Ideas about Vector Clocks for Proximity and Location to determine mobility

This note described what vector clocks are, and how they have been used in the past; how we can adapt this for use with location and proximity, and finally, what information these vector clocks can yield.

How vector clocks work

A vector clock describes the interaction between parts, in a time ordered sequence.

How can we adapt VCs to work with proximity

?

How can we adapt VCs to work with location

?

What metrics can we derive from vector clocks to drive routing

- Out of date,
- essential links
- backbone network.

Notes (transcribed from my notebook)

Vector clocks for oppnets
Aquire vector clos as nodes discover other nodes, and swaps clock info. (each node has a unique id)

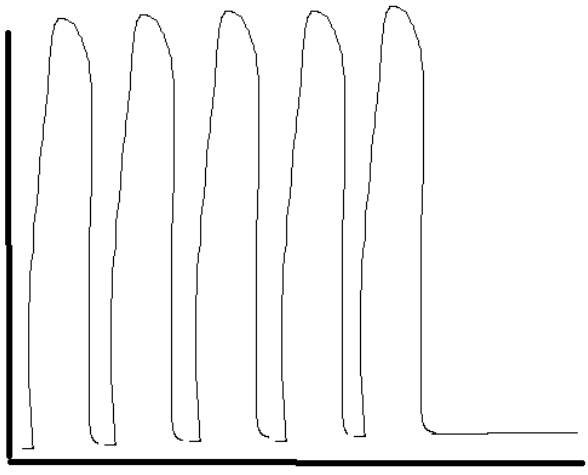
Observations of vector clocks:

- 1st compute for proximity then worry about location
- Bear in mind that ??? will be two way, so who does first update?
- Do we include out-nodes (i.e. ones that we do not have full records for)?

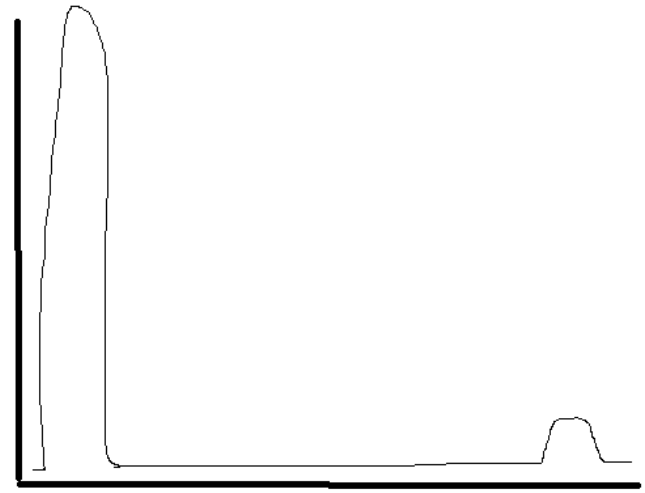
Likelyhoods of contacting a node again:

Friend – who must be identified
Drops off over time in a steady pattern, always likely to see a friend again

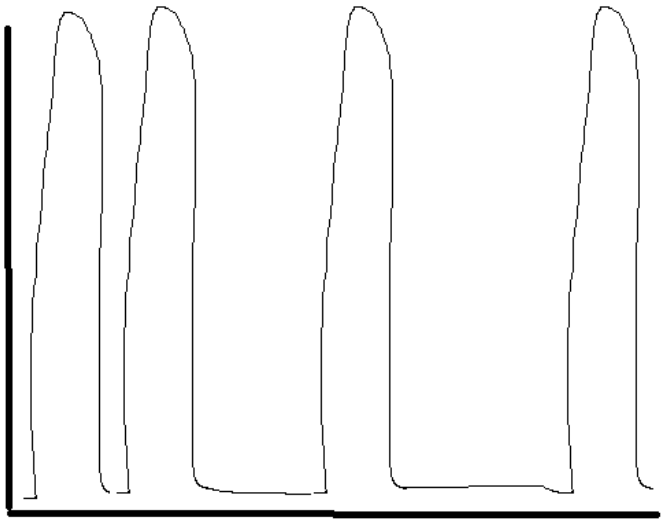
Colleague
Depends on day of week, with a regular pattern – you are very likely to see a work colleague during working hours, but much less likely during non working hours.



Unknown new contact
New contacts are unlikely to be seen again, and have a sharp drop-off, (perhaps a slim chance of seeing that person again at the same time next week?



Regular other contact
A regular contact might be seen on a some days at a regular place – e.g. a coffee shop
Contact pattern may be regular or semi-regular



On top of guestimation of contact, hold a routing table to next hop.
When i next see my friend, i'll ask if he knows X,
When i see new peoples, see who they know (largest hub)

<u>Issues</u>

- # Messages
- Storing of routing paths / known hosts
- Likelyhood of seeing that node again
- Direction of recipient
- Likeleyhood of a given node to deliver the message, or pass it on

If you know who friends are, you can also determine who colleagues and random aquaintences are.

Top Down
Routing using local node information
why: so nodes do not have to be connected to infrastructure – lower overhead?

How:

- Periodicity hunting
  - Patterns of proximity
  - Patterns of location?
- Vector Clocks
  - For proximity
  - For location
  - Can build metrics for
    - out of date nodes
    - range
    - update rate
    - update amount
    - ball of radius (based in out of date)
- How to pass messages:
  - Location prediction
  - Context aware routing
    - Using metrics derived from metrics above
- Find patterns
  - Datasets
    - Collected from n95
    - Reality mining

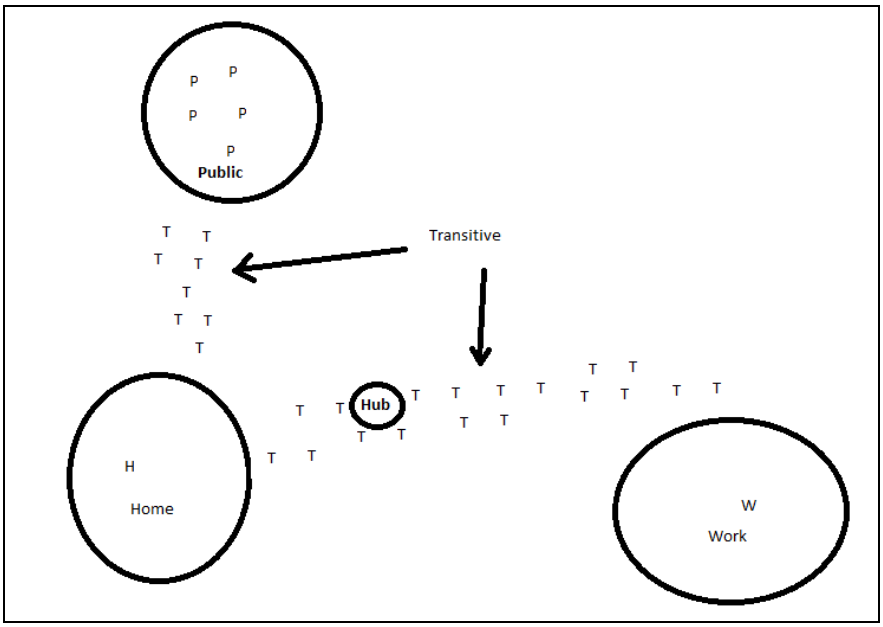
Vector Clocks with location
Set of locations *L*
Set of nodes *V*
Time period [0, *T*]
Co-location events
(*v,w,t,l*)
Node *v* saw node *w* at time *t* in location l
Location events
(v,l,t)
V visited location l at time t

ϕ v,t(l)	=	v's view of location l at time t
ϕ v,t (u)	=	v's view of node u at time t
ϕ v,l(u)	=	v's view of u at location l
ϕ v,t,l(u)	=	v's view of u at location l at time t

How to define a location

- For each node, track common location by measuring the number of readings at a location, and cluster them into a group which defines an area, OR

- Implicitly define a ‘location’ to be any place that a node visits regularly, defined as some precision of co-ordinate system e.g. using GPS decimal: 53.25, -6.14 is the center of a square approximately 100m by 100m
    - Location can be built up of adjacent squares
    - i.e. the node visits a general area regularly
- Define types of location
    - Home location is a place that a node considers to be its home location, or base location; this could be implicit, or derived by examining common locations associated with time (and maybe calendar)
    - Work locations are locations which nodes often go to, characterised by multiple un-connected nodes in the same area – e.g. city centre is built up of multiple public locations that are adjacent
    - Transitive locations are locations that one node often tracks through, but does not linger e.g. a route to work.
        - One nodes transitive location can be another nodes public/work/home location



    - Hub Location is a special type of public transitive location, and can be characterised by many nodes stopping for a short period of time before moving to some other destination. For example an airport or bus station.
        - Nodes here might be good candidates for propagation to diverse networks
    -

A nodes knowledge of another node is recorded over time

TV = {t,t ..... t’}

~~Tu = timestamps of ‘when’ new knowledge was received and from who~~
?u = {tv, (unfinished)

What is interesting and can be detected locally?

1. Build up knowledge of network by sharing contact/proximity knowledge
2. Vector clocks of proximity and location can tell us about the utility of a node for message carrying
3. Periodicity of nodes proximity is telling of their degree
4. Range – is the amount – a (unfinished)
5. Information latency

Entropy = amount of possible states see

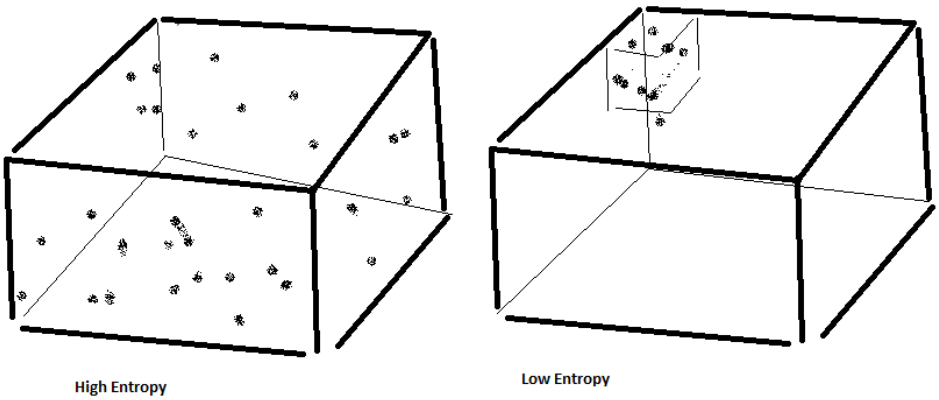


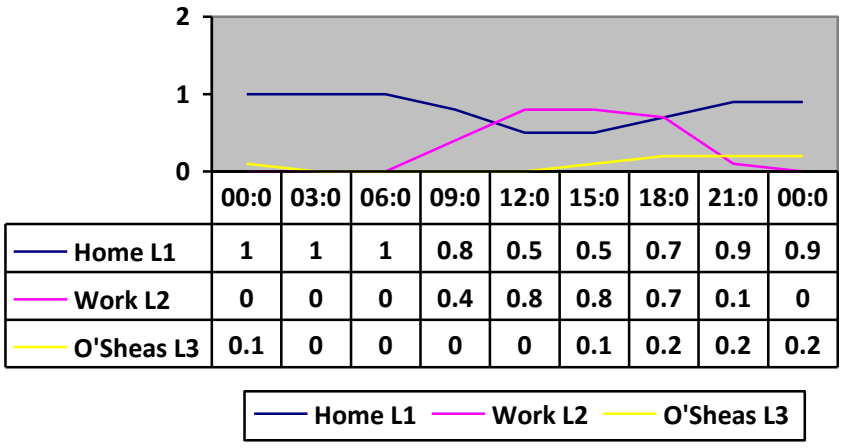
Figure 1 - Entropy

Location based routing (30 Jun 2010)

v = self node
w = other node

- Predict the next expected meeting for each known node
    - Or prediction model for time
- Predict next meeting place for each known node
- alternative known routes
- Known locations
    - Probability of visiting
    - Probability of co-location
- Which nodes visit locations that I don't and that destination node does
- Locations learned by query,
    - Can you (y) deliver to X?
      Yes, *n* hops, estimates at *t* time, via route (r, r', r'')
    - Store as route to X along with known co-location with y
    -
- If v and w trust each other, they share high fidelity location data, if not, they track only co-locations with each other.
- Nodes share routing information for other nodes
  e.g. I can deliver to X via y,z,a,b with next possible time of t

- Each node (v) generates a key/hash[1], that can be used to calculate a probability at any given time of day, that he will be at location L. He shares this with his trusted friends. And optionally specifies the extent to which it can be further shared.
- When node x is asked about delivering to node v, x uses his own knowledge his location history to calculate possible co-locations with v, using node v's hash for each of its known location and compares them to x's own locations, to predict the next time they will be co-located.
- When two nodes meet they can choose to share location hashes. If they choose not to, node v can calculate location probability for node w implicitly based on v's own knowledge of co-location with w.
- Timestamp Offsets! V and w may have different clock times. So at each interaction, they compare the current value of their clock information, and offset any data timestamps using the clock difference.
    - Node A has time DDMMYY 10h:43m:12s
    - Node B has time DDMMYY 10h:46m:58s
    - Difference of 226 seconds
    - Data passed from A is transformed by B, adding 226 seconds to any timestamped data it receives
    - Data passed from B is transformed by A, subtracting 226 seconds from any timestamped data it receives
- Nodes hold a record of their own location history, to some relevant period.[2]
    - Perhaps for as long as there is a unique reading of a location (e.g. 1 visit per year to a place, sets the period to 1 year)
- They use their location history to describe their probability of being at a location over time see Figure 2.



	00:0	03:0	06:0	09:0	12:0	15:0	18:0	21:0	00:0
Home L1	1	1	1	0.8	0.5	0.5	0.7	0.9	0.9
Work L2	0	0	0	0.4	0.8	0.8	0.7	0.1	0
O'Sheas L3	0.1	0	0	0	0	0.1	0.2	0.2	0.2

Figure 2 - Probability of being at a given location

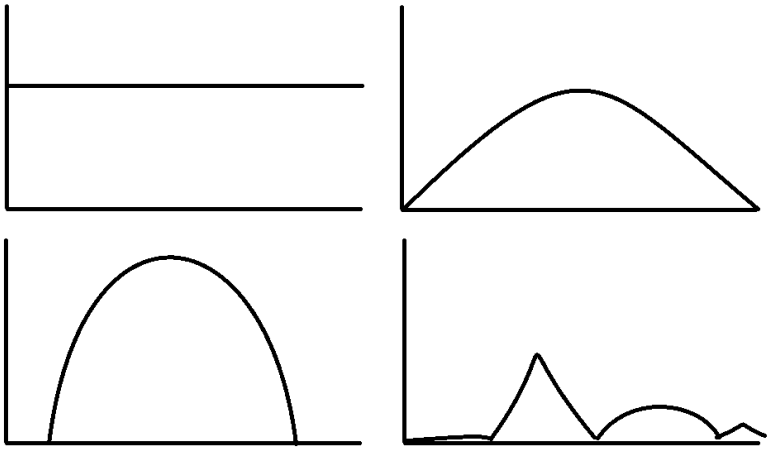
- This can be encoded as follows:

A	B	C	D	E
L1,	24,	7,	9eq,	1,1,1,0.8,0.5,0.5,0.7,0.9,0.9

[1] See Location based routing (30 Jun 2010) on page 6

[2] See Relevant periods to store data on page 7

- A = The location in question, this needs to be an agreed location, perhaps implied by some calculation on the GPS co-ordinates (see **How to define a location.**)
- B = period in hours over which the probabilities range; possible ranges: 24 (1 day), ?, 168 (7 days), 720 (30 days). It is the number of hours from midnight today, this week, this month that the probability spans
- C = period of ranges that are averaged to get he probability. This is a multiplier for B, e.g. 24 x 7 is an average over 7 days, for each 24 hour period of readings in those 7 days.
- D = Time period divisor, is a divisor for A, e.g. A/9eq meand the probabilities are split equally between 24 hours, in 9 parts.Possible values for subtext:
  - Eq = equal split
  - Cw = center weighted contracts the period to center of the day
  - Db = daily block concentrates readings into common daily working hours
  - Custom = specifies time ranges that apply



- E = List of probabilities that v will be at given location, this must be exactly |D| values (e.g. 24/9 = 9 probability values)

- Alternative is to give a series , with a list of probabilities of being at a location as in the table of Figure 2.
- Consider when two nodes meet, when they are not in a known location, e.g. when they meet en-route between locations. Could this be defined in a vector clock? Are between A & B, e.g. Vector clock for AB, a Pseudo Location – one which is not classed as a location – see How to define a location
- What does a transitive location look like? –

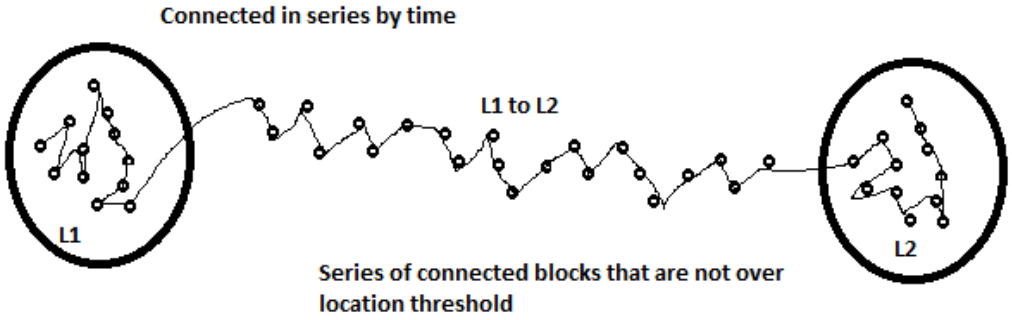


Figure 3 - What transitive location looks like

- a series of points between known locations

        - Series of connected blocks that are not over location threshold[3].
        - Pattern may also be reapeated, but not enough for a location,
        - Known as a 'route', perhaps with a 'direction'.
    - A location may be a cluster of readings repeated over time

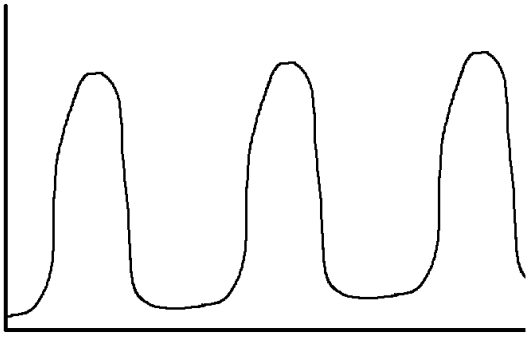


Figure 4 - cluster of readings over time

How to cluster readings into a location

Requirements:

- Must allow for occasional mis-readings
- Must not encompass too large an area
- Should consider sub-areas
    - Perhaps interfaced with Brightkite, Fireeagle, Gowalla, Foursquare etc.
- Over time can be labelled

i. Needs to have a learning phase, but on the fly can detect a new location if lingering for some pre-determined amount of time
ii. Triggers for 'location' or 'route' can be when detecting another node (possible collapse if too many nodes define all locations as a location)

- Nodes maintain state, e.g. current state might be 'in location' or 'en-route' or for privacy, 'incognito'
- When nodes transfer data, they may limit the query to just nodes they know. i.e. X says tell me about v1,v2,v99,v33
- How do nodes identify each other?
    - Bluetooth MAC address
    - Authentication mechanism
        - Perhaps to share data, nodes must know each other
- Simplified Probability sharing – simple list of the locations that node V1 might be at a given time of day.
  V1,24,7,9eq, (L1), (L1),(L1,L2),(L2),(L2,L3),(L2),(L3,L2),(L3,L1), (L3,L1),
- When nodes share information they can either just ask
    - for info about a node they have a message for, and get a delivery time in response.
    - oR ask for a list of nodes, and next probably delivery time.
    - Or a list of nodes with probability distribution over locations

[3] Number of reading required at a place to define it as a lingering location. See How to cluster readings into a location on page 9 and **Error! Reference source not found.** on page 11.

    - Or a combination
- The other way is to share all information and calculate the probability on each node – this might result in a very large amount of data being transferred, and a lot of calculations being made
- Clock updates / probability lists should have some notion of validity – time or authenticity.
    - Time is important because updates will become out of date
    - Authenticity is important (and hard) to avoid disruption of the system by nefarious nodes.

Dealing with Metrics

????

Relevant periods to store data

Period of location history
Probability updates validity

Graph Generation - Proximity

- For each reading,
    - if a node is in proximity to any other node via Bluetooth, add a directed edge
    - if any other node is within (x = 100m) distance, add a directed edge
    - if any node has also seen the same wifi point , add a directed edge
- Multiple edges are allowed (verification?)

V-> W

While

- If expected meeting with W is imminent
    - Keep Message
- Else if node in range has expected meeting within t time
    - Pass message with probability
- Else if node in range has expected meeting time sooner than own
    - Pass message with some probability

- Else if other node has a knowledge of other nodes that can deliver faster the itself of self
    - Pass message with some probability
- Else
    - Keep message

Expected Meeting

If my expected future crosses theirs, and that is {soon}

If i am going to the same place, but not at the same time, then i might see someone else who will cross paths with them

Algorithm has a ramp-up phase, which can be avoided with training data (google latitude, foursquare locations, gowalla locations, etc)

- At any given time, what is the probability of node being at location X
- At any given location, what is the probability of node being there at time X

Node Probabilities

If a node has been at a location in a time period, then record the number of times

When the count is low, user a different range to capture the overall pattern. E.g. consider using a monthly overage over a week to capture knowledge about less frequented locations.

Plan

- Build a skeleton graph of all locations for each user
- Build a skeleton graph for all interaction between users (e.g. proximity)
- Build a skeleton graph of all locations of all users
    - Edge is formed between a and b when user moves between a and b without an intermediate hop

Overview of techniques

	Positive	Negative

Periodicity Location/Proximity		Calculate over time
VC Location	Shared knowledge of network is distributed Local – record of ?????	Complexity increases with size of network
Location Prediction	Shared knowledge of location used to predict a given nodes future position	Must know about nodes
VC Proximity	Shared knowledge of social structure	complexity

Table 1 - Overview of techniques

1) Knowledge of locations is only useful when nodes visit locations independently
    a) As nodes can predict locations of co-locations
    b) Versus just knowing proximity

Hypotheses

1) Human mobility patterns are predictable
2) Human proximity patterns are predictable
3) Knowledge of proximity ***and location*** makes opportunistic routing more efficient than proximity alone.
4) There are low complexity algorithms based on vector clocks that can be used for routing
5) Any given node will only need to communicate with other nodes that they know
    a) *Most (?%) communications are to nodes within {x} hops locally, or {x++} hops globally*