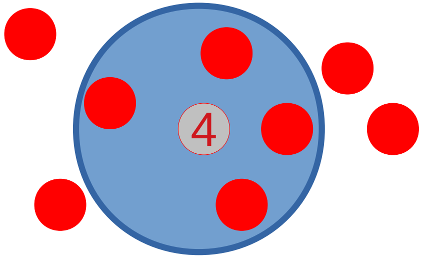
Efficient Density Estimation Algorithm for Ultra Dense Wireless Networks



Thierry Arrabal, Dominique Dhoutaut, **Eugen Dedu**

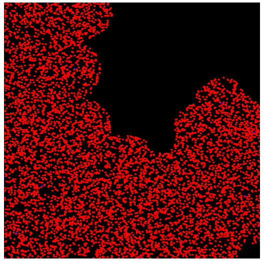
http://eugen.dedu.free.fr

Univ. Bourgogne Franche-Comté, FEMTO-ST Institute, CNRS, **France**

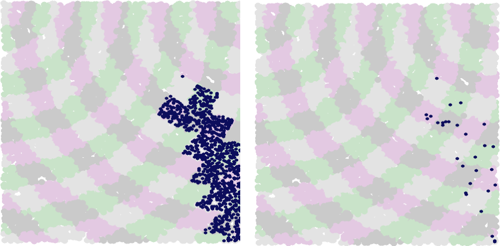
ICCCN conference
Hangzhou, China, July-Aug. 2018

Applications which could benefit from node density information

- Synonyms: node density, node degree, node cardinality, number of neighbours
- Send a message to all nodes in the network
  - use pure flooding, problem: huge traffic
  - better: probabilistic flooding, but which probability? How to prevent die out? Needs density
  - even better: backoff flooding, needs density to compute the backoff window size
- SLR, an addressing and routing protocol for nanonetworks
  - nodes decide themselves if they forward the packet received, contrary to IP where forwarder is decided by previous emitting node (based on routing table); needs density to reduce number of retransmitters
- Content centric networking (CCN) – how many nodes have a given content so that they can sleep for more or less time; needs density
- ...



Die out problem

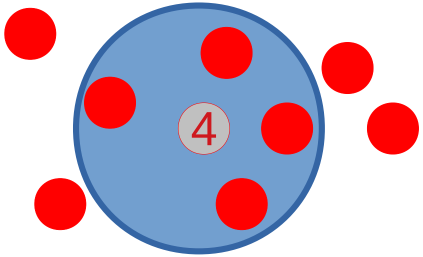


Many vs few nodes involved in SLR routing

How to count neighbours, current solutions

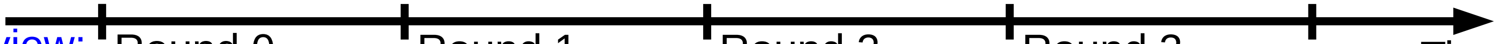
- So:
  - it is about neighbour counting, not about their identification, position, distance etc.
  - about counting all nodes, not only the communicating ones (got using collision probability)
- Count people in this room (imagine numerous participants)
  - hello method: I say "ping!", and each of you replies "pong!"
    - drawbacks: 1 packet per node, collisions (depending on message length compared to number of persons)
  - better: answering probability of 10% => .1 sent packets per node
    - drawbacks: still collisions; but which is the best probability? chicken and egg problem!
  - the more the neighbours (i.e. the bigger the density), the bigger the drawbacks!

Properties of the estimator



- Executed each time is needed:
  - either during network setup
  - or at hard coded intervals of time
  - or upon reception of a 1-hop signal received from a macro equipment
  - or when a given node broadcasts a message...
- Not precise, but gives an (maximum likelihood) estimation
- Tunable: application can set required confidence and error range of estimation
- Estimation obtained by only one node, or by all the nodes
- Small overhead in terms of number of exchanged packets
- Unaffected by background traffic
- Tiny memory footprint: only a counter per node is needed
- Targets high density networks, but works in low densities too

Algorithm of the estimator



A node's view:

Round 0
p0=.01
10 pkts rcvd

Round 1
p1=.03
28 pkts rcvd

Round 2
p2=.09
87 pkts rcvd

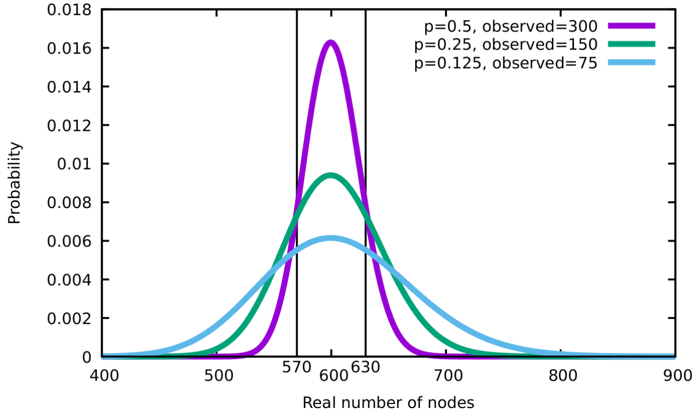
Round 3
p3=.27
264 pkts rcvd
=> 977 nodes
conf=95% with 10% error

Time

- Rounds of identical duration, known by all nodes
- Each round i, each node, after a backoff, sends a packet with probability pi
  - initially, p0 close to 0
  - pi grows exponentially with the round: pi = p0*growthRate^i, with growthRate>1
- The algorithm ends when:
  - either the number of packets received k during last round exceeds a threshold precomputed as a function of the desired confidence and pi
  - or pi≥1
- Then, each node estimates the number of neighbours as k/pi

Computation of the estimation

- Input: cmin, emax, pi, k (nb of pkts received)
- Output: n (number of nodes in reality) with given confidence and error
- Most likely k/pi neighbours, but is it within required error and confidence?
- We want n be in the interval:
  - n_min = k/pi * (1+emax)
  - n_max = k/pi * (1-emax)
- with a confidence (probability) P ≥cmin
  - where Pr is the probability that there are n nodes in reality

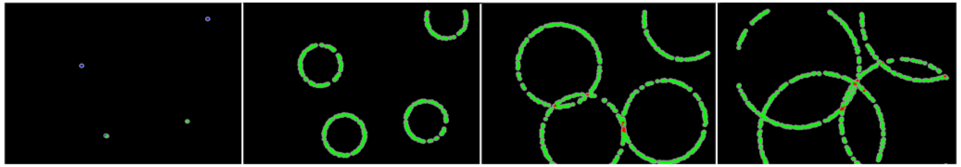


$$P = \frac{\sum_{n=n^{real}_{min}}^{n=n^{real}_{max}} Pr(k,n,p^{trans})}{\sum_{n=0}^{n=\infty} Pr(k,n,p^{trans})}$$

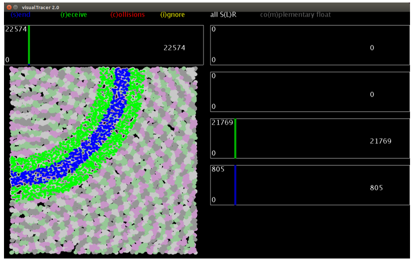
$$Pr(k,n,p) = Pr(X=k) = \binom{n}{k} p^k \, (1-p)^{n-k}$$

BitSimulator

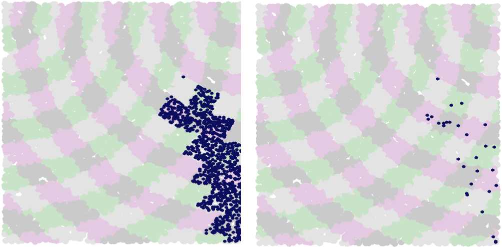
- The only simulator usable in dense networks, e.g. hundreds of thousands of nodes/neighbours
- Implements nanonetwork (network of nanonodes) features: TS-OOK modulation scheme based on pulses, packet overlapping, propagation delay, geographic bit collision etc.
- Deterministic results using several RNGs, for node positions, sending nodes, traffic pattern etc.
- Visualisation tools built in



30000 neighbours receiving 4 bits over time
(4 very small intervals of time)



Histograms for sent, received, and collided bits in an interval of time

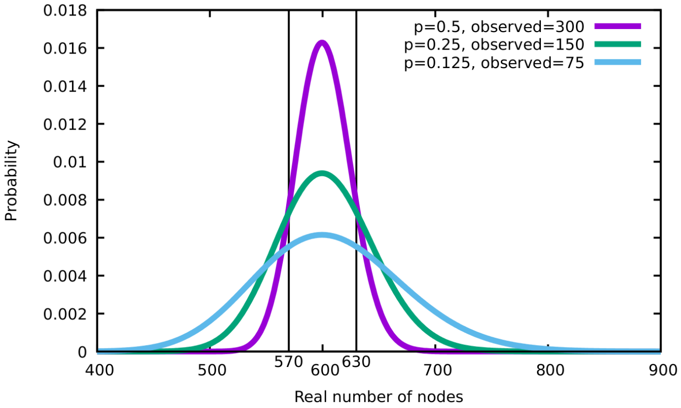


Many vs few nodes involved in SLR routing

D. Dhoutaut, T. Arrabal, E. Dedu. *BitSimulator, an electromagnetic nanonetworks simulator*. NanoCom 2018
http://eugen.dedu.free.fr/bitsimulator

The application chooses confidence and estimation error

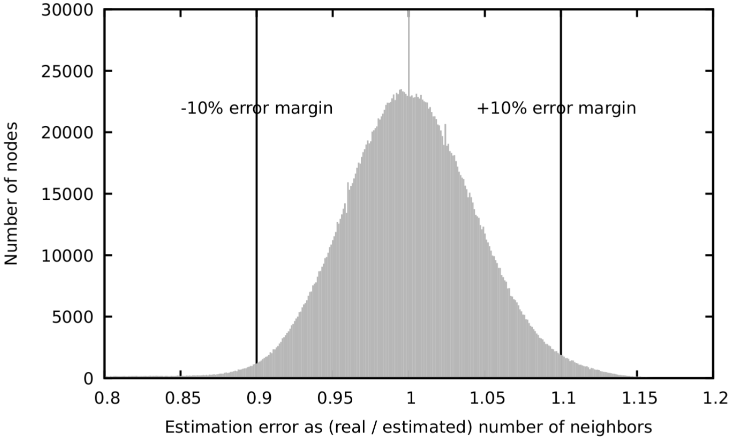
Theory: probability distribution of the confidence based on pi ("p" in the figure) and number of received packets k ("observed" in the figure)



- 600 neighbours most probably
- vertical lines: 5% error
- the confidence increases with p

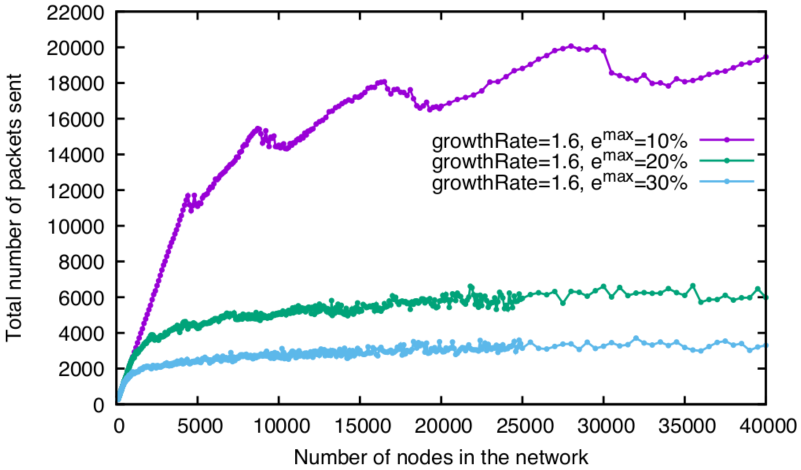
Simulation, similar results:

- 2.5mm x 2.5mm, comm. range 0.5mm
- emax=10%, confidence=95%, growthRate=1.6
- sum of 202 scenarios, nb. neighbours 10..4200



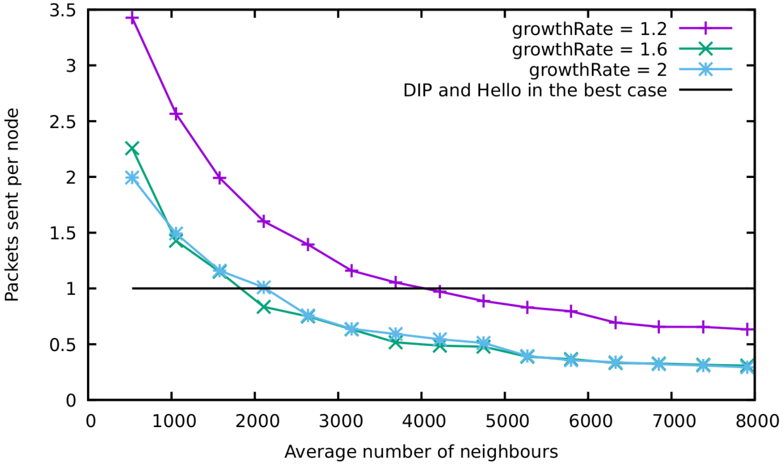
Overhead of the estimator: number of exchanged packets

- 2.5mm x 2.5mm, comm. range 0.5mm
- 200 scenarios, 100..40000 nodes placed randomly



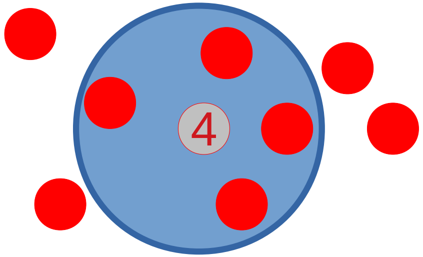
- emax has a big influence on the number of exchanged packets

emax = 10%



- Smaller overhead for dense networks => less bandwidth used, fewer collisions, ...
- The overhead gets smaller with the density

Conclusions



- Proposed an algorithm allowing nodes to estimate the number of their neighbours
- Useful to flooding, various transmission schemes, sleep decisions etc.
- Functioning: several rounds, with increasing probability of replying
- Tunable confidence and estimation error
- Has a tiny memory footprint, and a small overhead (number of packets exchanged)
- The denser the network, the better the results
- Future work: continuous estimation, use it to reduce network congestion