The Quorum Deployment Problem

Seth Gilbert

Grzegorz Malewicz

Outline

1. Background on quorum systems
2. The Quorum Deployment Problem
3. Hardness of quorum deployment
4. Special cases

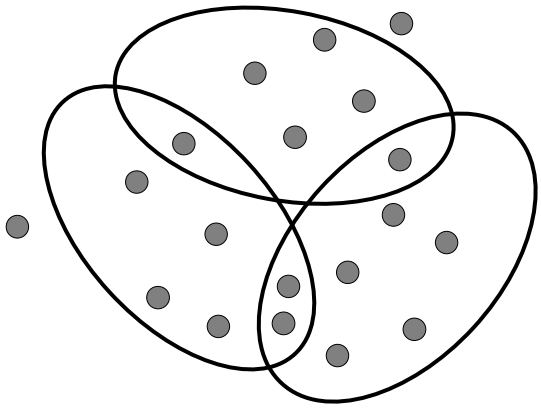
Outline

1. **Background on quorum systems**
2. The Quorum Deployment Problem
3. Hardness of quorum deployment
4. Special cases

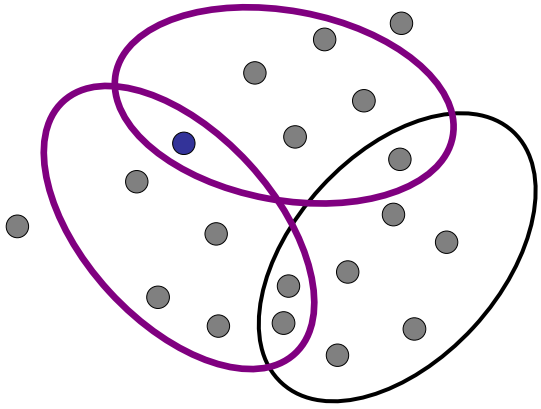
Quorum Systems

- A *quorum system* is a collection of sets where all pairs of sets intersect.
- Notation
  - Q, a quorum system
  - $q \in Q$, a quorum
- Intersection
  - $q_1 \bigcap q_2 \neq \varnothing$

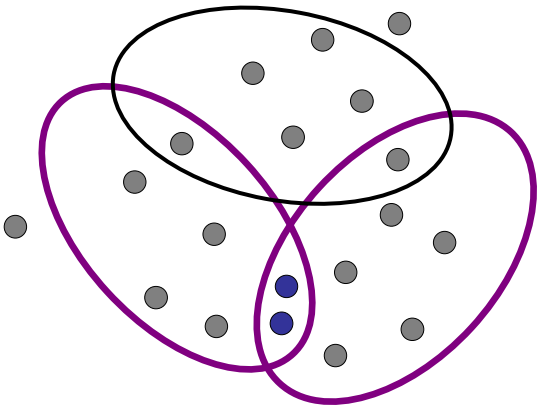
Example Quorum System



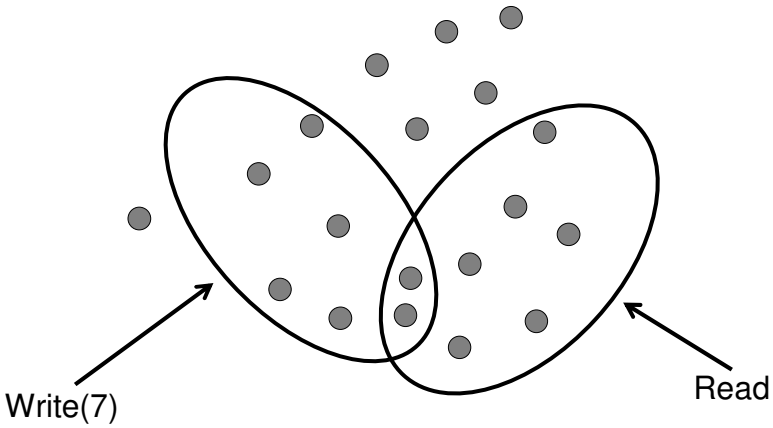
Example Quorum System



Example Quorum System



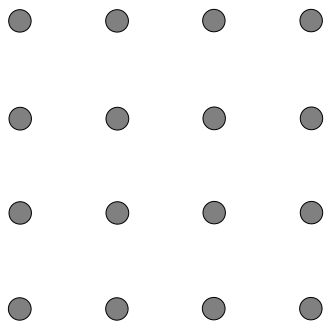
Using Quorums



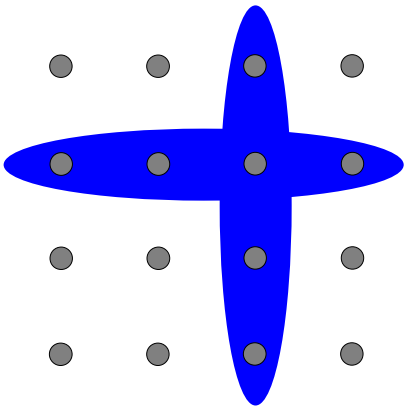
Theoretical Quorum Systems

- Good properties
  - Fault Tolerance
  - Probe Complexity
  - Load
  - And others…
- Much research
  - Grid quorums
  - Hierarchical quorums
  - Paths quorums
  - etc.

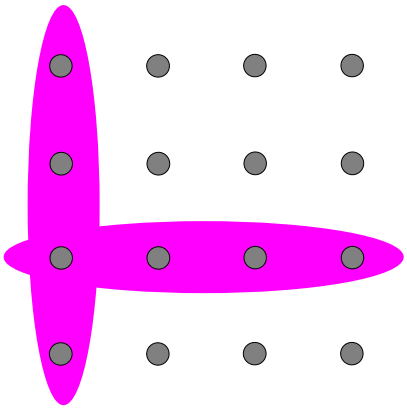
Example: Grid



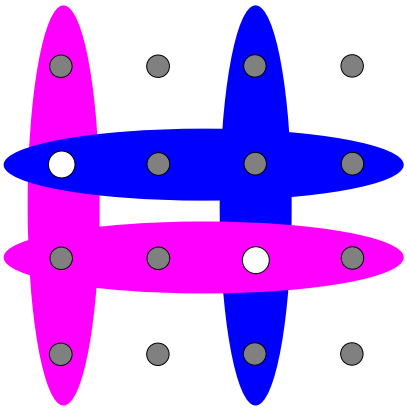
Example: Grid



Example: Grid

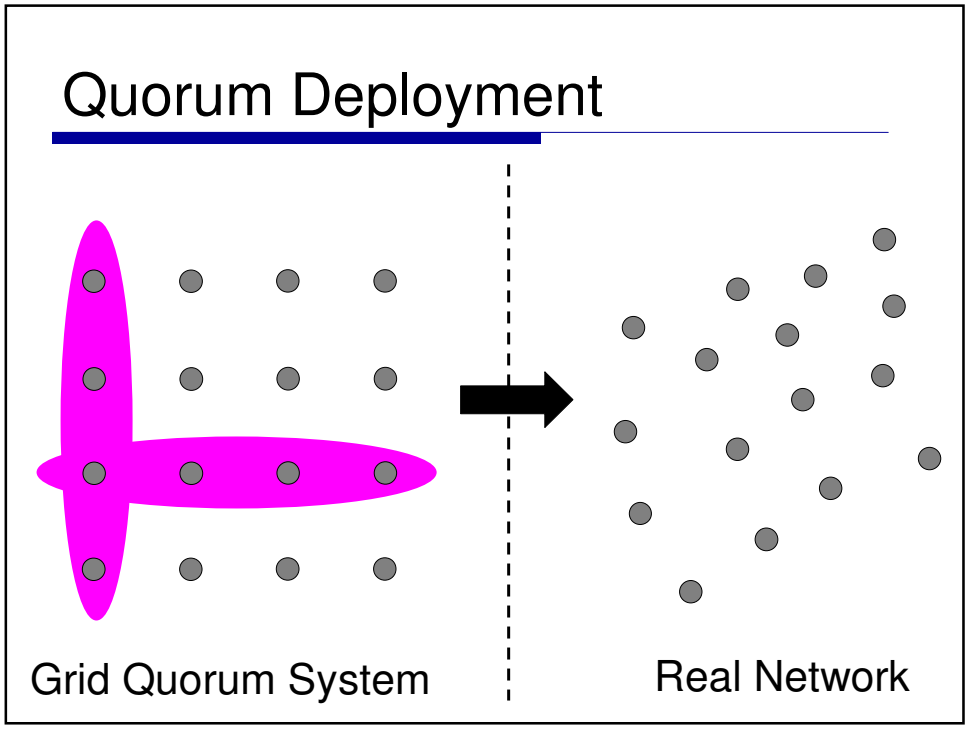


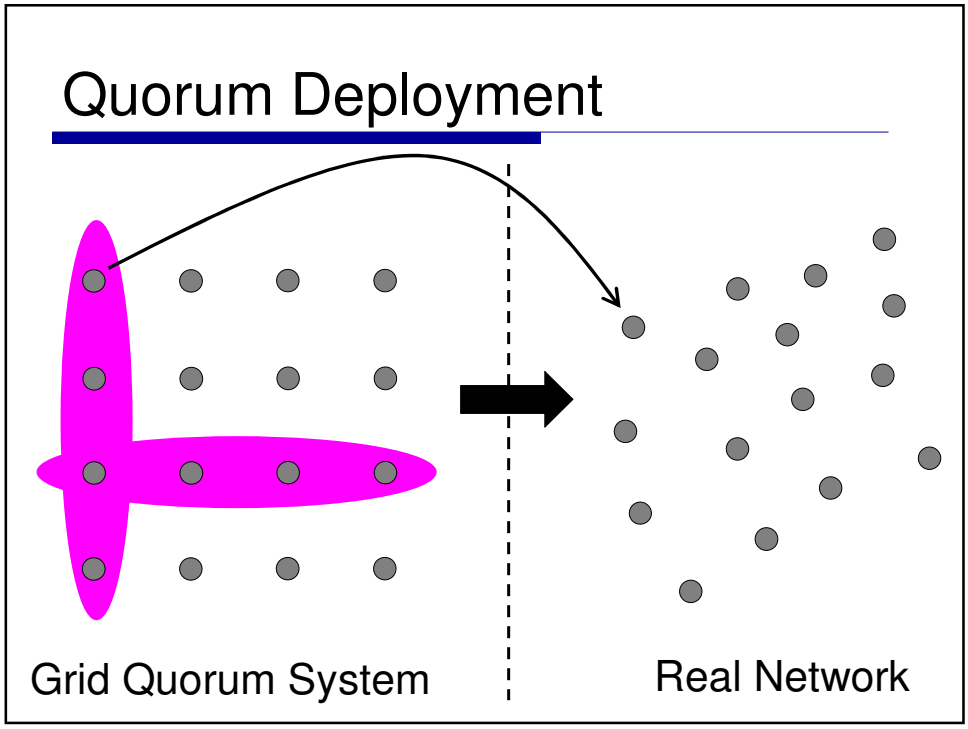
Example: Grid

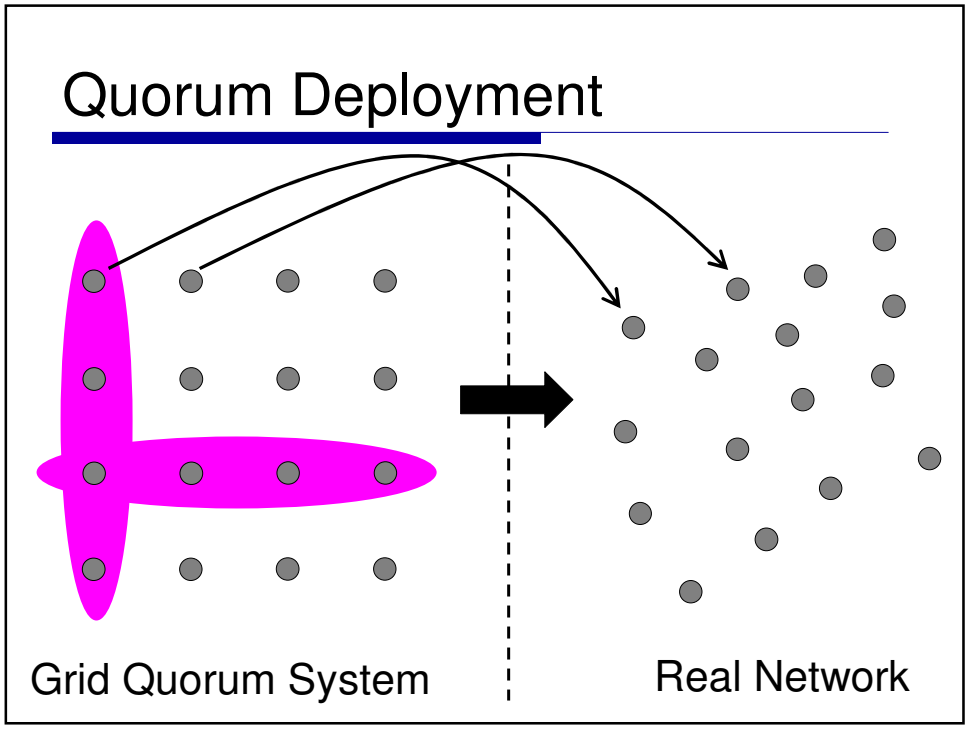


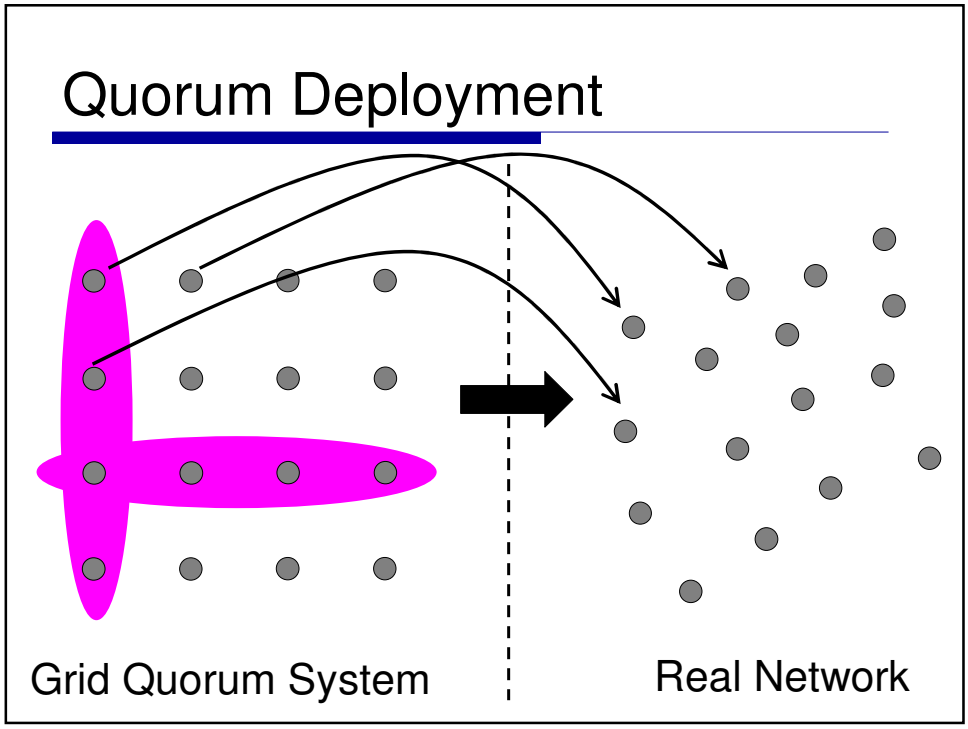
Outline

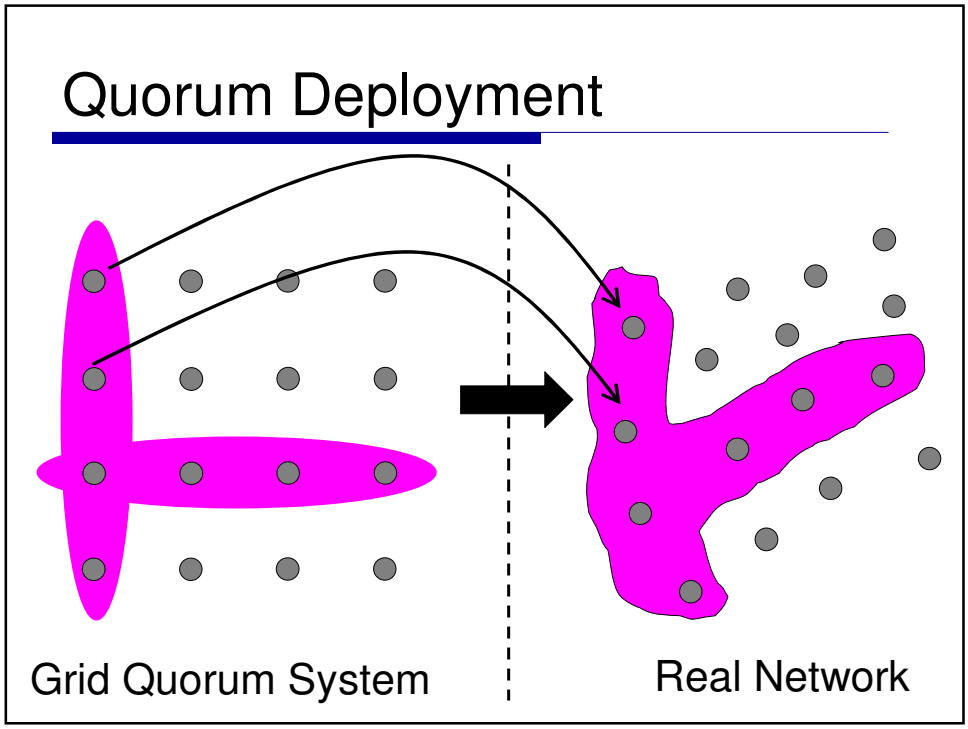
1. Background on quorum systems
2. **The Quorum Deployment Problem**
3. Hardness of quorum deployment
4. Special cases

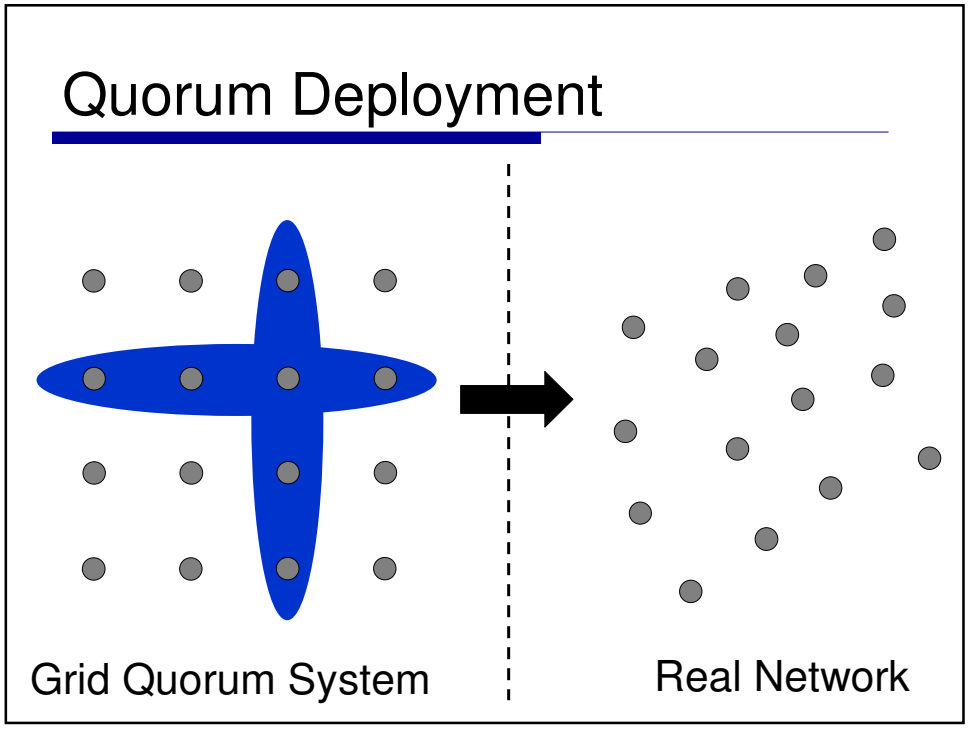

Quorum Deployment
Grid Quorum System
Real Network

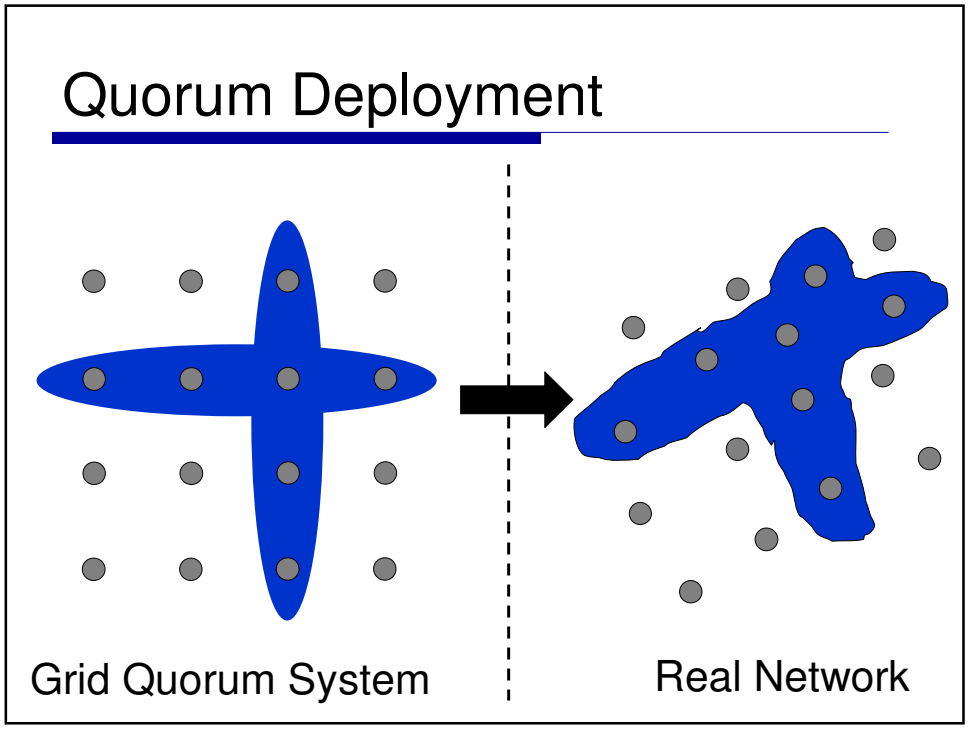

Quorum Deployment
Grid Quorum System
Real Network

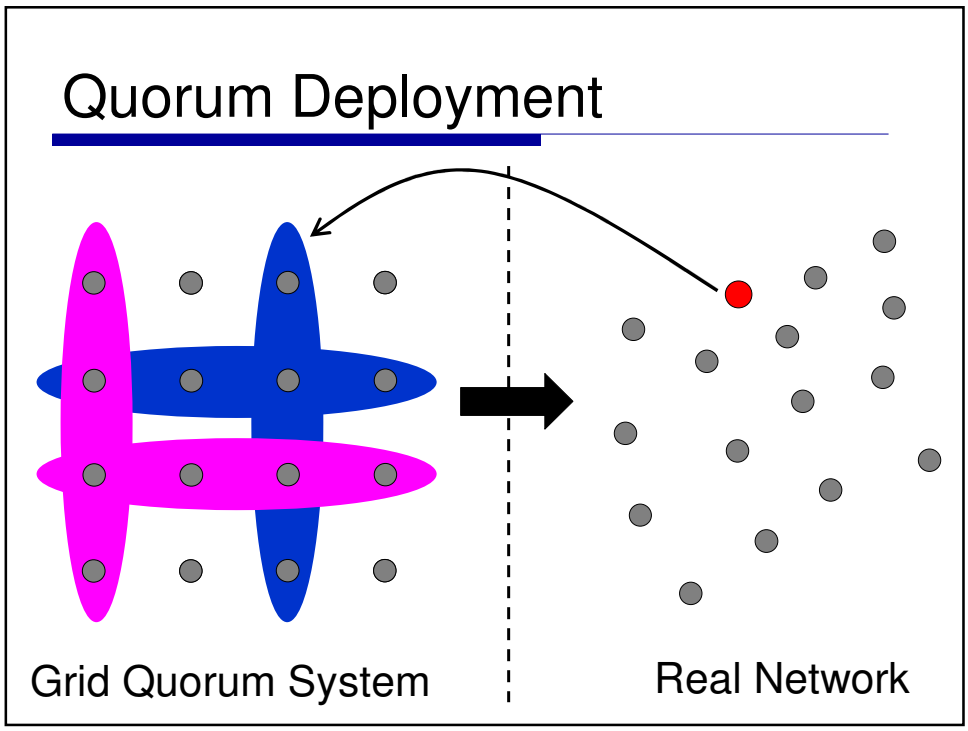

Quorum Deployment
Grid Quorum System
Real Network


Quorum Deployment
Grid Quorum System
Real Network

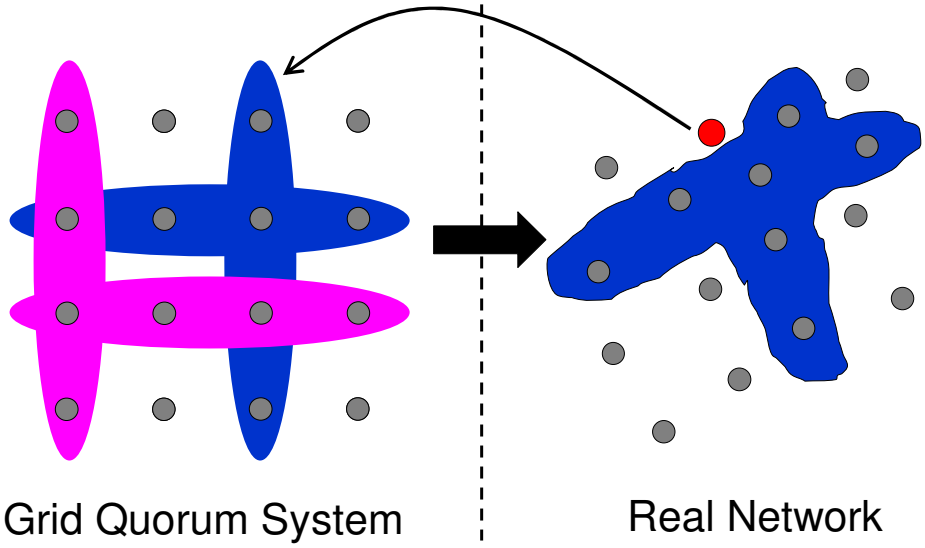

Quorum Deployment
Grid Quorum System
Real Network


Quorum Deployment
Grid Quorum System
Real Network


Quorum Deployment
Grid Quorum System
Real Network

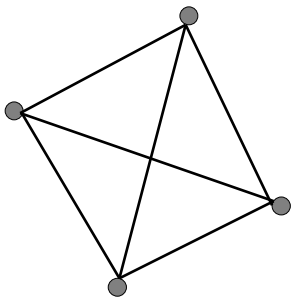

Quorum Deployment
Grid Quorum System
Real Network

Quorum Deployment



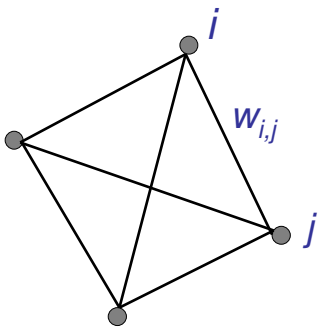
Quorum Deployment Problem

- Given:
  - Graph *G* with *n* vertices



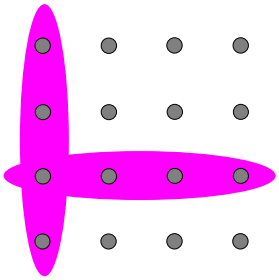
Quorum Deployment Problem

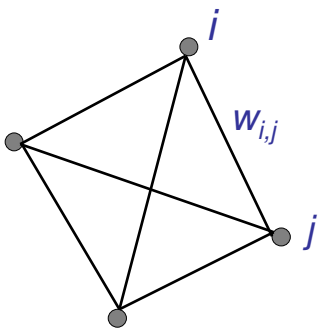
- Given:
  - Graph G with n vertices
  - Weighted edges; $w_{i,j}$ the weight of edge (i,j)



Quorum Deployment Problem

- Given:
  - Graph G with n vertices
  - Weighted edges; $w_{i,j}$ the weight of edge (i,j)
  - Quorum system Q





Quorum System Notation

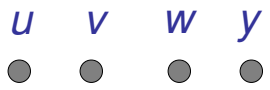
- Quorum system, Q

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$Q(p, j) = \begin{cases} 0 \text{ if node } j \text{ is } \textbf{not} \text{ in quorum } p \\ 1 \text{ if node } j \text{ is in quorum } p \end{cases}$$

Quorum System Notation

- Quorum system, Q

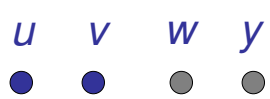


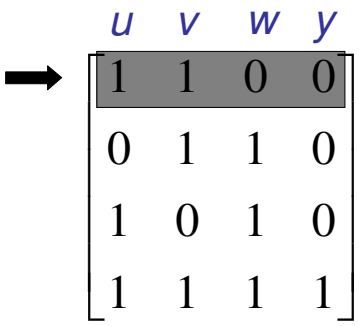
$$\begin{matrix} u & v & w & y \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$Q(p, j) = \begin{cases} 0 \text{ if node } j \text{ is } \textbf{not} \text{ in quorum } p \\ 1 \text{ if node } j \text{ is in quorum } p \end{cases}$$

Quorum System Notation

- Quorum system, Q

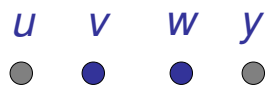


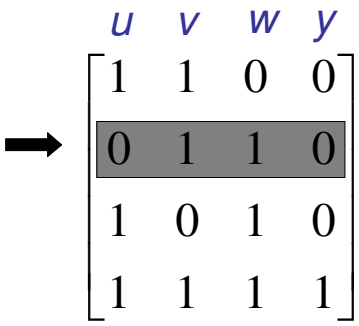


$$Q(p,j) = \begin{cases} 0 & \text{if node } j \text{ is } \textbf{not} \text{ in quorum } p \\ 1 & \text{if node } j \text{ is in quorum } p \end{cases}$$

Quorum System Notation

- Quorum system, Q





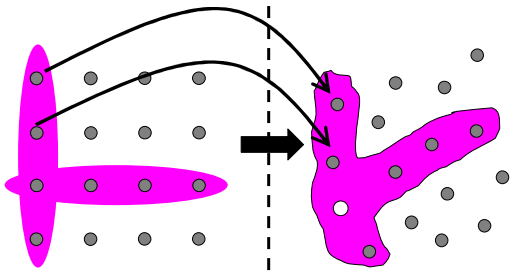
$$Q(p,j) = \begin{cases} 0 & \text{if node } j \text{ is } \textbf{not} \text{ in quorum } p \\ 1 & \text{if node } j \text{ is in quorum } p \end{cases}$$

Quorum Deployment Problem

- Given:
  - Graph G with n vertices
  - Weighted edges; $w_{i,j}$ the weight of edge (i,j)
  - Quorum system Q
- Output:
  - Optimal deployment of Q for graph G

Optimal Quorum Deployment

- Two parts
  - β, mapping quorum system to real nodes
  - α, mapping from nodes to quorums



Optimal Quorum Deployment

- Two parts
  - β, mapping quorum system to real nodes
  - α, mapping from nodes to quorums
- Minimize the deployment cost:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} \cdot Q(\alpha(i), \beta(j))$$

Optimal Quorum Deployment

- Two parts
  - β, mapping quorum system to real nodes
  - α, mapping from nodes to quorums
  - Similar to Quadratic Assignment Problem
    - 2 degrees of freedom, instead of 1
    - QAP only has one permutation

Optimal Quorum Deployment

- Two parts
  - β, mapping quorum system to real nodes
  - α, mapping from nodes to quorums
- Preserve good properties of quorum system
  - fault tolerance, availability, probe complexity

Outline

1. Background on quorum systems
2. The Quorum Deployment Problem
3. **Hardness of quorum deployment**
4. Special cases

Main Result

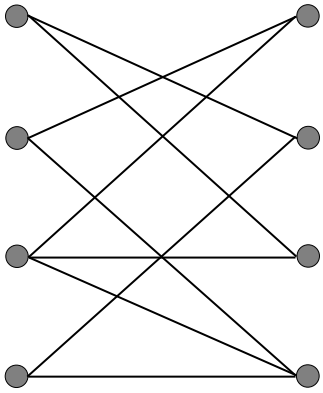
- **Theorem 1**:

  For any $\delta > 0$, it is NP-hard to approximate the optimal Quorum Deployment within a factor of n^{δ}.

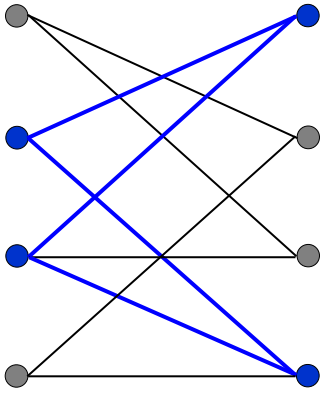
Balanced Complete Bipartite Subgraph

- Given:
  - Bipartite graph, $G = (V,E)$
    - Left nodes, L
    - Right nodes, R
  - parameter, k
- Output:
  - Complete, bipartite subgraph, G'
    - k left nodes
    - k right nodes

Balanced Complete Bipartite Subgraph



Balanced Complete Bipartite Subgraph



Reduction: BCBS → Deployment

- Given BCBS, {*G=(V,E), k*}
- Output instance of Quorum Deployment
  - *Q, w,* $n = |V| + 1$
  - Gap creating:

$$(G,k) \in \text{BCBS} \Rightarrow \exists \alpha, \beta : \text{deploy cost} \le n^2$$

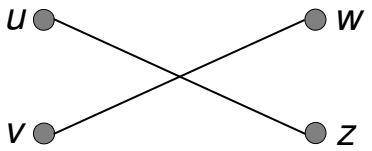
$$(G,k) \notin \text{BCBS} \Rightarrow \forall \alpha, \beta : \text{deploy cost} > n^x$$

Reduction: BCBS → Deployment

$$w_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ and } i, j < n \\ n^x & \text{if } (i,j) \notin E \text{ and } i, j < n \\ 1 & \text{if } (i = n) \text{ or } (j = n) \end{cases}$$

Reduction: BCBS → Deployment

$$w_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ and } i,j < n \\ n^x & \text{if } (i,j) \notin E \text{ and } i,j < n \\ 1 & \text{if } (i = n) \text{ or } (j = n) \end{cases}$$



	u	v	w	z	
u	n^x	n^x	n^x	1	1
v	n^x	n^x	1	n^x	1
w	n^x	1	n^x	n^x	1
z	1	n^x	n^x	n^x	1
	1	1	1	1	1

Reduction: BCBS → Deployment

$$Q(p,i) = \begin{cases} 1 & \text{if } p,i \le k \\ 1 & \text{if } i = n \\ 1 & \text{if } p = n \\ 0 & \text{otherwise} \end{cases}$$

Reduction: BCBS → Deployment

$$Q(p,i) = \begin{cases} 1 & \text{if } p,i \le k \\ 1 & \text{if } i = n \\ 1 & \text{if } p = n \\ 0 & \text{otherwise} \end{cases}$$

$$k = 2 \quad \Rightarrow \quad Q = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Proof: Case 1

- Assume there exists a bipartite complete subgraph of size *k*
- Construct a deployment:
  - map complete subgraph to the "all 1's" corner
  - map last row/column to itself
- Resulting cost: $k^2+2n-1 \le n^2$
  - k^2 cost for "all 1's" corner
  - $2n-1$ cost for the last row/column

Proof: Case 2

- Assume **no** complete bipartite subgraph of size *k*
  - Every deployment must include an expensive edge. Otherwise, we can find a complete bipartite subgraph

$$Q = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Metric Cost Deployment

- **Theorem 2**:

  If the weight is a *distance metric* (*symmetric* and *triangle inequality*) then it is NP-hard to find an optimal deployment.

Metric Cost Deployment

- **Theorem 2**:
  If the weight is a *distance metric* (*symmetric* and *triangle inequality*) then it is NP-hard to find an optimal deployment.
- Proof:
  Same reduction, except metric weights.

$$\left[\begin{array}{cc|ccc} 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 2 & 1 \\ \hline 2 & 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

Relaxed Metric Cost Deployment

- Relax assumptions that quorums intersect
  - Consider a quorum system to be a collection of sets.
- Assume the weight is a metric (symmetric and triangle inequality).

Relaxed Metric Cost Deployment

- **Theorem 3**:

  For any $\delta > 0$, it is NP-hard to approximate the optimal Metric Cost Deployment within a factor of n^{δ}.

- Proof ideas:
  - Reduction from 3-Partition Problem
    - Partition graph into pieces
  - Extension from Queyranne'86 reduction for the Quadratic Assignment Problem

Outline

1. Background on quorum systems
2. The Quorum Deployment Problem
3. Hardness of quorum deployment
4. **Special cases**
   - Bounded Cost Networks
   - Partial Deployment
   - Hyperbolic Quorum Systems

Bounded Cost Networks

- Assume that U is an upper bound on the distance between two nodes.
- Assume that L is a lower bound on the distance between two nodes.
- Trivial: all deployments are a $\frac{U}{L}$ approximation of optimal.

Partial Quorum Deployment

- Recall: deployment consists of two parts
  - β, mapping quorum system to real nodes
  - α, mapping from nodes to quorums
- Assume quorum system specification includes α.
  - E.g., node 1 uses quorum 1, node 2 uses quorum 2, etc.

Partial Quorum Deployment

- **Theorem 4:**

  We can find an optimal Partial Deployment in polynomial time.

Partial Quorum Deployment

- **Theorem 4:**

  We can find an optimal Partial Deployment in polynomial time.

- Proof:
  - Construct bipartite graph — matching in graph is a permutation.
  - Weight of edge is cost of assigning node to that quorum .
  - Find minimum weight matching.

Deployable Quorum Systems

- Are there quorum systems that are easy to deploy (for all networks)?

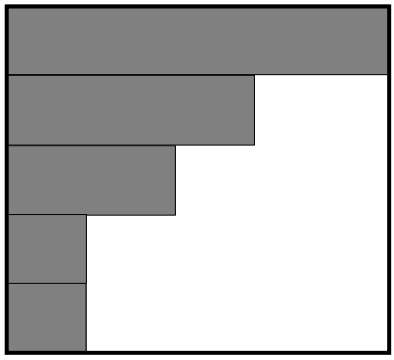
Deployable Quorum Systems

- Are there quorum systems that are easy to deploy (for all networks)?
  - Majorities:
    - A quorum is a majority of nodes in the system
    - Impossibility results do not apply since it is not a quorum system, by our formal definition.
  - Others??

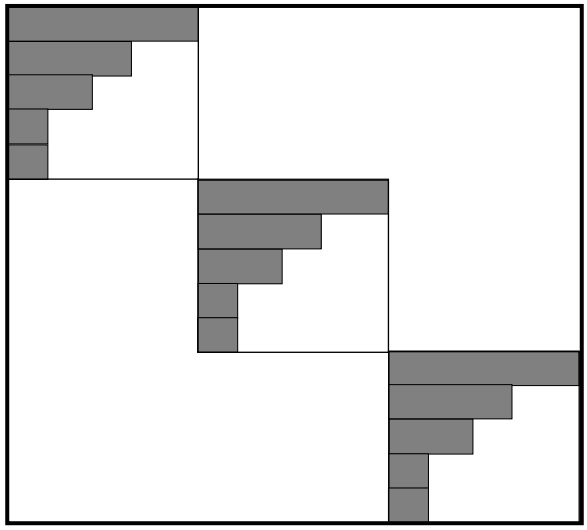
Hyperbolic Quorum Systems

- Advantages:
  - (Approximately) efficient to deploy
  - Help to understand complexity of quorum systems
- Disadvantages
  - NP-hard to deploy optimally
  - Not fault tolerant

Hyperbolic Quorum Systems



Hyperbolic Quorum Systems



Hyperbolic Quorum Systems

- **Theorem 5:**

  We can find an approximately optimal deployment for a hyperbolic quorum system.

Hyperbolic Quorum Systems

- **Theorem 5:**

  We can find an approximately optimal metric cost deployment for a hyperbolic quorum system.

- Note:

  The running time and the approximation depend on the complexity of the quorum system.

Open Questions

Open Questions

- Can we deploy:
  - Grid quorum systems
  - Paths quorum systems
  - Others?

Open Questions

- Can we deploy:
  - Grid quorum systems
  - Paths quorum systems
  - Others?
- Can we deploy quorum systems in:
  - growth-restricted metrics
  - metric spaces (if they intersect)
  - on a linear network

Open Questions

- Can we develop a quorum system with good fault tolerance, good probe complexity, and good load — that can also be deployed efficiently?

Conclusions

Conclusions

If you want to use quorums, use majorities — or other simple, deployable quorum systems.

Conclusions

If you want to ***use*** quorums, use majorities — or other simple, deployable quorum systems.

If you want to ***develop*** quorums, think about quorum deployment.