# Fault Tolerance Logical Network Properties of Irregular Graphs

ICA3PP'12, Fukuoka, Japan

Christophe Cérin ♣ , **Camille Coti** ♣ , Michel Koskas ♠

♣ Université Paris 13
♠ Institut national de la recherche agronomique

*Sept 6th* 2012

# Context: cloud computing

Cloud computing is using the Internet to interconnect resources

- Connecting together HUGE amounts of computing and storage resources
- Designed to be resilient

Resilient?

- In March 2011 Armenia was disconnected from the Internet by a 75yo Georgian woman who stole a wire for copper ("the hacker with the shovel")
- In August Wikipedia was disconnected from the Internet because a data center in Florida was disconnected by an accidental cable cut

Cloud computing is using the Internet to interconnect resources

- Connecting together HUGE amounts of computing and storage resources
- Designed to be resilient

Resilient?

- In March 2011 Armenia was disconnected from the Internet by a 75yo Georgian woman who stole a wire for copper ("the hacker with the shovel")
- In August Wikipedia was disconnected from the Internet because a data center in Florida was disconnected by an accidental cable cut

# Context: cloud computing

Cloud computing is using the Internet to interconnect resources

- Connecting together HUGE amounts of computing and storage resources
- Designed to be resilient

Resilient?

- In March 2011 Armenia was disconnected from the Internet by a 75yo Georgian woman who stole a wire for copper ("the hacker with the shovel")
- In August Wikipedia was disconnected from the Internet because a data center in Florida was disconnected by an accidental cable cut

How can we characterize a netwrk topology wrt expectations for parallel applications?

- **Diameter** : longest shortest path between any two vertices from the grah. In other words: maximum number of hops made by a message to reach its destination.
- **Node connectivity** : minimum number of vertices that must be removed to disconnect the graph.
  In other words: how many nodes can fail before we can expect connectivity to be lost.
- **Link connectivity** : minimum number of links that must be removed to disconnect the graph.
  In other words: how many links (cables...) can be broken before we can expect connectivity to be lost.
- **Fault diameter** : diameter of the graph, given the maximum number of failure before the graph becomes bipartite ($\kappa - 1$, if $\kappa$ is the node connectivity)
  In other words: how the diameter evolves in degraded conditions.

# Characteristics of networks

How can we characterize a netwrk topology wrt expectations for parallel applications?

- **Diameter** : longest shortest path between any two vertices from the grah. In other words: maximum number of hops made by a message to reach its destination.
- **Node connectivity** : minimum number of vertices that must be removed to disconnect the graph.
  In other words: how many nodes can fail before we can expect connectivity to be lost.
- **Link connectivity** : minimum number of links that must be removed to disconnect the graph.
  In other words: how many links (cables...) can be broken before we can expect connectivity to be lost.
- **Fault diameter** : diameter of the graph, given the maximum number of failure before the graph becomes bipartite ($\kappa - 1$, if $\kappa$ is the node connectivity)
  In other words: how the diameter evolves in degraded conditions.

How can we characterize a netwrk topology wrt expectations for parallel applications?

- **Diameter** : longest shortest path between any two vertices from the grah.
  In other words: maximum number of hops made by a message to reach its destination.
- **Node connectivity** : minimum number of vertices that must be removed to disconnect the graph.
  In other words: how many nodes can fail before we can expect connectivity to be lost.
- **Link connectivity** : minimum number of links that must be removed to disconnect the graph.
  In other words: how many links (cables...) can be broken before we can expect connectivity to be lost.
- **Fault diameter** : diameter of the graph, given the maximum number of failure before the graph becomes bipartite ($\kappa - 1$, if $\kappa$ is the node connectivity)
  In other words: how the diameter evolves in degraded conditions.

How can we characterize a netwrk topology wrt expectations for parallel applications?

- **Diameter** : longest shortest path between any two vertices from the grah. In other words: maximum number of hops made by a message to reach its destination.
- **Node connectivity** : minimum number of vertices that must be removed to disconnect the graph.
  In other words: how many nodes can fail before we can expect connectivity to be lost.
- **Link connectivity** : minimum number of links that must be removed to disconnect the graph.
  In other words: how many links (cables...) can be broken before we can expect connectivity to be lost.
- **Fault diameter** : diameter of the graph, given the maximum number of failure before the graph becomes bipartite ($\kappa - 1$, if $\kappa$ is the node connectivity)
  In other words: how the diameter evolves in degraded conditions.

Regular graphs have some symmetries

- Some properties can be extracted to simplify the computation of these metrics
- *e.g.*,to compute the diameter

Here we are talking about **irregular graphs**

- No such propertie to simplify the computation

Large-scale graphs

- Large number of nodes!
- The complexity of the algorithms matters a lot

This algorithm answers two major questions:

- Is our set of vertices/nodes connected or disconnected?
- How many connected components do we have?

Algorithm: based of the Breadth First Search (BFS) algorithm.

1: Start with the first non visited vertex
2: Visit its connected component
3: Restart until there is no more vertex to visit (*i.e.*,all vertices have been visited)
4: The number of times we do step 3 = number of connected components

If all the vertices are visited during the first pass of the algorithm: the graph is connected.

This algorithm answers two major questions:

- Is our set of vertices/nodes connected or disconnected?
- How many connected components do we have?

Algorithm: based of the Breadth First Search (BFS) algorithm.

1: Start with the first non visited vertex
2: Visit its connected component
3: Restart until there is no more vertex to visit (*i.e.,*all vertices have been visited)
4: The number of times we do step 3 = number of connected components

If all the vertices are visited during the first pass of the algorithm: the graph is connected.

# Node connectivity and link connectivity

Naive approach:

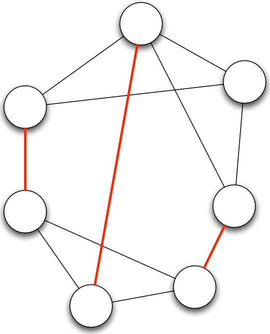
**for** $i = 1$ to $n$ **do**
    **for** for NumChoice $= 1$ to MaxChoiceVertices **do**
        choose $i$ vertices among $n$;
        cancel the vertices;
        **if** graph is not connected **then**
            **return** $i$
        **end if**
    **end for**
**end for**

Problem: exponential complexity!

# Node connectivity and link connectivity

Dichotomous approach:

- Remove $n/2$ vertices
- Check if the graph is still connected
- Remove more or less vertices depending on the previous try

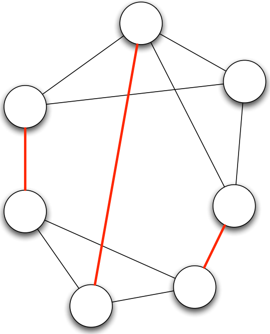


**Require:** Low = 1 ; Up = MAX.
**Require:** (MAX = NbVertices or MinDegree).

```
set K = log n
while Up − Low > 1 do
    m = (Low + Up) / 2
    repeat
        Remove m vertices (or edges) randomly
        if graph not connected then
            Up = m
        end if
    until K times
    if all graphs are connected then
        Low = m
    end if
end while
return Low + 1
```

Dichotomous approach:

- Remove $n/2$ vertices
- Check if the graph is still connected
- Remove more or less vertices depending on the previous try



**Require:** Low = 1 ; Up = MAX.
**Require:** (MAX = NbVertices or MinDegree).

set $K = \log n$
**while** $Up - Low > 1$ **do**
    m = (Low + Up) / 2
    **repeat**
        Remove m vertices (or edges) randomly
        **if** graph not connected **then**
            Up = m
        **end if**
    **until** K times
    **if** all graphs are connected **then**
        Low = m
    **end if**
**end while**
return Low + 1

# Diamater of the graph

Doing an extensive enumeration of all the vertices of the graph would be too expensive

- Randomized algorithm

```
set Diameter = 0
repeat
    Select randomly a vertex, name it 'current vertex'
    and mark it as visited. Set 'Current diameter' to 0
    while current vertices have non visited neighbors do
        a) Compute the non visited neighbors of current vertices
        b) Replace the current vertices by their non visited vertices
        c) Add 1 to 'Current diameter'
    end while
    if 'Current diameter' > 'Diameter' then
        'Diameter' = 'Current diameter'
    end if
until 'some' vertices have been visited
return 'Diameter'
```

Several benchmark files (see our paper for all 4 of them)
Topology of the Internet: obtained from a campaign of traceroute calls and dedicated tooks (WebGraph, eDonkey, MetroSec).

- web: 1 719 037 vertices and 11 095 298 edges
- p2p: 5 792 297 vertices and 142 038 401 edges
- web: 39 459 925 web pages (vertices) and 783 027 125 links (edges)
- ip: 2 250 498 vertices and 19 394 216 edges

Characteristic of the Internet: end-users have a degree of 1.

Comparison with bounds given by Magnien et al

Table: Comparison between estimated diameters

| | Magnien et al's results<br>tlb - dslb - hdtub - rtub - tub | Our results |
|---|---|---|
| inet | 29-31-34-34-38 | 25 (24) |
| p2p | 8-9-10-10-10 | 8 (7) |
| web | 26-32-33-33-34 | 22 (23) |
| ip | 9-9-9-9-10 | 8 (7) |

Table: Metrics for Fault Tolerance ($\delta_{min} > 1$)

| | Link co. | Node co. | Fault diameter |
|---|---|---|---|
| inet | 2 | 18 | 24 |
| p2p | 2 | 1054 | 7 |
| web | 2 | 36 | 23 |
| ip | 2 | 391 | 7 |

- Computing metrics such as diameter and node connectivity of large-scale, irregular graphs is too expensive to be done extensively
- Probabilistic algorithms to approach these values
- Our tool can be used for any graph, no assumption on the topology

Conclusions on the resilience of the Internet

- Quite resilient (was the initial goal)
- End users are still subjet to disconnections