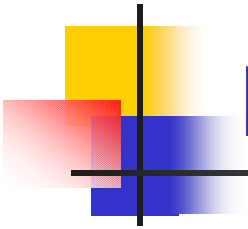
Towards Fast Decentralized Construction of Locality-Aware Overlay Networks

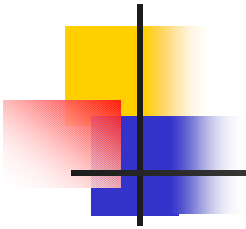
Alex Slivkins

Microsoft Research
Silicon Valley Center



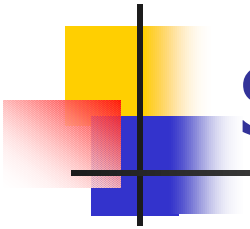
Locality-awareness

- idea: keep communication as local as possible
  - tasks complete faster, consume less resources
- examples:
  - download the nearest file replica
  - on DHT lookup, avoid traversing many long overlay links
  - multicast reaches a cluster of nearby nodes via one long link, then spreads locally
- concrete notion of locality: latencies (round-trip times)
  - almost a metric
  - relatively easy to measure in practice



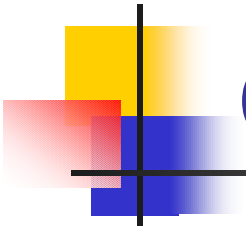
Locality-aware primitives

- focus on three locality-aware (low-stretch) primitives
  - name-independent routing schemes:
    look at routing table, choose next hop
    stretch ($u \rightarrow v$ path) = length(path) / d(u,v)
  - distance labels: decode distances from short node labels
    assuming $d(u,v) \leq$ estimate(u,v),
    stretch (u,v) = estimate(u,v) / d(u,v)
  - multicast trees: low-degree and low-depth spanning trees
    stretch (node) = stretch (root -> node path)



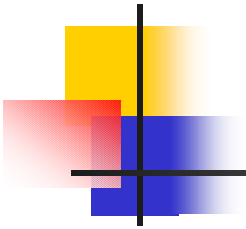
Scalable constructions

- prior work on routing schemes and distance labels:
  - focuses on stretch vs storage trade-off
  - does not (really) address scalable distributed constructions
- existing "polynomial-time" constructions do not scale
  - polynomial time: too long
  - full info: infeasible to measure all distances
  - centralized: infeasible to gather all data at one node
- our goal: scalable distributed constructions for large networks
  - "scalable" = running time poly-log in #nodes



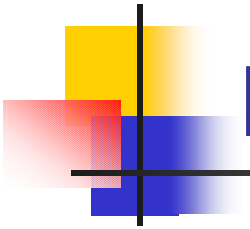
Our results

- Scalable distributed constructions for low-stretch primitives
  - distance labeling schemes, multicast trees, name-independent routing schemes
  - also: k-closest node discovery, estimating #nodes in B(u,r)
- Common framework
  - layers of common functionality
- provable guarantees for growth-constrained metrics
  - running times $(\delta^{-1} \log n)^{O(1)}$ for stretch $1+\delta$
  - almost optimal storage-stretch trade-offs



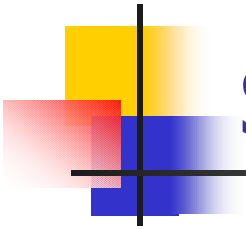
Plan

- ✓ Intro
- ❑ Setting
- ❑ Construction (sketch)



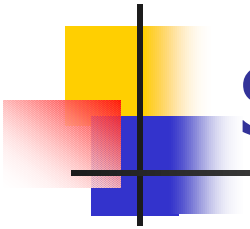
Peer-to-peer setting

- nodes communicate via Internet
  - any two nodes can communicate directly
  - sender must know receiver's ID ("ip-address")
  - black box: nodes have no knowledge of topology
- node IDs are arbitrary
  - reveal no info about the node, cannot be guessed
  - either given externally or passed between nodes
  - u has a link to v $\Leftrightarrow$ u stores node ID of v



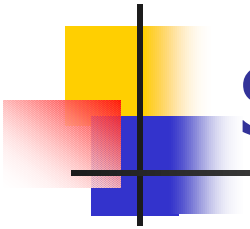
Setting: sending packets

- unit-size packets
  - unit time to send, unit time to receive
- for running times, assume each packet takes unit time en route
  - meaningful results: poly-log(n) $\times$ max RTT
  - accounting for latencies would not help much:
    must send many packets over long distances
- packet transmission induces load only on sender and receiver
  - abstracts away the underlying Internet



Setting: initialization

- start with non-locality-aware network and make it locality-aware
  - initially each node has some links (stores some node IDs)
  - assume low-degree, high expansion
- how to construct the initial overlay?
  - addressed in prior work, beyond our scope

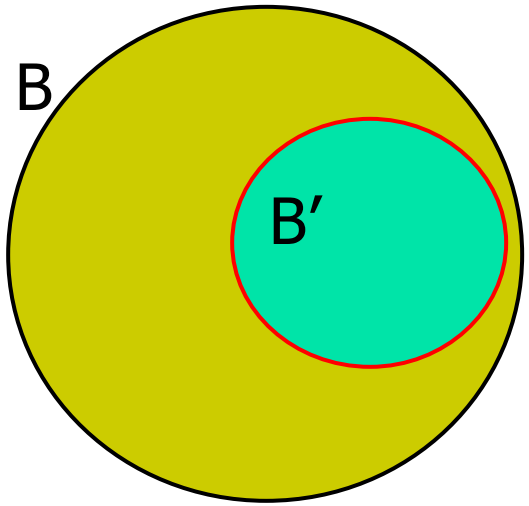


Setting: latencies

- latencies form a metric
  - ignore triangle inequality violations
- "growth-constrained metrics"
  - def ball of twice the radius has $\leq 2^d$ times as many points
  - generalize grids in d-dimensional Euclidean space

Setting: latencies

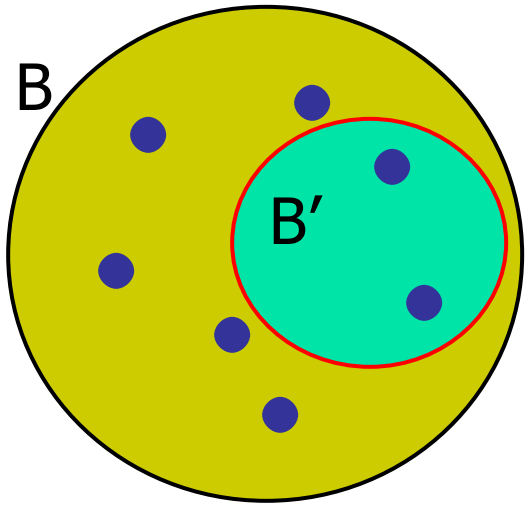
- latencies form a metric
  - ignore triangle inequality violations
- "growth-constrained metrics"
  - def ball of twice the radius has $\leq 2^d$ times as many points
  - generalize grids in d-dimensional Euclidean space



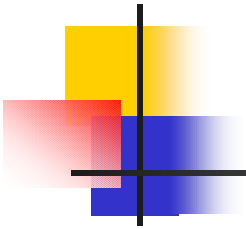
- if radius(B′) = radius(B)/2 then $|B'| \geq \Omega(|B|)$

Setting: latencies

- latencies form a metric
  - ignore triangle inequality violations
- "growth-constrained metrics"
  - def ball of twice the radius has $\leq 2^d$ times as many points
  - generalize grids in d-dimensional Euclidean space



- if radius(B′) = radius(B)/2 then $|B'| \geq \Omega(|B|)$
- $\Theta(\log n)$ random samples in B $\Rightarrow \geq 1$ lands in B′ w.h.p.

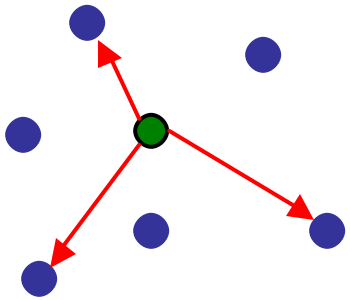


Plan

- ✓ Intro
- ✓ Setting
- ❑ Construction (sketch)
  - focus on common framework
  - example: distance labels

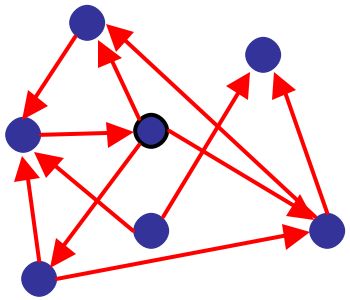
Network of peers

- each node stores distances to some other nodes: its peers
  - some peers can be far, some can be close



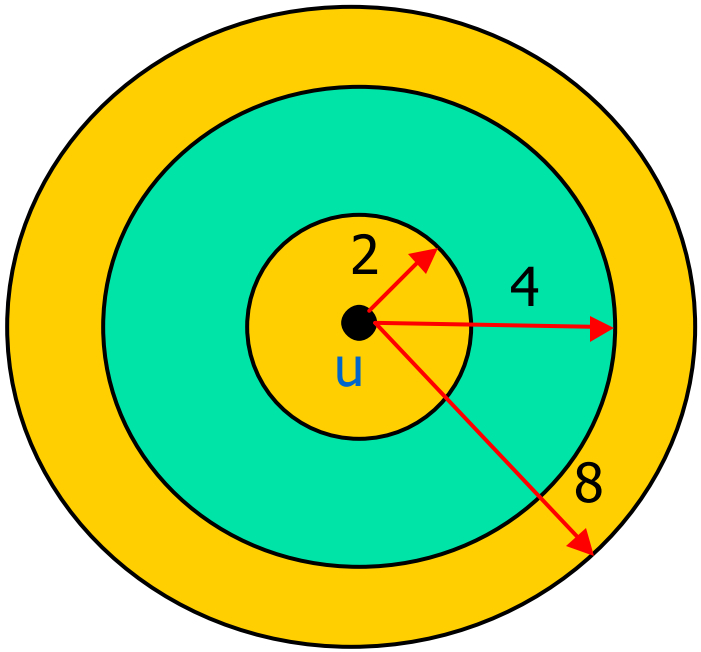
Network of peers

- each node stores distances to some other nodes: its peers
  - some peers can be far, some can be close
- peers induce a a directed graph: network of peers



Rings of peers

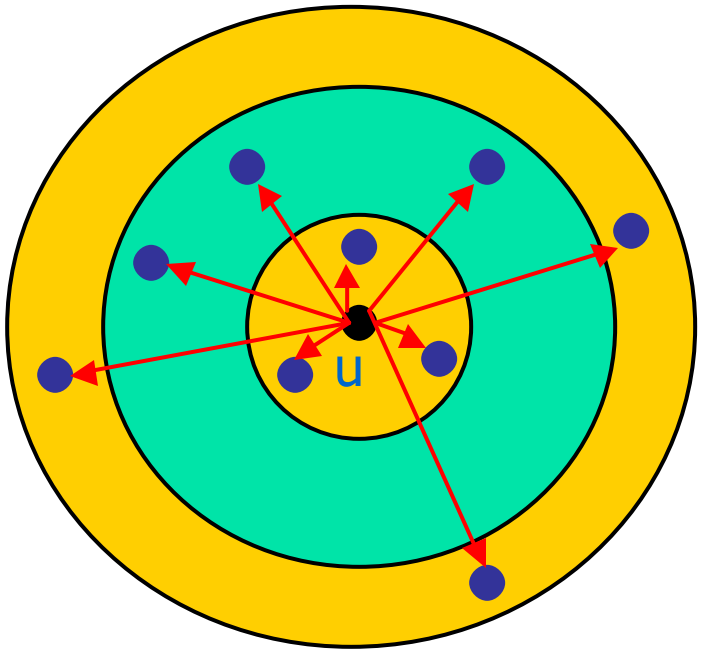
Peers are organized in rings of exponentially increasing radii



Rings of peers

Peers are organized in rings of exponentially increasing radii

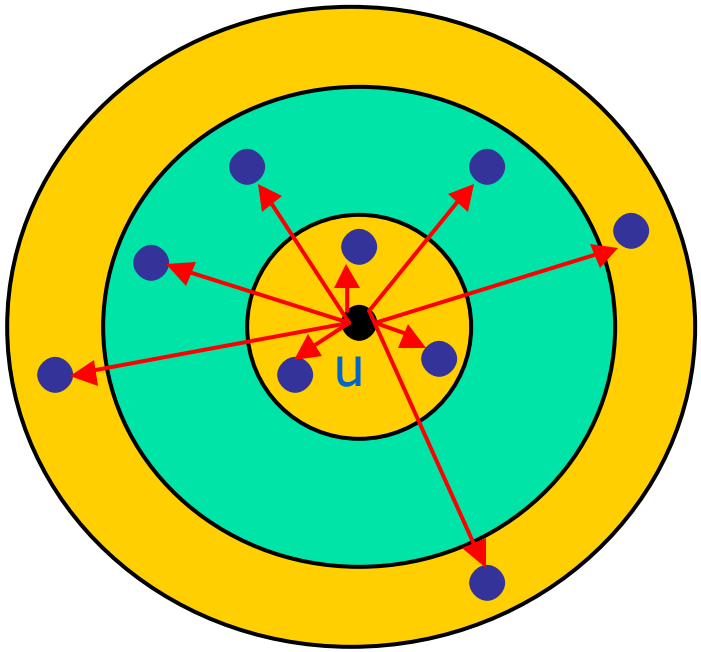
- ring #i: $\delta^{-O(1)}$ (log n) peers distributed near-uniformly in B(u, 2^i)
- peers are independent (for all nodes and all rings)



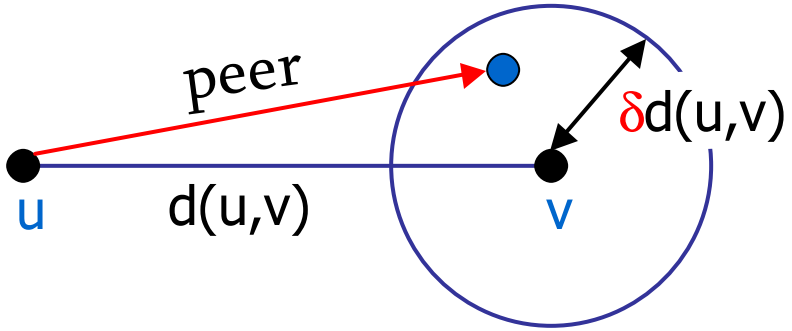
Rings of peers

Peers are organized in rings of exponentially increasing radii

- ring #i: $\delta^{-O(1)}$ (log n) peers distributed near-uniformly in B(u, 2^i)
- peers are independent (for all nodes and all rings)

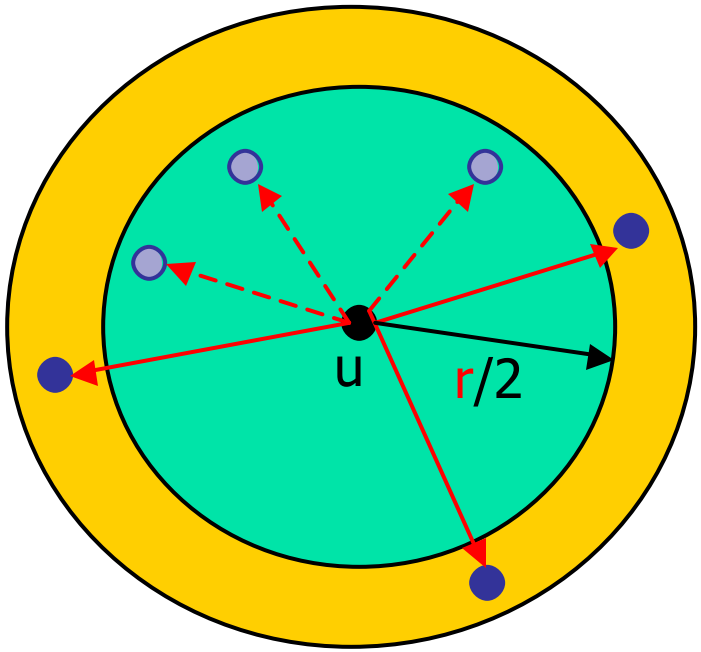


- Useful property: for any node v node u has a peer in B(v, δd(u,v))

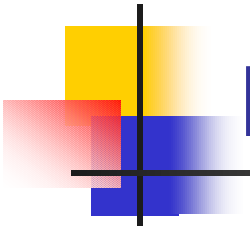


Rings of peers: construction

- Top-down construction
  - outer ring = random samples
  - use radius r peers of all nodes to construct radius r/2 peers



- how to sample from B(u, r/2)?
  - graph G(u, r) on B(u, r/2) edges are radius r peers
    - low-degree expander
  - each node u initiates random walk on G(u, r)
    - uniform in O(log n) steps

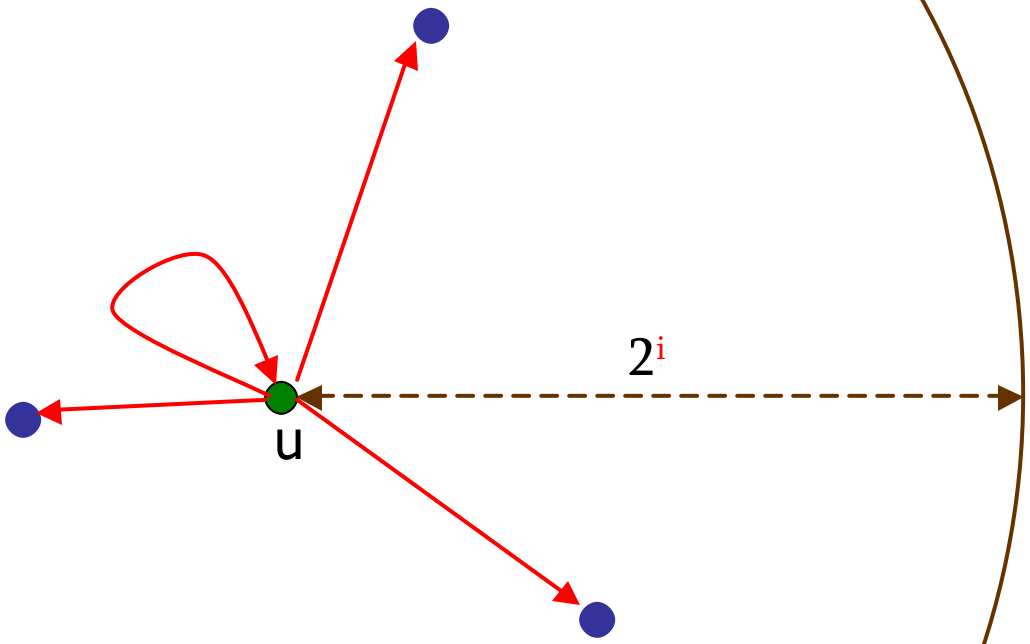


Beacon selection

- node is a beacon if it is its own peer

Beacon selection

- node is a level-i beacon if it is its own radius-2^i peer

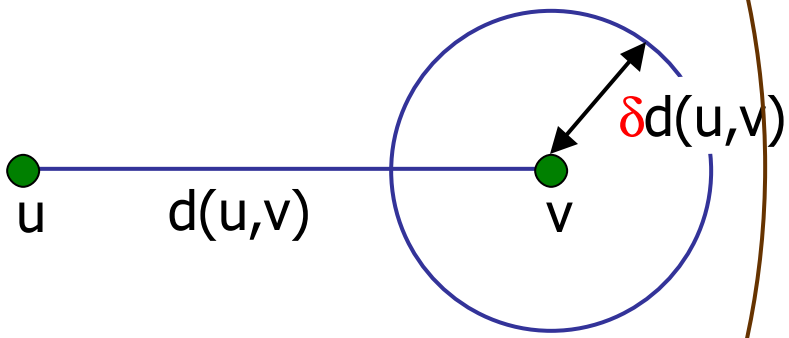


Beacon selection

- node is a level-i beacon if it is its own radius-2^i peer
- $\leq \delta^{-O(1)}$ (log n) level-i beacons in B(u, 2^i)

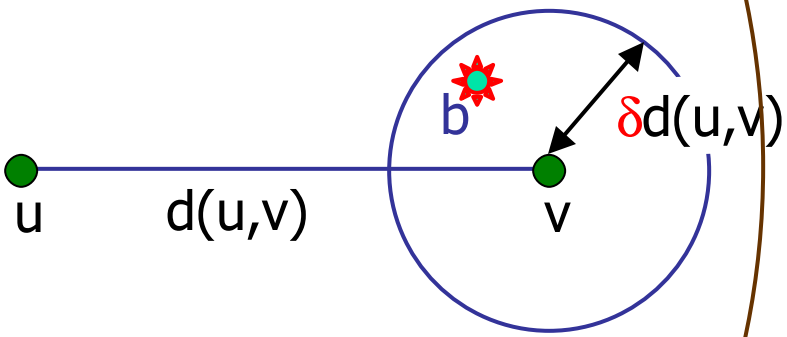
Beacon selection

- node is a level-i beacon if it is its own radius-2^i peer
- $\leq \delta^{-O(1)}$ (log n) level-i beacons in B(u, 2^i)
- for each v such that $2^{i-1} \leq (1+\delta)\, d(u,v) \leq 2^i$



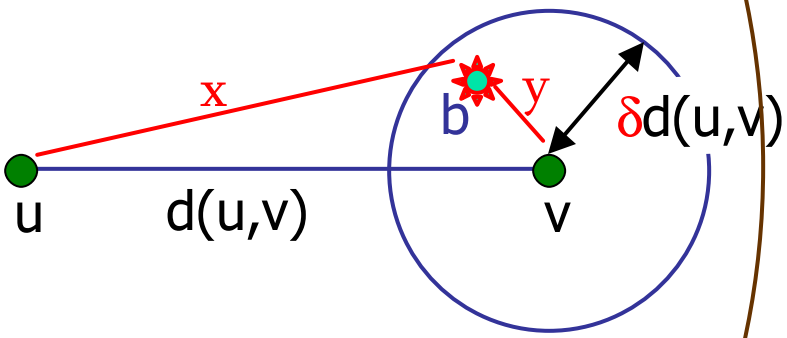
Beacon selection

- node is a level-i beacon if it is its own radius-2^i peer
- $\leq \delta^{-O(1)}$ (log n) level-i beacons in B(u, 2^i)
- for each v such that $2^{i-1} \leq (1+\delta)\, d(u,v) \leq 2^i$
  - some level-i beacon lies within $\delta d(u,v)$ from v

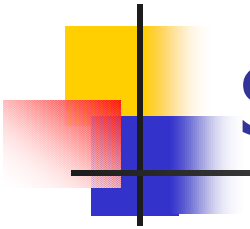


Beacon selection

- node is a level-i beacon if it is its own radius-2^i peer
- $\leq \delta^{-O(1)} (\log n)$ level-i beacons in $B(u, 2^i)$
- for each v such that $2^{i-1} \leq (1+\delta)\, d(u,v) \leq 2^i$
  - some level-i beacon lies within $\delta d(u,v)$ from v
  - $x - y \leq d(u,v) \leq x + y$



good distance estimates
PROBLEM:
each node must know
all level-i beacons
within radius 2^i
(for every i)

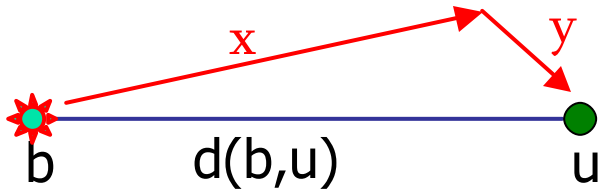


Special broadcast

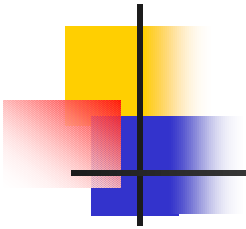
- Beacons announce themselves via a special broadcast that spreads via links between peers
- the broadcast from each level-i beacon:
  - reaches each node within radius 2^i
  - stays inside radius 4×2^i

Special broadcast

- Beacons announce themselves via a special broadcast that spreads via links between peers
- the broadcast from each level-i beacon:
  - reaches each node within radius 2^i
  - stays inside radius 4×2^i
  - uses "nice" paths $\Rightarrow$ good estimates on distance to beacon
    - $x - y \le d(b,u) \le x + y$



broadcasts from all beacons run concurrently, yet no node is overloaded; all complete in time $(\delta^{-1} \log n)^{O(1)}$

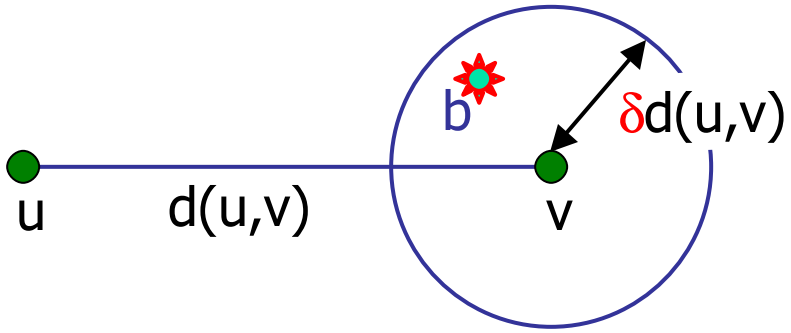


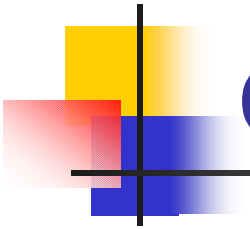
Putting it all together

- rings of peers
- beacon selection
- beacons announce themselves via broadcast
- distance label = {all relevant beacons}

Putting it all together

- rings of peers
- beacon selection
- beacons announce themselves via broadcast
- distance label = {all relevant beacons}
- take two distance labels:
  some beacon gives a good distance estimate





Conclusions

- Scalable distributed constructions of location-aware primitives
  - setting: complete communication, growth-constrained metrics
  - notion of scalability: poly-log in #nodes
  - type of guarantees: (uniformly) low stretch
  - applications: distance labeling, routing schemes, multicast
  - common multi-layer framework: rings of peers, beacon
- High-level contribution: bringing these ingredients together
  - allows for a clean mathematical treatment