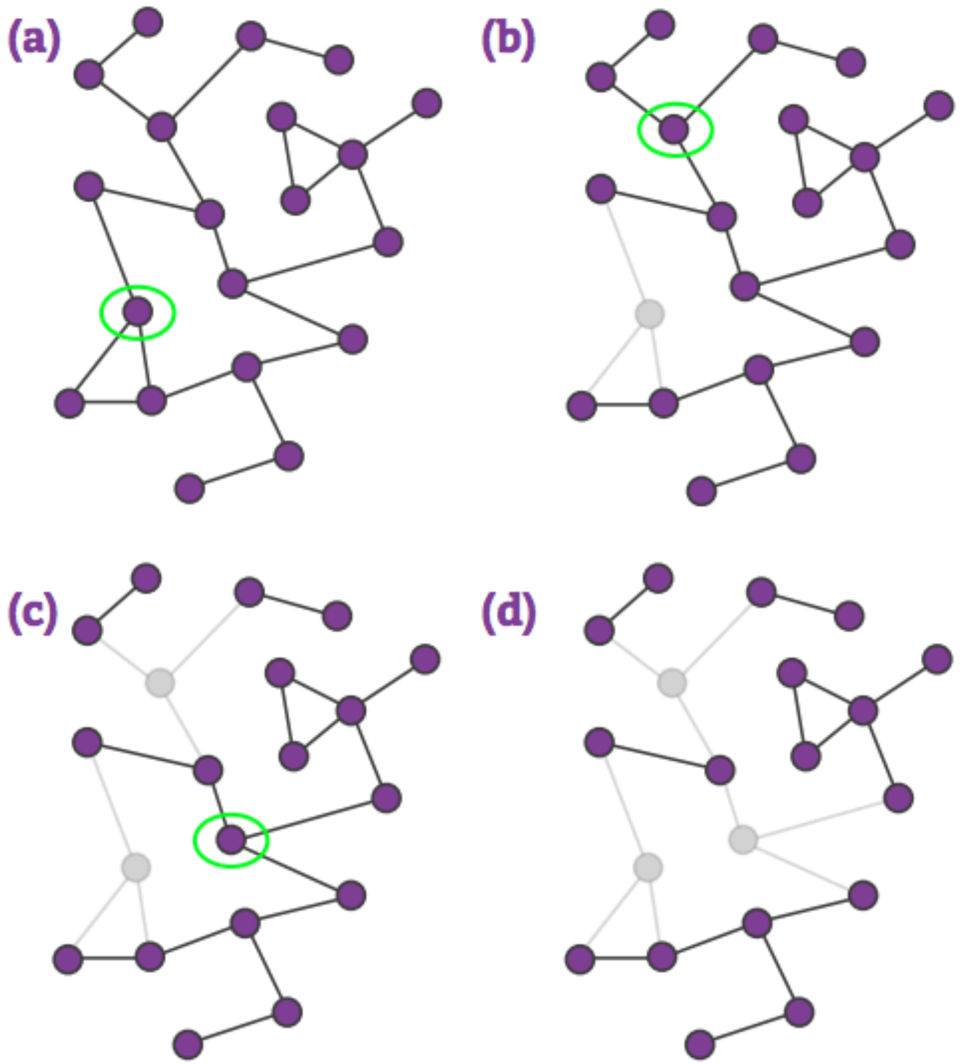
Network Robustness

Famim Talukder

Introduction

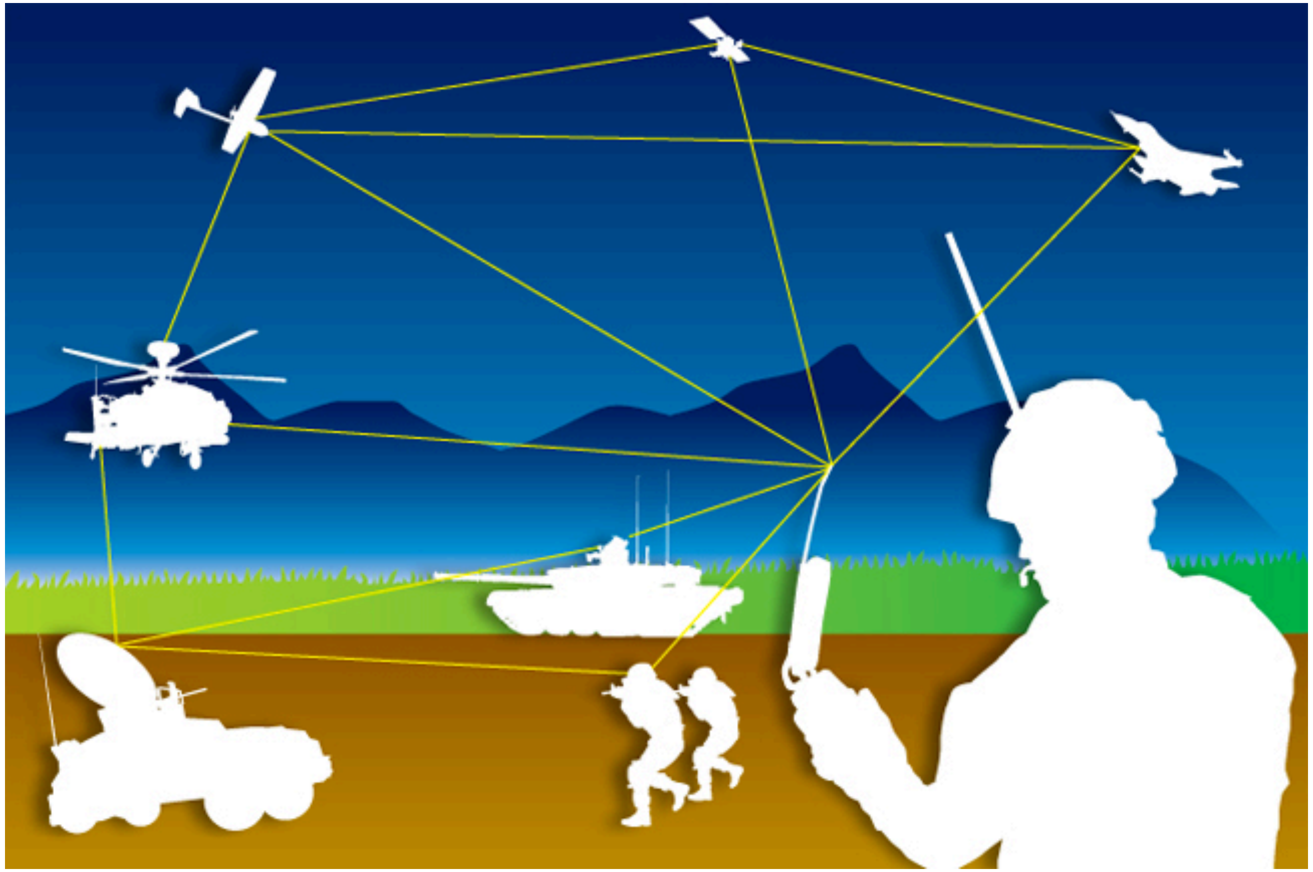
- Network Robustness – the network's structure plays a role in its ability to survive
  - Random failures
  - Deliberate attacks
- Cascading failures



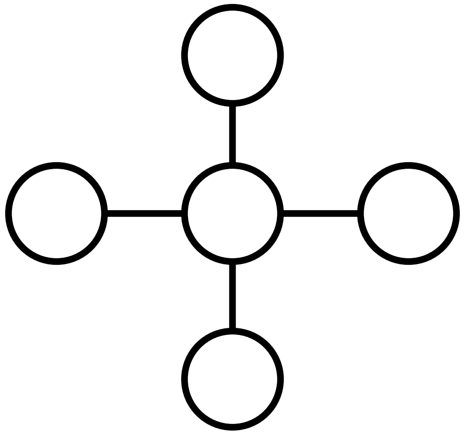
Motivations

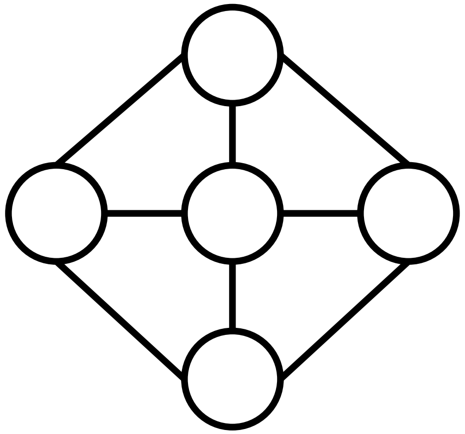
- Biology: some mutations lead to diseases while others do not
- Ecology: failure of an ecosystem based on human activity
- Engineering: communication systems, power grids, and component failures

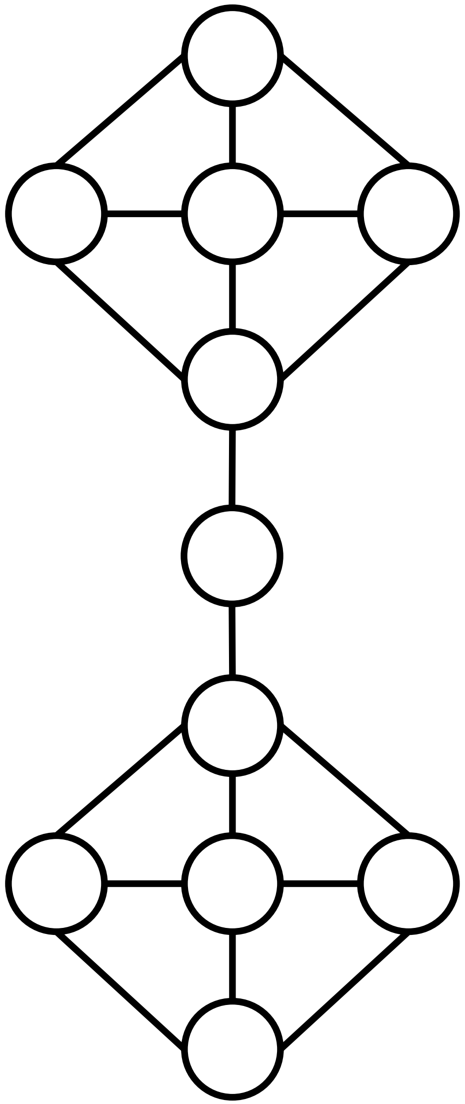
Military Communication Network



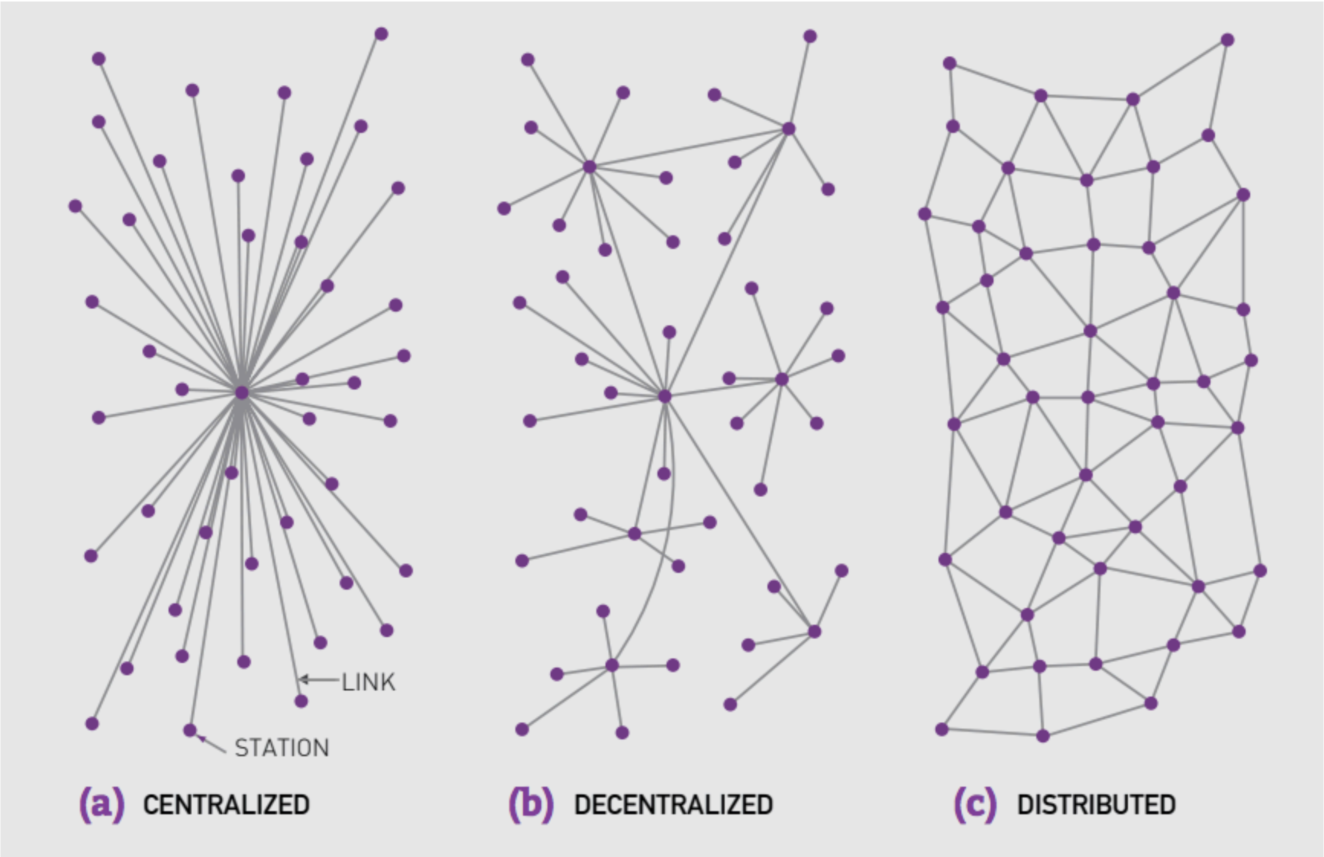
Example Networks





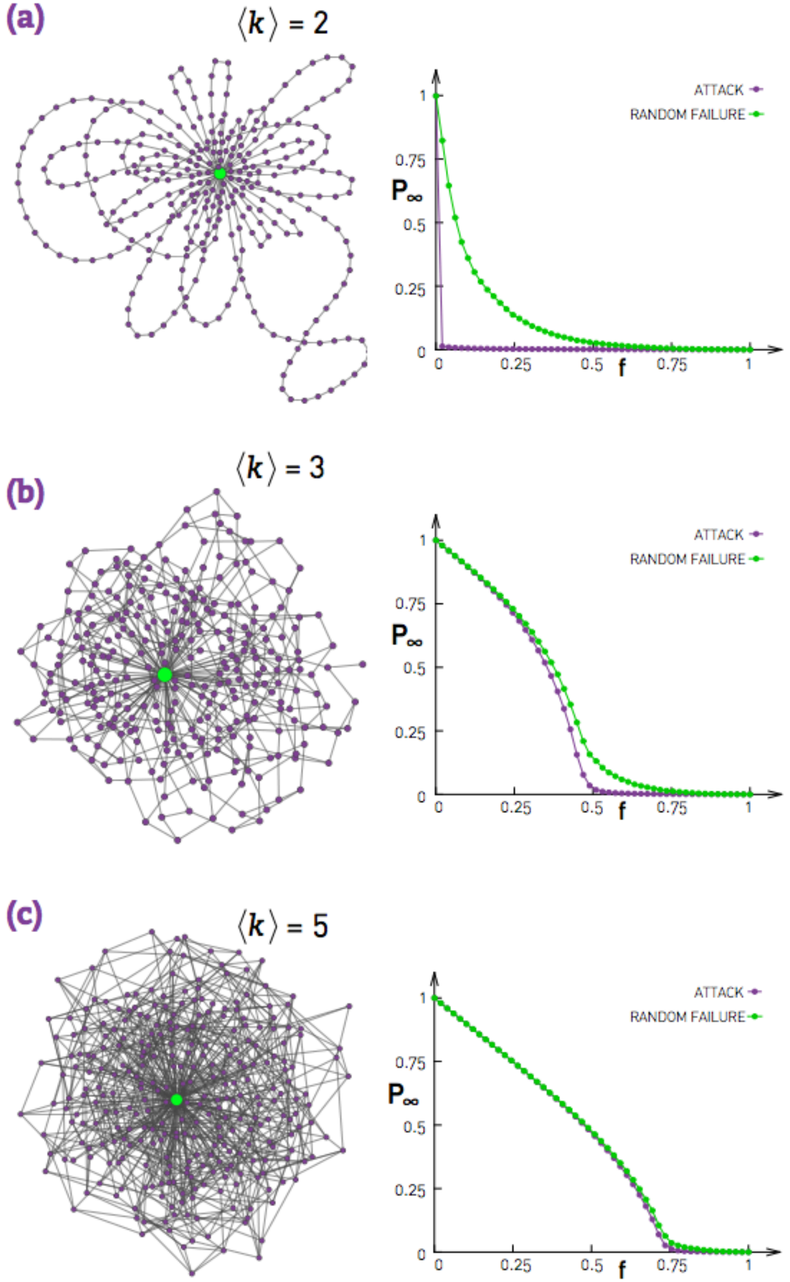


Communication Networks



Building Robustness

- Building robustness takes time
- How can we contain damages?
  - Remove nodes/edges
- Network topology is paramount in in network robustness



Vulnerability Metrics

Centrality Metrics

- Degree
- Closeness
- Centroid
- Eccentricity
- Betweenness
- Eigenvector

Robustness Measure

- Average path length
- Efficiency (power grids)
- Largest connected component

Pseudocode – Brandes Algorithm

Algorithm 1: Betweenness centrality in unweighted graphs

```
C_B[v] ← 0, v ∈ V;
for s ∈ V do
    S ← empty stack;
    P[w] ← empty list, w ∈ V;
    σ[t] ← 0, t ∈ V;    σ[s] ← 1;
    d[t] ← −1, t ∈ V;   d[s] ← 0;
    Q ← empty queue;
    enqueue s → Q;
    while Q not empty do
        dequeue v ← Q;
        push v → S;
        foreach neighbor w of v do
            // w found for the first time?
            if d[w] < 0 then
                enqueue w → Q;
                d[w] ← d[v] + 1;
            end
            // shortest path to w via v?
            if d[w] = d[v] + 1 then
                σ[w] ← σ[w] + σ[v];
                append v → P[w];
            end
        end
    end
    δ[v] ← 0, v ∈ V;
    // S returns vertices in order of non-increasing distance from s
    while S not empty do
        pop w ← S;
        for v ∈ P[w] do δ[v] ← δ[v] + σ[v]/σ[w] · (1 + δ[w]);
        if w ≠ s then C_B[w] ← C_B[w] + δ[w];
    end
end
```

- Uses BFS for unweighted graphs
- Runtime: O(m*n) on unweighted
- Space: O(m+n)

- Weighted networks:
- Runtime: $O(m*n+n^2\log(n))$

Largest Connected Component

Algorithm 2: Largest Connected Component

Data: Graph $G = (V, E)$
Result: Largest Connected Component of unweighted graphs
$Vis[v] \leftarrow 0, v \in V$;
for $v \in V$ **do**
  if $Vis[v] == 0$ **then**
    $DFSuntil(v)$;
  end
end

Data: Node v, $Vis[]$, Graph $G = (V, E)$
$v == 1$ **for** u *adjacent to* v **do**
  if $Vis[u] == 0$ **then**
    DFSuntil($u, Vis[], G$);
  end
end

Can be performed with either BFS or DFS

Runtime: O(m+n)

Questions\Comments