signalframe

Building Temporal Graphs and Embeddings

A Practitioner's Approach

Srdjan Marinovic

February, 2020

About me

- Research background in security and non-monotonic systems
- Currently CTO at a DC-startup SignalFrame

SignalFrame

- Indexing public WiFi/Bluetooth infrastructure

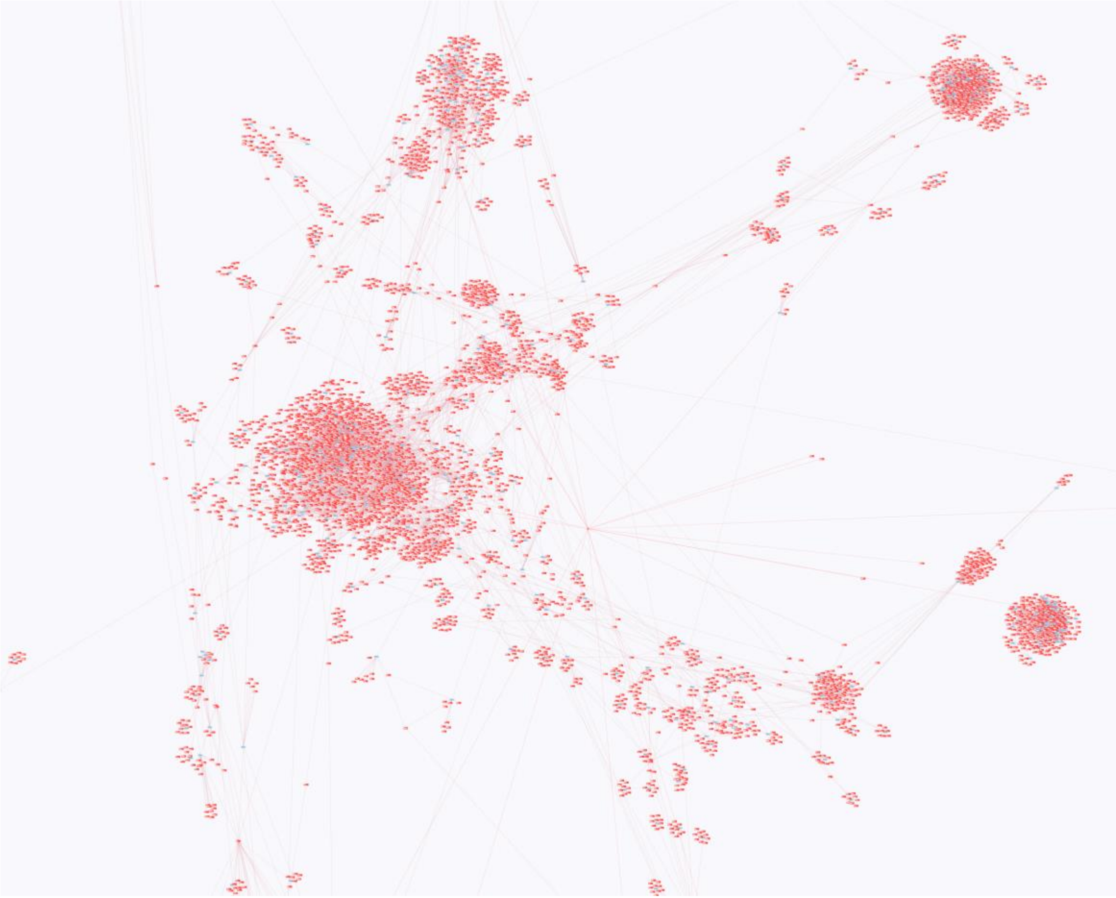
- Analyzing temporal changes and relationships between spaces and devices
  - Supplementing satellite image analysis
  - 2nd Factor Authentication
  - Market intelligence

SignalGraph

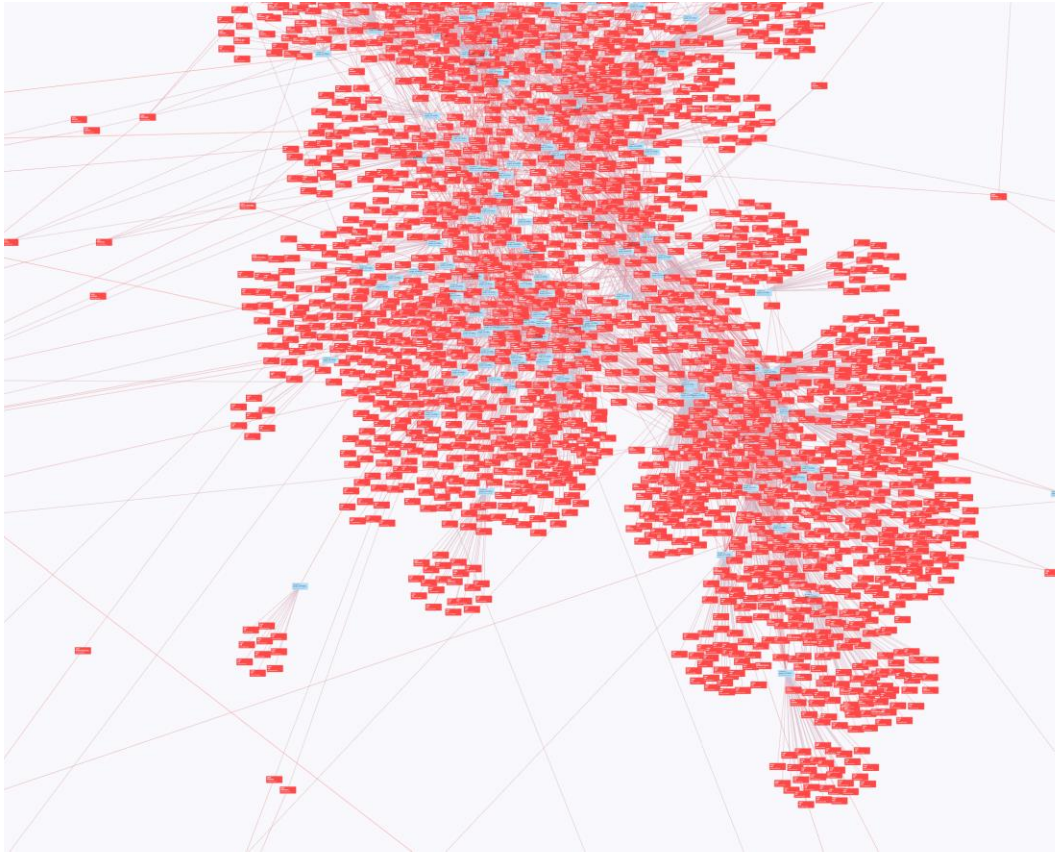
- Signals are nodes in a streaming temporal graph
- **~ 6 billion nodes**
- **~ 100 billion edges**

- **~ 300 million updated nodes** per day
- **~ 1 billion edge** updates per day

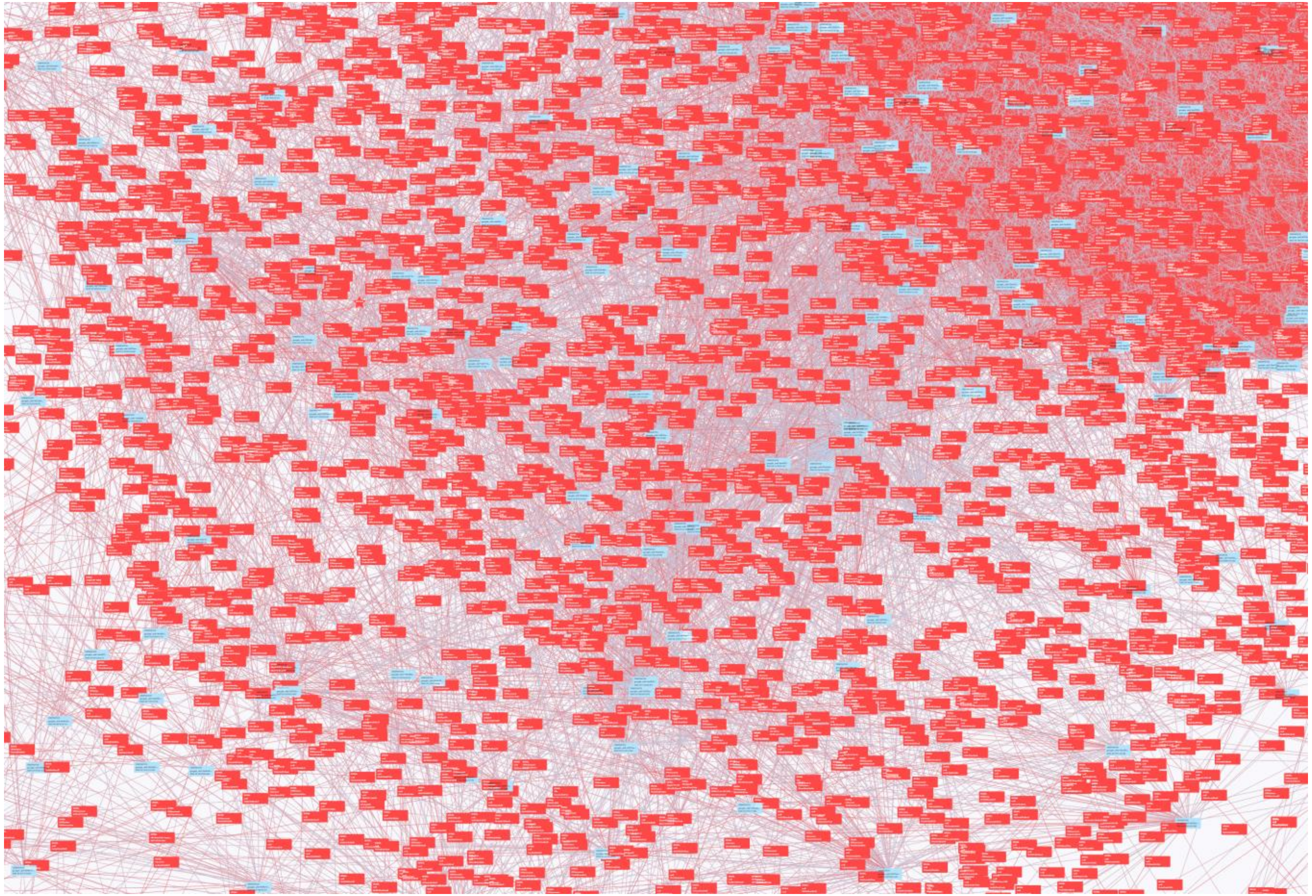
SignalGraph (GWU wifi @ 1 week Feb)



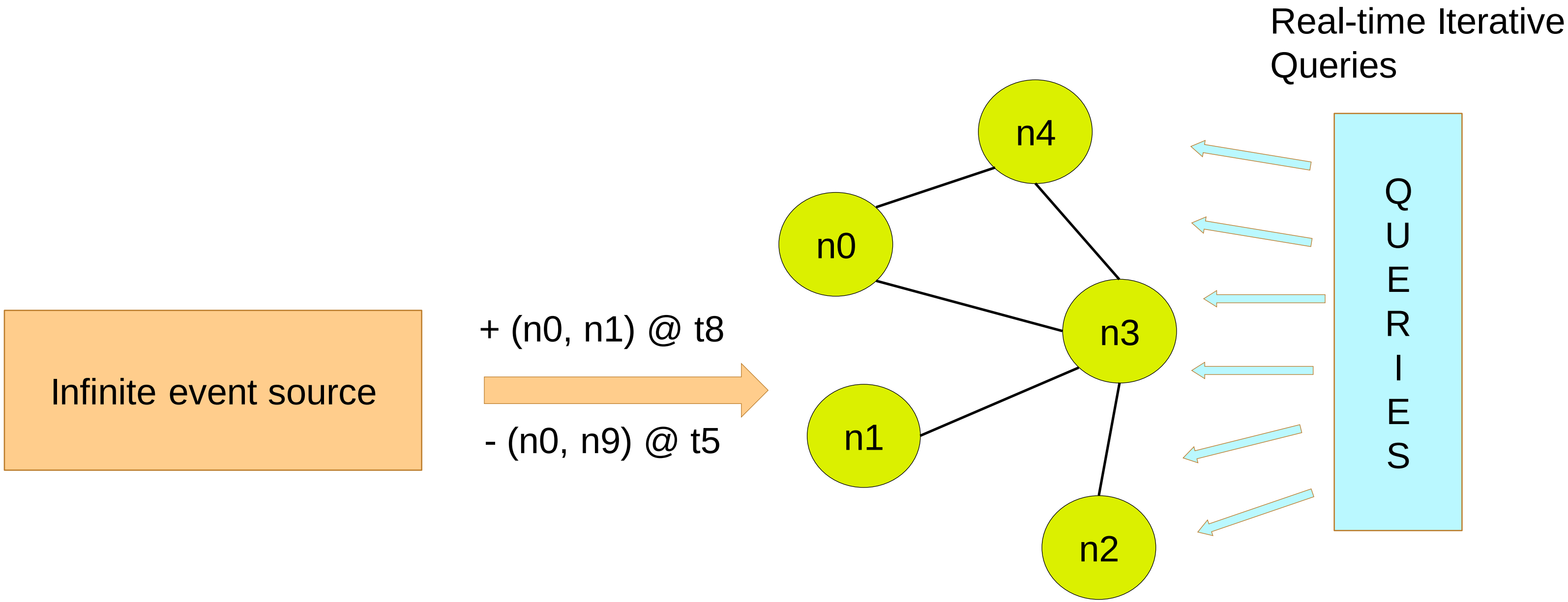
SignalGraph (GWU wifi @ 1 week Feb)



SignalGraph (GWU wifi @ 1 week Feb)



Temporal (Streaming) System Model



Temporal (Streaming) Systems

- Network analytics
  - Intrusion detection
- Recommendation engines
- User rankings
- Geo-temporal analytics

Practitioner's proposition

Model and analyze temporal graphs via explicit temporal nodes and edges.

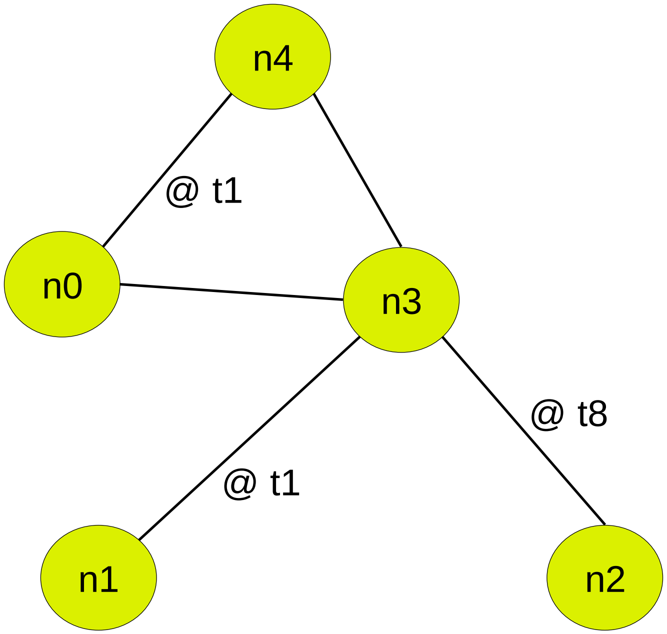
01

Temporal Graph Schema

Schema Goals

- Queries (lock-free)* parallelizable over time
- Implement on-top of existing DBs
- Maintain constant in-memory size
- Reduce time-filtering in batch queries
    - i.e. process only the necessary data

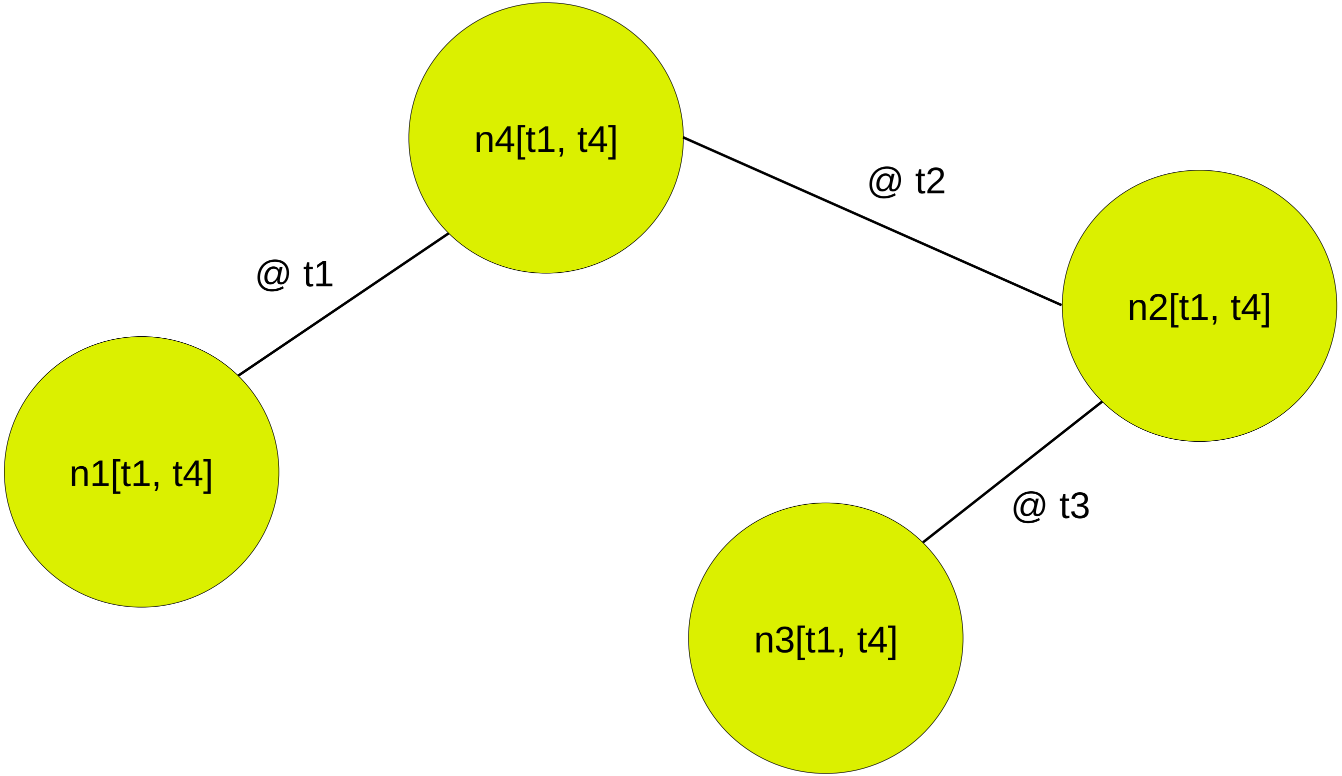
Strawman: Time as a (multi-)edge property



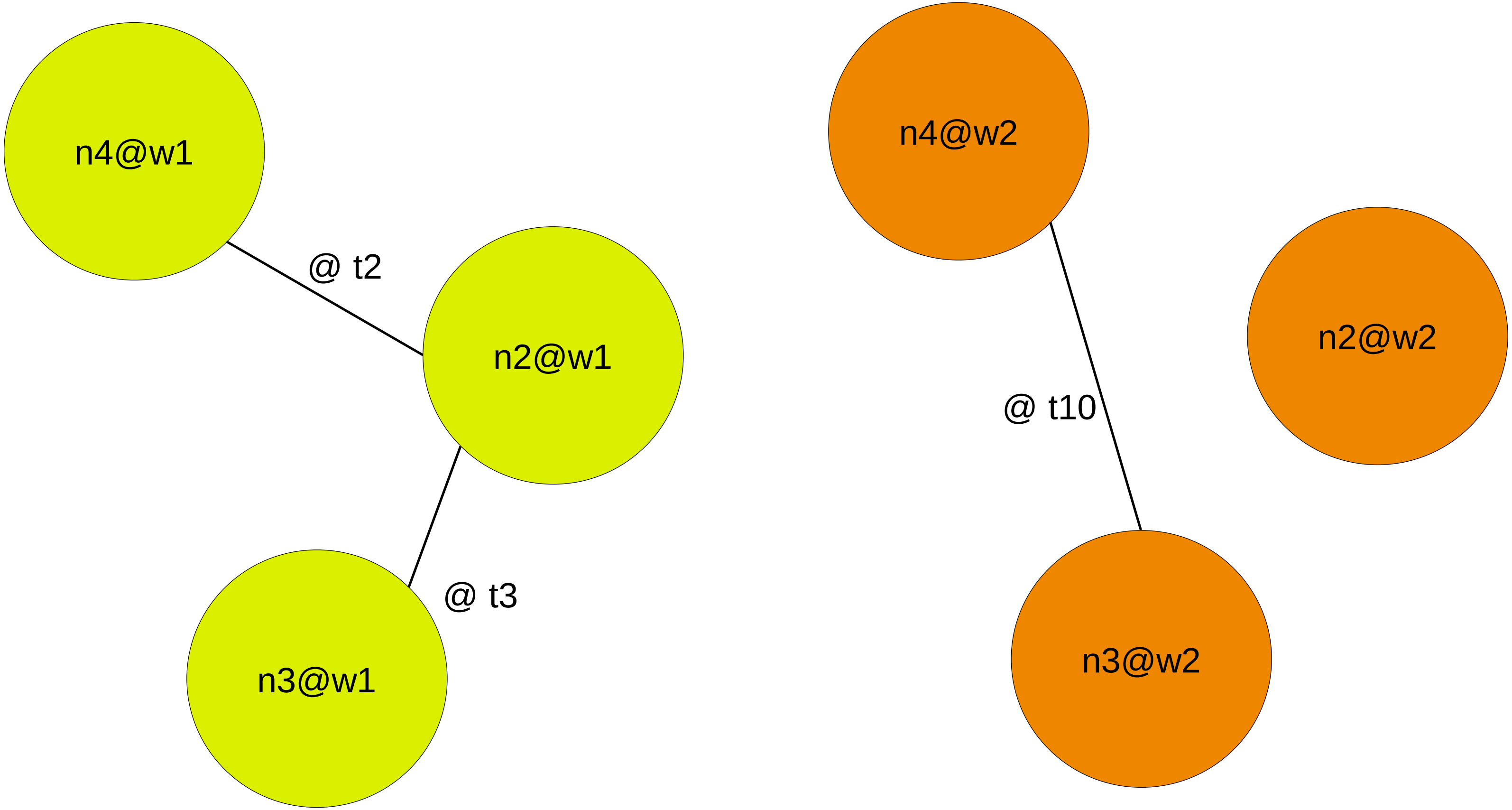
Strawman: Time as a (multi-)edge property

- Density edges/node increases with time
  - Limits the scalability for real-time and batch queries
- Limited concurrent access for reads and writes
- Makes time-constraints hard to implemented and scale

Timed Nodes: Time window part of a node's id

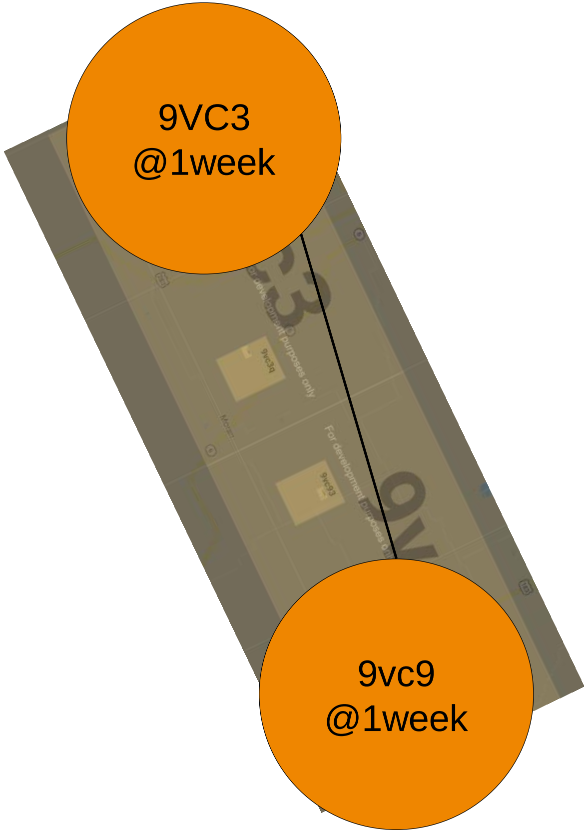


Timed Nodes: Time window part of a node’s id

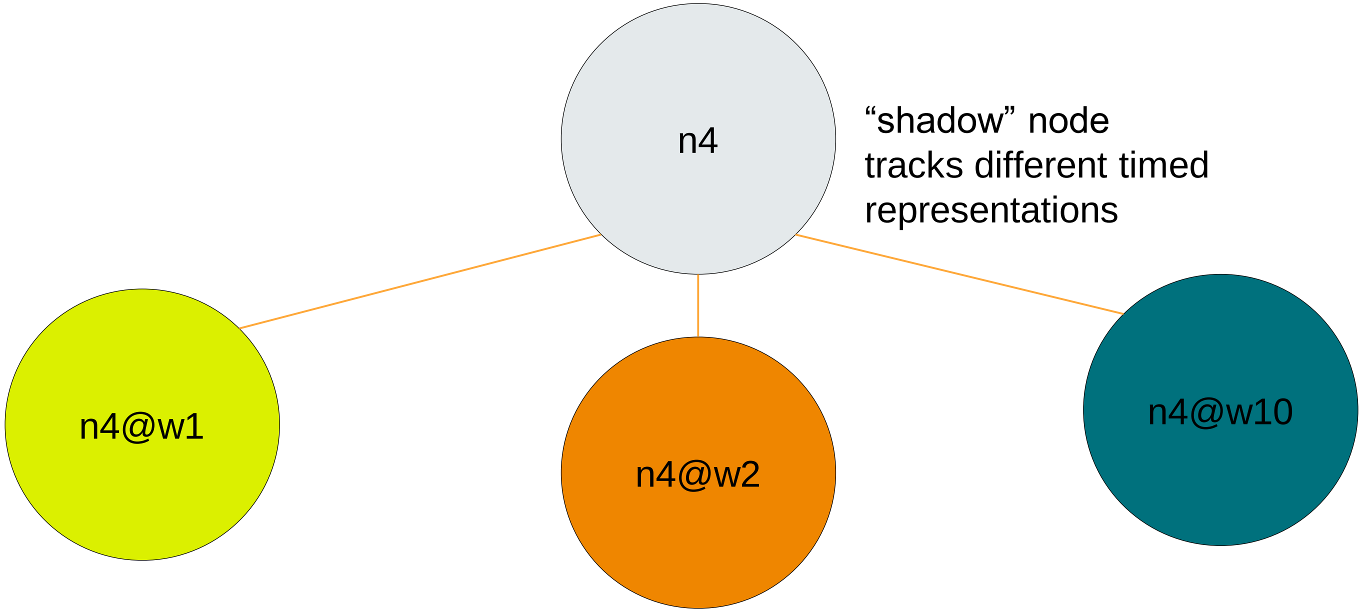


Windows need not be the same:

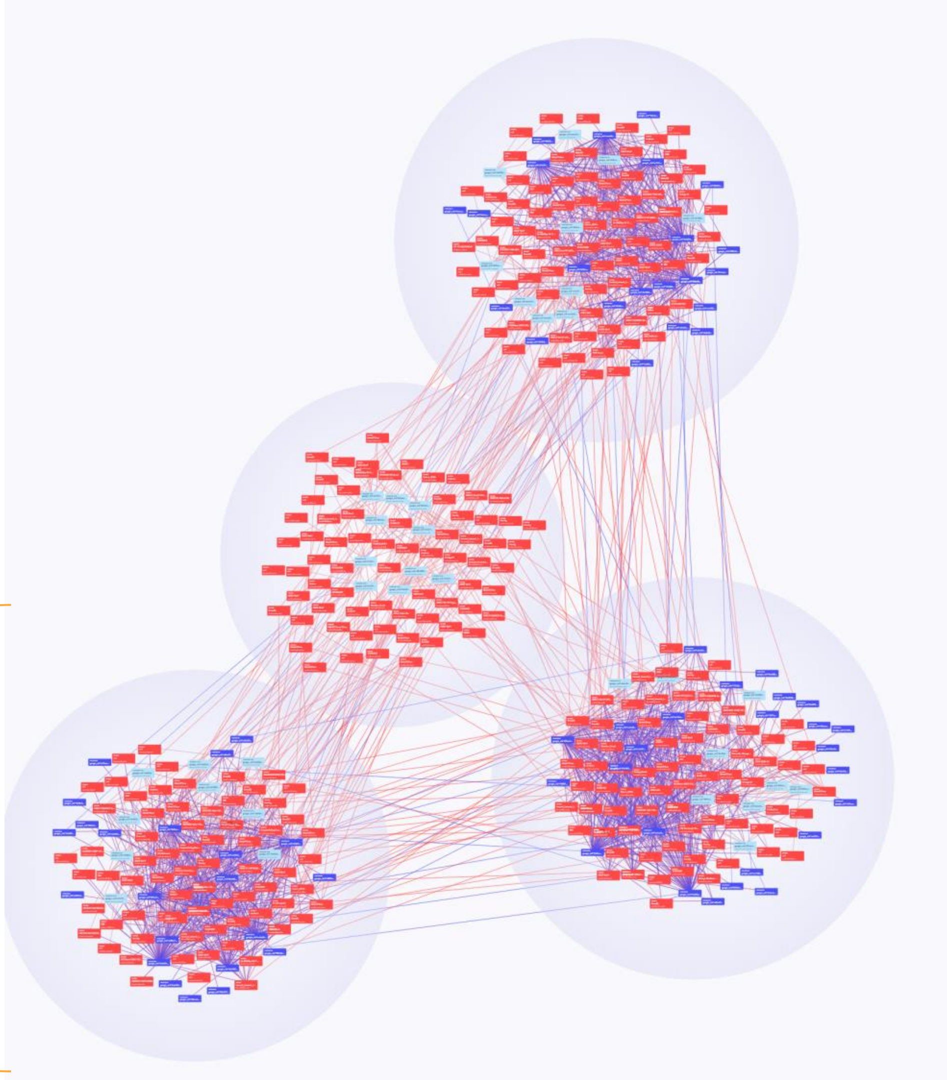
Geo-temporal analysis



Timed Nodes: Time window part of a node's id



Note: suited for vertex-centric (adjacency list) storage

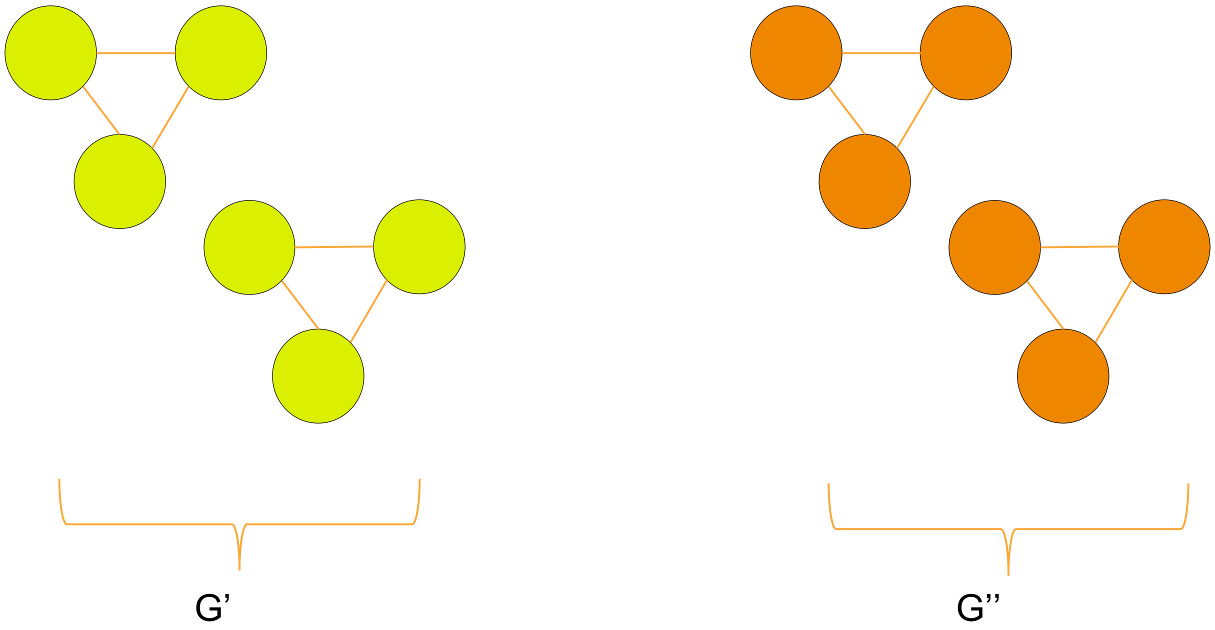


February 3-10

February 11-20

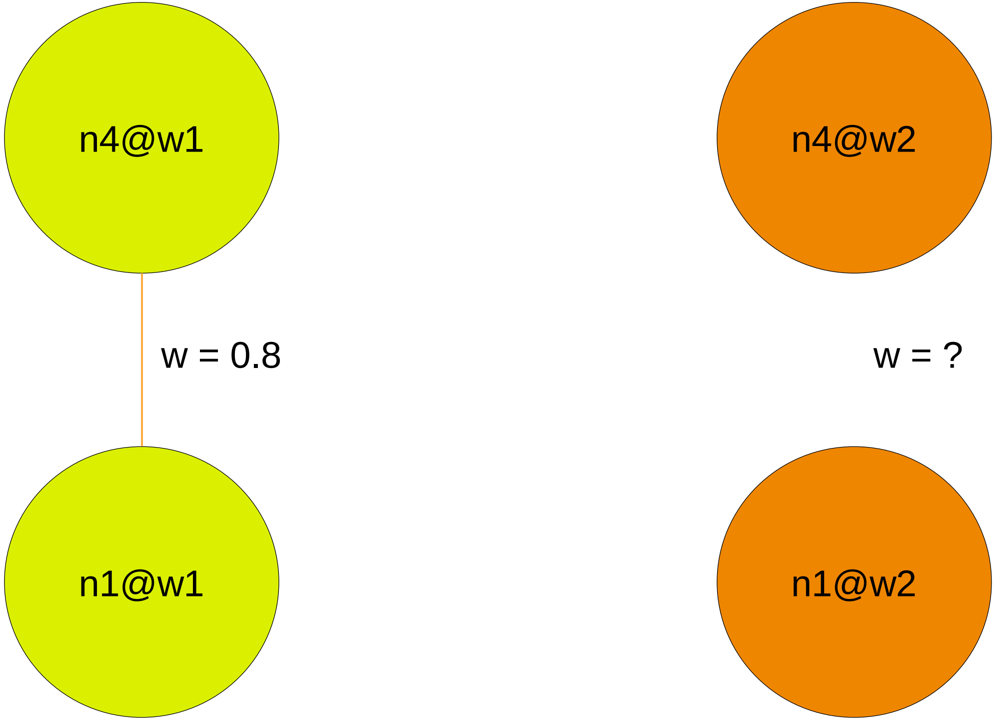
Timed Graphs:

Group windows into graphs for batch processing



Timed Nodes: Open-World Assumption

Are two nodes connected?



Timed Nodes: Open-World Assumption

- Assume Closed-World
  - Set w = 0 if edge does not exist in past *N* windows
- Create snapshots of aggregated past windows
  - Propagate aggregated edges as a new edge
  - Can be done in a lazy (amortized) fashion

Related Work on Modelling/Processing Temporal Graphs

- Chronos: A Graph Engine for Temporal Graph Analysis
  - [Han et al 2014]
- **GraphOne: A Data Store for Real-time Analytics on Evolving Graphs**
  - **[Kumar et al 2019]**
- GraphTau: Time-Evolving Graph Processing at Scale
  - [Iyer et al 2016]
- Kineograph: Taking the Pulse of a Fast-Changing and Connected World
  - [Cheng et al 2012]
- A Foundation of Lazy Streaming Graphs
  - [Dexter et al 2019]
- KickStarter: Fast and Accurate Computations on Streaming Graphs via Trimmed Approximations
  - [Vora et al 2017]

Timed Nodes Schema: Summary

- Nodes sharded across time windows.
- Length of windows can be learnt from the stream.

- Pro: Can be implemented on top of existing Graph/KV DBs
- Pro: Well suited for concurrent reads/writes
- Pro: Reduces density edges/nodes
- Pro: Easy to drop past data and have a constant in-mem size

- Con: Requires an additional query layer
- Con: Requires dealing with Open-World and snapshots

02

Temporal Embeddings

Embedding Goals

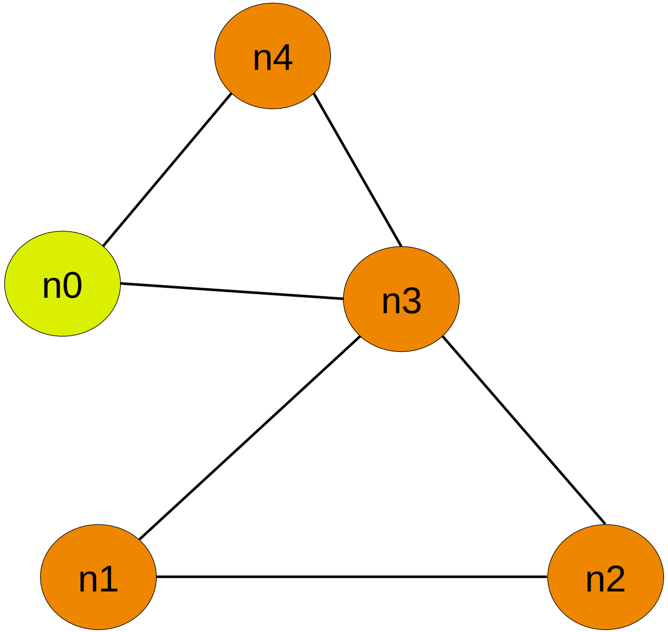
- Expose changes in a node’s behaviour over arbitrary time windows.
- Account for different levels of activity across time.
- Deal with infinite node sets.
    - or at least billions of nodes

(Static) Embeddings

- $f_{embed} : Node \to R^d$
  - (Ideally, d << number of nodes)

- Two main approaches:
  - Laplacian Eigenvectors
  - Random-walk skip-gram models

Random-walk skip-gram



Multiple walks per node

e.g.

Walk = [n4,n3,n2,n1]

Skip-window-1 = [

(n4,n3)

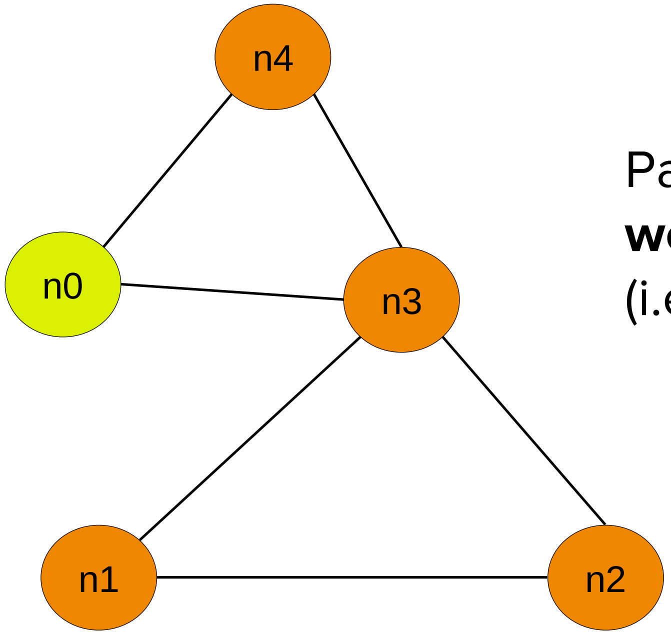
(n3,n4)

(n3,n2)

(n1,n2)

]

Random-walk skip-gram



Pairs can be fed into an encoder, ala **word2vec**, to produce the embeddings (i.e. the net's inner layer).

Strawman Temporal Embedding 1

- Train on random walks across all time windows to produce one embedding per node.

- Does not model change over time.
- Does not differentiate between different levels of activity over time.

Strawman Temporal Embedding 2

- Apply skip-gram model to each timed node
  - Add regularization to “shadow” (non-temporal) nodes
  - Use strawman-1 embeddings as priors

- Still need to deal with:
  - “infinite” (streaming) graphs?
  - no activity?
  - different levels of activity?

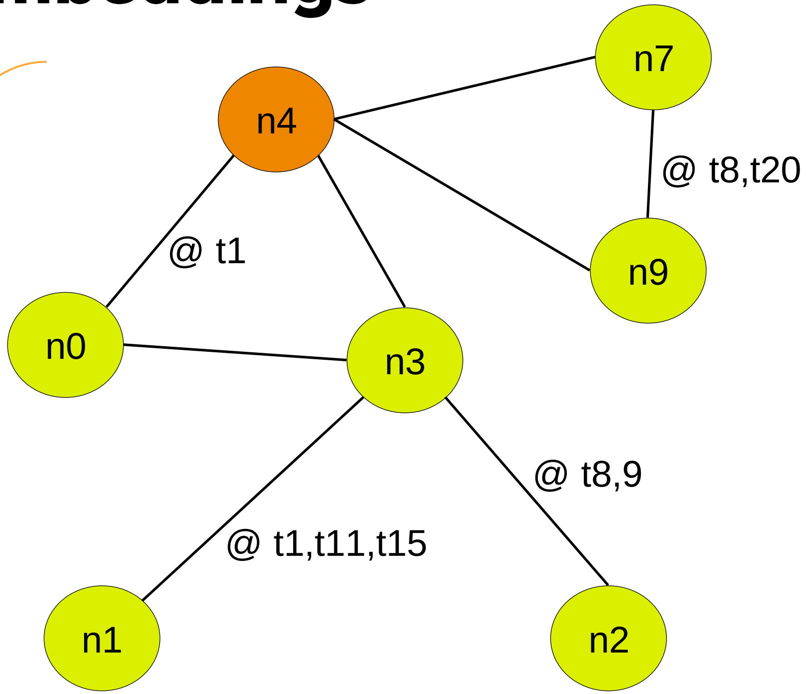
SignalFrame’s Quasi-Embeddings

1. Build random-walks per node per sliding windows
2. **Aggregate random-walks from connected components into a sparse vector**
    - **NLP/IR: Each vector is a document with nodes as dimensions**.
3. Collect all sparse vectors per connected components per sliding windows

SignalFrame’s Quasi-Embeddings

[t0, t_n]
[t0+step, t_n+step]
[t0+2*step, t_n+2*step]

Build embeddings for [t0, t_m] with some step.
Step and size hyper-params can result in a smoother-transition between embeddings.

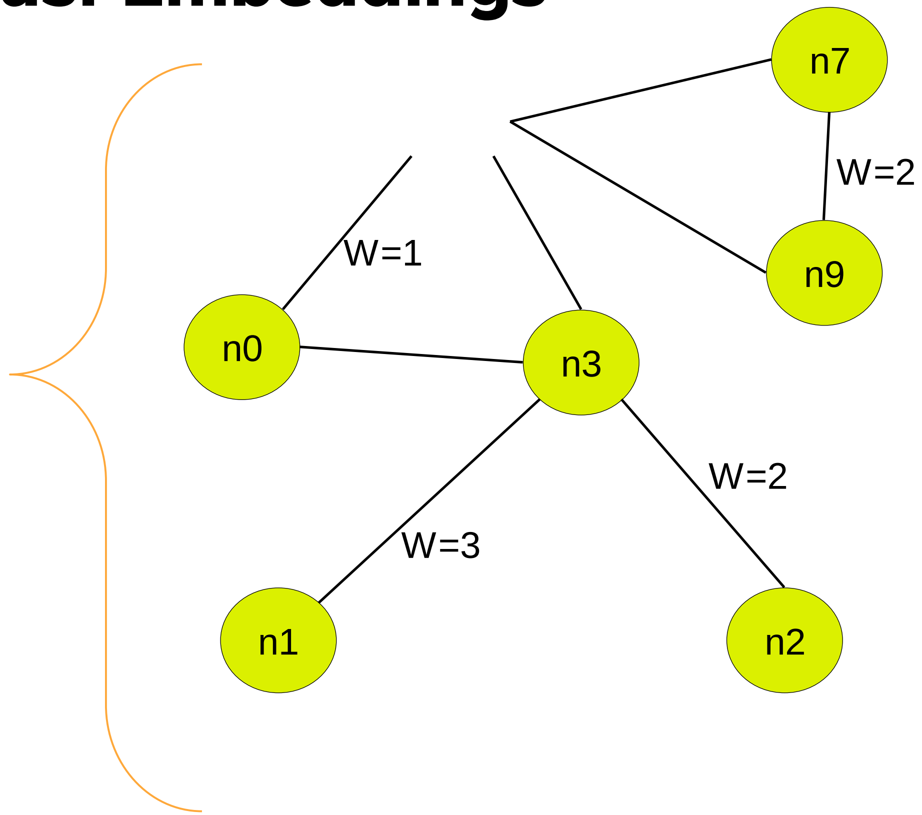


SignalFrame's Quasi-Embeddings

[t0, t_n]

[t0+step, t_n+step]

[t0+2*step, t_n+2*step]



SignalFrame's Quasi-Embeddings

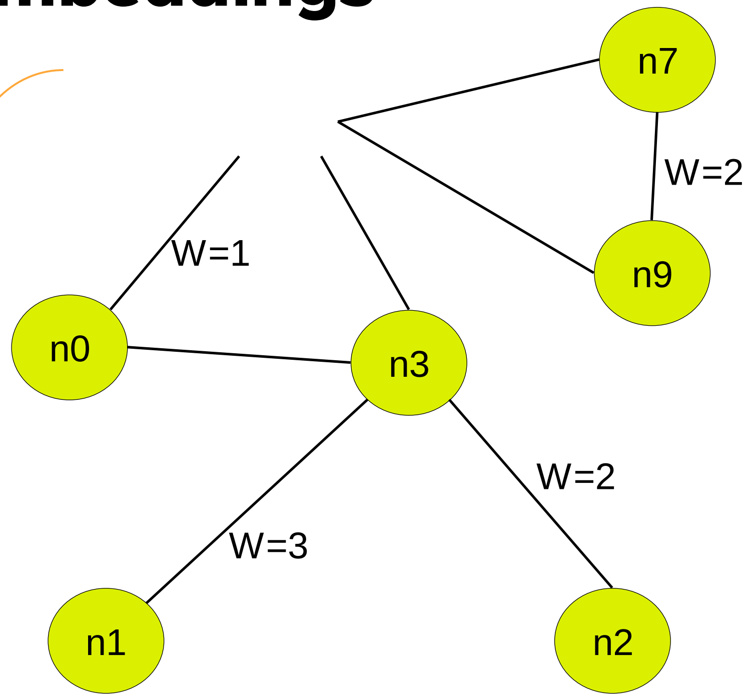
Generate weighted random walks per connected component.
Starting at the hidden node (in this case n4)

(n0, n3, n1)
(n0, n3, n2)
…

∑ "document" vector

(n7, n9)
(n7, n9)
…

∑ "document" vector

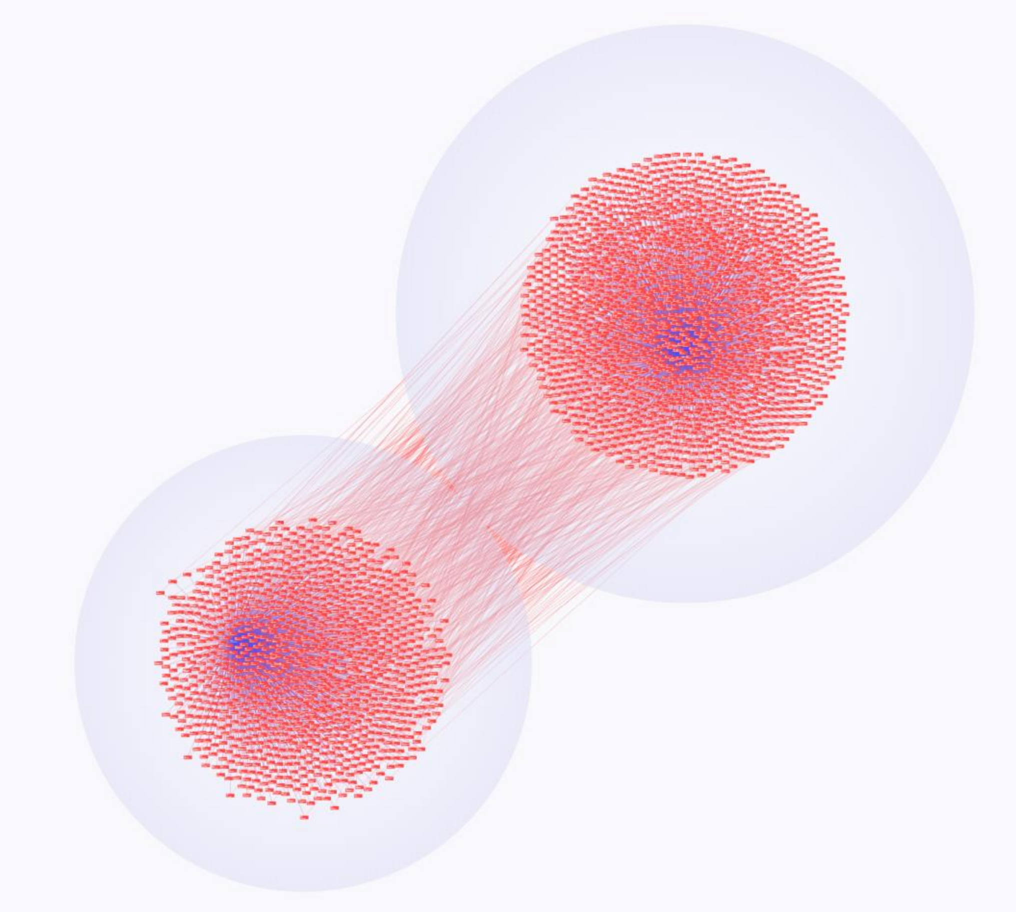


SignalFrame's 2nd Factor Authentication

A bubble is a time window of 14 days, with a 3-day overlap.

A bubble represents all 1-hop neighbours of a device that we want to authenticate.

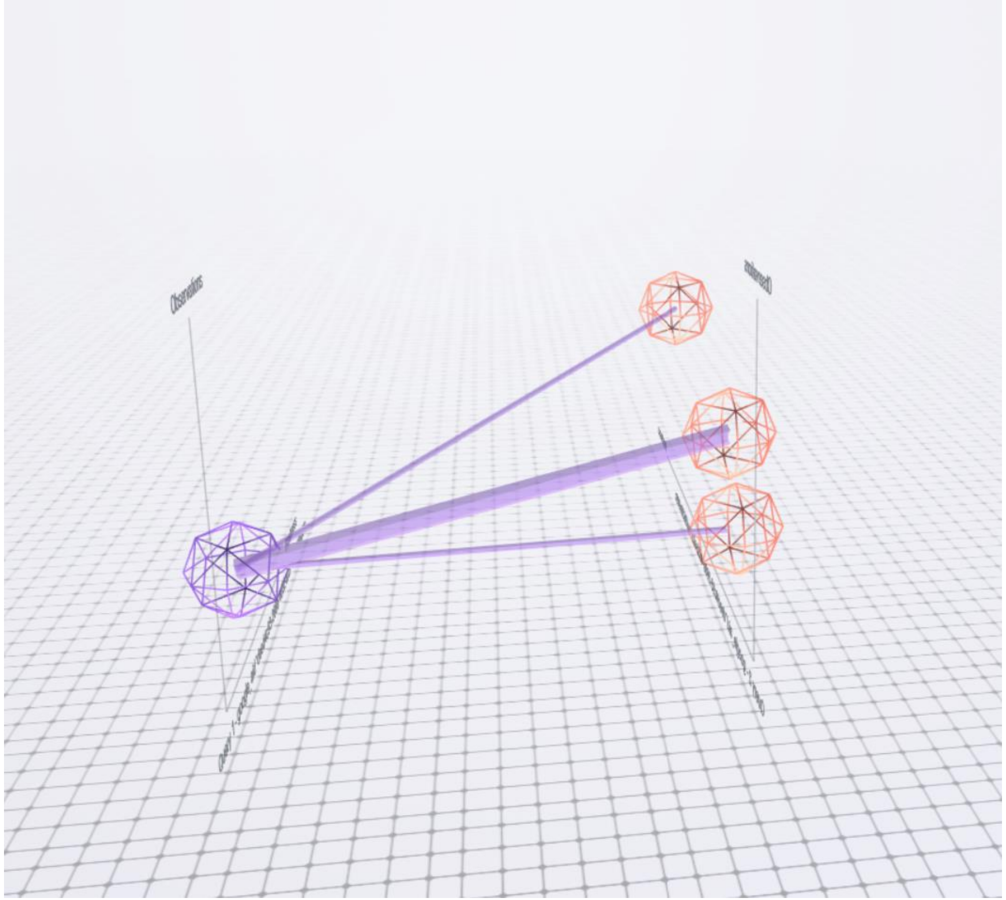
Has the behaviour changed?



SignalFrame's 2nd Factor Authentication

Reduction to temporal embeddings.

Has the behaviour changed?



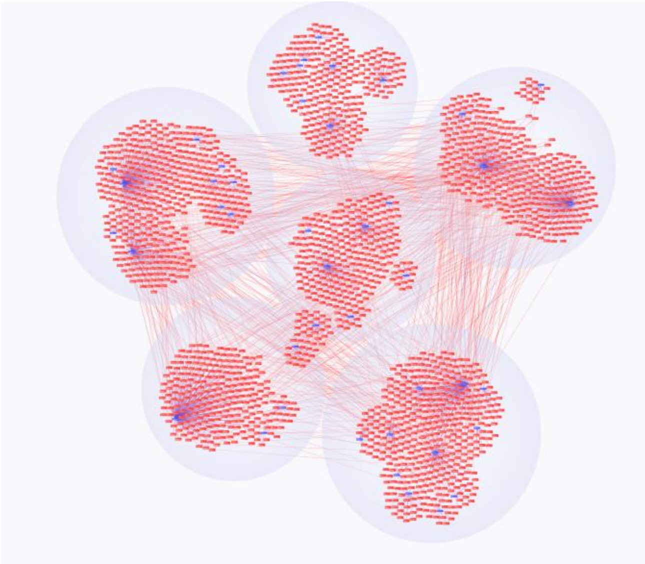
SignalFrame's 2nd Factor Authentication

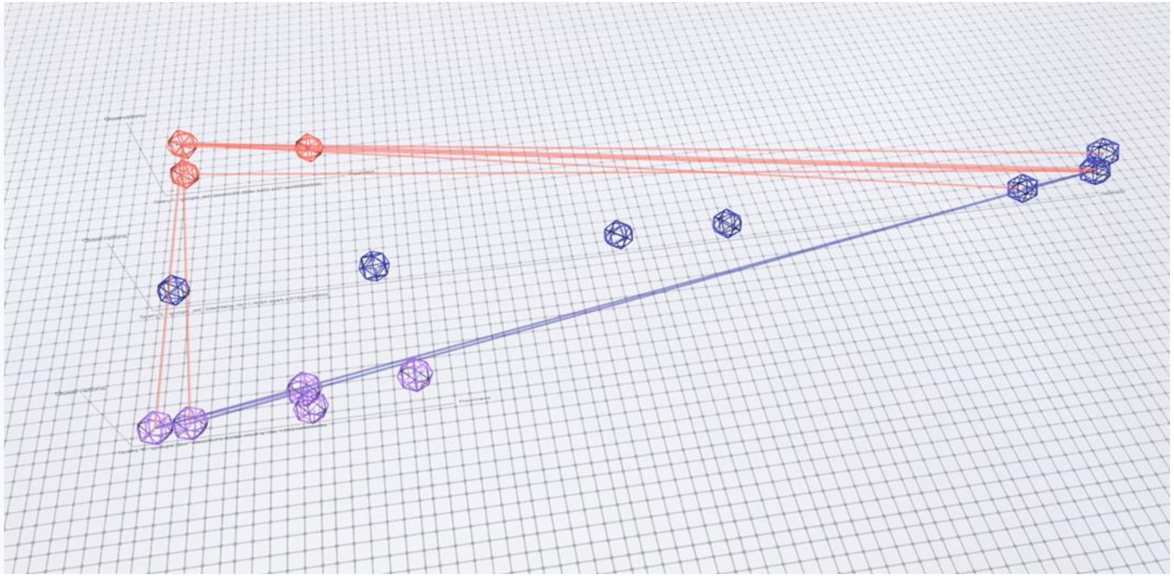
Embedding is a "signal" document.

All other signals are noise.

Similarity between sets of temporal embeddings

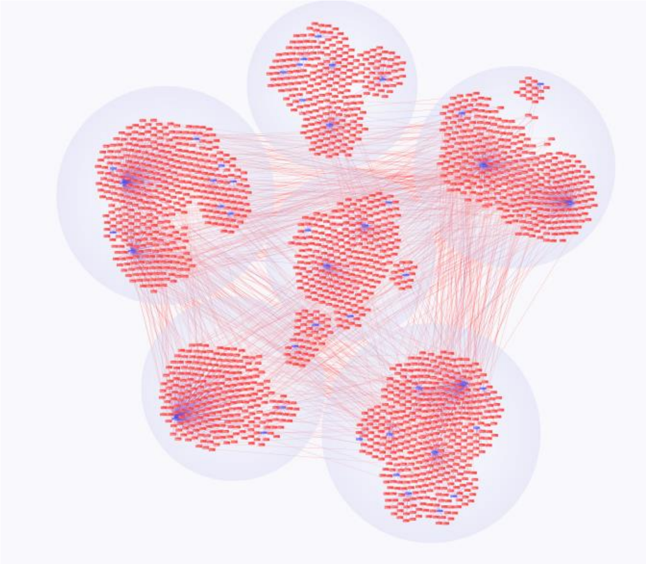
- Still need to address:
  - Different amount of evidence for the activity during a time window
  - Different set sizes, i.e. presence and absence of activity

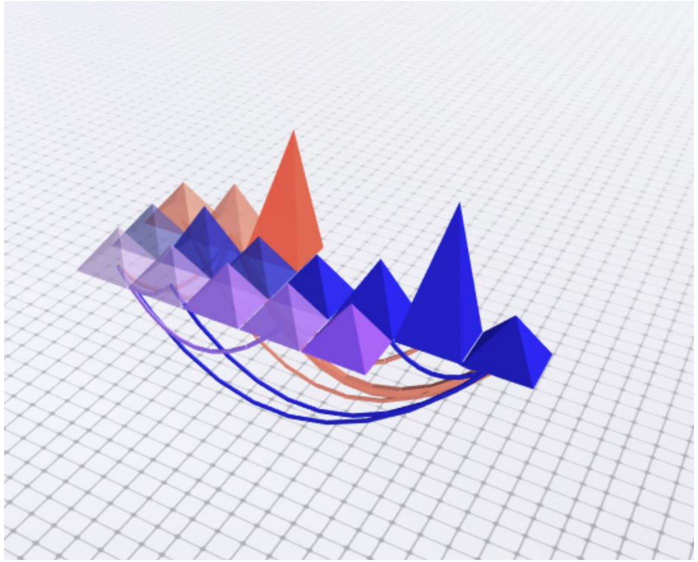




3 devices; 2 temporal communities per device

Embeddings per device (derived from sliding over temporal communities)





3 devices; 2 temporal communities per device

Embeddings per device (derived from sliding over temporal communities)

Similarity between sets of temporal embeddings

- $f : 2^{Embedding}\ X\ 2^{Embedding} \rightarrow R$
- Input:
  - M(n,m) – pairwise cosine between sets A, B
  - weights_a – weights associated to members of A
  - weights_b – weights associated to members of B

Sketch

1. W, where w(i,j) = **min**(weights_a(i), weights_b(j))
2. S = M ∘ W // *Hadamard product*
3. *score* = **max**($\sum_i^n$ **max**(S(i,.)), $\sum_j^m$ **max**(S(.,j)))
4. *decay* = **max**(**0norm**(Max_i^n M(i,.)), **0norm**(Max_j^m M(.,j)))
    1. 0norm(vector) := (len_non_zero(vector) + 1)/(len(vector) +1)
5. **return** *score * decay*

Related Work on Graph Embeddings

- Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering
    - [Belkin et al 2000]
- DeepWalk: Online Learning of Social Representations
    - [Perozzi et al 2014]
- node2vec: Scalable Feature Learning for Networks
    - [Grover et al 2016]
- struc2vec: LearningNodeRepresentationsfromStructural Identity
    - [Ribeiro et al 2017]
- **Is a Single Embedding Enough? Learning Node Representations that Capture Multiple Social Contexts**
    - **[Epasto et al 2019]**

Temporal (Quasi-)Embeddings Summary

- Pro: Can be done in pseudo real-time for some use-cases
- Pro: Explicit similarity model for sets of embeddings
- Pro: Process new nodes in a streaming mode

- Con: Dimensions are not reduced
- Con: No explicit cost function

Future Work

- Focus on structural embeddings (ala struct2vec) but with infinite inputs
    - Use quasi-embeddings as “syntactic” embedding
- Explore how/if Graph NNs can be used for structural analysis

signalframe

Thanks.