Epidemic data flooding in peer-to-peer systems

Hannu Reittu

Internet publishing/ single file distribution

- A server based solution:expensive, limited bandwidth -> ‘linear performane’
- p2p: BitTorrent: file is distributed among the peers in small ‘chunks’
- sharing uplink resource of peers -> ‘exponential performance’
- a web server, ‘tracker’ coordinates the peers
- a point of failure and a threshold for publishing
- -> a need for ‘trackerless’ file sharing system

Several prototypes:

- eXeem, Azureus…(beta)
- tracker-> DHT
- PAN-NET client, DHT=Chord, random encounter

Example

- A company network with 100 000 PCs
- distribute a 4 MB file with a server with bandwidth 100 Mb/s
- unicast:100 000 x 4 MB/100Mb/s = 10 hours
- p2p: 1 minute (server-> seeder with 10 contacts at the time)

Simulations and analysis:

- Massoulié and Vojnović: Coupon Replication Systems (analysis)
- Felber, Biersack: Self-scaling Networks for Content Distribution (simulation)
- Balakrishna Prabhu: simulations (Ercim fellow at VTT)

Three models

- Push model: deliver a file to a fixed population
- pull model: Internet flash-crowd for a popular file
- constant arrival rate

Push model with large number of peers

- Extreme case of the flash-crowd, possible important, worst case
- random peer selection
- chunk selection: random, rarest first?
- first injected nodes replicate exponentially
- rare junks appear, have to get from the seeder (central serving…)

How does it look like?
A crude model (~Exp(1) delays)

1 Start: chunks *0* and *1*

2 select one chunk randomly and uniformly and duplicate it

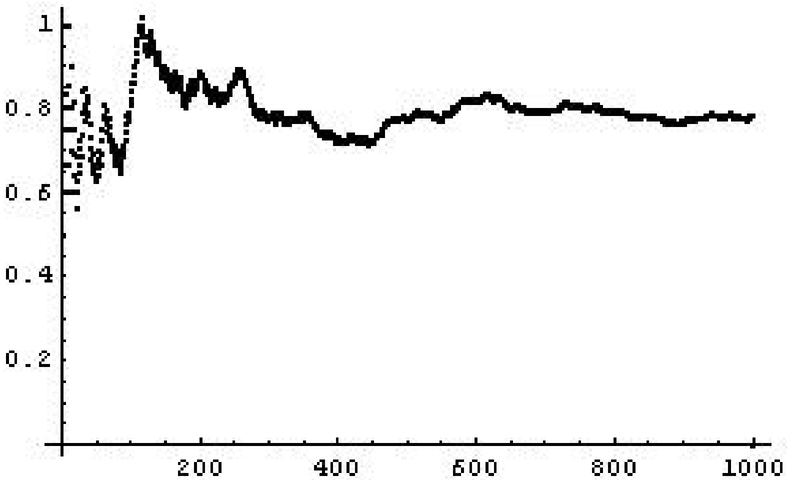
3 return to **2**

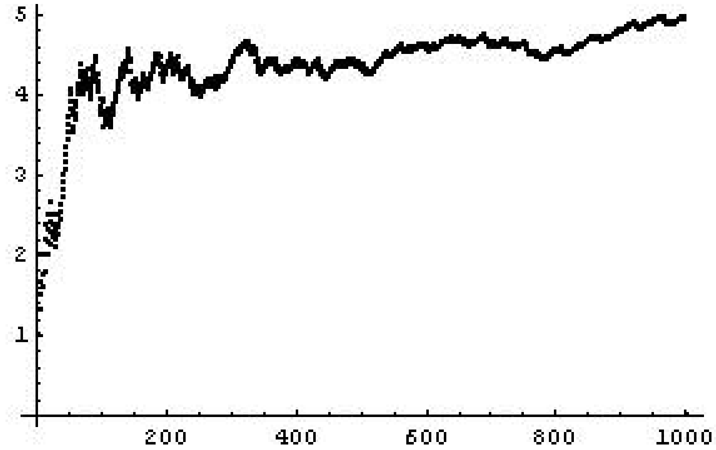
1 is more frequent, this times
in general, any ration with uniform probability!

0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,

1000 steps:

$$\frac{\#\,1}{\#\,0}$$





Corresponds to a Pólya urn model
(see W Feller, 'An Introduction to Probability…')

- An event: first *m1* times *1* then *m0* times *0*

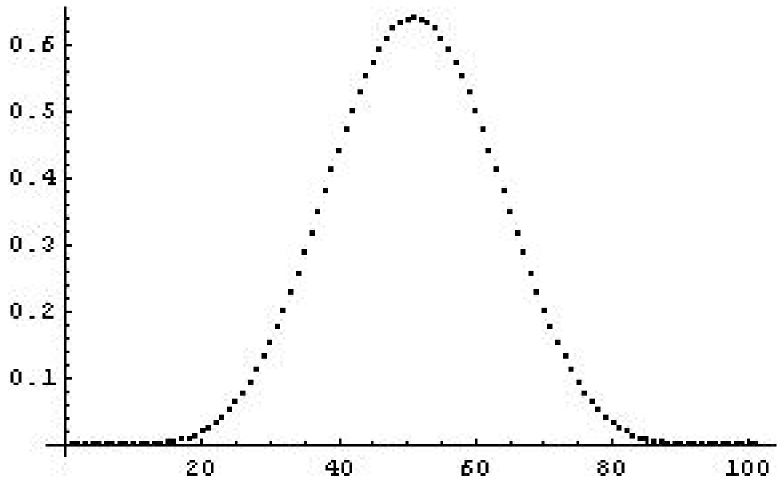
$$\Pr(m1 \mapsto m0) = \frac{1 \bullet 2 \bullet .... \bullet m1}{2 \bullet 3 \bullet ...(m1+1)} \frac{1 \bullet 2 \bullet ...m0}{(m1+2)...(m1+m0+1)} = \frac{m1!\,m0!}{(n+1)!}$$

Does not depend on the order, in general, {*m1*,*m0*}:

$$\Pr(m1, m0) = \binom{n}{m1} \Pr(m1 \mapsto m0) = \frac{1}{n+1}$$

Uniform distribution
However first selections matters

Start, say, with *c1=10* and *c0=10*, proceed, *n=100*



After *n* steps,

- *m1(n)/n* takes any value in {*0,1/n,…,1*} uniformly
- conditionally on *m1(n)/n*- remains further almost constant,
- a martingale:

$$\xi(n) = m1(n)/n$$

$$E(\xi(n)/\xi(n-1)) = E(\frac{a(n)+m1(n-1)}{n}/\xi(n-1)) = \frac{n-1}{n}\xi(n-1) + \frac{P(a(n)=1/\xi(n-1))}{n} =$$

$$= \frac{n-1}{n}\xi(n-1) + \frac{\xi(n-1))}{n} = \xi(n-1)$$

The same with the seeder

- Start from the seeder, seeder gives *1* with probability *1/2*
- seeder is also a peer (remains)
- encountering *1* conditionally on *m1, m0*

$$P(1/m1,m0)=\frac{1}{2}\frac{1}{n+1}+\frac{m1}{n+1}=\frac{2m1+1}{2(n+1)}$$

A Pólya urn model, tends (m1/(n+1) submartingale) to arcsin-law instead of uniform

$$P(m1,m0)=\frac{1}{\pi}\frac{\Gamma(m1+1/2)}{\Gamma(m1+1)}\frac{\Gamma(n-m1+1/2)}{\Gamma(n-m1+1)}\rightarrow\frac{1}{\pi n}\frac{1}{\sqrt{x(1-x)}},x=\frac{m1}{n}$$

Result: unpredictably unbalanced
A curiosity: can be relaxed

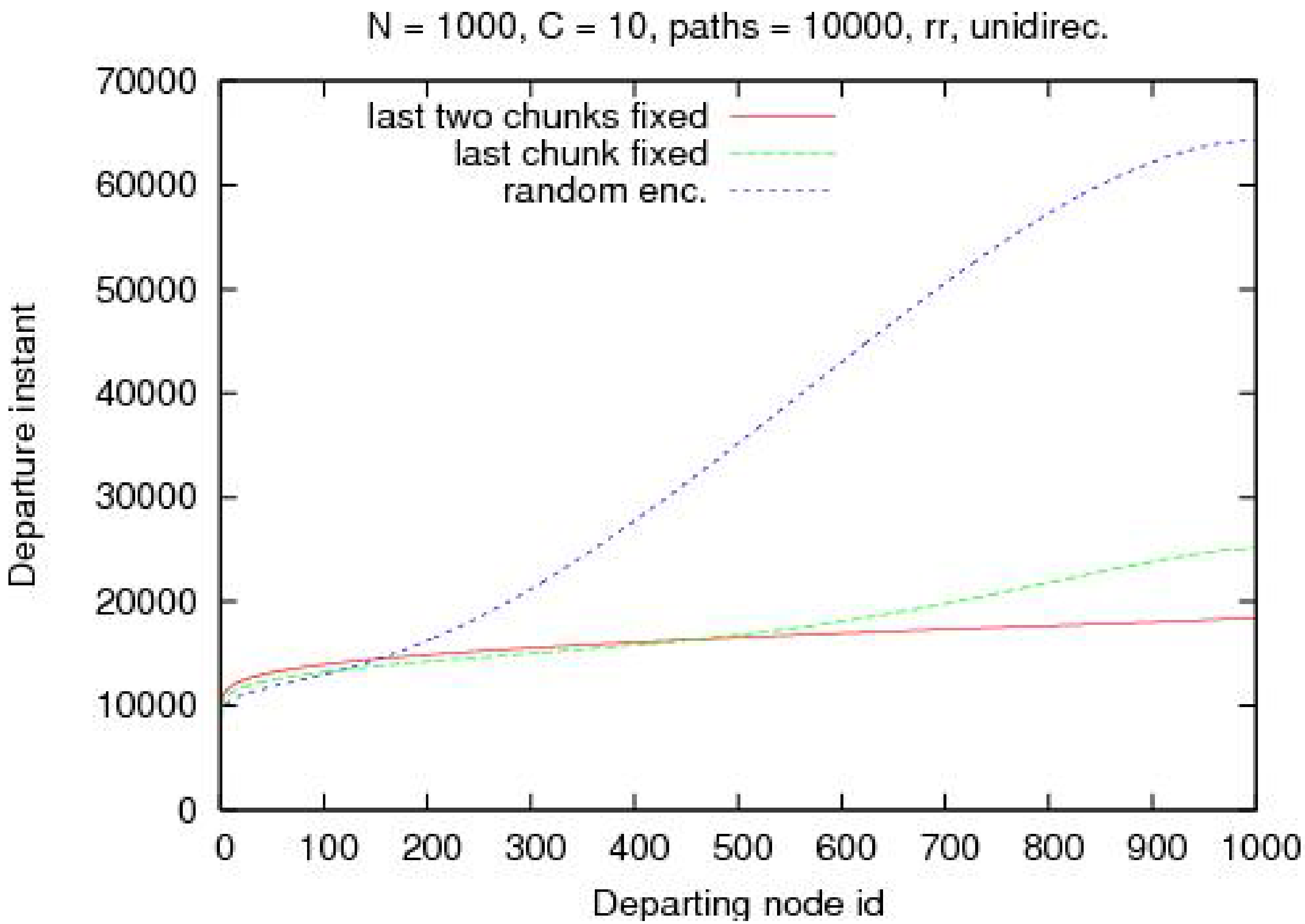
- Pick uniformly a ball (1 or 0), if 1 return it and add 0
- a bit similar to Ehrenfests urn model for ‘heat transfer’ (Feller)
- m0/m1 -> 1, with seeder:

For many (1000) chunks?

- Why unbalance is bad?
- the same last missing chunk (only one seeder)
- should use large numbers: chunks and peers and randomisation
- should outperform tracker (extra cost on network)

Try a simple solution first:

- Start: each node selects one chunk number uniformly and independently
- download any other missing random chunk unless only one is missing
- -> high diversity in the last missing chunk
- Balakrishna: seems to work quite well already with 10 chunks and 1000 nodes


N = 1000, C = 10, paths = 10000, rr, unidirec.
last two chunks fixed
last chunk fixed
random enc.
Departure instant
Departing node id
0
10000
20000
30000
40000
50000
60000
70000
100
200
300
400
500
600
700
800
900
1000