walk* in a graph G is a sequence of nodes in a graph $(n_1, n_2, \ldots, n_l)$ such that each adjacent pair is an arc.

Graph Theory – More Basic Definitions

A *path* is a walk with no repeated nodes.

A *weighted graph* $G = (\mathcal{N}, \mathcal{A})$ is one where each arc $(i, j) \in \mathcal{A}$ has associated with it a weight w_{ij}.

The *shortest path problem* is for a weight graph $G = (\mathcal{N}, \mathcal{A})$ finding a path from n_1 to n_l which minimises the sum $\sum_{i=1}^{l-1} w_{n_i n_{i+1}}$.

Naturally this is useful in routing algorithms.

Bellman-Ford algorithm

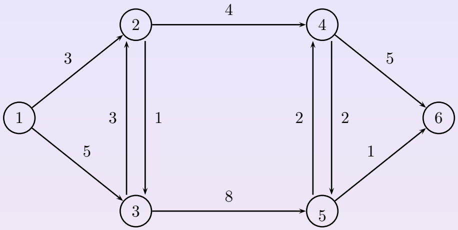
D_i^h is the distance of the shortest walk from node 1 to node i of h steps or less.
Set initially $D_i^0 = \infty$ for $i \neq 1$ and $D_1^h = 0$ for all h.
The Bellman-Ford Algorithm is then simply, for all $i \neq 1$,

$$D_i^{h+1} = \min_j [D_j^h + w_{ji}]$$

The algorithm terminates after h iterations if

$$D_i^h = D_i^{h-1} \quad \forall i$$

Example



i	D_1^i	D_2^i	D_3^i	D_4^i	D_5^i	D_6^i
1	0	3	5	∞	∞	∞
2	0	3	4	7	13	∞
3	0	3	4	7	9	12
4	0	3	4	7	9	10
5+	0	3	4	7	9	10

$D_i^0 = \infty$ for $i \neq 1$ and $D_1^h = 0$ for all h

$$D_i^{h+1} = \min_j [D_j^h + w_{ji}]$$

The ns-2 simulation

- ns-2 is a freely available event-driven simulator which simulates packet-level traffic.
- It is available from `http://www.isi.edu/nsnam/ns/`
- The simulator is written in C++ but uses tcl for simulations.
- The scripts used for the rest of this lecture are available at `http://www.richardclegg.org/lectures`

Final thoughts

- Select an appropriate level of modelling — if you need to model the whole internet you cannot do packet level modelling. If you need to model intricate protocol details for packets you cannot model the whole internet.
- Check against real data where possible that your modelling assumptions are justified.
- Is your experiment repeatable? Do you get similar results if you try slightly different starting scenarios?
- Remember sensitivity analysis: What happens if the bandwidth is a little less? What if the demand is a little more?
- Can statistical analysis of your results help?
- Remember that what you model today is out of date in a year and hopelessly obsolete in ten years.