Tiara
A Self-Stabilizing Deterministic Skip List*

Thomas Clouser
Mikhail Nesterenko
Christian Scheideler

* most results appeared in the proceedings of SSS'08

Overlay Networks and Stabilization

why stabilization is good for overlay networks

- peer-to-peer system organized as an overlay network is an effective way to distribute information at scale
- millions of users constantly leave and join the network (churn)
  - faults and inconsistencies are the norm
  - esoteric faulty states may be reached
  - large scale precludes centralized fault tolerance and initialization

why overlay networks are good for stabilization

- practical application
- network topology is under system control – intriguing algorithmic solutions

Outline

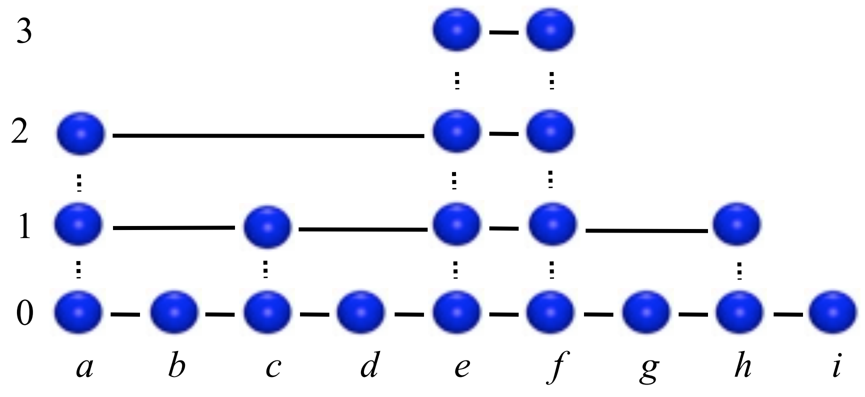
- overlay networks and programming model
- Tiara
  - bottom level
  - skip-list
  - searches, topology updates
- related literature
- extensions and future work

Outline

- overlay networks and programming model
- Tiara
  - bottom level
  - skip-list
  - searches, topology updates
- related literature
- extensions and future work

Overlay Network Terminology

- in overlay network any pair of peers can establish a link
- topology
  - unstructured – low maintenance, high search cost
  - structured – predictable performance, fast searches higher maintenance



- structure formation
  - randomized – usually simpler, may have better average case performance
  - deterministic – precise bounds
- structures optimize search and update costs – for best existing structures, it is logarithmic
- skip-list - leveled structure – enables logarithmic searches & updates
  - 0th is a sorted list of peers
  - only a fraction of the nodes is promoted to each subsequent level
- our contribution – a deterministic self-stabilizing skip-list

Execution Model

- unique ordered ids
- undirected links
- shared registers
- interleaving execution semantics
- high atomicity (can read neighbor's state and update its own)
- graph initially connected
  - stabilization presumes connectivity preservation

- notation
  - left process – lower id
  - right process – higher id
  - neighbor – process sharing an edge

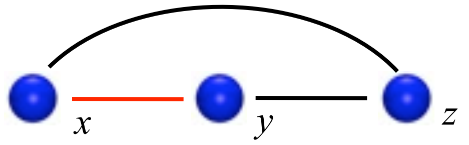
Outline

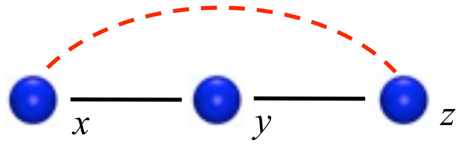
- overlay networks and programming model
- Tiara
  - bottom level
  - skip-list
  - searches, topology updates
- related literature
- extensions and future work

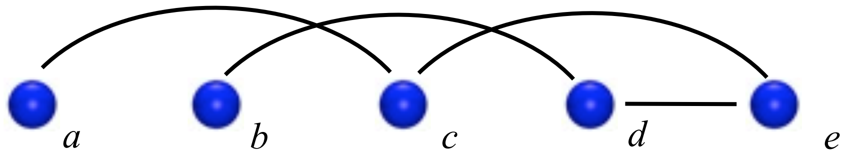
b-Tiara

linearizes a list of peers at the bottom level
actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





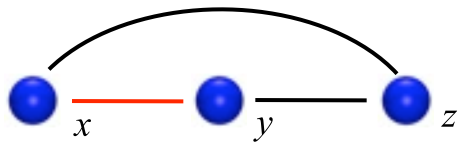


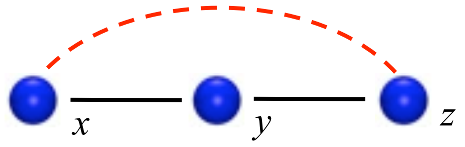
b-Tiara

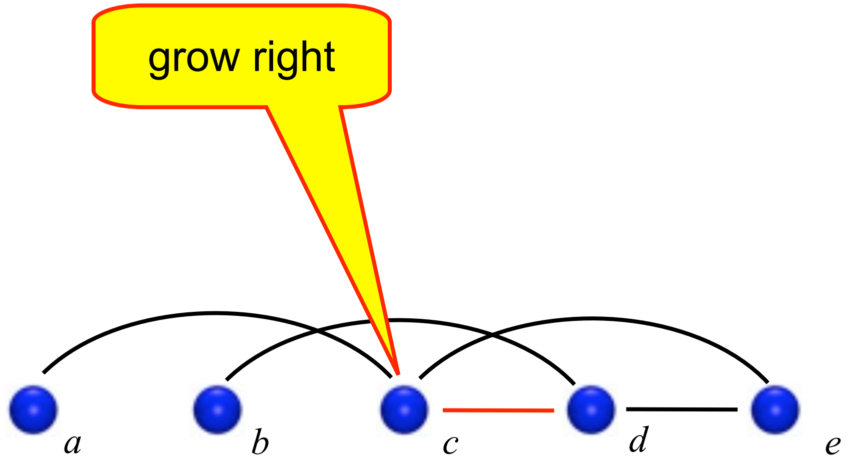
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar



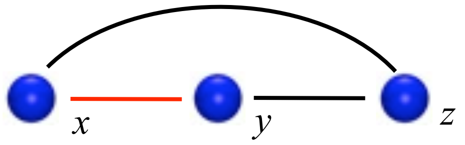


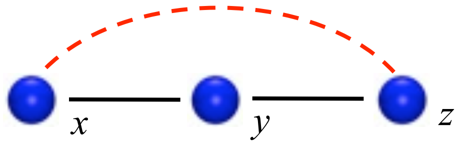


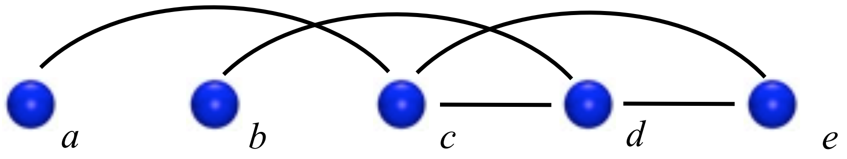
b-Tiara

linearizes a list of peers at the bottom level
actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





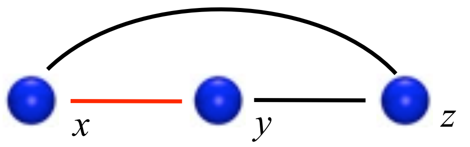


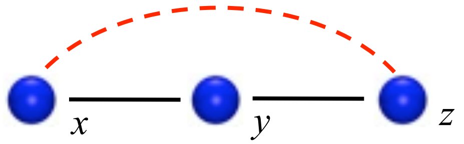
b-Tiara

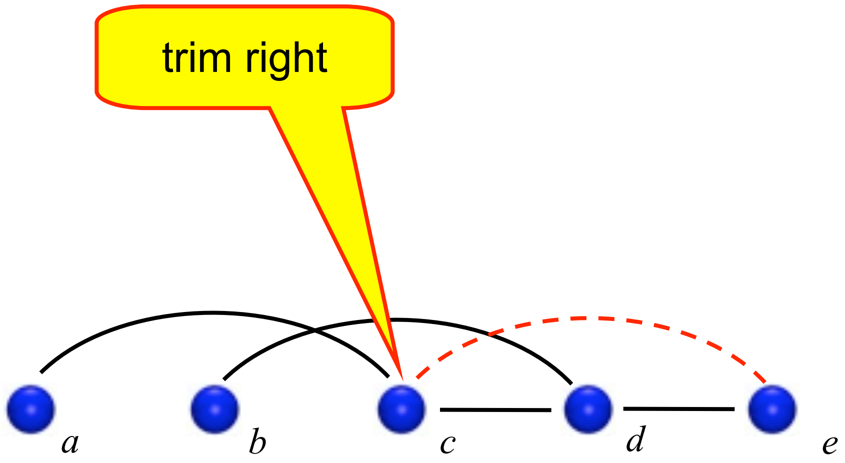
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





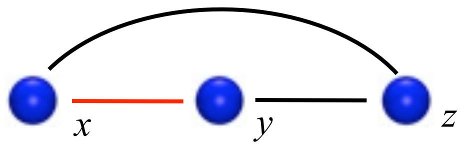


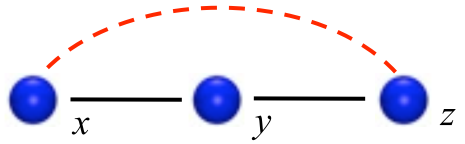
b-Tiara

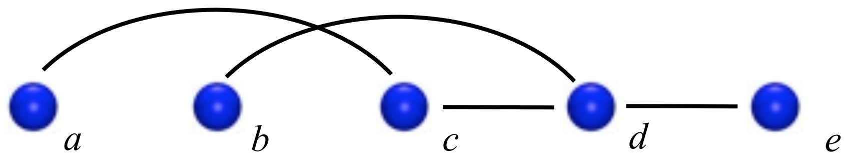
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





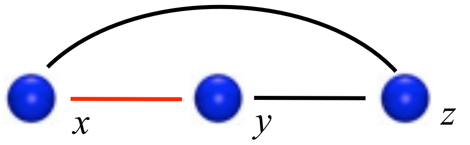


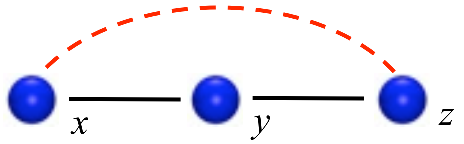
b-Tiara

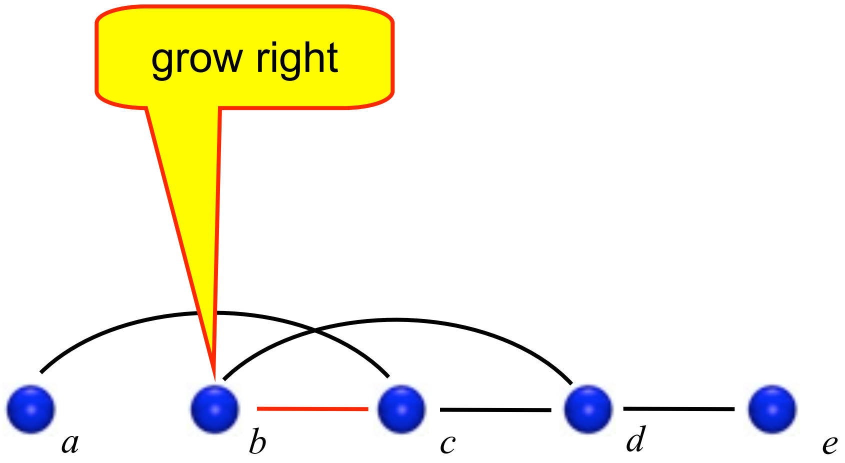
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





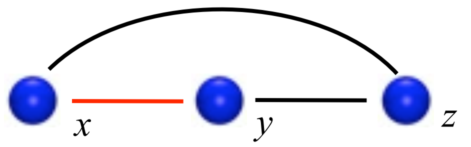


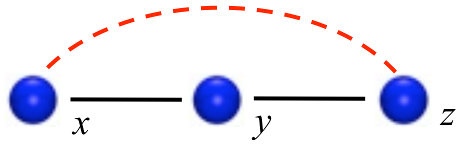
b-Tiara

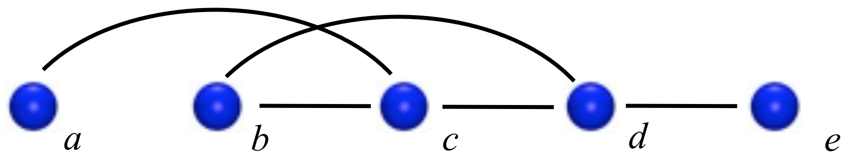
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





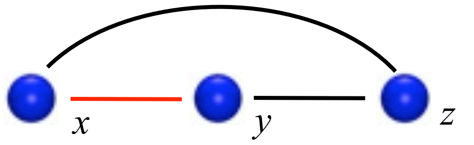


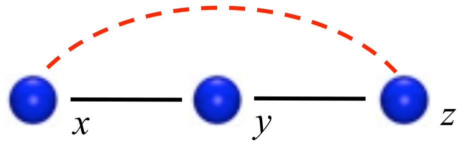
b-Tiara

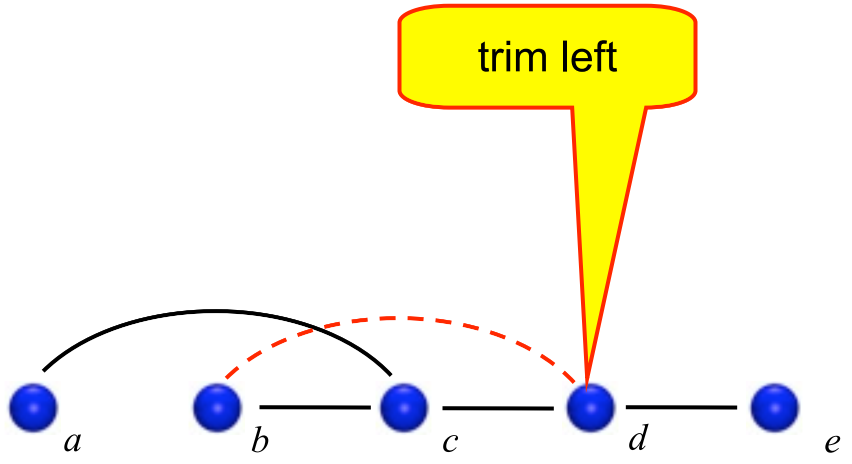
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





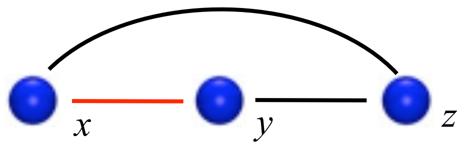


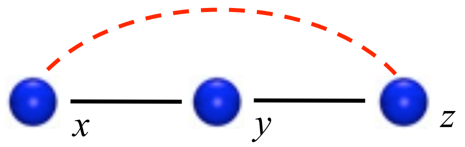
b-Tiara

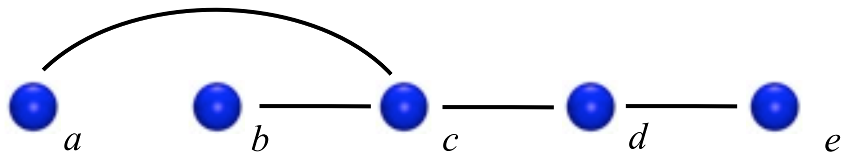
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





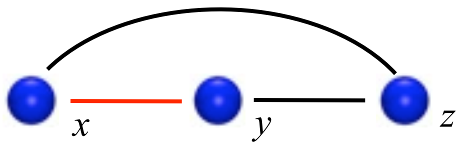


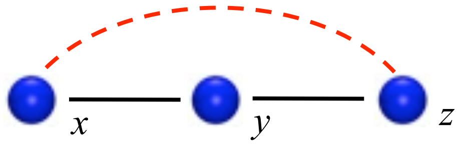
b-Tiara

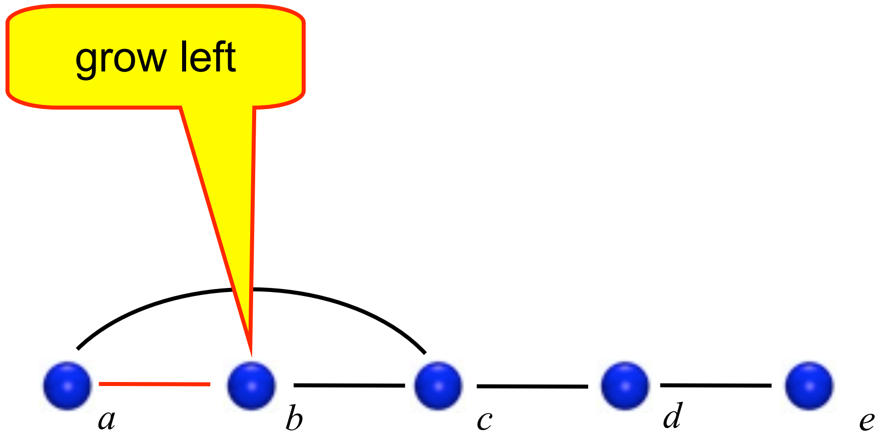
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





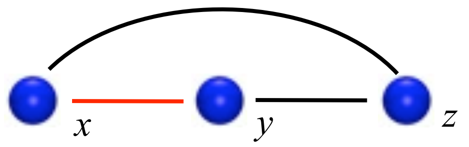


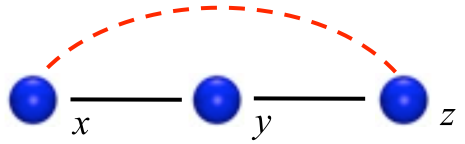
b-Tiara

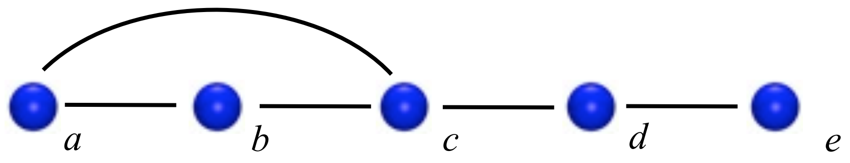
linearizes a list of peers at the bottom level

actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar



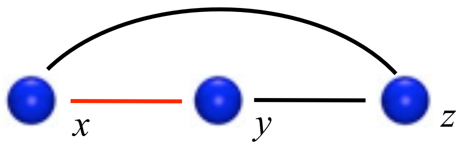


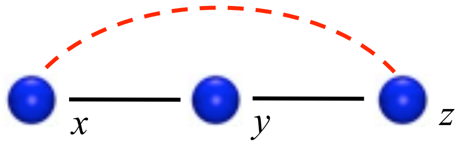


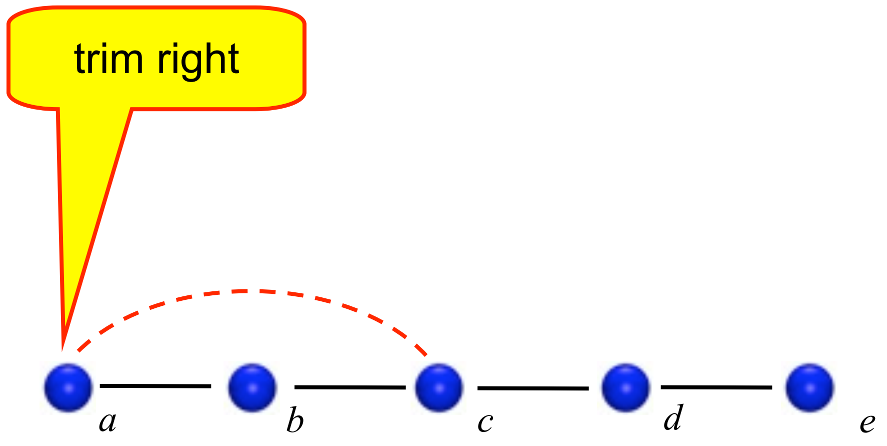
b-Tiara

linearizes a list of peers at the bottom level
actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





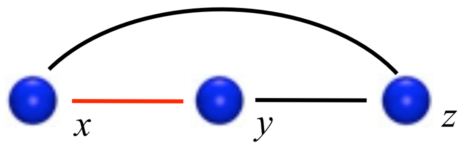


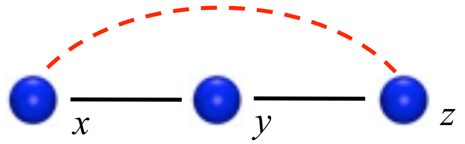
b-Tiara

linearizes a list of peers at the bottom level

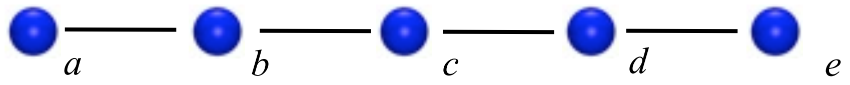
actions

- **grow right**: connect to a left neighbor of my right neighbor
- **grow left:** similar
- **trim right:** disconnect from right neighbor if there is a node in between connected to both
- **trim left:** similar





system is linearized



s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

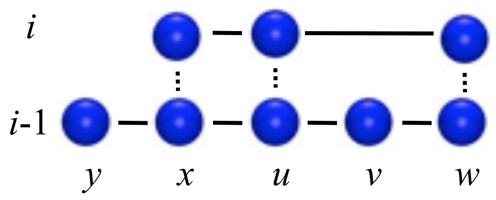
prune: remove all links but closest left and right neighbor and add to bottom level

downgrade right: remove and add to bottom left link from u if it does not link to either v or w

downgrade left: similar

downgrade center: unlink u if x and v are present at level i

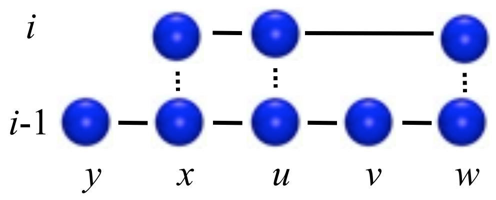
sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i

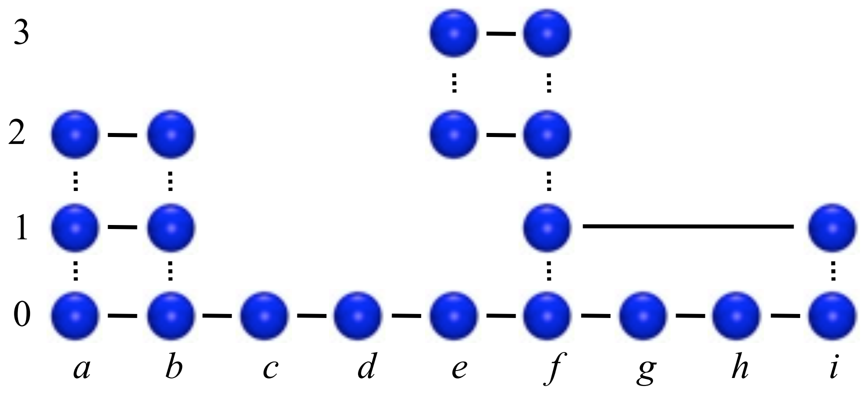


s-Tiara

actions
upgrade right: link u to w if v is down
upgrade left: similar
bridge left: link x to u if both exist at level i
bridge right: similar
prune: remove all links but closest left and right neighbor and add to bottom level
downgrade right: remove and add to bottom left link from u if it does not link to either v or w
downgrade left: similar
downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i





s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

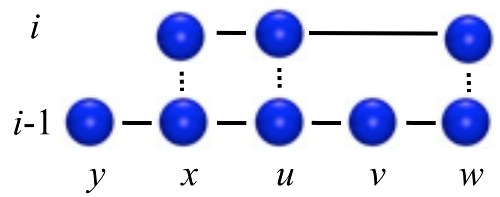
prune: remove all links but closest left and right neighbor and add to bottom level

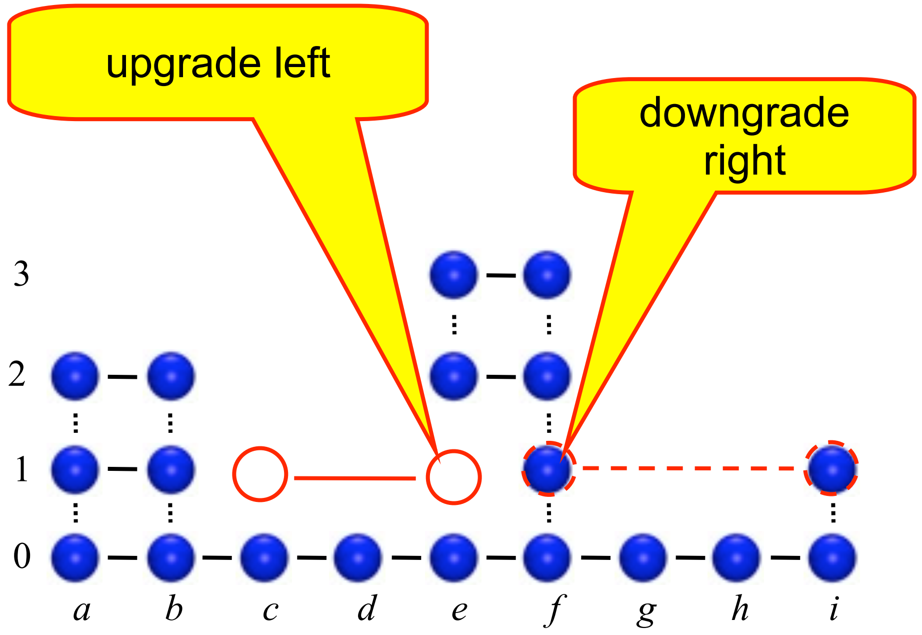
downgrade right: remove and add to bottom left link from u if it does not link to either v or w

downgrade left: similar

downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i





s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

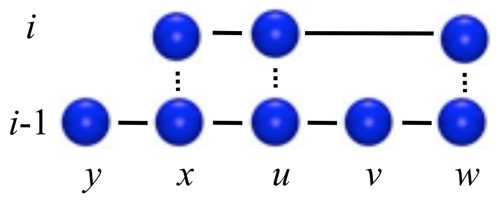
prune: remove all links but closest left and right neighbor and add to bottom level

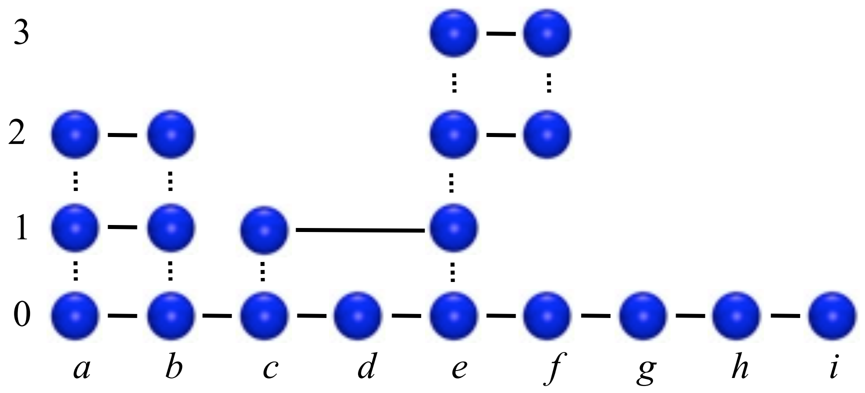
downgrade right: remove and add to bottom left link from u if it does not link to either v or w

downgrade left: similar

downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i





s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

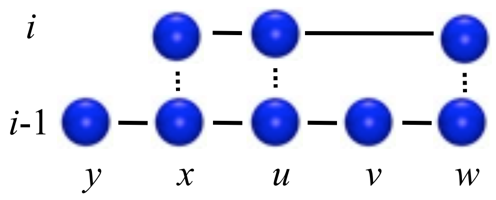
prune: remove all links but closest left and right neighbor and add to bottom level

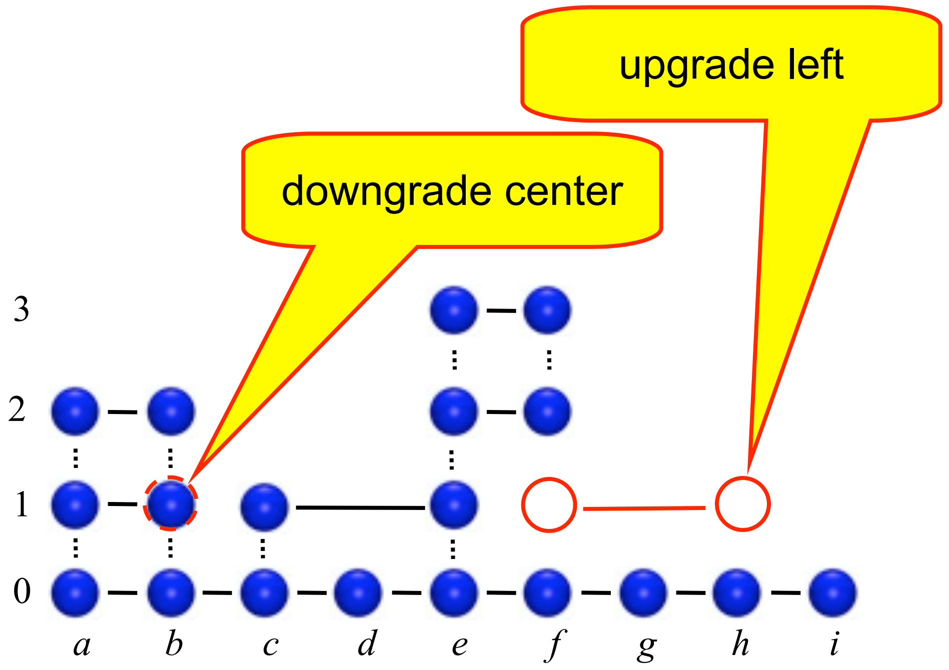
downgrade right: remove and add to bottom left link from u if it does not link to either v or w

downgrade left: similar

downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i





s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

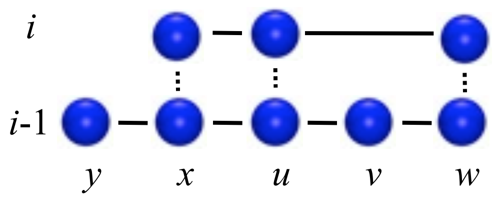
prune: remove all links but closest left and right neighbor and add to bottom level

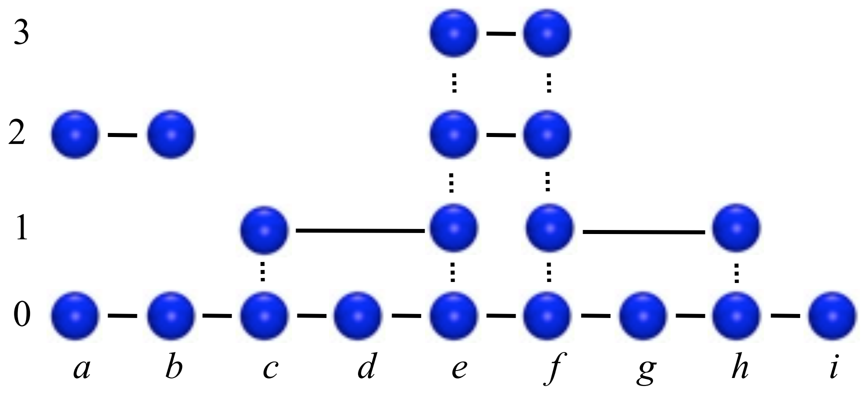
downgrade right: remove and add to bottom left link from u if it does not link to either v or w

downgrade left: similar

downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i





s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

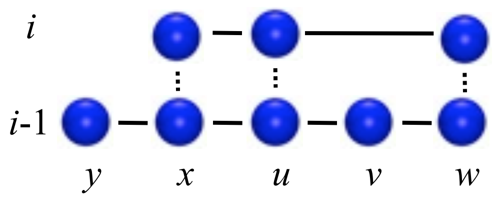
prune: remove all links but closest left and right neighbor and add to bottom level

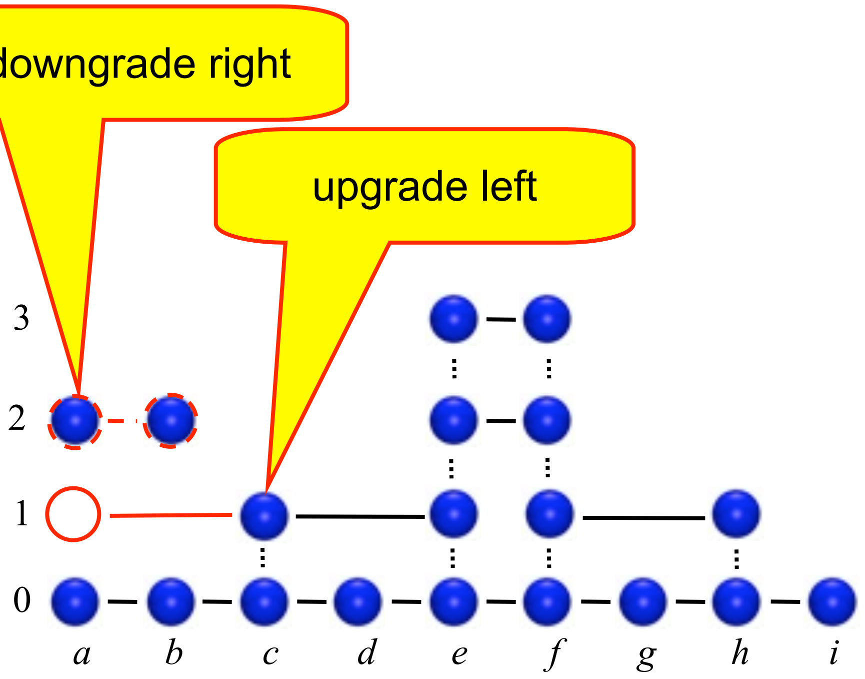
downgrade right: remove and add to bottom left link from u if it does not link to either v or w

downgrade left: similar

downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i





s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

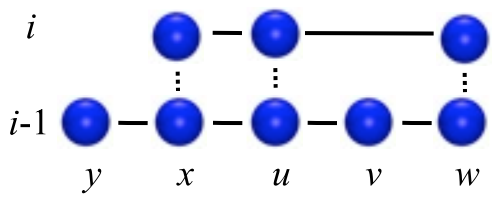
prune: remove all links but closest left and right neighbor and add to bottom level

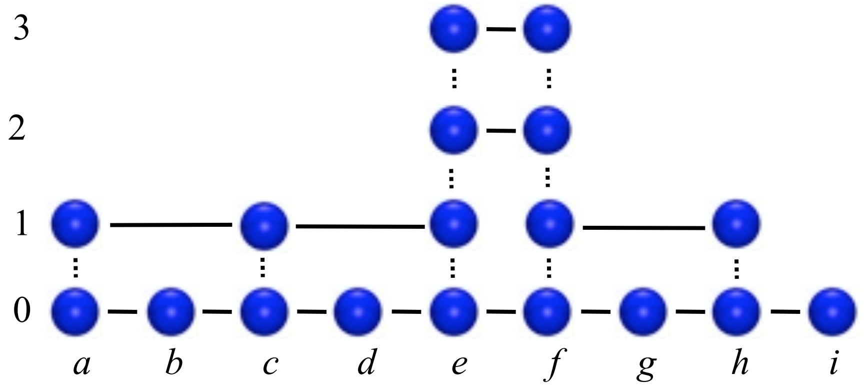
downgrade right: remove and add to bottom left link from u if it does not link to either v or w

downgrade left: similar

downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i





s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

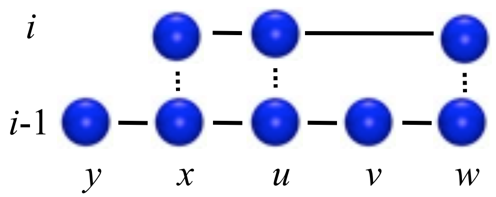
prune: remove all links but closest left and right neighbor and add to bottom level

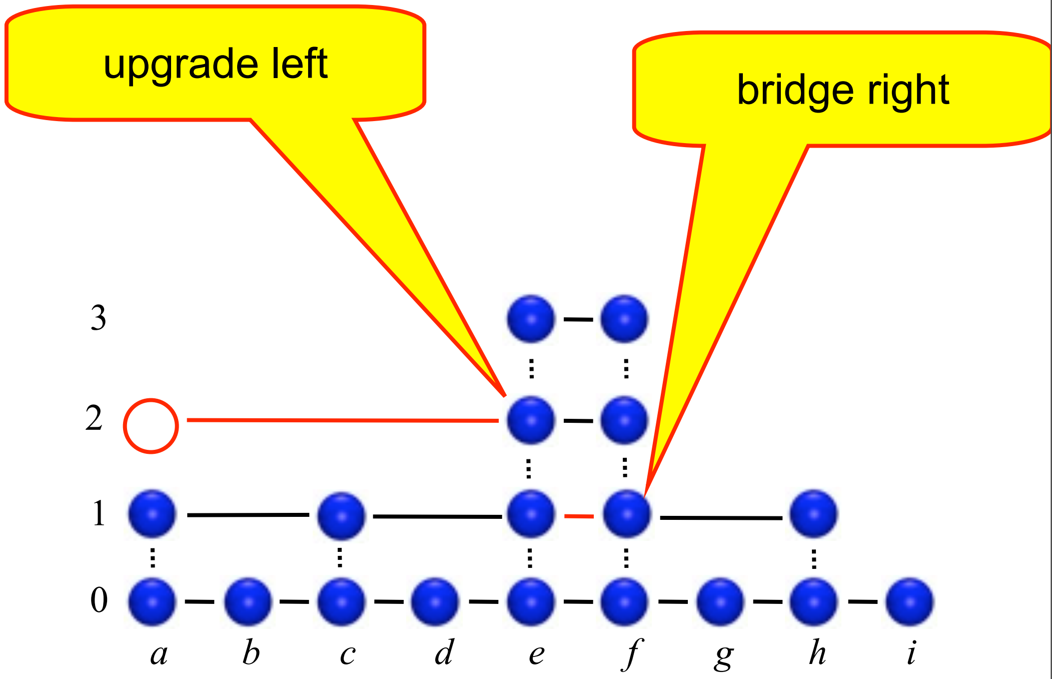
downgrade right: remove and add to bottom left link from u if it does not link to either v or w

downgrade left: similar

downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i





s-Tiara

actions

upgrade right: link u to w if v is down

upgrade left: similar

bridge left: link x to u if both exist at level i

bridge right: similar

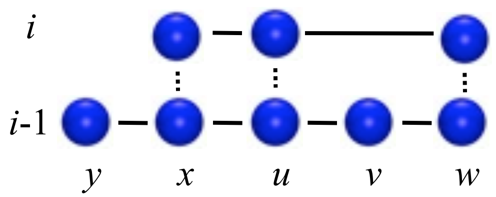
prune: remove all links but closest left and right neighbor and add to bottom level

downgrade right: remove and add to bottom left link from u if it does not link to either v or w

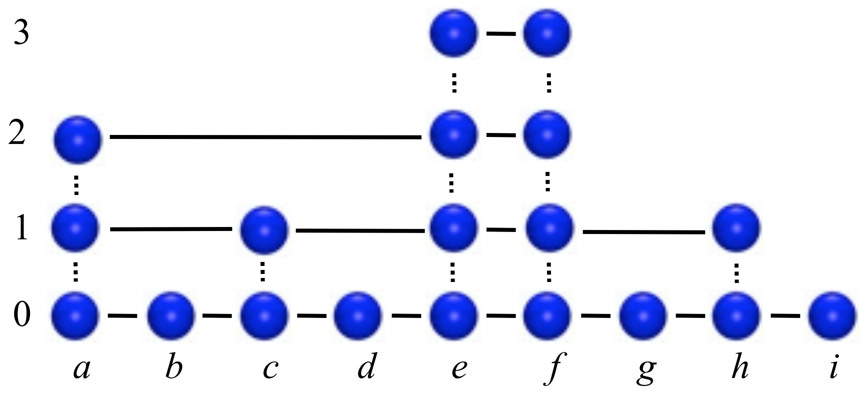
downgrade left: similar

downgrade center: unlink u if x and v are present at level i

sparse 0-1 skip list: out of three consecutive nodes at level i-1 at most two and at least one are on level i



skip list stabilized



Correctness Proof Outline

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

- assume bottom level is connected. Three stages
  - stabilization of grow in b-Tiara
    - closure – link connecting consequent nodes is never removed

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

- assume bottom level is connected. Three stages
  - stabilization of grow in b-Tiara
    - closure – link connecting consequent nodes is never removed
    - convergence – path connecting consequent nodes is always shortened
  - stabilization of s-Tiara (by level)
    - assume lower level(s) are stable,
    - closure – cage is indestructible

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

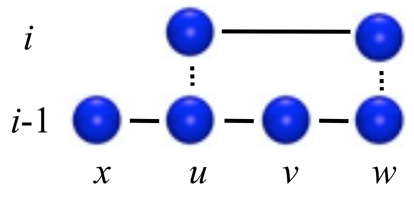
- assume bottom level is connected. Three stages
  - stabilization of grow in b-Tiara
    - closure – link connecting consequent nodes is never removed
    - convergence – path connecting consequent nodes is always shortened
  - stabilization of s-Tiara (by level)
    - assume lower level(s) are stable,
    - closure – cage is indestructible
    - convergence – eventually cages are formed and bridged

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

- assume bottom level is connected. Three stages
  - stabilization of grow in b-Tiara
    - closure – link connecting consequent nodes is never removed
    - convergence – path connecting consequent nodes is always shortened
  - stabilization of s-Tiara (by level)
    - assume lower level(s) are stable,
    - closure – cage is indestructible
    - convergence – eventually cages are formed and bridged

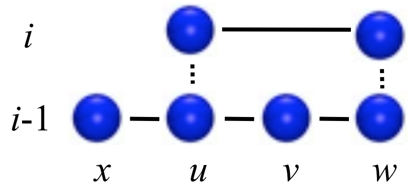


Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

- assume bottom level is connected. Three stages
  - stabilization of grow in b-Tiara
    - closure – link connecting consequent nodes is never removed
    - convergence – path connecting consequent nodes is always shortened
  - stabilization of s-Tiara (by level)
    - assume lower level(s) are stable,
    - closure – cage is indestructible
    - convergence – eventually cages are formed and bridged

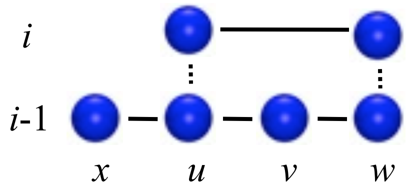


Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

$\Rightarrow$ level-by-level stabilization proof is not immediately applicable

- assume bottom level is connected. Three stages
  - stabilization of grow in b-Tiara
    - closure – link connecting consequent nodes is never removed
    - convergence – path connecting consequent nodes is always shortened
  - stabilization of s-Tiara (by level)
    - assume lower level(s) are stable,
    - closure – cage is indestructible
    - convergence – eventually cages are formed and bridged



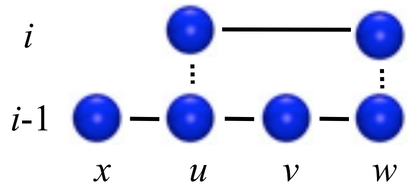
  - stabilization of trim in b-Tiara
    - assume grow of b-Tiara and complete s-Tiara are stable
    - closure – when b-Tiara is linearized link addition is not possible
    - convergence – when b-Tiara and s-Tiara are stable, no links are added to b-Tiara, outermost link enables trim which removes it

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

- assume bottom level is connected. Three stages
  - stabilization of grow in b-Tiara
    - closure – link connecting consequent nodes is never removed
    - convergence – path connecting consequent nodes is always shortened
  - stabilization of s-Tiara (by level)
    - assume lower level(s) are stable,
    - closure – cage is indestructible
    - convergence – eventually cages are formed and bridged



  - stabilization of trim in b-Tiara
    - assume grow of b-Tiara and complete s-Tiara are stable
    - closure – when b-Tiara is linearized link addition is not possible
    - convergence – when b-Tiara and s-Tiara are stable, no links are added to b-Tiara, outermost link enables trim which removes it

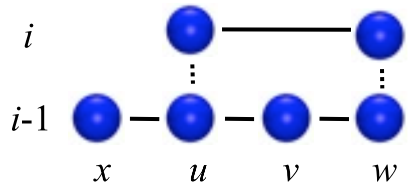
- if bottom level is disconnected (system connected at upper levels)

Correctness Proof Outline

complication – due to connectivity preservation, there is a feedback between s-Tiara and b-Tiara

⇒ level-by-level stabilization proof is not immediately applicable

- assume bottom level is connected. Three stages
  - stabilization of grow in b-Tiara
    - closure – link connecting consequent nodes is never removed
    - convergence – path connecting consequent nodes is always shortened
  - stabilization of s-Tiara (by level)
    - assume lower level(s) are stable,
    - closure – cage is indestructible
    - convergence – eventually cages are formed and bridged



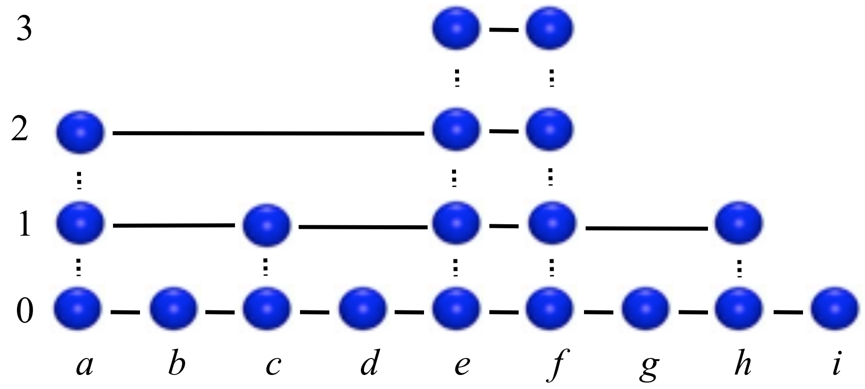
  - stabilization of trim in b-Tiara
    - assume grow of b-Tiara and complete s-Tiara are stable
    - closure – when b-Tiara is linearized link addition is not possible
    - convergence – when b-Tiara and s-Tiara are stable, no links are added to b-Tiara, outermost link enables trim which removes it

- if bottom level is disconnected (system connected at upper levels)
  - connected bottom-level components stabilize forcing extraneous links to drop to bottom level and connect

Outline

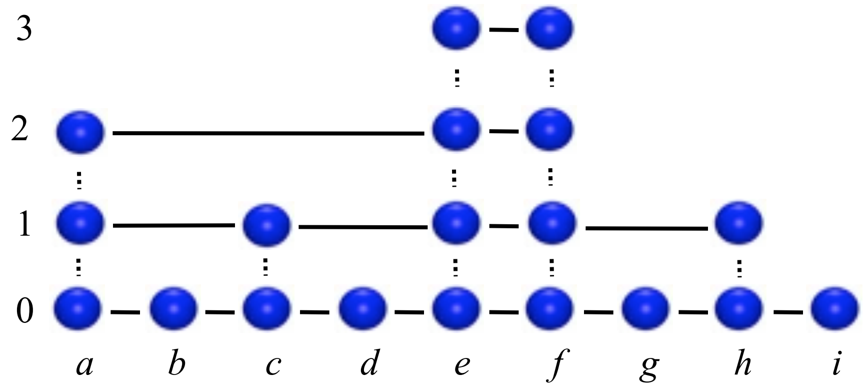
- overlay networks and programming model
- Tiara
  - bottom level
  - skip-list
  - searches, topology updates
- usage and extensions
- related literature
- extensions and future work

Usage, Implementation and Extensions



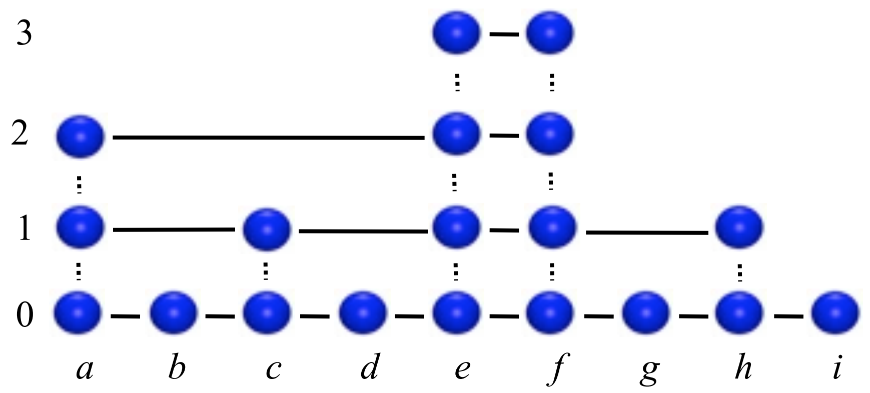
Usage, Implementation and Extensions

- searches: Tiara maintains a skip list – equivalent to a balanced search tree, a node at level i is responsible for ranges between its right and left neighbors, at level i-1, each range contains at most two subranges
  - proceed up until the node of correct range is found
  - proceed splitting ranges until target is found or target is in range of consequent nodes (search miss)
  - $\log(N)$ steps



Usage, Implementation and Extensions

- searches: Tiara maintains a skip list – equivalent to a balanced search tree, a node at level i is responsible for ranges between its right and left neighbors, at level i-1, each range contains at most two subranges
    - proceed up until the node of correct range is found
    - proceed splitting ranges until target is found or target is in range of consequent nodes (search miss)
    - $\log(N)$ steps



- topology updates (joins and leaves)
    - to join the node conducts search and merges at bottom level
    - to leave the node signals intent to leave, left/right neighbors link
- extension to ring (to imitate ring-based structures like Chord)
    - need to establish wraparound link – node without left neighbor (potentially smallest id) searches over b-Tiara for node without right neighbor (potentially largest)
        - this procedure succeeds after grow of b-Tiara stabilizes
    - higher levels link over the wraparound link

Related Work

- M. Onus, A. Richa, C. Scheideler, “Linearization: Locally Self-Stabilizing Sorting in Graphs”, *ALENEX* 07 – high-atomicity stabilizing linearization
- A. Shaker, D.S. Reeves, “Self-Stabilizing Structured Ring Topology P2P Systems”, *P2P* 05 – stabilizing ring
- E. Caron, F. Desprez, F. Petit, C. Tedesci, “Snap-Stabilizing Prefix Tree for Peer-to-Peer Systems”, *SSS 07* - snap-stabilizing prefix tree for P2P systems
- S. Bianchi, A. Datta, P. Felber, M. Gradinariu, “Stabilizing Peer-to-Peer Spatial Filters”, *ICDCS 07* – stabilizing search tree for overlay networks optimized for content filters
- S. Dolev, R.I. Kat, “HyperTree for Self-Stabilizing Peer-to-Peer Systems” *Distributed Computing* 20(5) - randomized search structure for P2P
- D. Dolev, E. Hoch, R. van Renesse, *OPODIS 07*, “Self-Stabilizing and Byzantine Fault Tolerant Overlay Network” – stabilizing randomized synchronous intrusion tolerant overlay network

Future Work

- decrease atomicity (possible)
- decrease congestion (extended to deterministic skip-graphs in journal version)
- stabilize structures with high expansion and small diameter
- skip lists with smaller fraction of nodes promoted to higher levels (possible)
- fortify against churn

Thank You

Questions?