אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

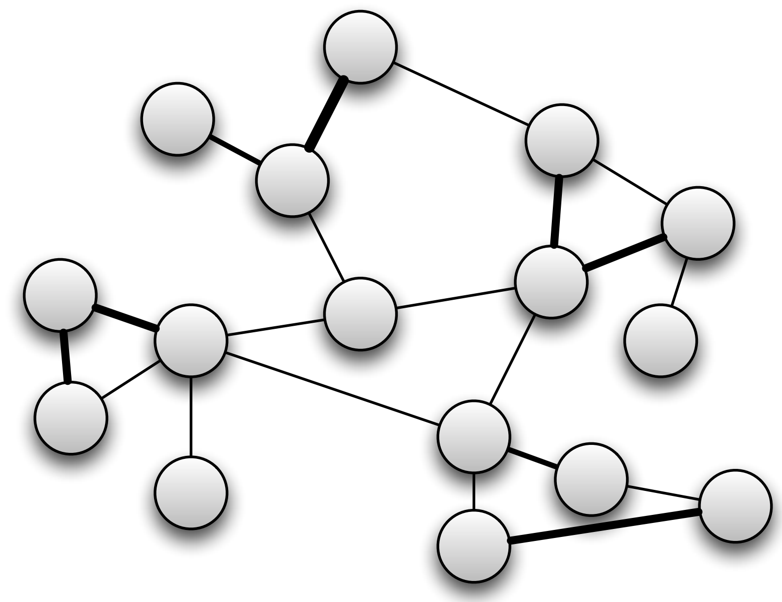
Online Balanced Repartitioning

Chen Avin

Ben Gurion University of the Negev

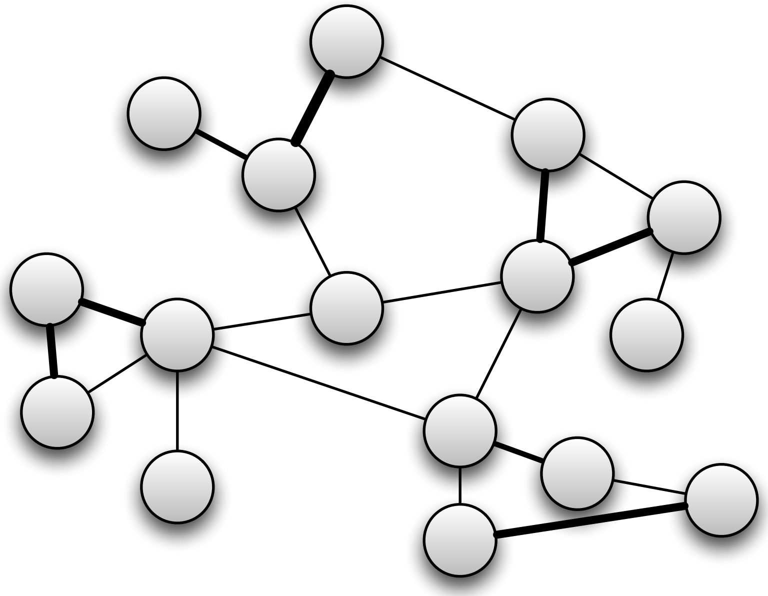
Joint work with Andreas Loukas, Maciej Pacut & Stefan Schmid

Motivation



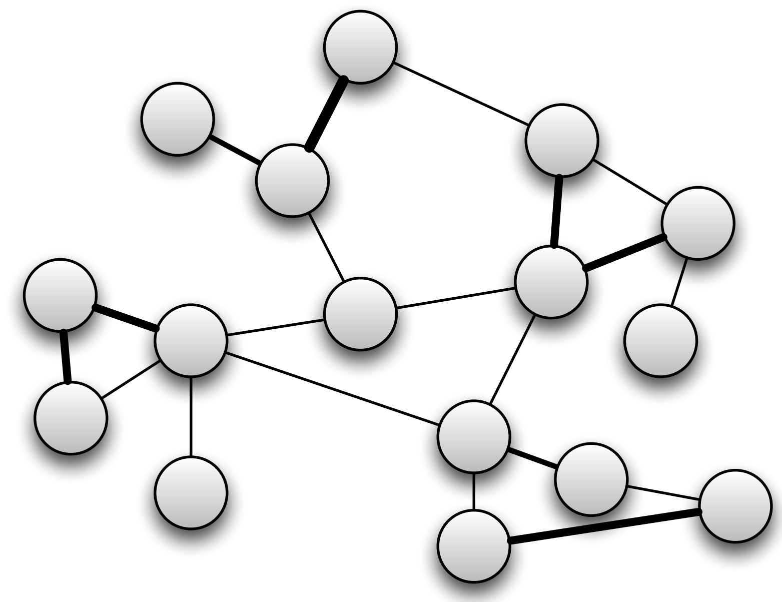
Motivation

- Graph partitioning problems



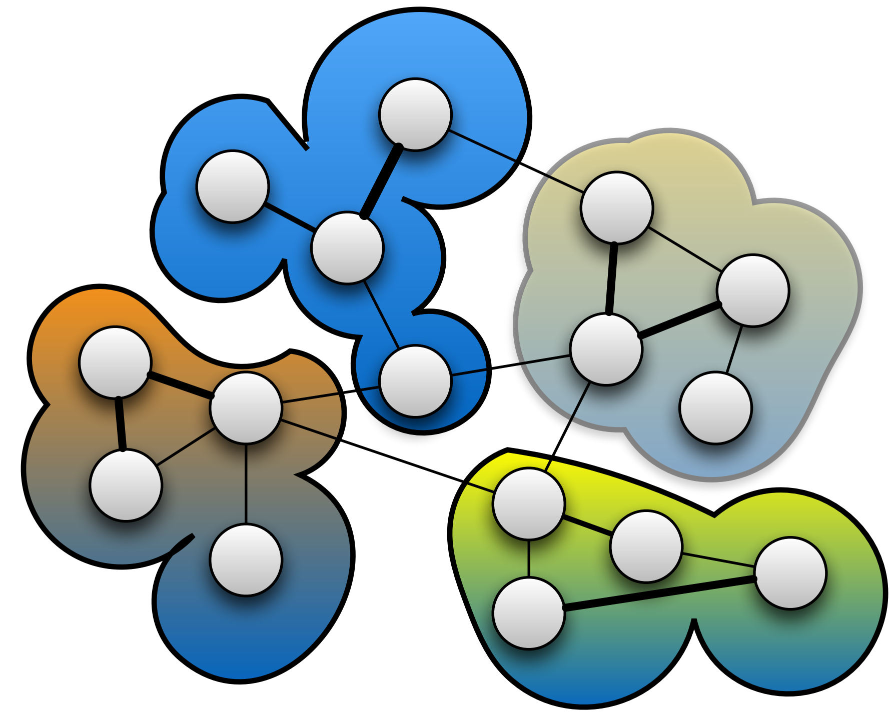
Motivation

- Graph partitioning problems
    - ℓ clusters, each of size k



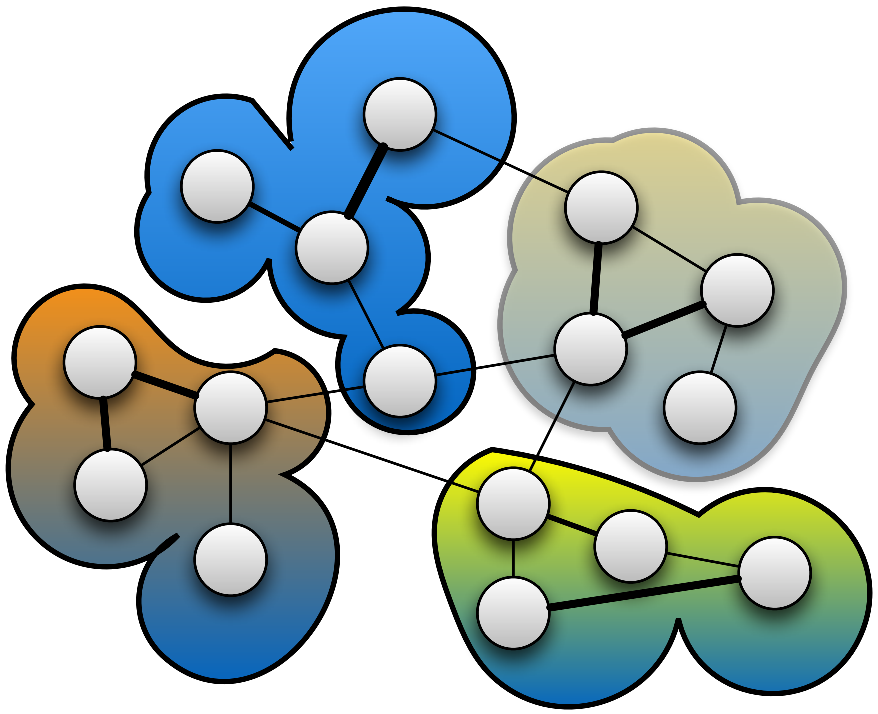
Motivation

- Graph partitioning problems
  - ℓ clusters, each of size k



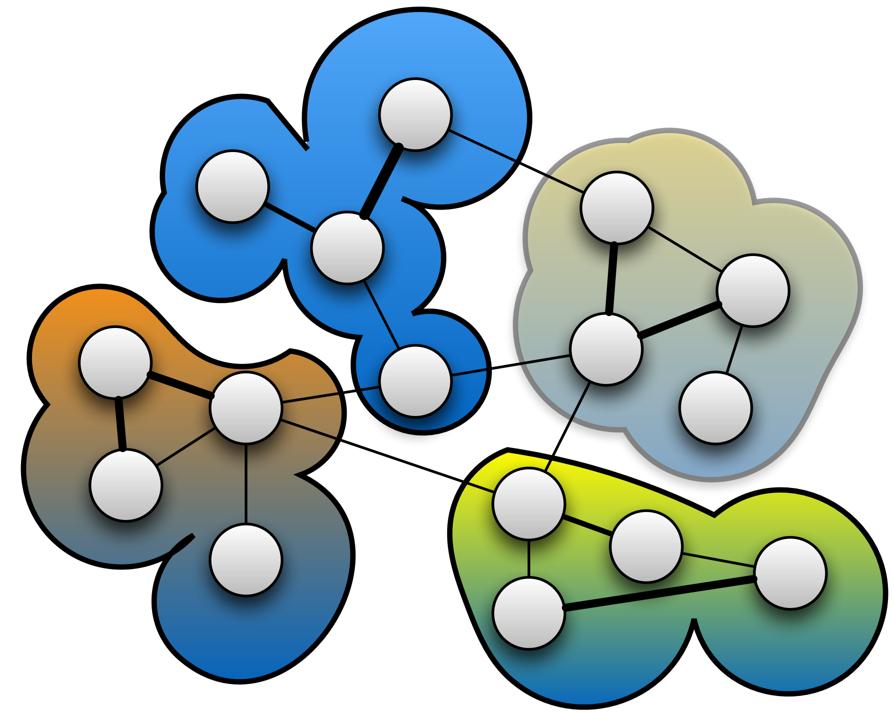
Motivation

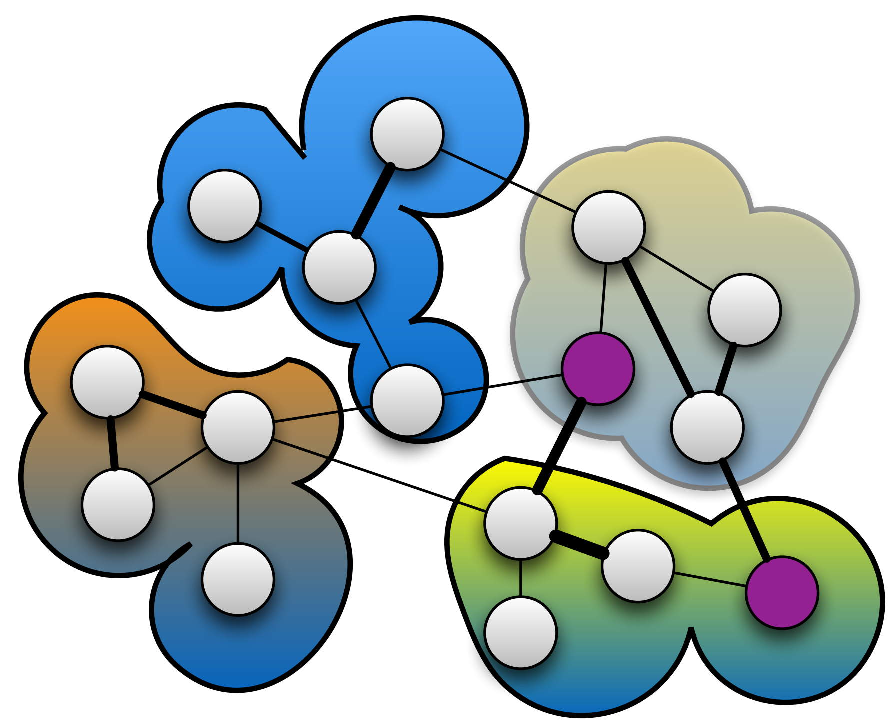
- Graph partitioning problems
  - ℓ clusters, each of size k
- Online graph re-partitioning



Motivation

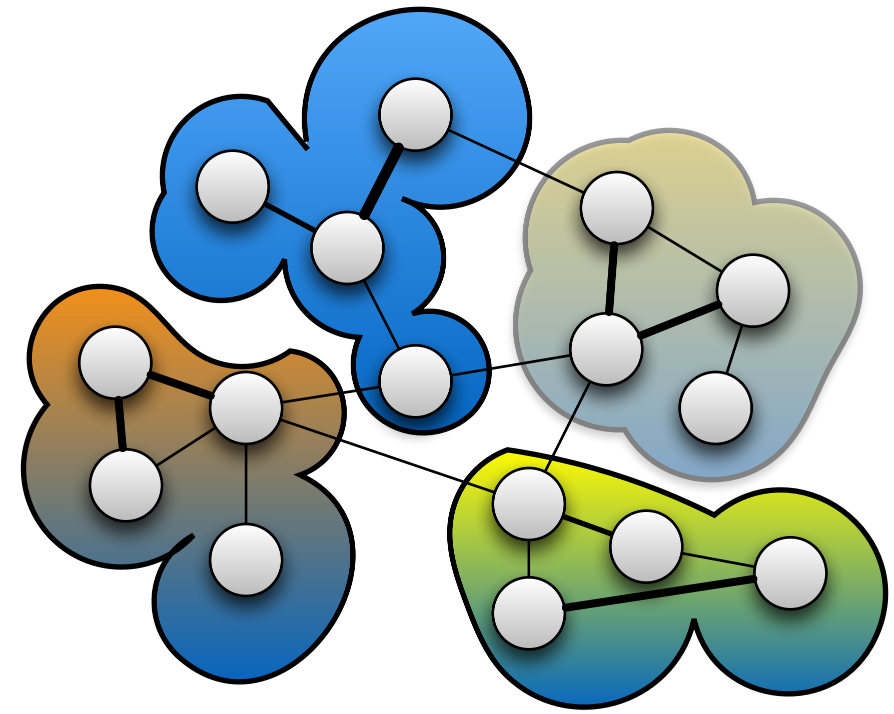
- Graph partitioning problems
    - ℓ clusters, each of size k
- Online graph re-partitioning
    - Edges are updated

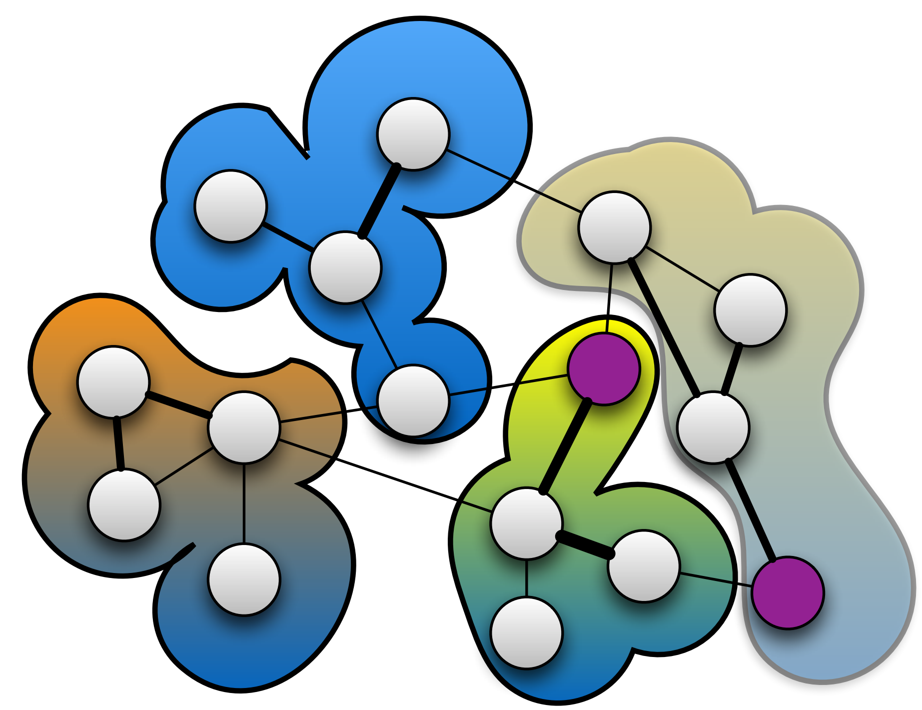




Motivation

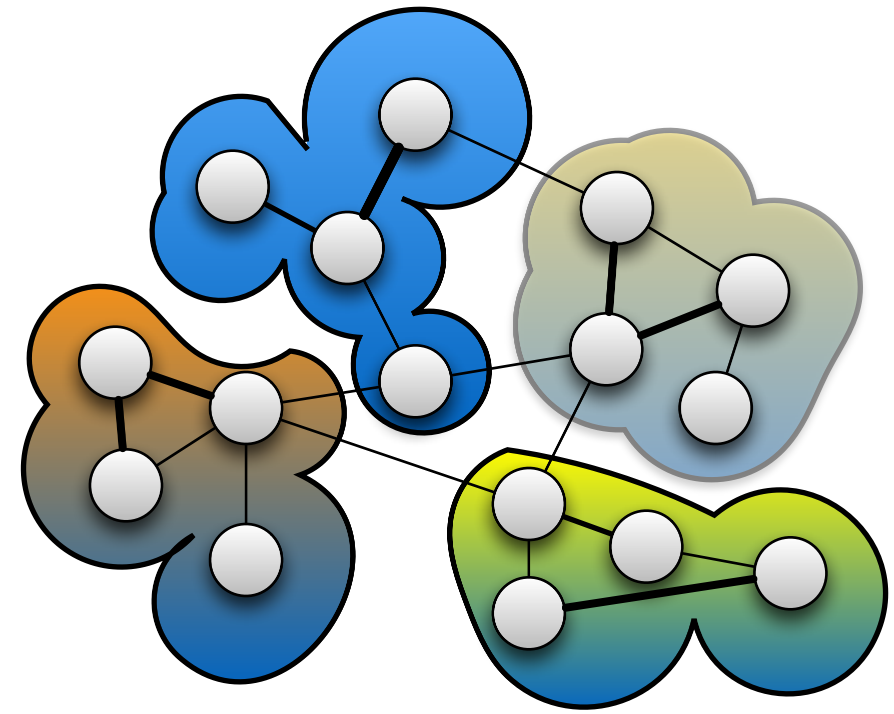
- Graph partitioning problems
    - ℓ clusters, each of size k
- Online graph re-partitioning
    - Edges are updated
    - Clustering is updated

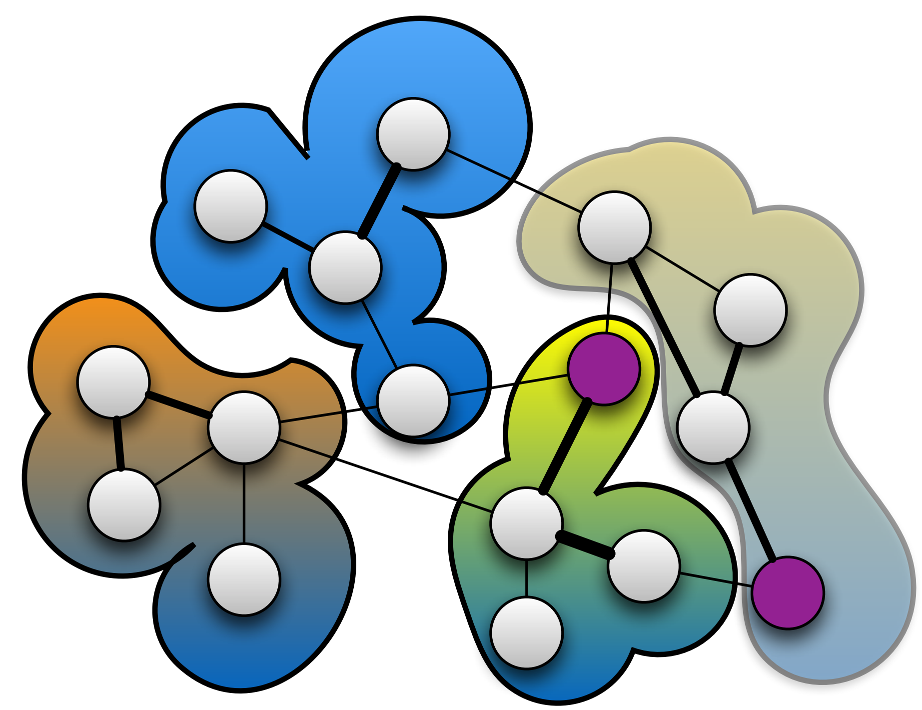




Motivation

- Graph partitioning problems
  - ℓ clusters, each of size k
- Online graph re-partitioning
  - Edges are updated
  - Clustering is updated
  - At a cost





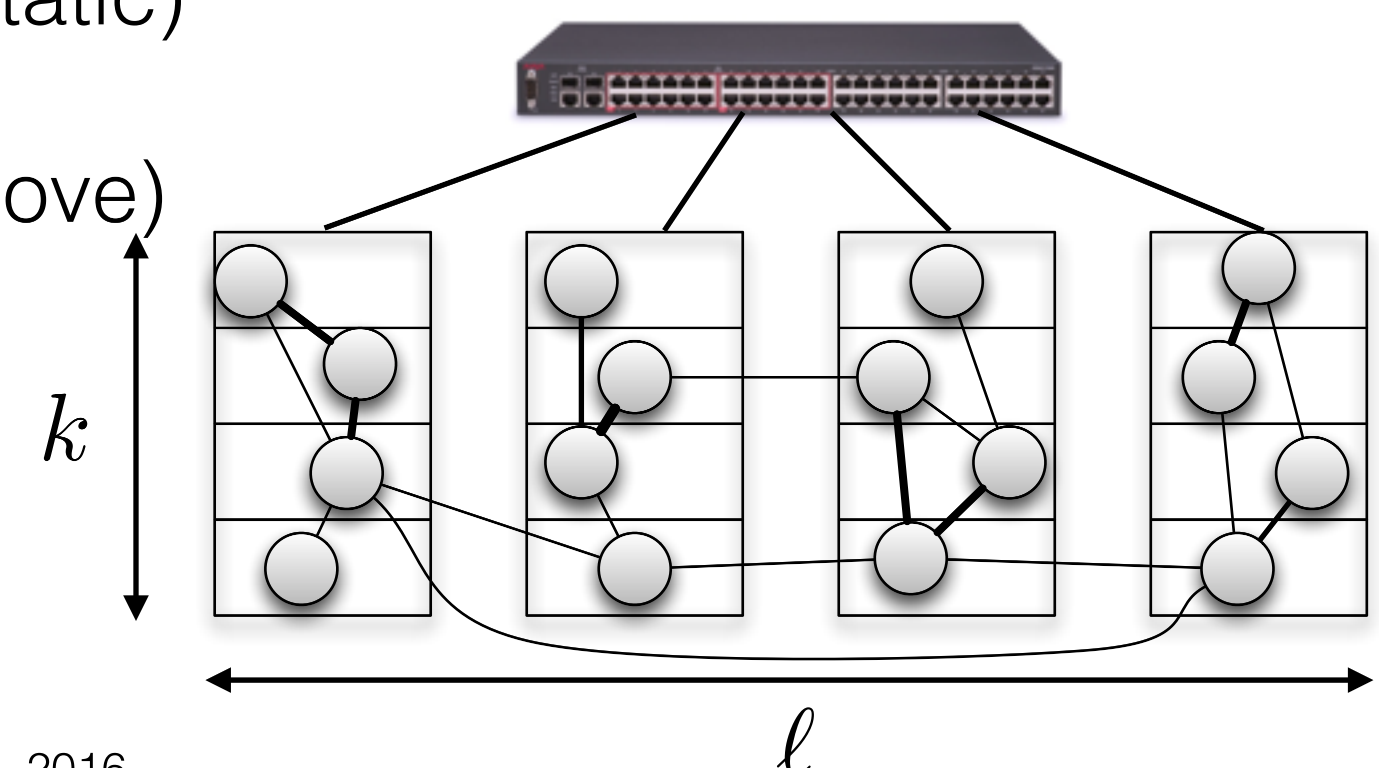
Motivation

Motivation

- Practical motivation
  - Data centres
  - Reduce network traffic

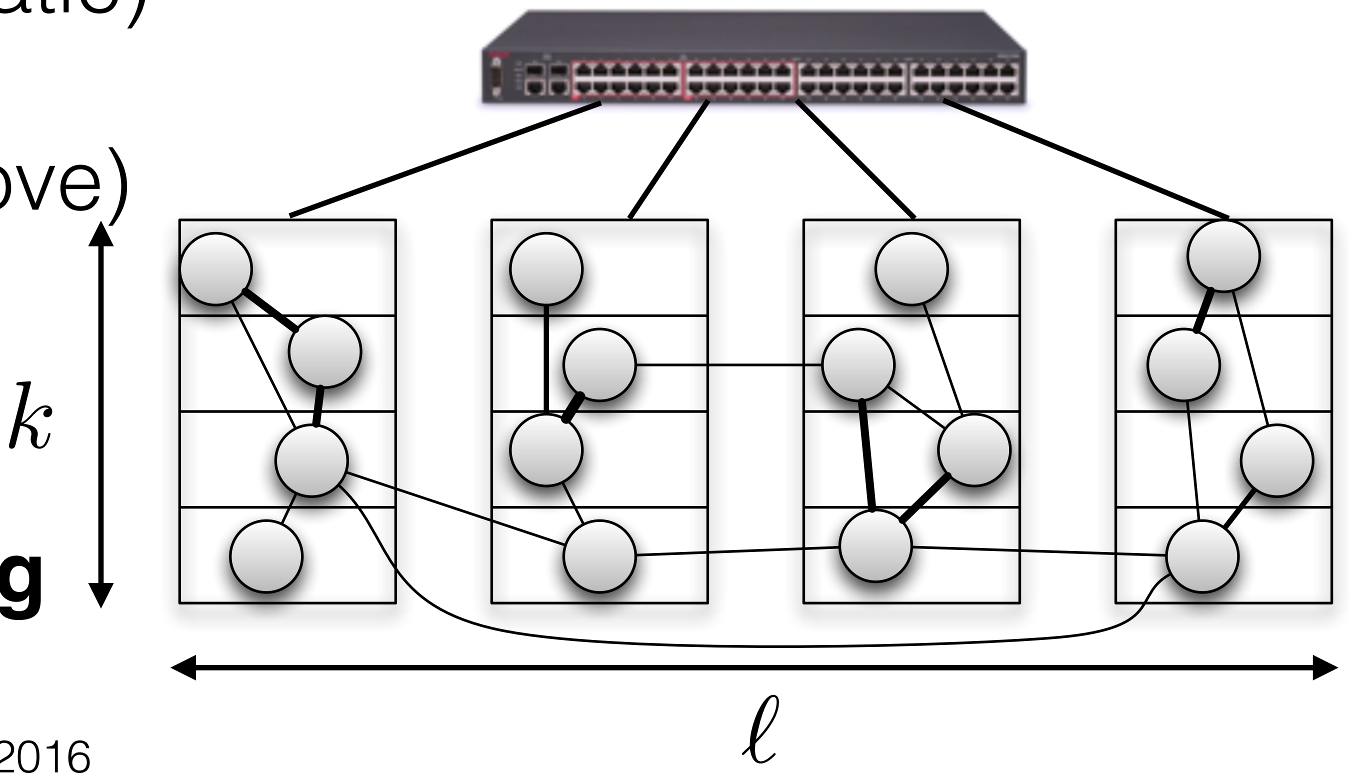
Motivation

- Practical motivation
  - Data centres
  - Reduce network traffic
  - Clusters as **servers** (static)
  - Nodes as **VMs** (can move)
  - $k << \ell$



Motivation

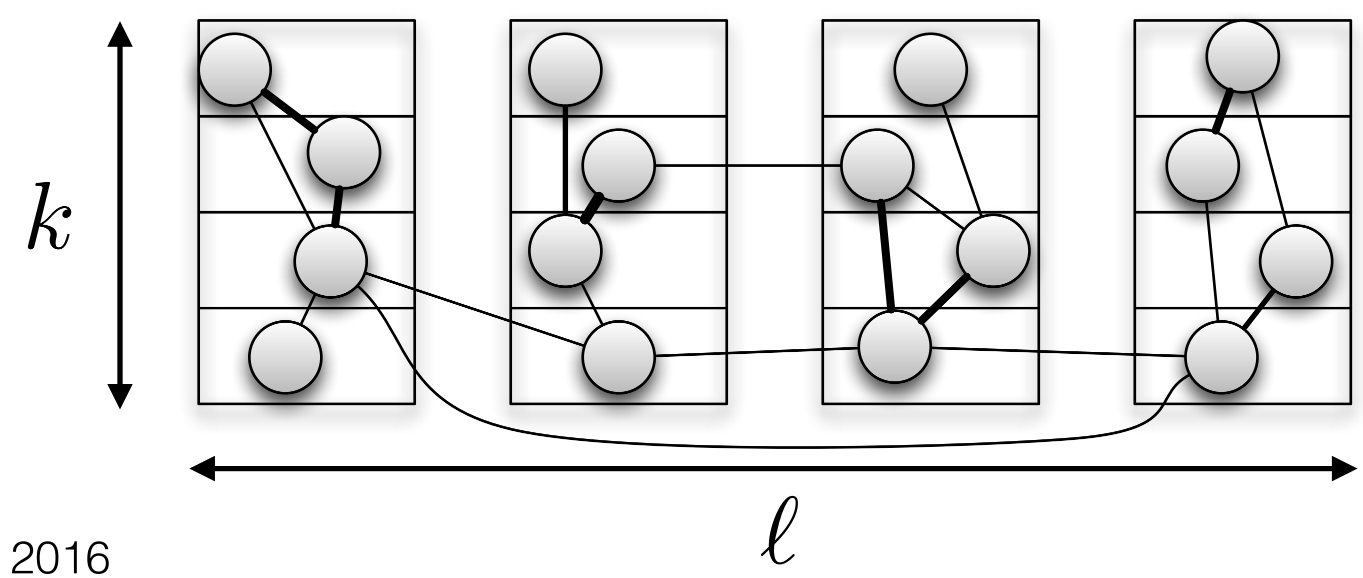
- Practical motivation
  - Data centres
  - Reduce network traffic
  - Clusters as **servers** (static)
  - Nodes as **VMs** (can move)
  - $k << \ell$
- **Traffic-Aware Networking**



Overview

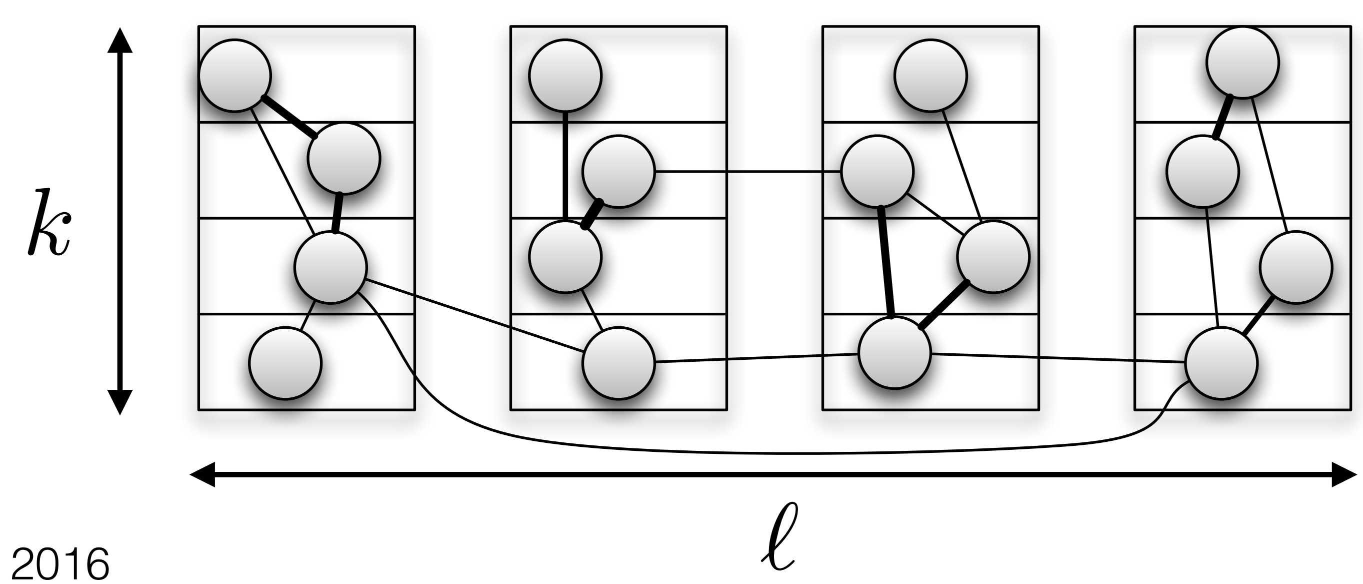
- Motivation
- Model and Problem definition
- Examples
- Some results
- Future work and open questions

Model



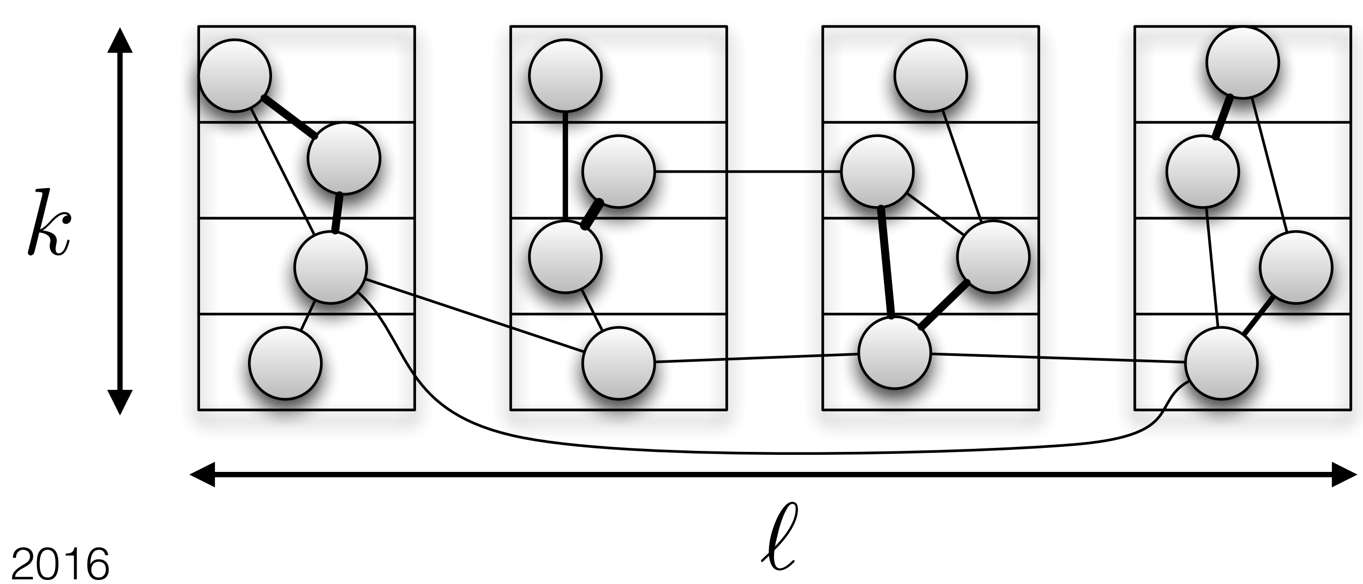
Model

- ***Balanced RePartitioning*** **(BRP)**



Model

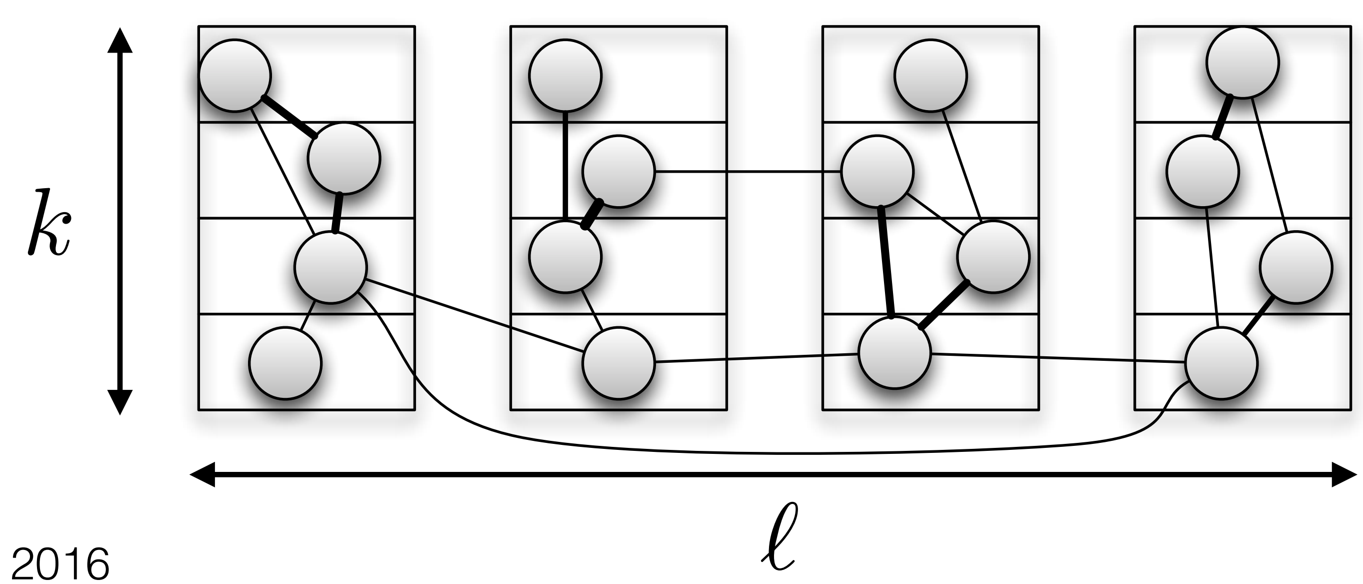
- ***Balanced RePartitioning*** **(BRP)**
    - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size k



Model

- ***Balanced RePartitioning* (BRP)**
  - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size k
  - (online) pairwise communication requests

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

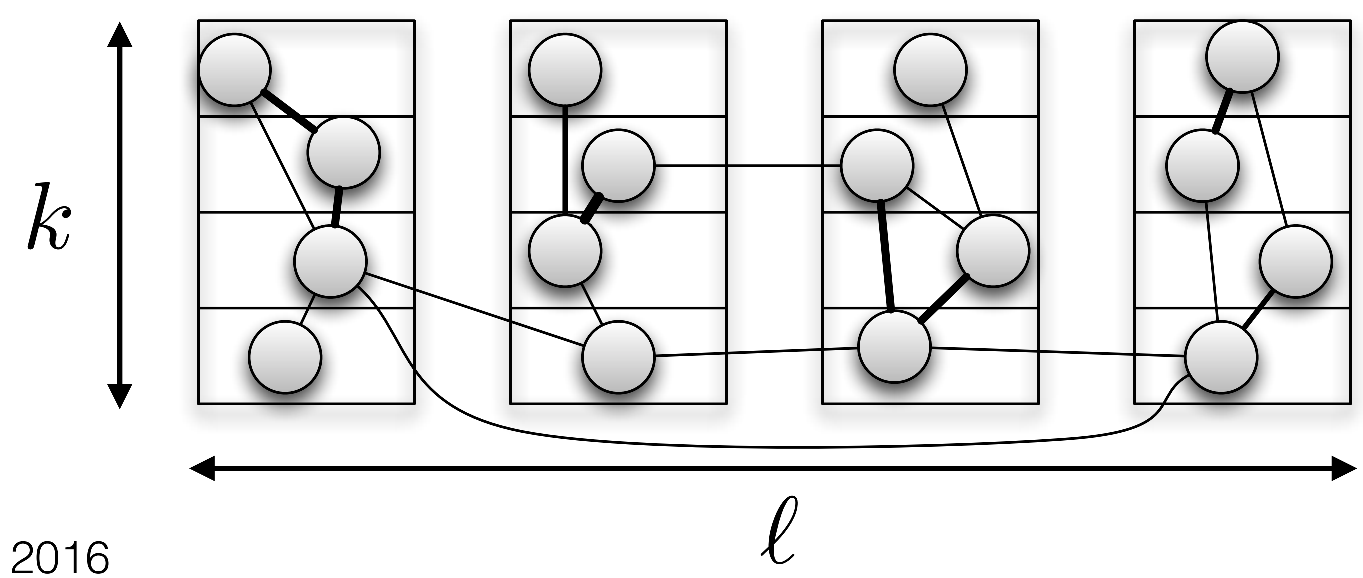


Model

- ***Balanced RePartitioning*** **(BRP)**
  - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size k
  - (online) pairwise communication requests

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

  - Serving **costs** for $\sigma_t = \{u, v\}$:

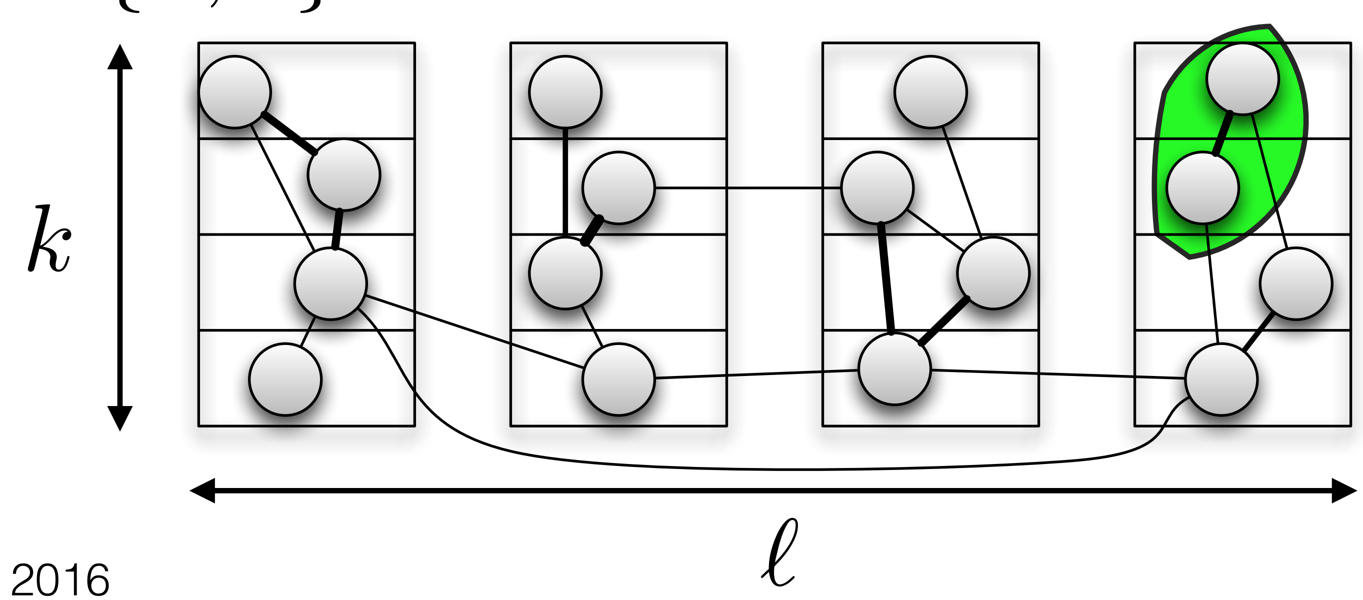


Model

- ***Balanced RePartitioning* (BRP)**
  - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size k
  - (online) pairwise communication requests

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

  - Serving **costs** for $\sigma_t = \{u, v\}$:
    - **intra-cluster:** 0

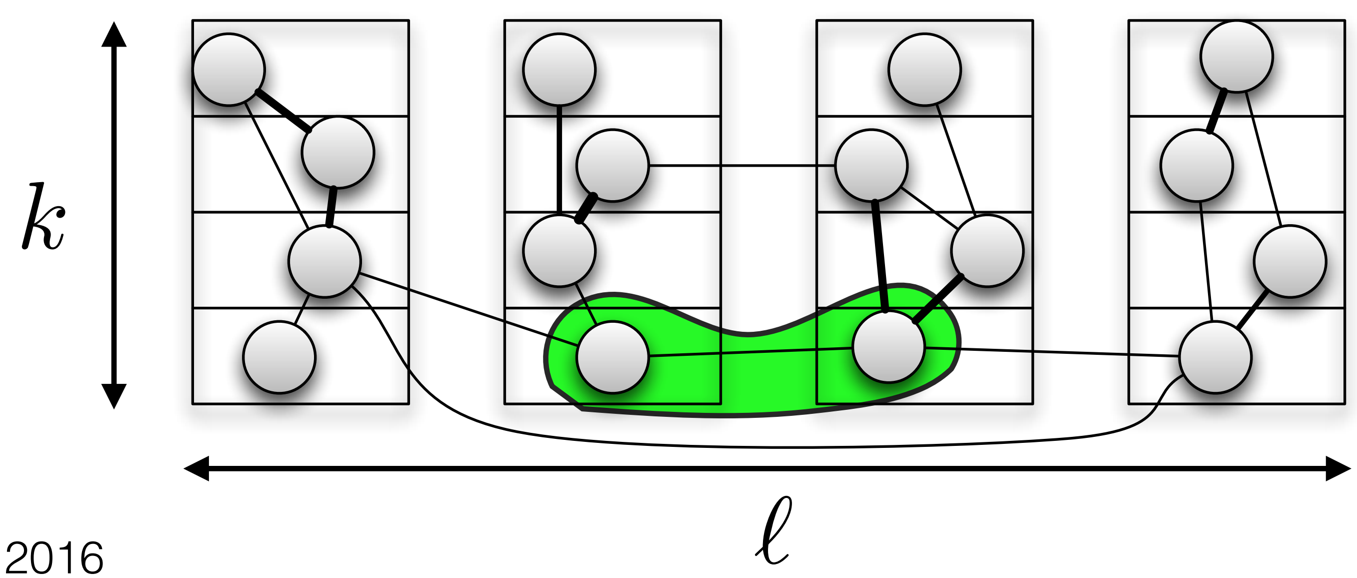


Model

- ***Balanced RePartitioning* (BRP)**
  - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size k
  - (online) pairwise communication requests

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

  - Serving **costs** for $\sigma_t = \{u, v\}$:
    - **intra-cluster:** 0
    - **inter-cluster:** 1

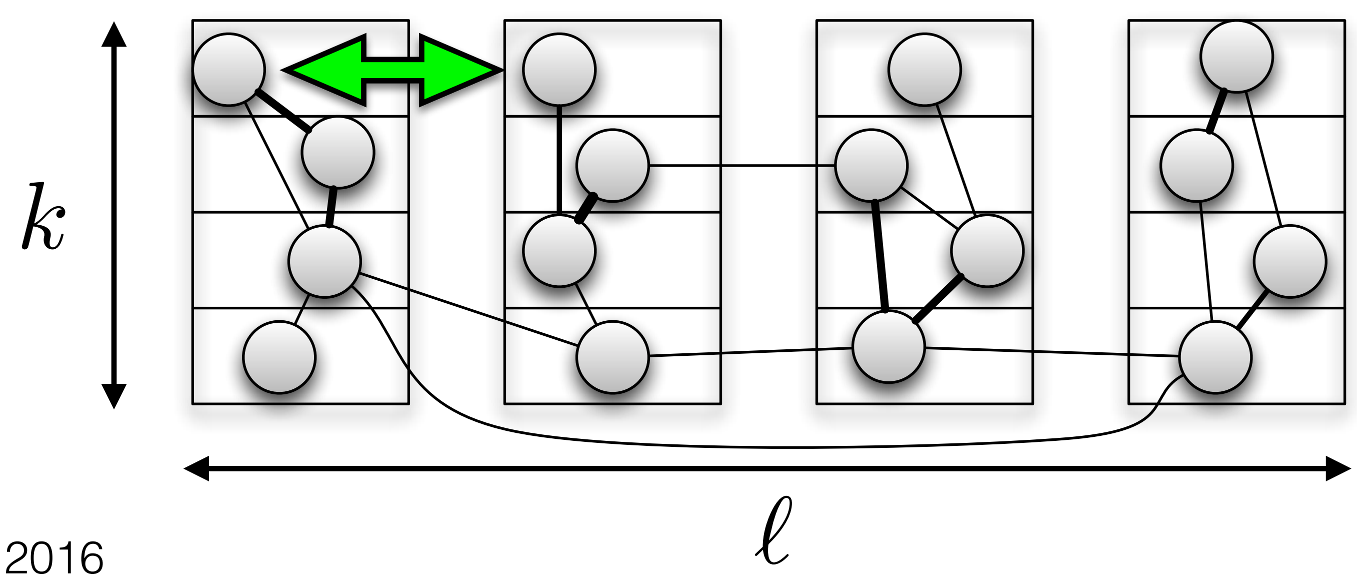


Model

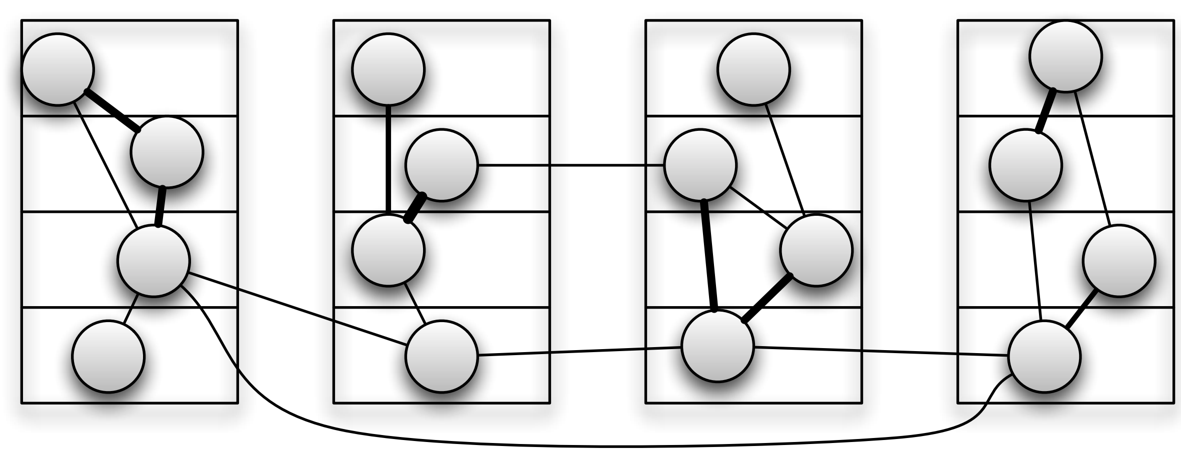
- ***Balanced RePartitioning*** **(BRP)**
  - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size k
  - (online) pairwise communication requests

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

  - Serving **costs** for $\sigma_t = \{u, v\}$:
    - **intra-cluster:** 0
    - **inter-cluster:** 1
    - **migration**: α



Problem Defintion

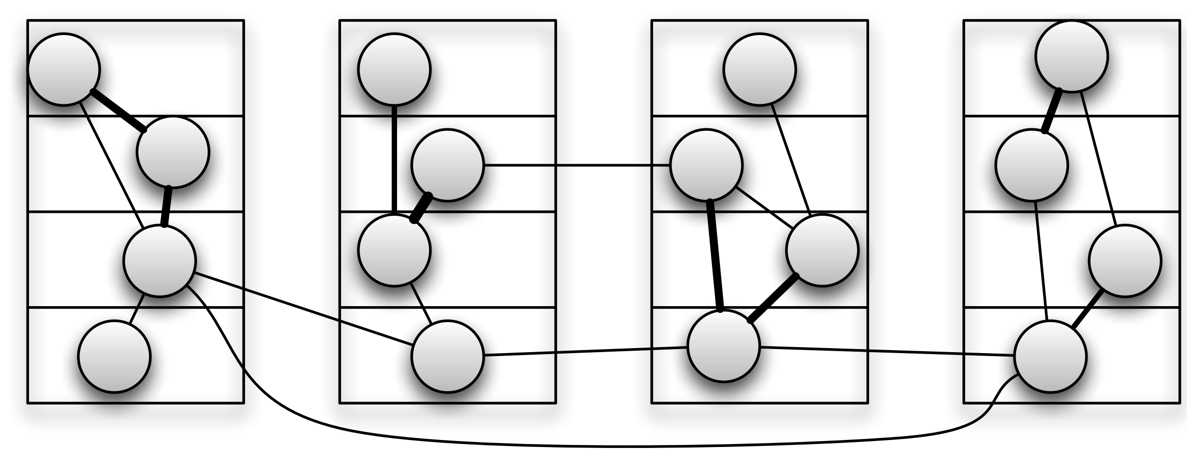


Problem Defintion

- The Cost of **ALG**

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

$$\text{ALG}(\sigma) = \sum_{t=1}^{|\sigma|} \text{mig}(\sigma_t; \text{ALG}) + \text{com}(\sigma_t; \text{ALG})$$

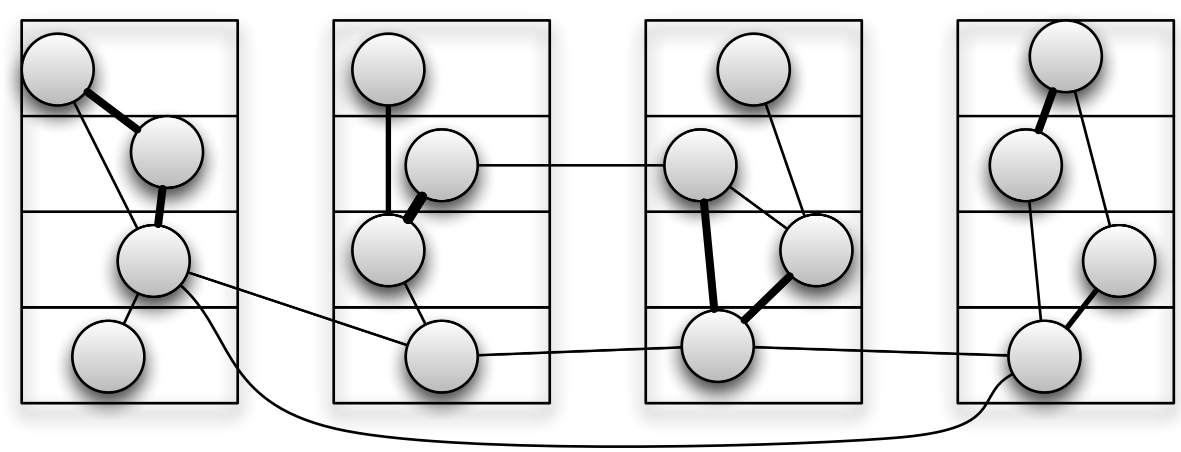


Problem Defintion

- The Cost of **ALG**

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

$$\mathrm{ALG}(\sigma) = \sum_{t=1}^{|\sigma|} \mathrm{mig}(\sigma_t; \mathrm{ALG}) + \mathrm{com}(\sigma_t; \mathrm{ALG})$$

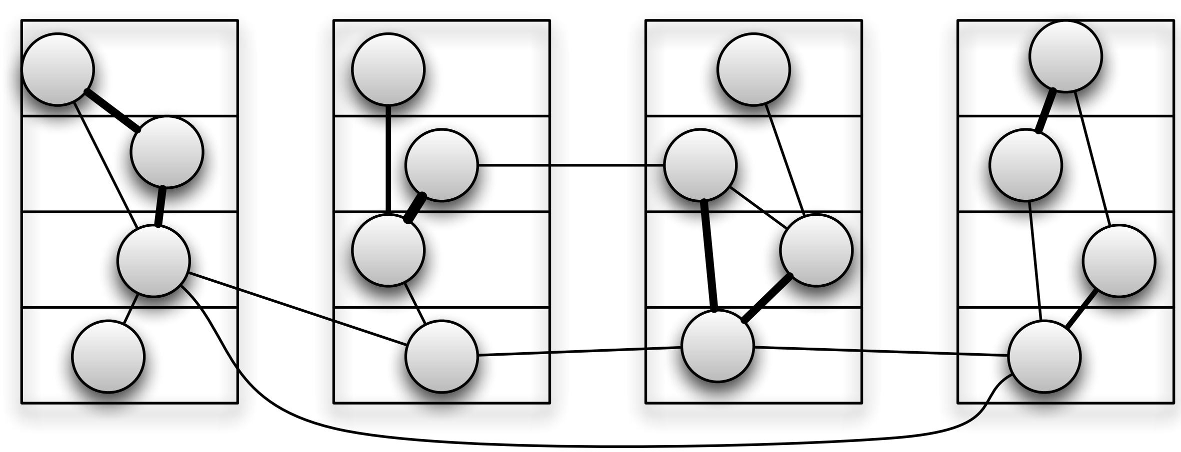


Problem Defintion

- The Cost of **ALG**

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

$$\mathrm{ALG}(\sigma) = \sum_{t=1}^{|\sigma|} \mathrm{mig}(\sigma_t; \mathrm{ALG}) + \mathrm{com}(\sigma_t; \mathrm{ALG})$$



Problem Defintion

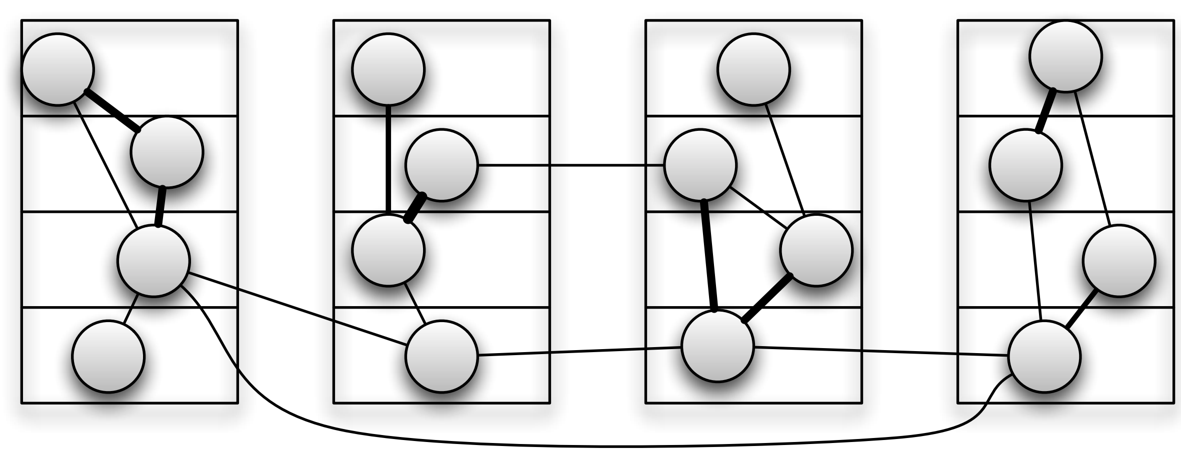
- The Cost of **ALG**

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

$$\mathrm{ALG}(\sigma) = \sum_{t=1}^{|\sigma|} \mathrm{mig}(\sigma_t; \mathrm{ALG}) + \mathrm{com}(\sigma_t; \mathrm{ALG})$$

- What is the **competitive ratio**

$$\rho(\textsc{On}) = \max_{\sigma} \frac{\textsc{On}(\sigma)}{\textsc{Off}(\sigma)}$$



Problem Defintion

- The Cost of **ALG**

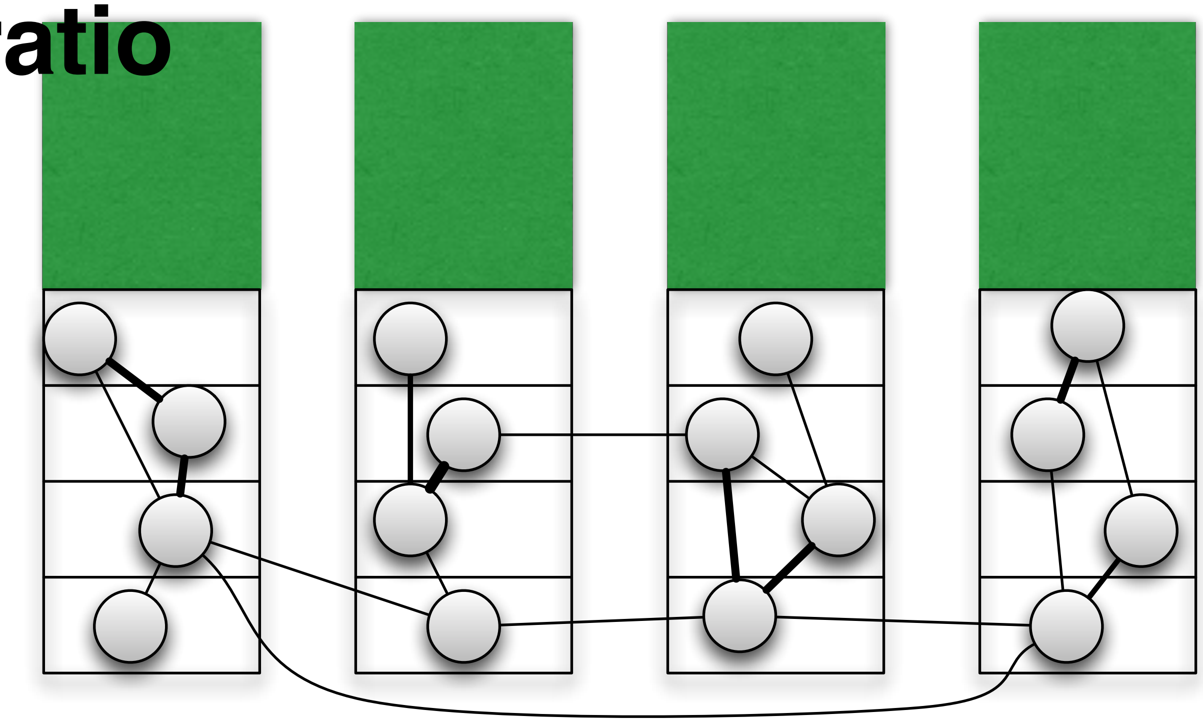
$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

$$\mathrm{ALG}(\sigma) = \sum_{t=1}^{|\sigma|} \mathrm{mig}(\sigma_t; \mathrm{ALG}) + \mathrm{com}(\sigma_t; \mathrm{ALG})$$

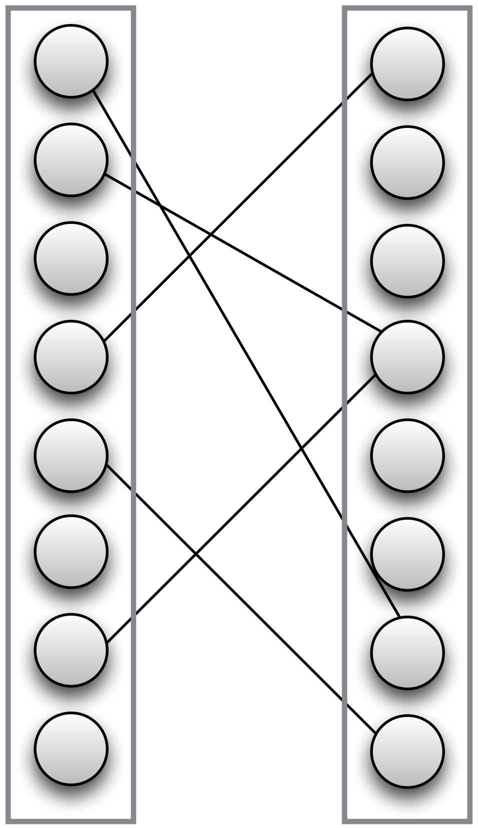
- What is the **competitive ratio**

$$\rho(\textsc{On}) = \max_{\sigma} \frac{\textsc{On}(\sigma)}{\textsc{Off}(\sigma)}$$

- w/o **Augmentation**

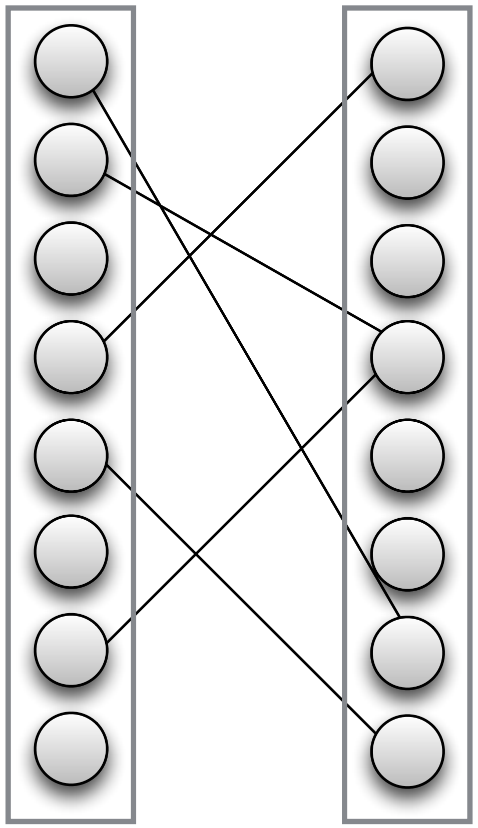


Example I: $\ell = 2$



Example I: $\ell = 2$

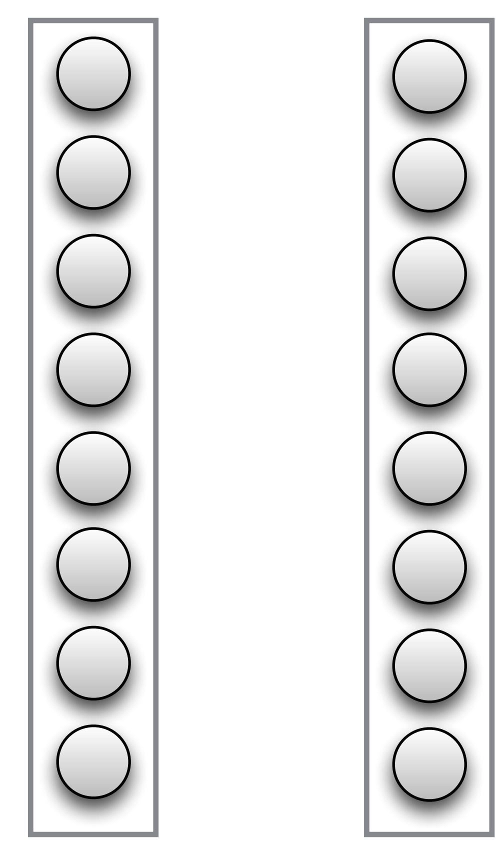
- The static variant corresponds to the minimum bisection problem - hard, but approx



Example I: $\ell = 2$

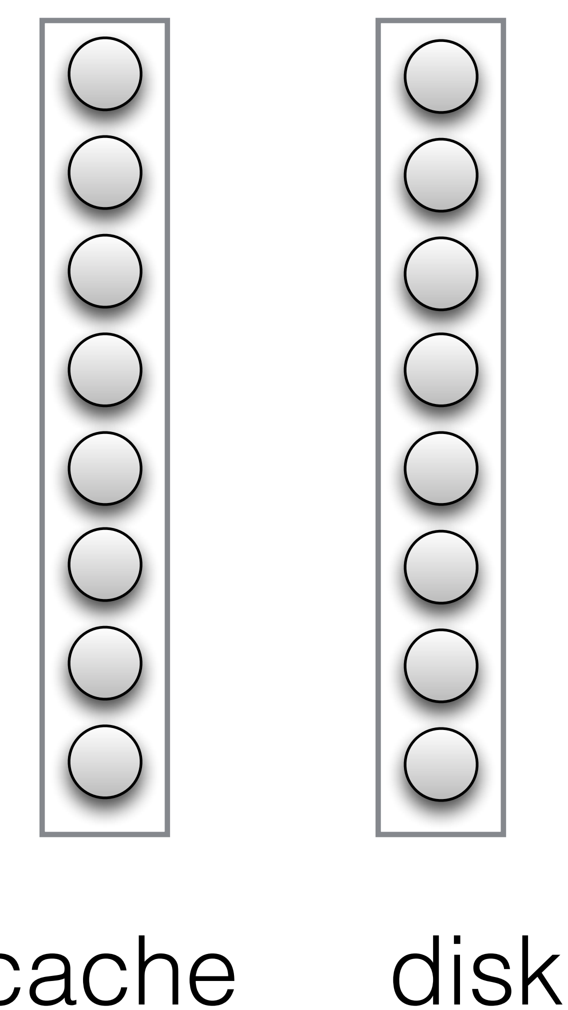
- The static variant corresponds to the minimum bisection problem - hard, but approx

- The dynamic case is a generalization of online paging



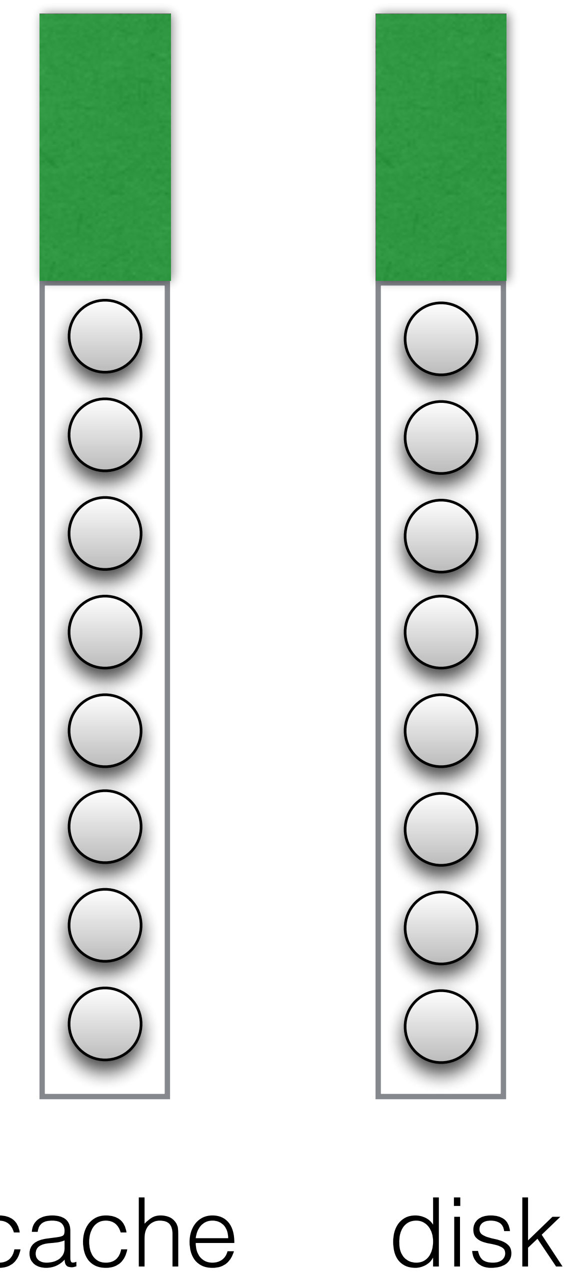
Example I: $\ell = 2$

- The static variant corresponds to the minimum bisection problem - hard, but approx
- The dynamic case is a generalization of online paging
- Imply k lower bound (deterministic)

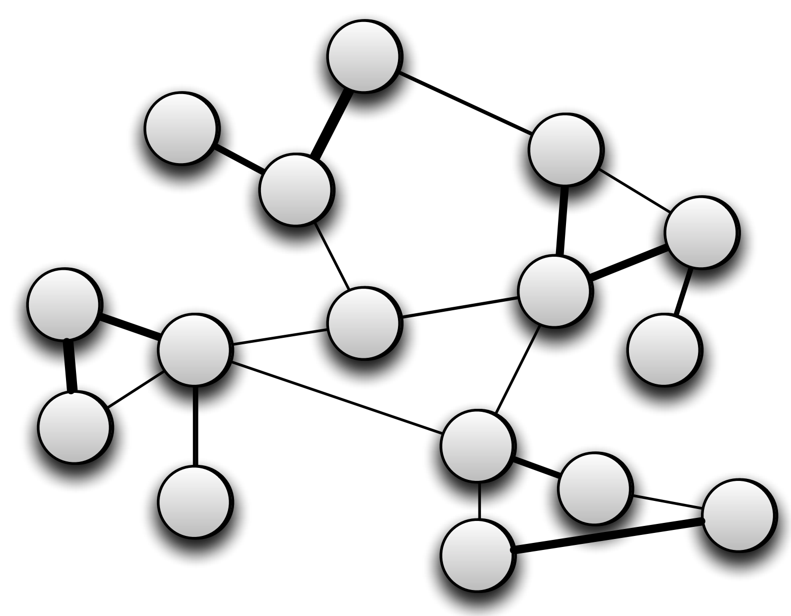


Example I: $\ell = 2$

- The static variant corresponds to the minimum bisection problem - hard, but approx
- The dynamic case is a generalization of online paging
- Imply k lower bound (deterministic)
- With augmentation it's different….

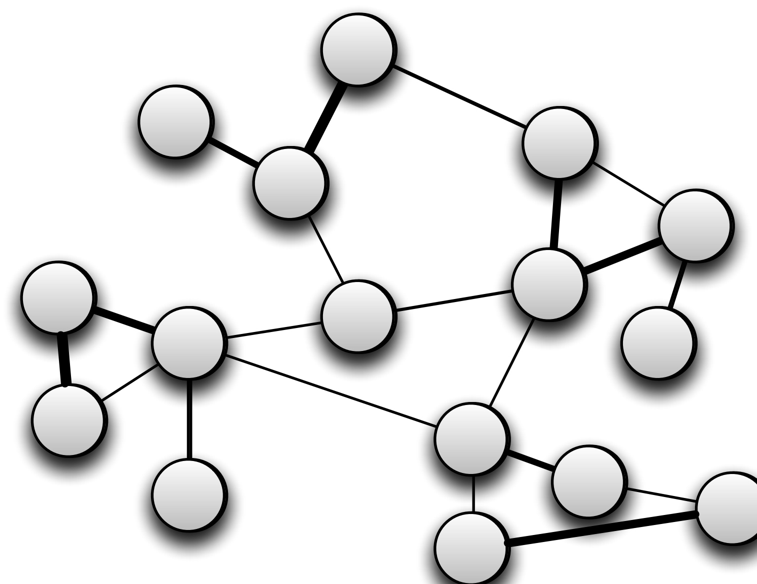


Example II: $k = 2$



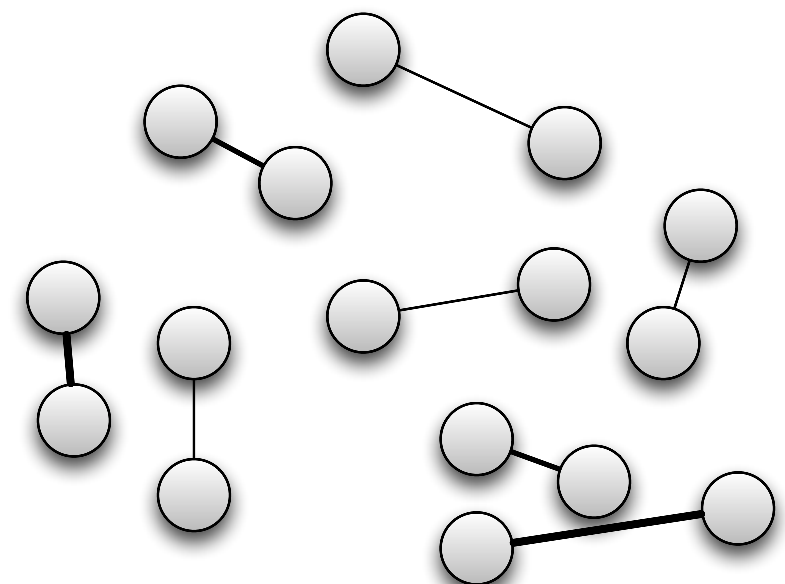
Example II: $k = 2$

- The iid variant corresponds to the maximum matching problem (minimum cut)



Example II: $k = 2$

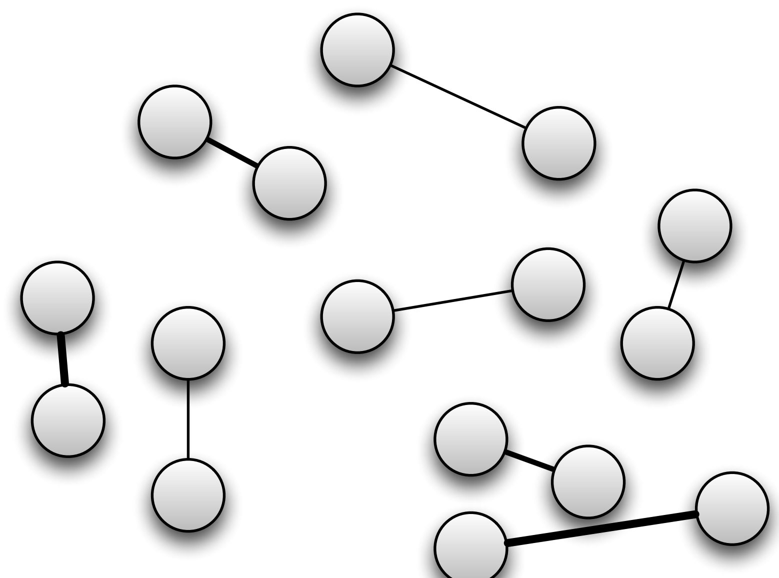
- The iid variant corresponds to the maximum matching problem (minimum cut)



Example II: $k = 2$

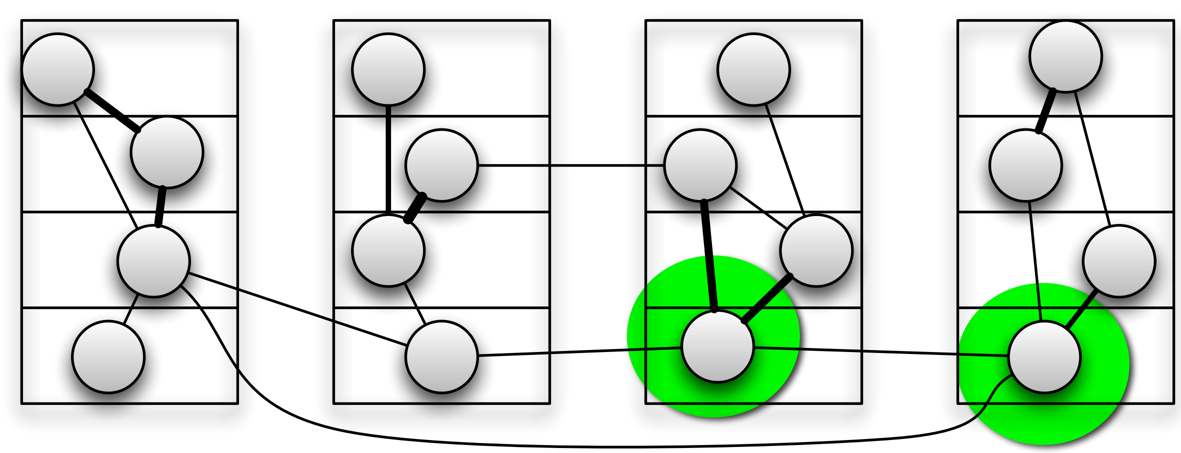
- The iid variant corresponds to the maximum matching problem (minimum cut)

- A novel online version of maximum matching



Algo Guidelines

- Serve remotely or migrate (``rent or buy'')?
- Where to migrate, and what?
- Which nodes to evict?



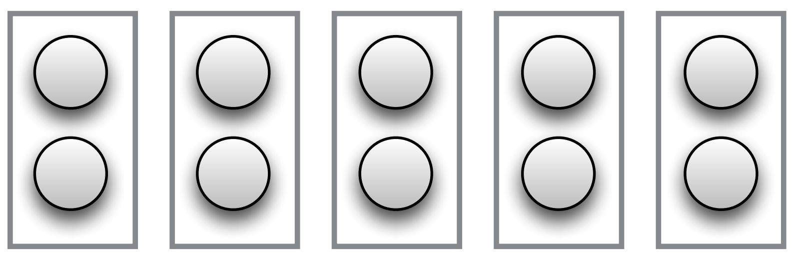
Related work

- Similar in spirit to many classical on-line problems:
  - ski rental, page and server migration, k-server, caching, bin packing
- However, does not fit to the online metrical task system scenario
  - both ends of the communication requests can move
  - every request only reveals partial and and limited information about the optimal configuration
  - large space
- Caching models *with bypassing*

Results overview

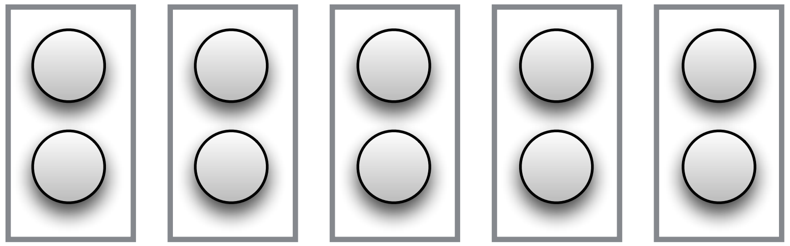
- Bounds for **deterministic** algorithms
- k=2 - constant competitive bound
- Lower bound (with augmentation) - $\Omega(k)$
- Upper bound (with augmentation) - $O(k \log k)$

$$k = 2$$



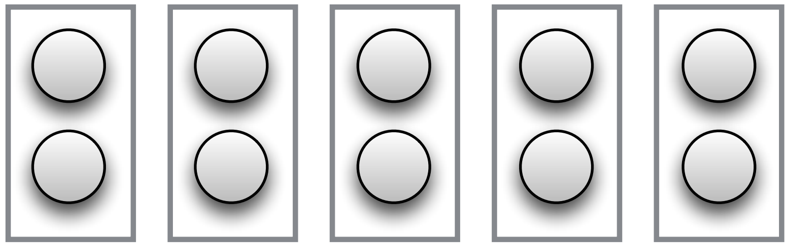
$k = 2$

- We show a lower bound of 3-competitive



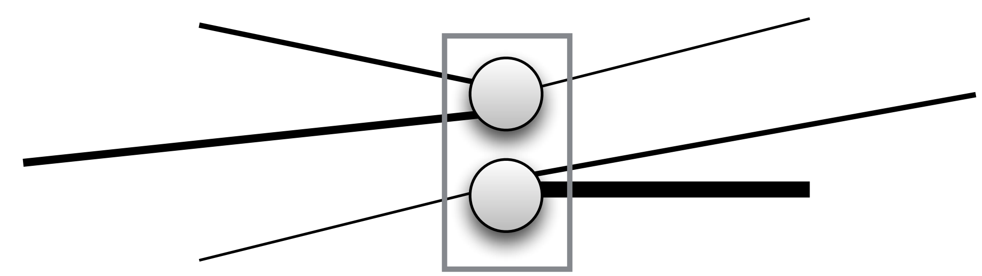
$$k = 2$$

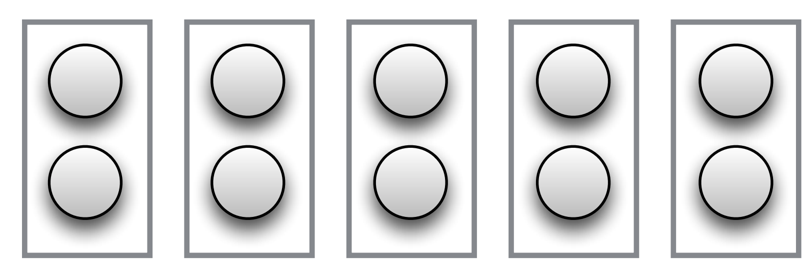
- We show a lower bound of 3-competitive
- No eviction problem 😀



$k = 2$

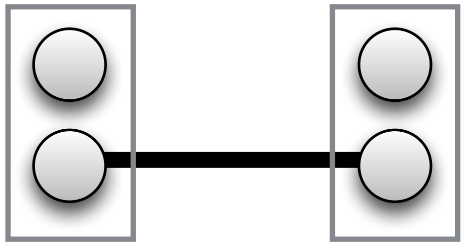
- We show a lower bound of 3-competitive
- No eviction problem 😀
- A **greedy** algorithm:

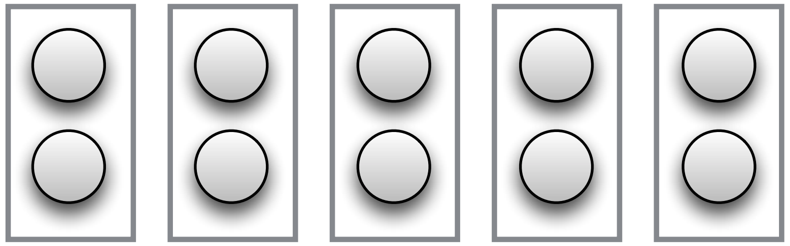




$$k = 2$$

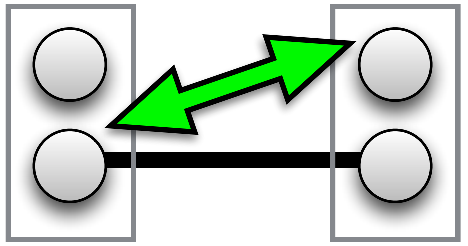
- We show a lower bound of 3-competitive
- No eviction problem 😀
- A **greedy** algorithm:
  - When outside traffic > 3α

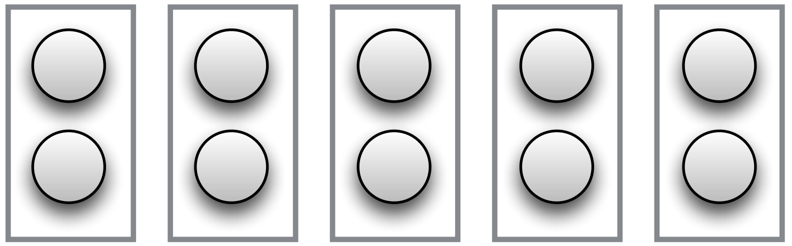




$k = 2$

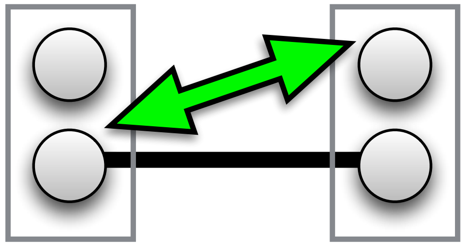
- We show a lower bound of 3-competitive
- No eviction problem 😀
- A **greedy** algorithm:
    - When outside traffic > 3α

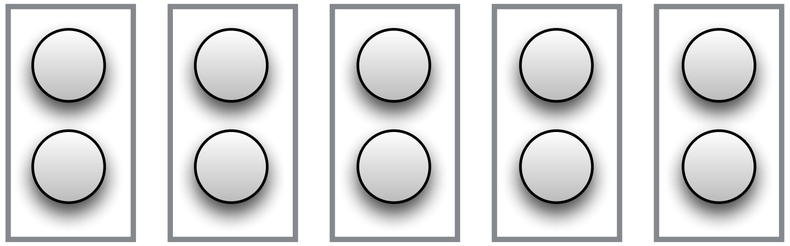




$k = 2$

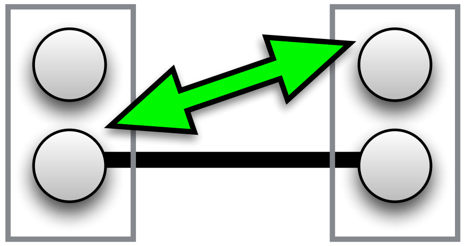
- We show a lower bound of 3-competitive
- No eviction problem 😀
- A **greedy** algorithm:
    - When outside traffic > 3α
    - Identify and migrate to best cluster

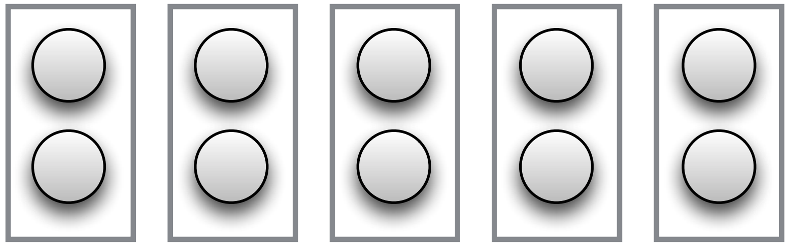




$k = 2$

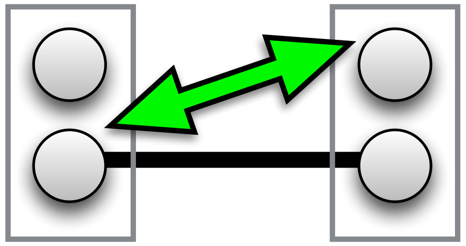
- We show a lower bound of 3-competitive
- No eviction problem 😀
- A **greedy** algorithm:
  - When outside traffic > 3α
  - Identify and migrate to best cluster
- An upper bound of 7α

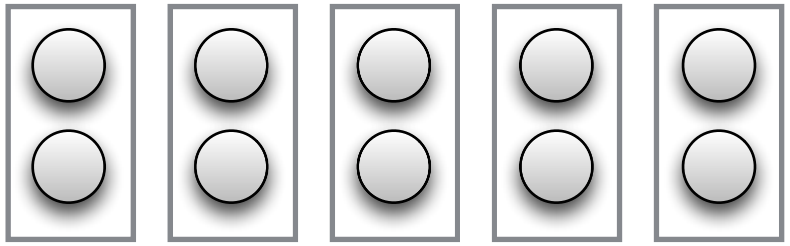




$k = 2$

- We show a lower bound of 3-competitive
- No eviction problem 😀
- A **greedy** algorithm:
  - When outside traffic > 3α
  - Identify and migrate to best cluster
- An upper bound of 7α
- no augmentation





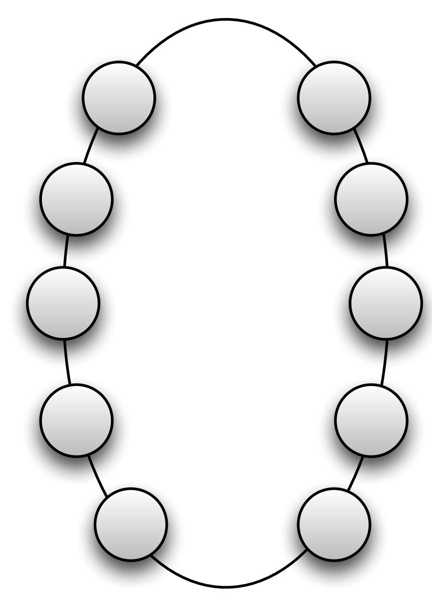
Lower Bound

Lower Bound

- Any non trivial augmentation (all fit in one)

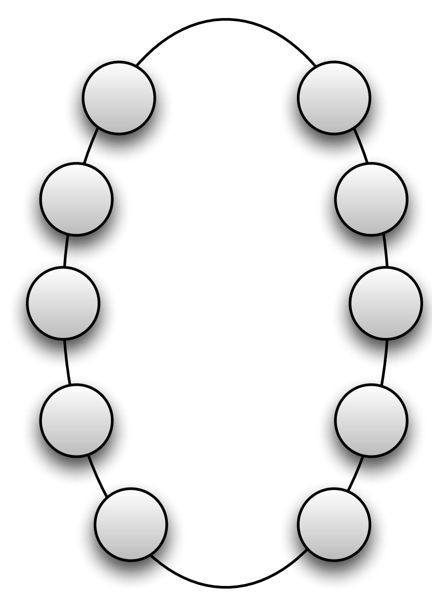
Lower Bound

- Any non trivial augmentation (all fit in one)
- A simple cycle like request sequence



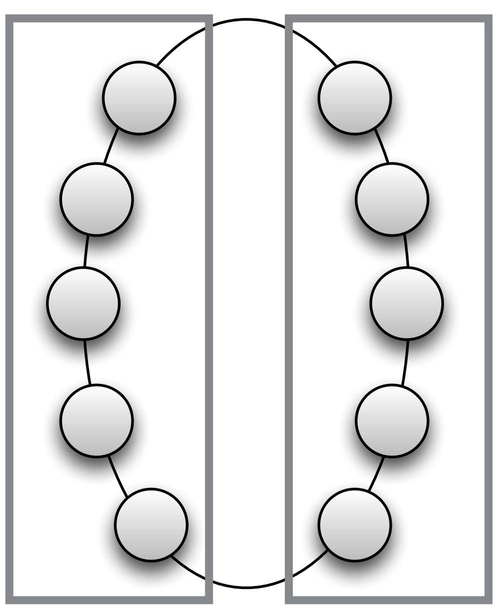
Lower Bound

- Any non trivial augmentation (all fit in one)
- A simple cycle like request sequence
- Always an **inter-cluster** edge to ask



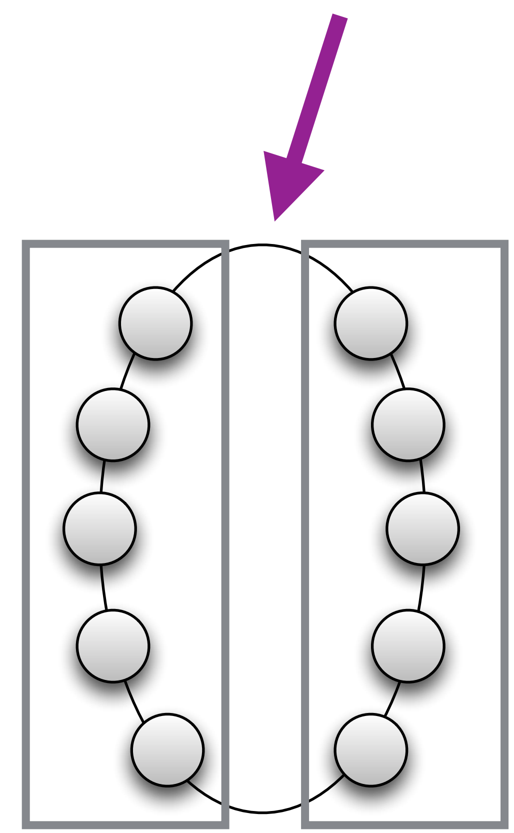
Lower Bound

- Any non trivial augmentation (all fit in one)
- A simple cycle like request sequence
- Always an **inter-cluster** edge to ask



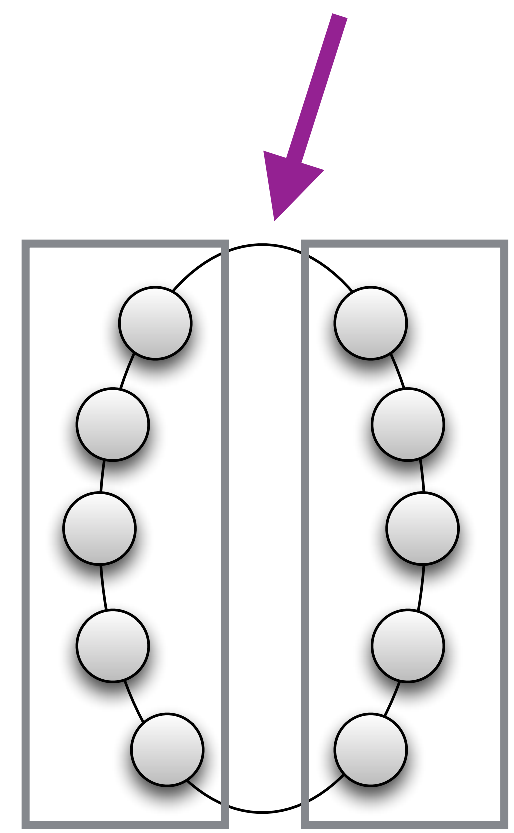
Lower Bound

- Any non trivial augmentation (all fit in one)
- A simple cycle like request sequence
- Always an **inter-cluster** edge to ask

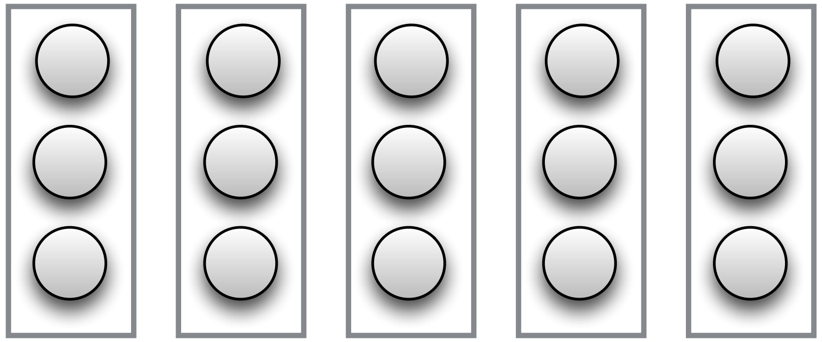


Lower Bound

- Any non trivial augmentation (all fit in one)
- A simple cycle like request sequence
- Always an **inter-cluster** edge to ask
- Leads to a lower bound > k

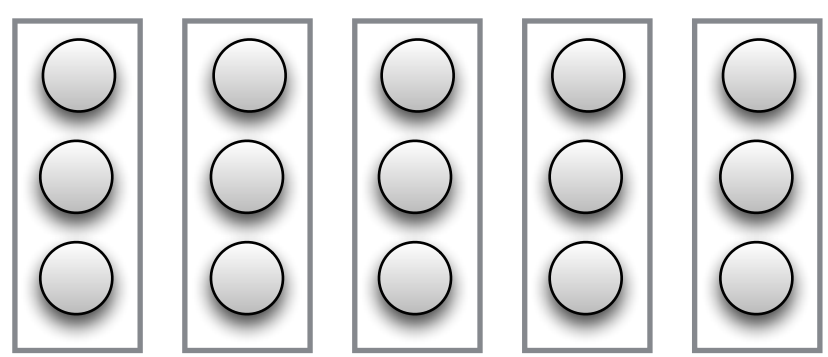


Upper Bound



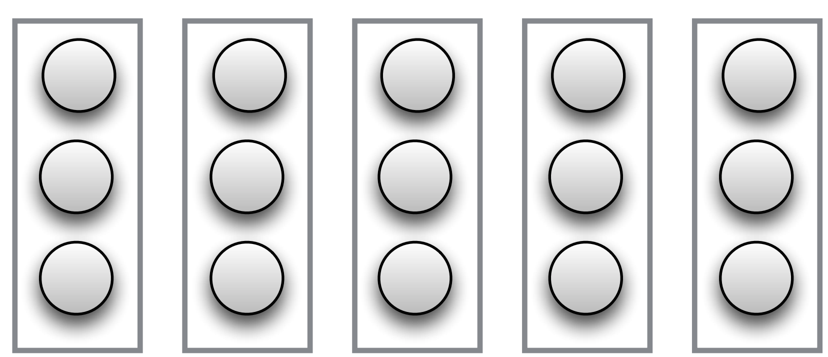
Upper Bound

- C-REP algorithm (Component-based)



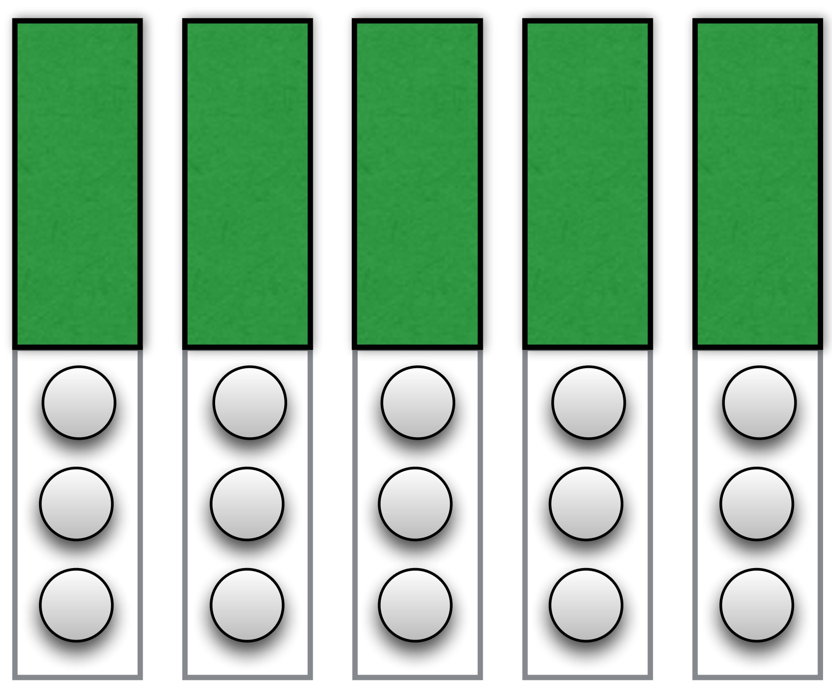
Upper Bound

- C-REP algorithm (Component-based)
- 4 Augmentation



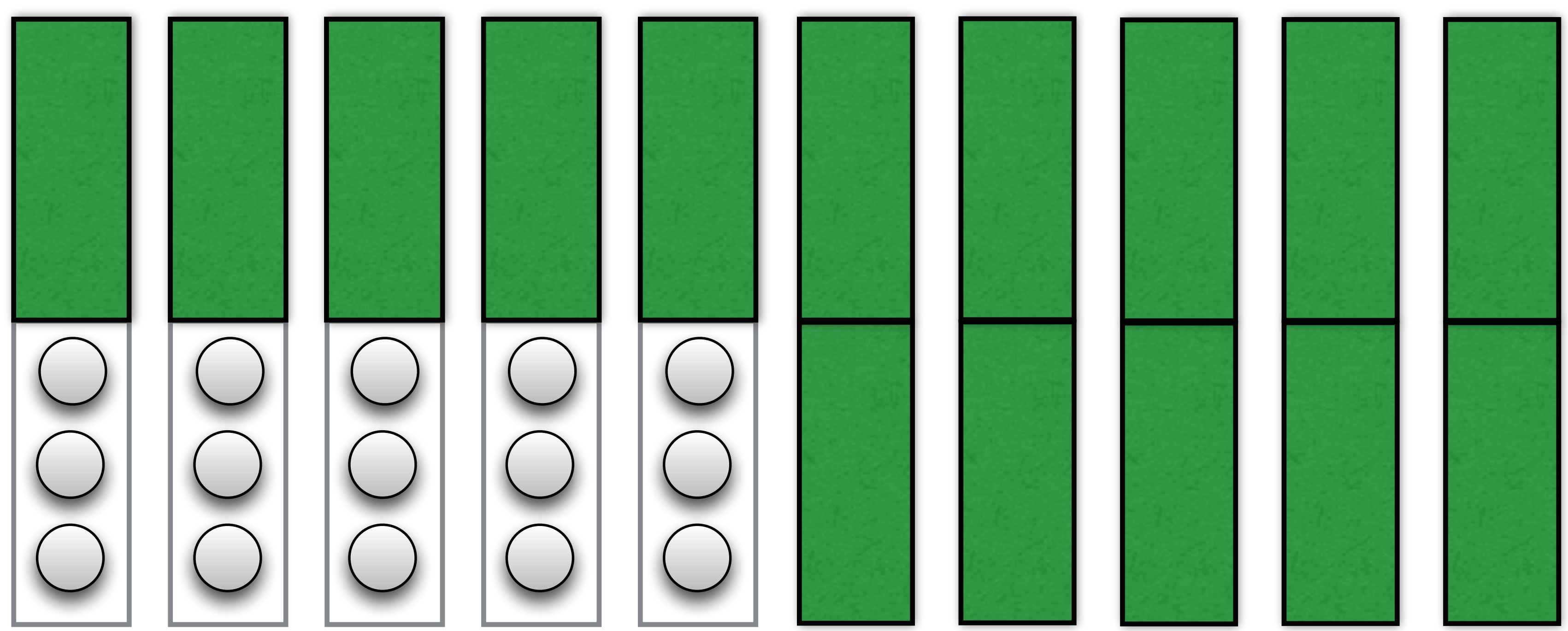
Upper Bound

- C-REP algorithm (Component-based)
- 4 Augmentation



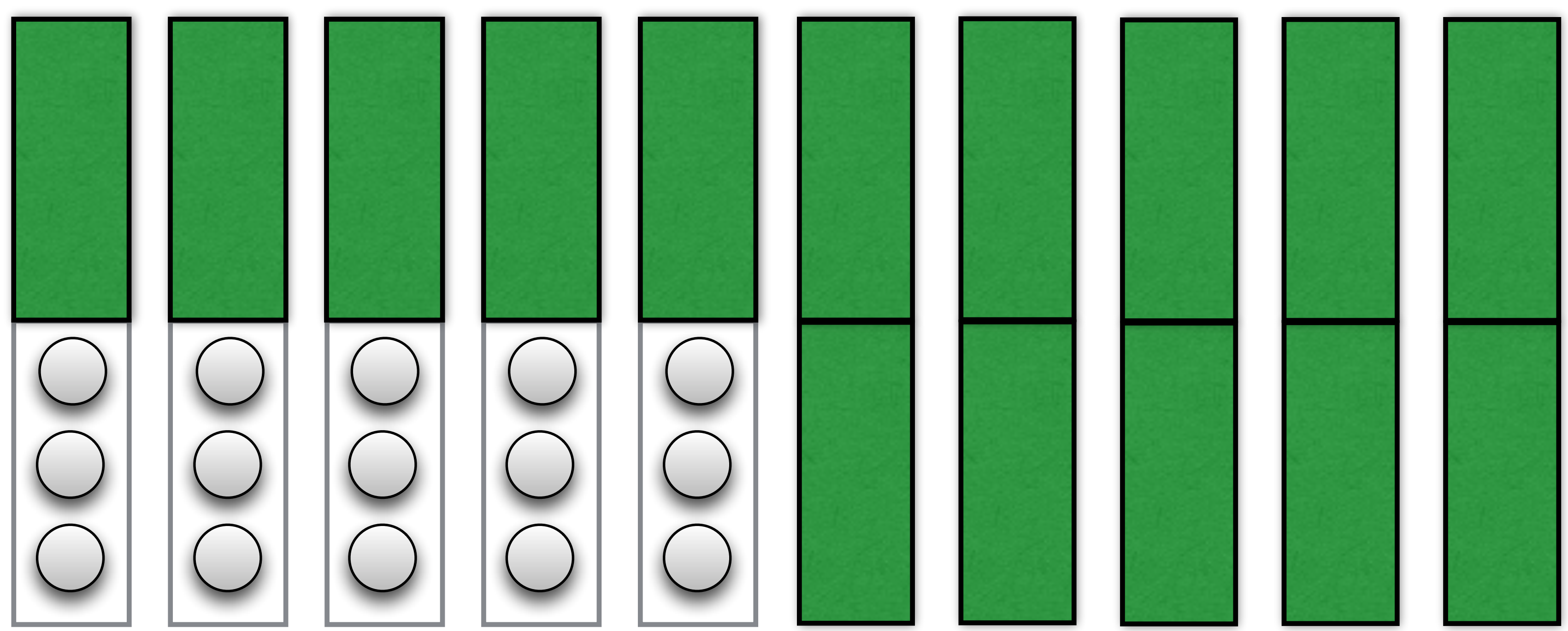
Upper Bound

- C-REP algorithm (Component-based)
- 4 Augmentation

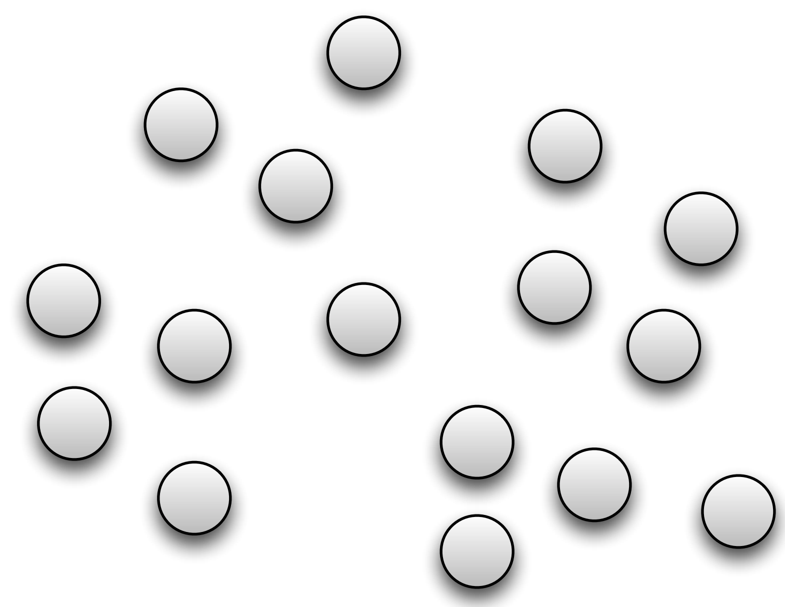


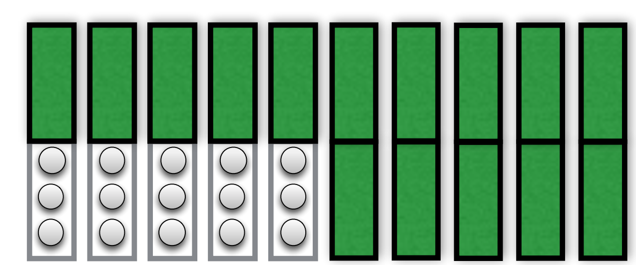
Upper Bound

- C-REP algorithm (Component-based)
- 4 Augmentation
- **Theorem**: CREP is $O(k \log k)$ competitive.



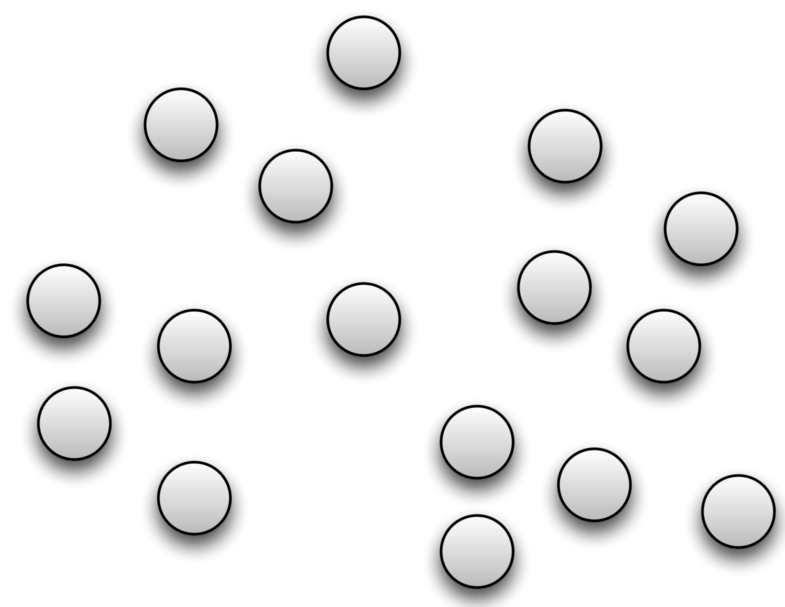
Component based

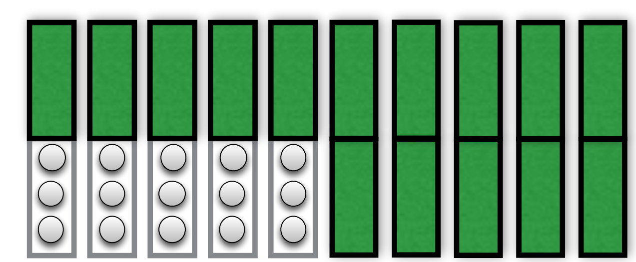




Component based

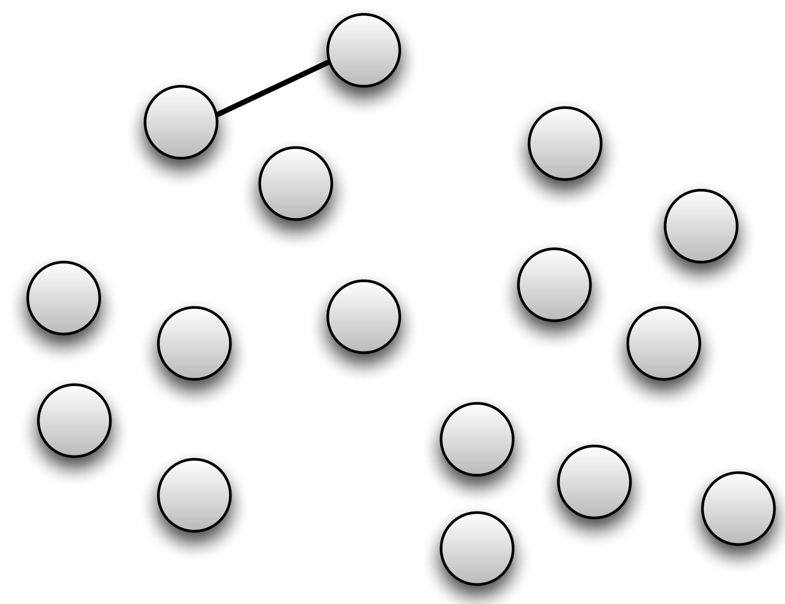
- Communication components

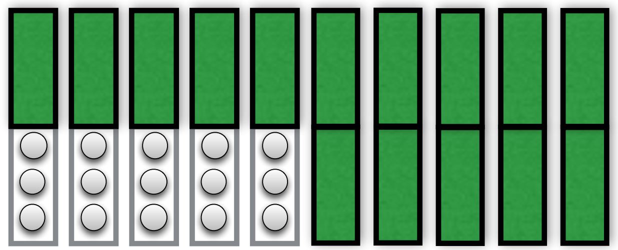




Component based

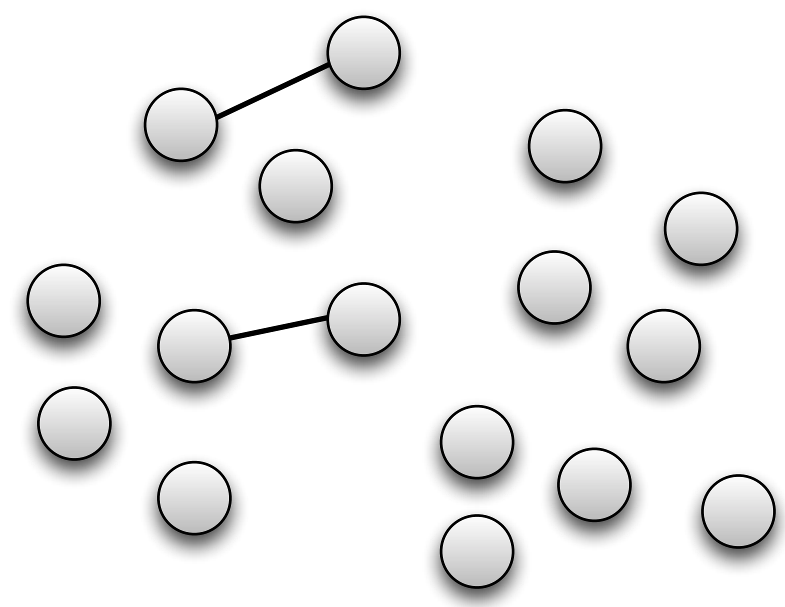
- Communication components

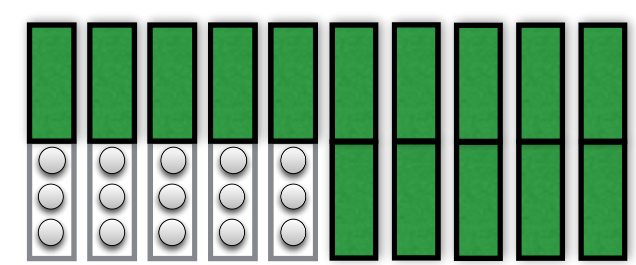




Component based

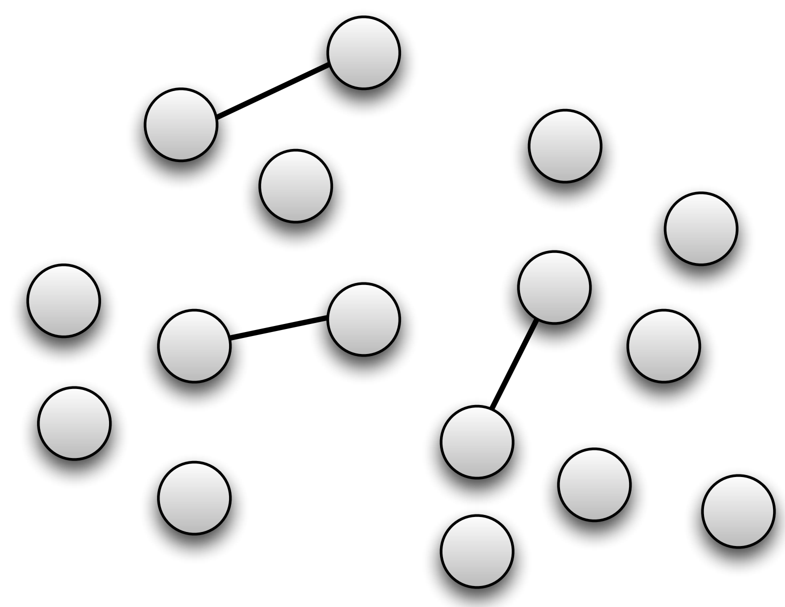
- Communication components

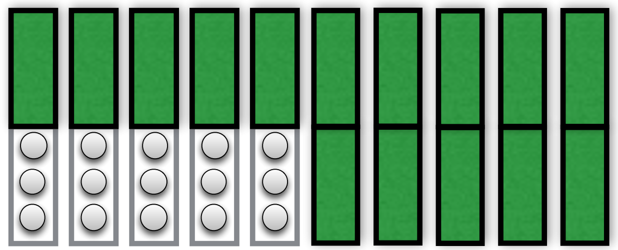




Component based

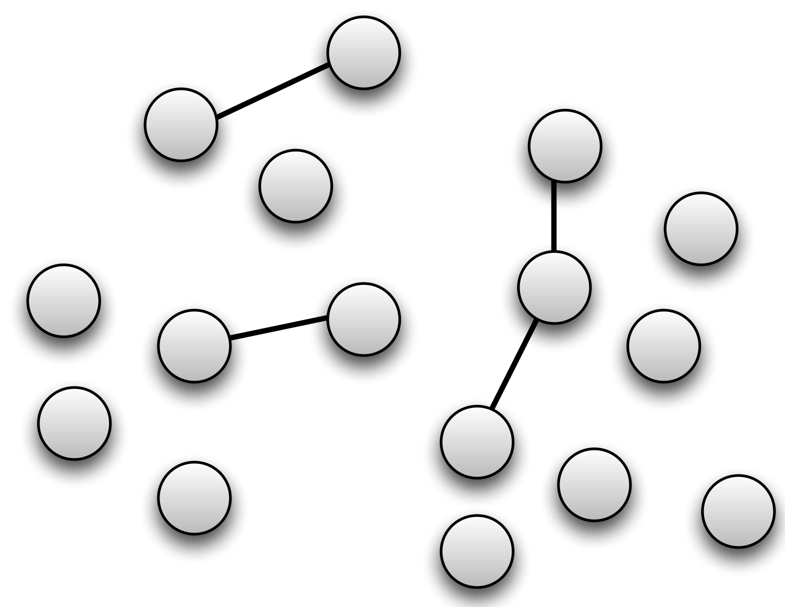
- Communication components

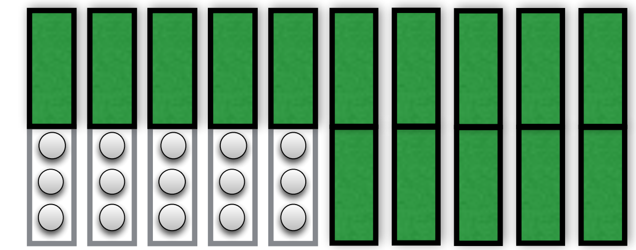




Component based

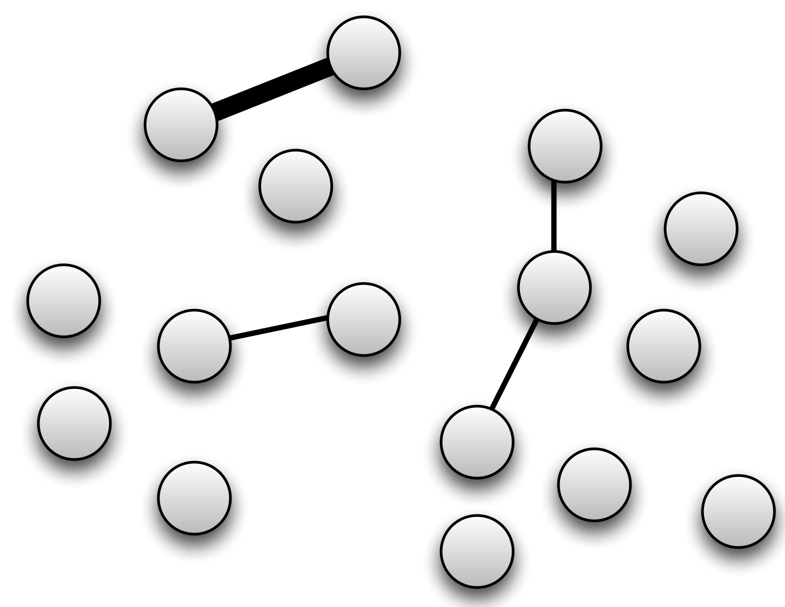
- Communication components

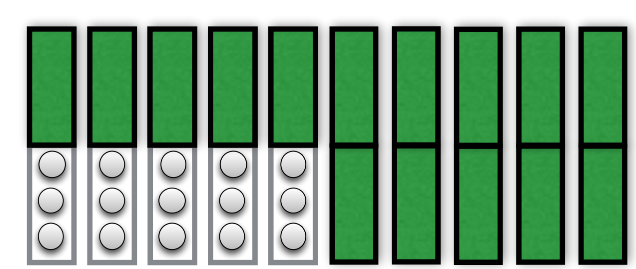




Component based

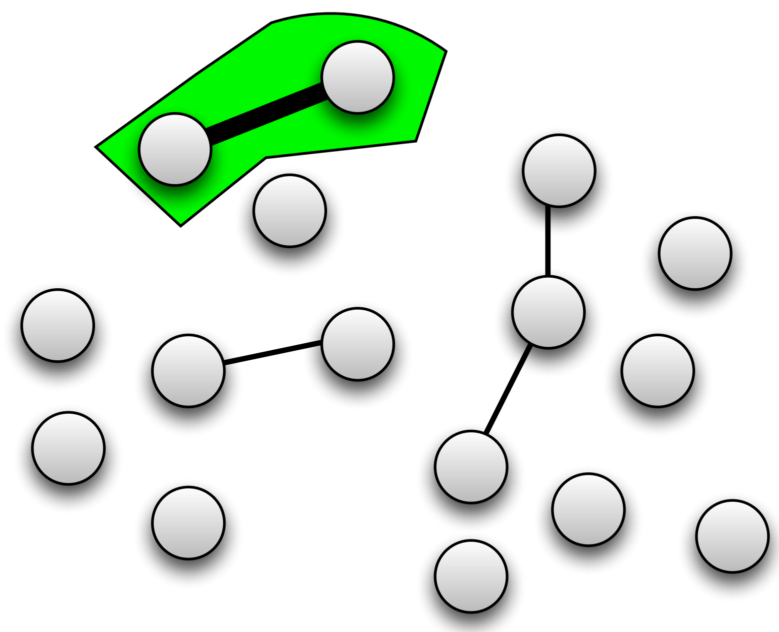
- Communication components

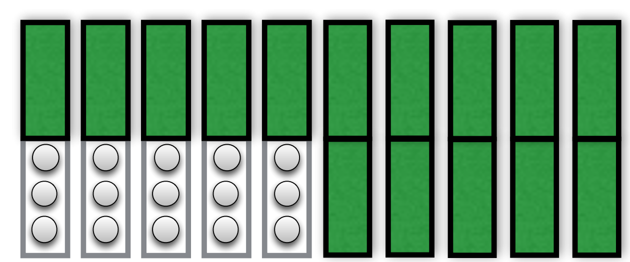




Component based

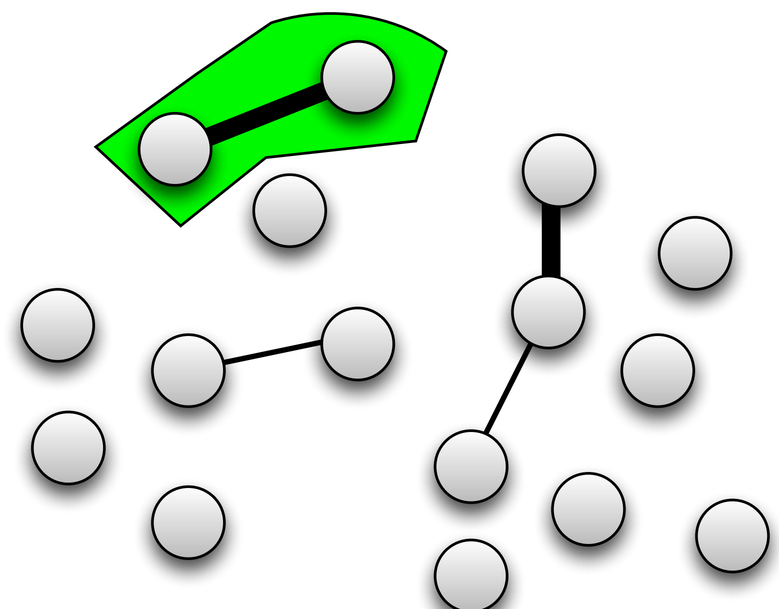
- Communication components

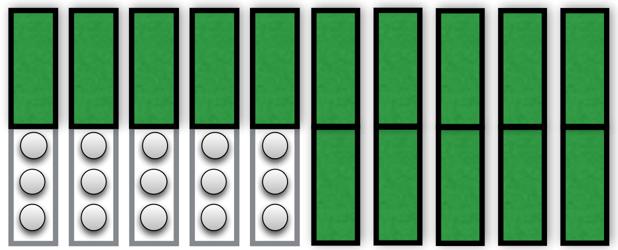




Component based

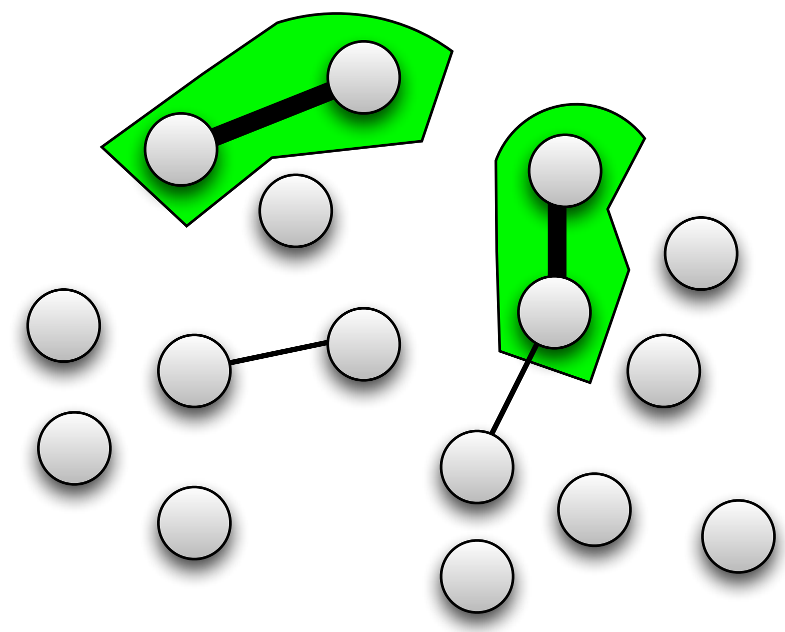
- Communication components

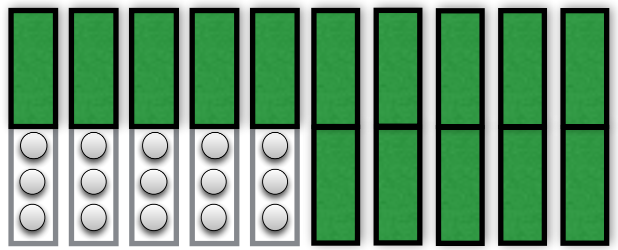




Component based

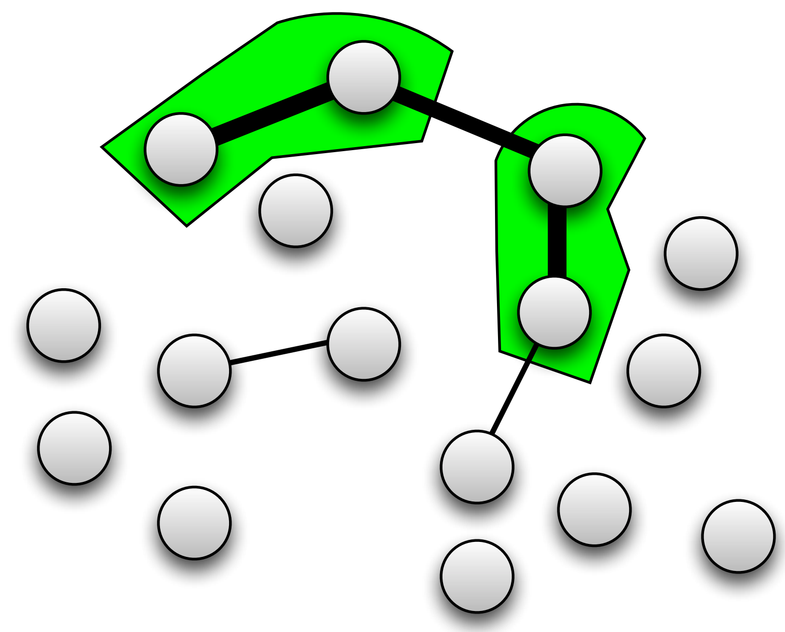
- Communication components

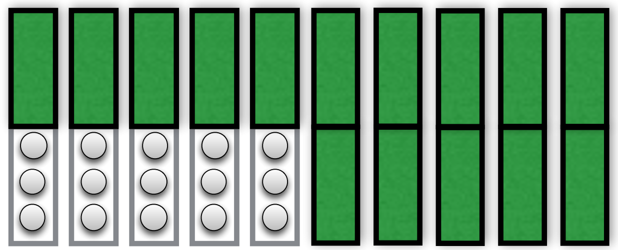




Component based

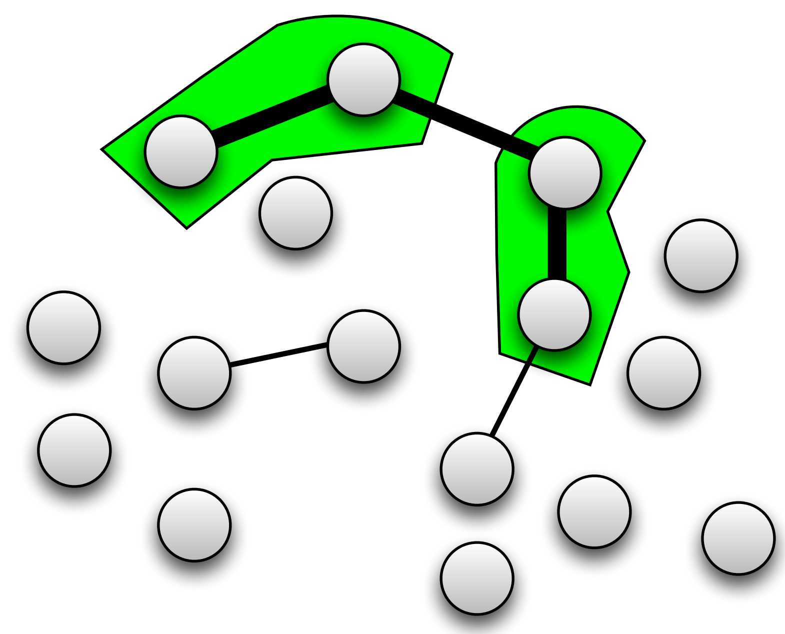
- Communication components

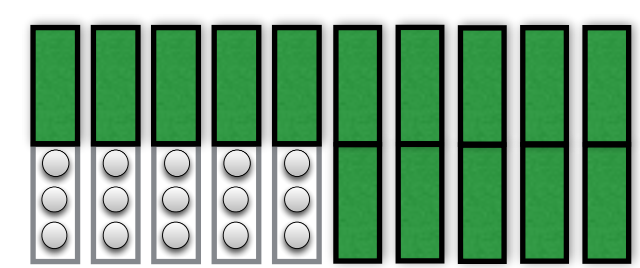




Component based

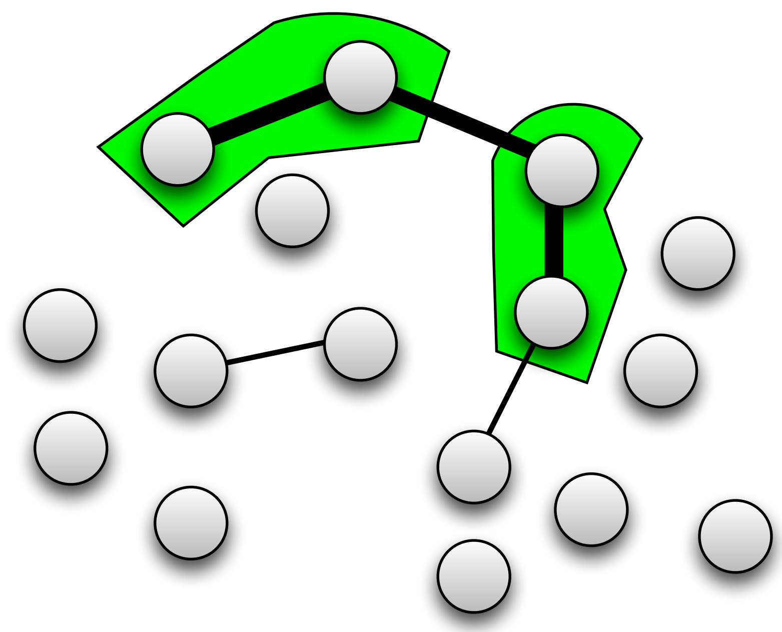
- Communication components
- Merge while you can

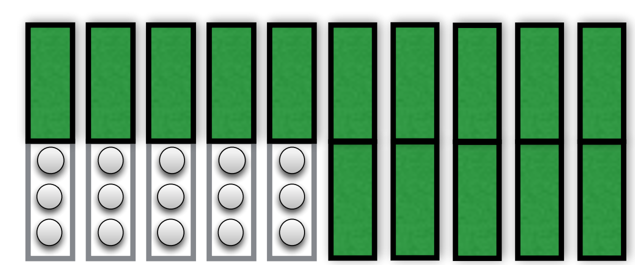




Component based

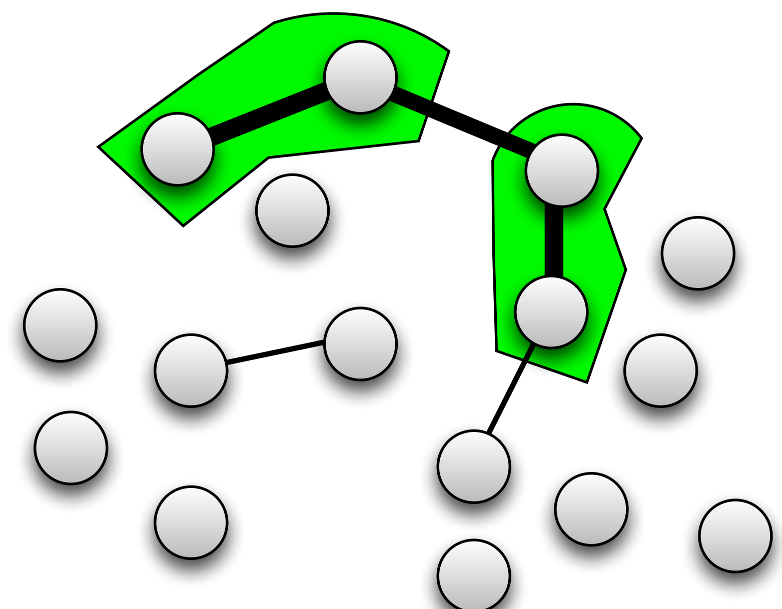
- Communication components
- Merge while you can
- Small-to-large component

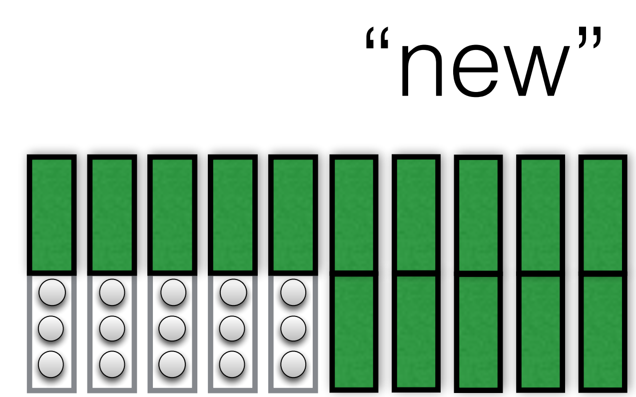




Component based

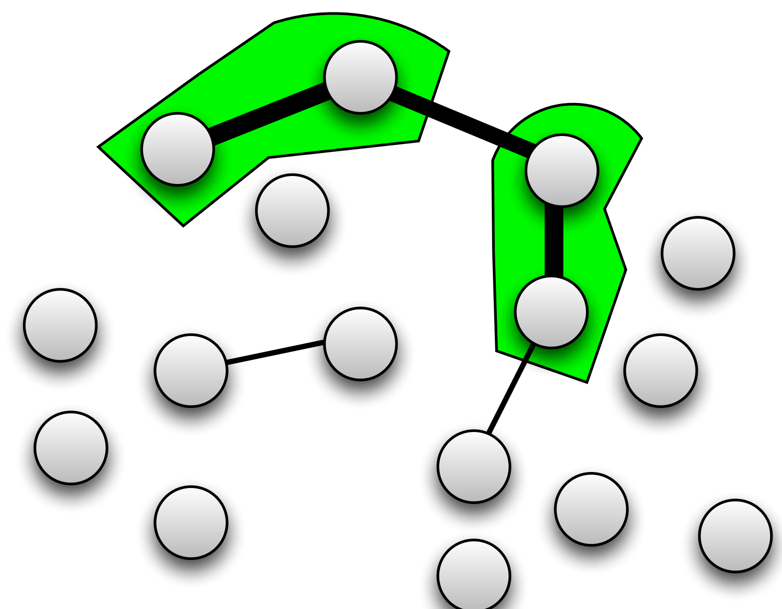
- Communication components
- Merge while you can
- Small-to-large component
- Only move once to a “new” cluster

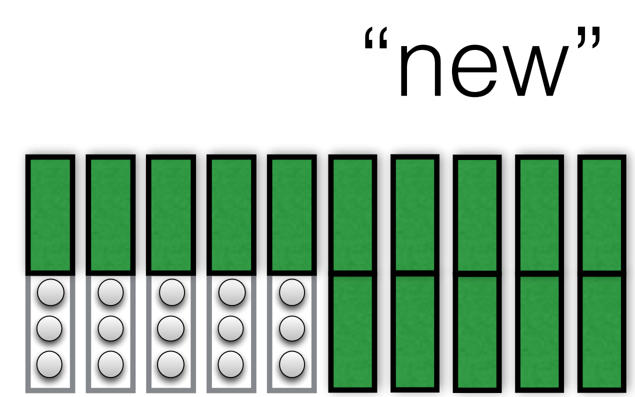




Component based

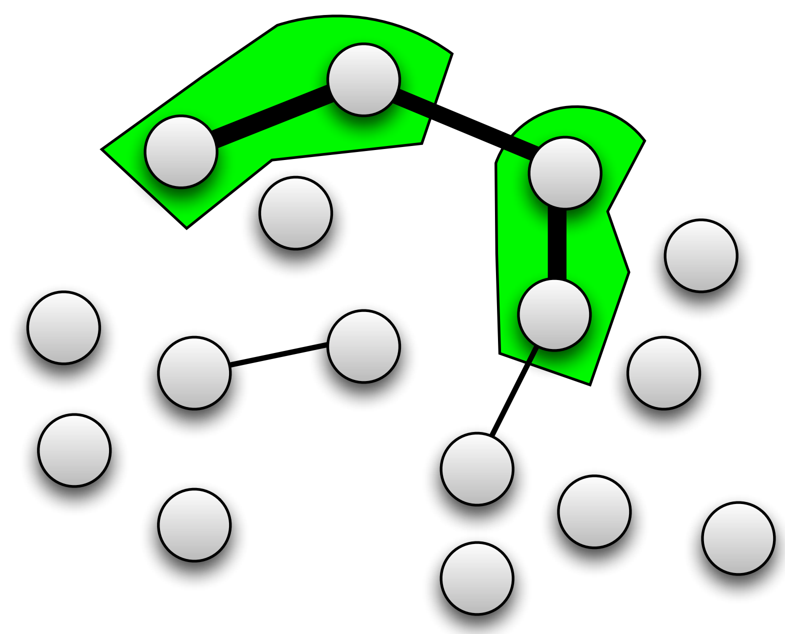
- Communication components
- Merge while you can
- Small-to-large component
- Only move once to a “new” cluster
- Component larger than k we can safely charge OFF



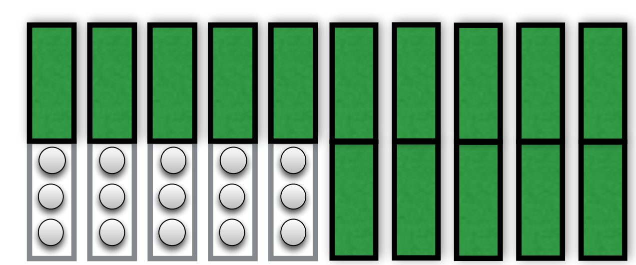


Component based

- Communication components
- Merge while you can
- Small-to-large component
- Only move once to a “new” cluster
- Component larger than k we can safely charge OFF
- Epoch ends…. split cluster to singletons

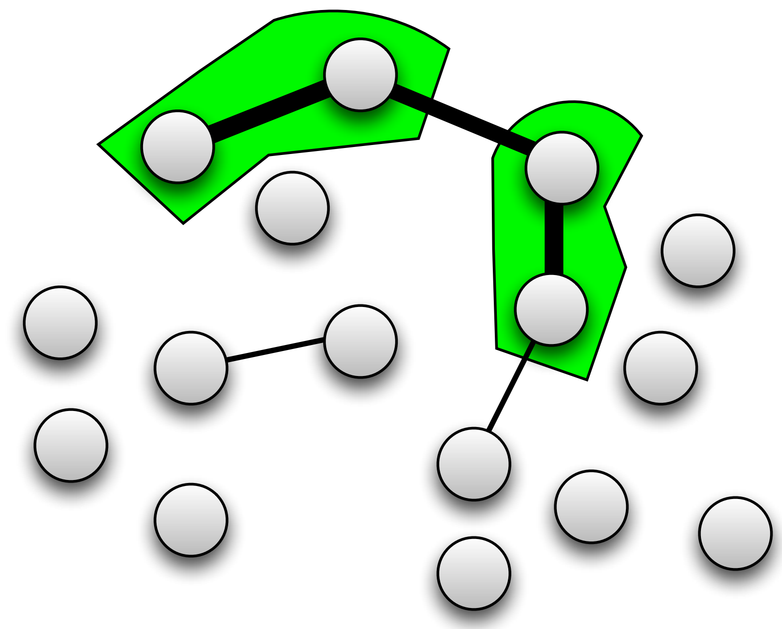


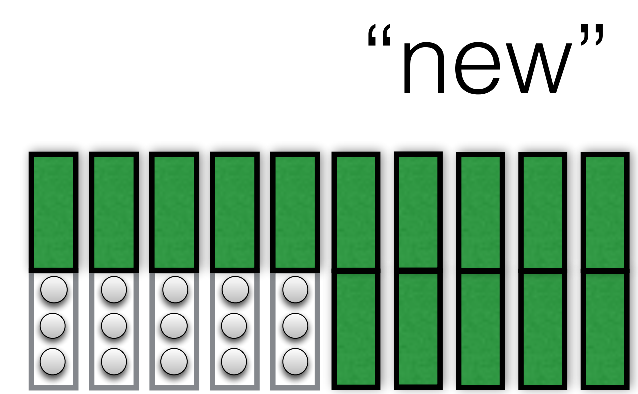
“new”



Component based

- Communication components
- Merge while you can
- Small-to-large component
- Only move once to a “new” cluster
- Component larger than k we can safely charge OFF
- Epoch ends…. split cluster to singletons
- Need to be careful on the condition to merge





Algorithm 1 CREP with 4 Augmentation

Construct graph $G = (\Psi, E, w)$ with singleton components: one component per node. Set $w_{ij} = 0$ for all $\{v_i, v_j\} \in \binom{V}{2}$. For each component ϕ_i, reserve space reserve$(\phi_i) = 1$.

for each new request $\{u_t, v_t\}$ **do**

▷ Keep track of communication cost.

  Let $\phi_i = \Phi(u_t)$ and $\phi_j = \Phi(v_t)$ be the two components that communicated.
  if $\phi_i \neq \phi_j$ **then**
    $w_{ij} \leftarrow w_{ij} + 1$
  end if

▷ Merge components.

  Let X be the largest cardinality set with $\mathrm{vol}(X) \leq k$ and $\mathrm{com}(X) \geq (|X| - 1) \cdot \alpha$
  if $|X| > 1$ **then**
    Let $\phi_0 = \bigcup_{\phi_i \in X} \phi_i$ and for all $\phi_j \in \Phi \backslash X$ set $w_{0j} = \sum_{\phi_i \in X} w_{ij}$.
    Let $\phi \in X$ be the component having the largest reserved space.
    if reserved$(\phi) \geq \mathrm{vol}(X) - |\phi|$ **then**
      Migrate ϕ_0 to the cluster hosting ϕ
      Update reserved$(\phi_0) =$ reserved$(\phi) - (\mathrm{vol}(X) - |\phi|)$
    else
      Migrate ϕ_0 to a cluster s with spare$(s) \geq \min(k, 2|\phi_0|)$
      Set reserved$(\phi_0) = \min(k - |\phi_0|, |\phi_0|)$
    end if
  end if

▷ End of a Y-epoch.

  Let Y be the smallest components set with $\mathrm{vol}(Y) > k$ and $\mathrm{com}(Y) \geq \mathrm{vol}(Y) \cdot \alpha$
  if $Y \neq \emptyset$ **then**
    Split every $\phi_i \in Y$ into ϕ_i singleton components and reset the weights of all edges involving at least one newly created component. Reserve one additional space for each newly created component. If necessary, migrate at most $\mathrm{vol}(Y)/2 + 1$ singletons to clusters with spare space.
  end if
end for

Algorithm 1 CREP with 4 Augmentation

Construct graph $G = (\Psi, E, w)$ with singleton components: one component per node. Set $w_{ij} = 0$ for all $\{v_i, v_j\} \in \binom{V}{2}$. For each component ϕ_i, reserve space reserve$(\phi_i) = 1$.

for each new request $\{u_t, v_t\}$ **do**

$\qquad\qquad$ ▷ Keep track of communication cost.

- Let $\phi_i = \Phi(u_t)$ and $\phi_j = \Phi(v_t)$ be the two components that communicated.
- **if** $\phi_i \neq \phi_j$ **then**
  - $w_{ij} \leftarrow w_{ij} + 1$
- **end if**

$\qquad\qquad$ ▷ Merge components.

- Let X be the largest cardinality set with vol$(X) \leq k$ and com$(X) \geq (|X| - 1) \cdot \alpha$
- **if** $|X| > 1$ **then**
  - Let $\phi_0 = \bigcup_{\phi_i \in X} \phi_i$ and for all $\phi_j \in \Phi \setminus X$ set $w_{0j} = \sum_{\phi_i \in X} w_{ij}$.
  - Let $\phi \in X$ be the component having the largest reserved space.
  - **if** reserved$(\phi) \geq$ vol$(X) - |\phi|$ **then**
    - Migrate ϕ_0 to the cluster hosting ϕ
    - Update reserved(ϕ_0) = reserved$(\phi) -$ (vol$(X) - |\phi|$)
  - **else**
    - Migrate ϕ_0 to a cluster with spare$(s) \geq \min(k, 2|\phi_0|)$
    - Set reserved$(\phi_0) = \min(k - |\phi_0|, |\phi_0|)$
  - **end if**
- **end if**

$\qquad\qquad$ ▷ End of a Y-epoch.

- Let Y be the smallest components set with vol$(Y) > k$ and com$(Y) \geq$ vol$(Y) \cdot \alpha$
- **if** $Y \neq \emptyset$ **then**
  - Split every $\phi_i \in Y$ into ϕ_i singleton components and reset the weights of all edges involving at least one newly created component. Reserve one additional space for each newly created component. If necessary, migrate at most vol$(Y)/2 + 1$ singletons to clusters with spare space.
- **end if**

end for

illegal slide

Open questions

- Randomize algorithms (lower and upper bounds)
    - Some initial results
- A better network model than one-switch network
- Similar models that fits better in practice (e.g., MapReduce. etc.)
- Open Postdoc position (Beer-Sheva and Berlin) to work on these problems… feel free to talk to me.

Thank you !