Adversarial VNet Embeddings: A Threat for ISPs?

Yvonne-Anne Pignolet, Gilles Tredan, **Stefan Schmid**

April, 2013

CloudNets = Virtual Networking Cloud Resources

Success of Node Virtualization

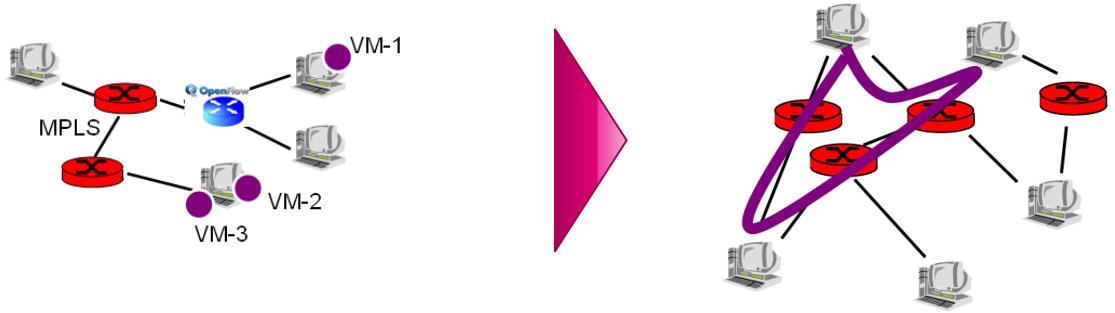
- a.k.a. end-host virtualization
- VMWare revamped server business
- OpenStack
- VM = flexible allocation, migration..
- «**Elastic computing**»

Trend of Link Virtualization

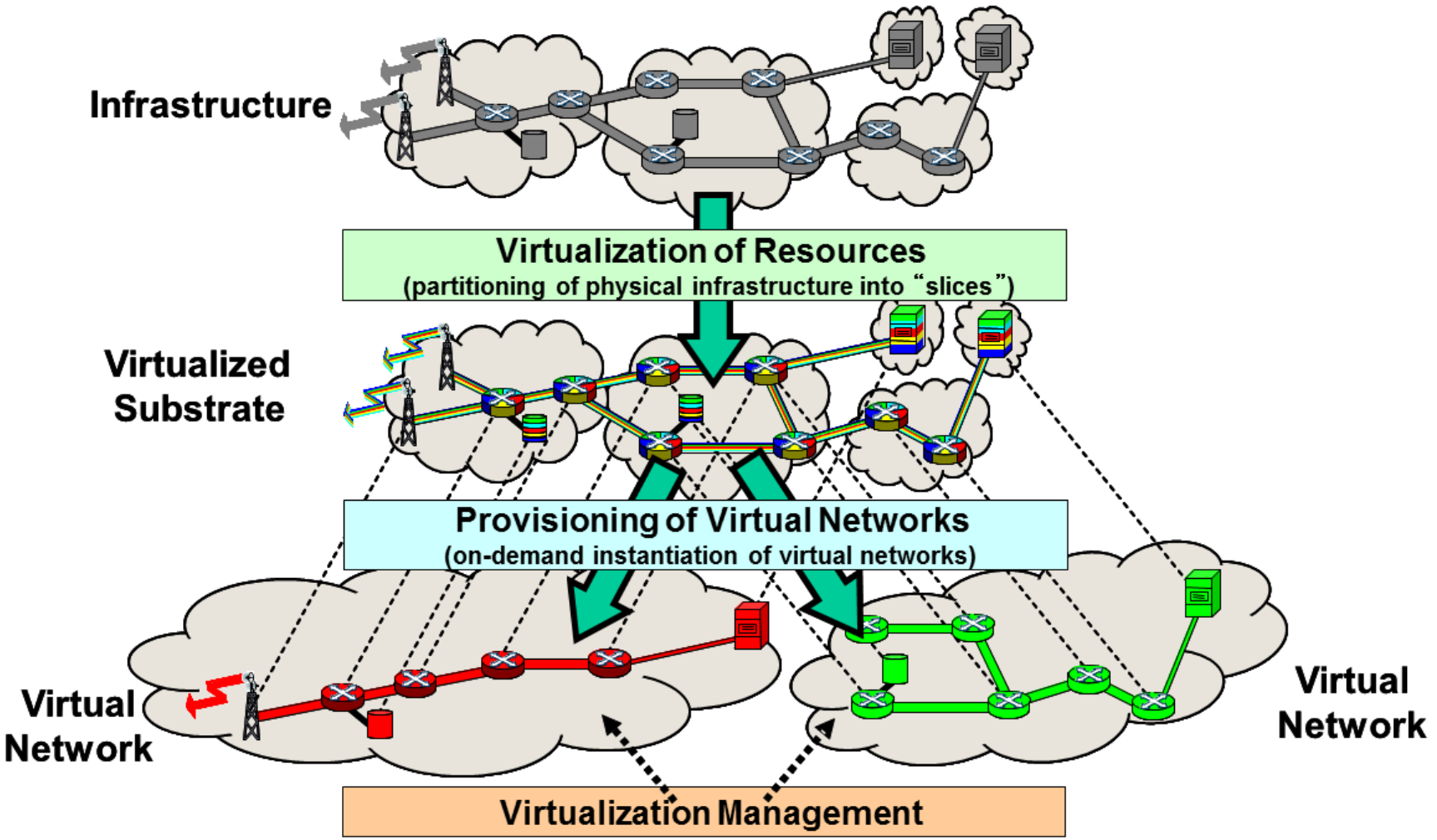
- MPLS, VPN networks, VLANs
- Software Defined Networks (SDN), OpenFlow, ...
- «The VMWare of the net»
- «**Elastic networking**»

Unified, fully virtualized networks: **CloudNets**

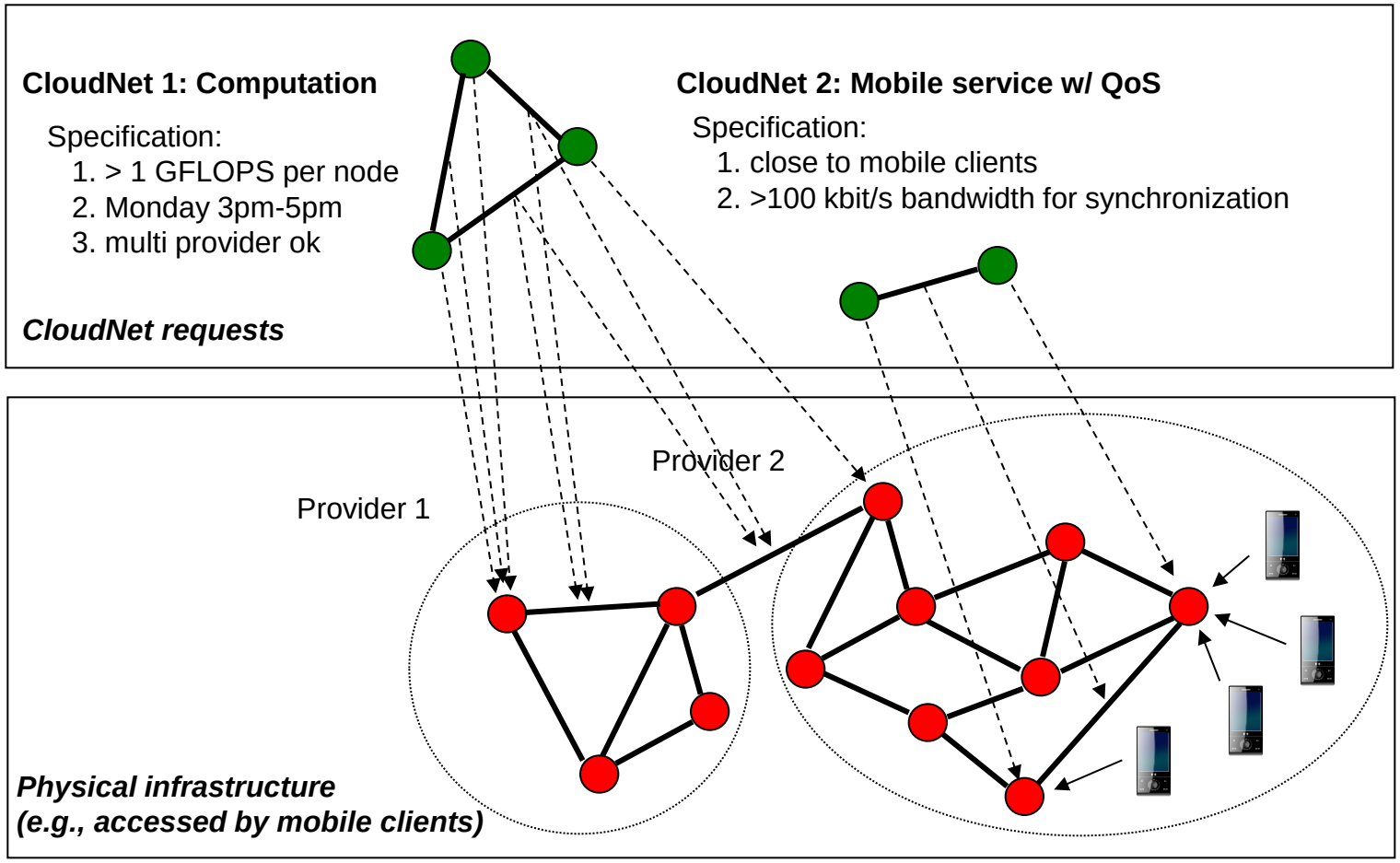
„Combine networking with heterougeneous cloud resources (e.g., storage, CPU, ...)!“



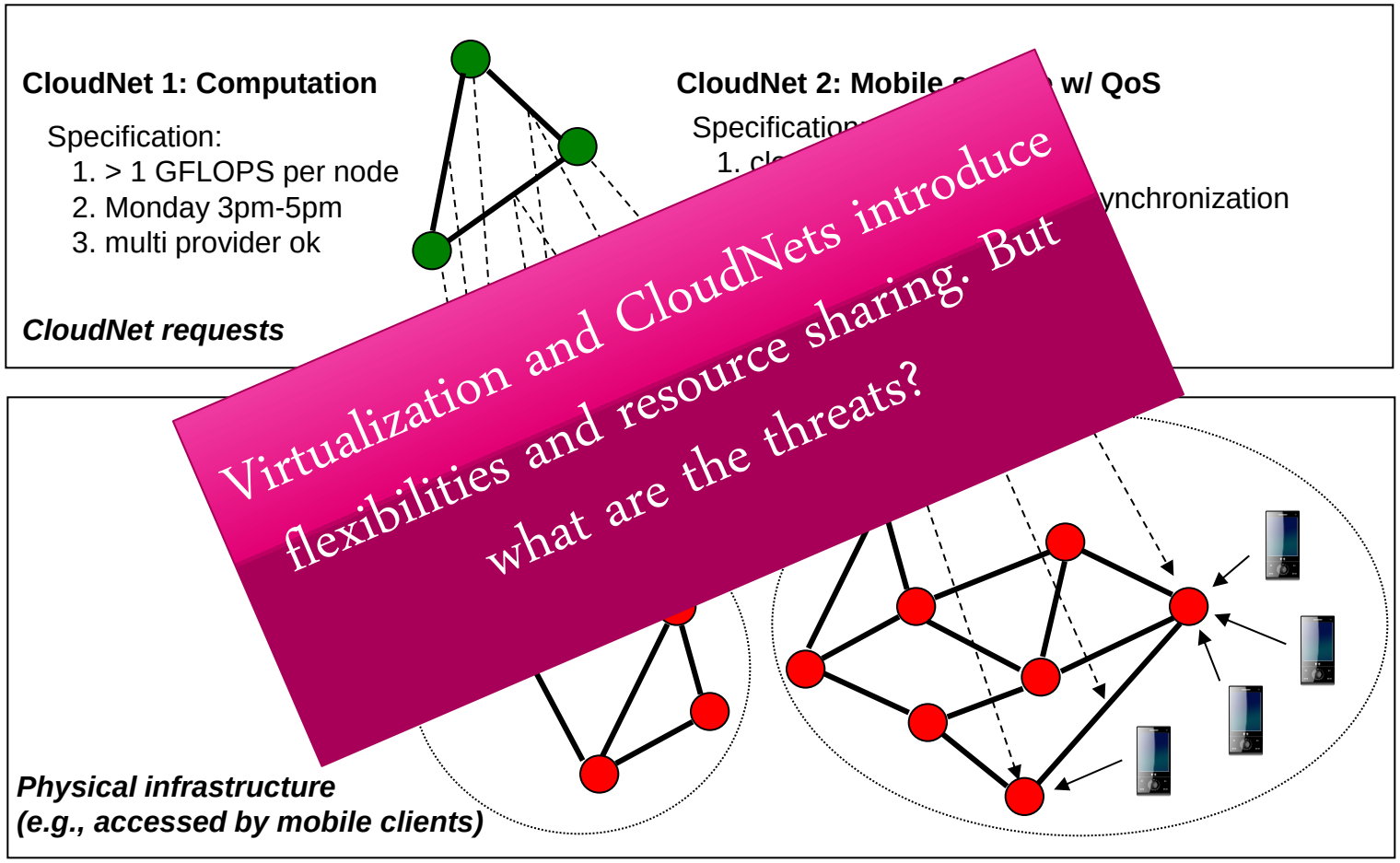
The Vision: Sharing Resources.



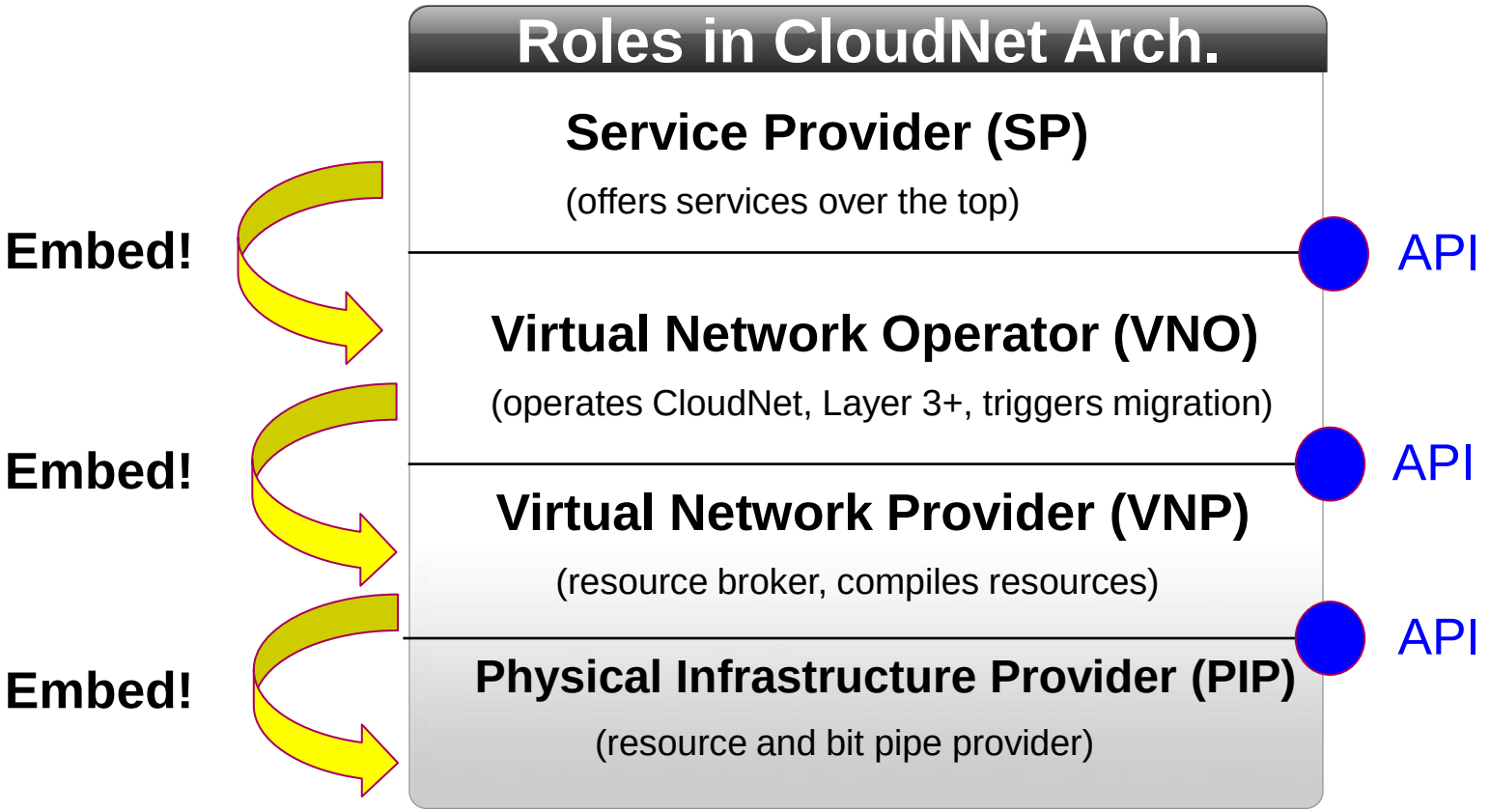
The Vision: Flexible CloudNets.



The Vision: Flexible CloudNets.

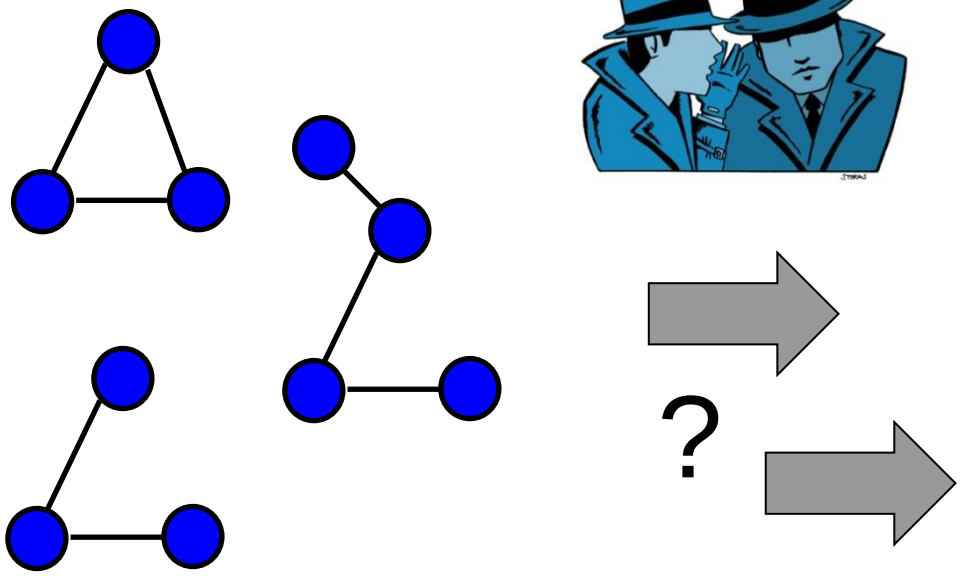


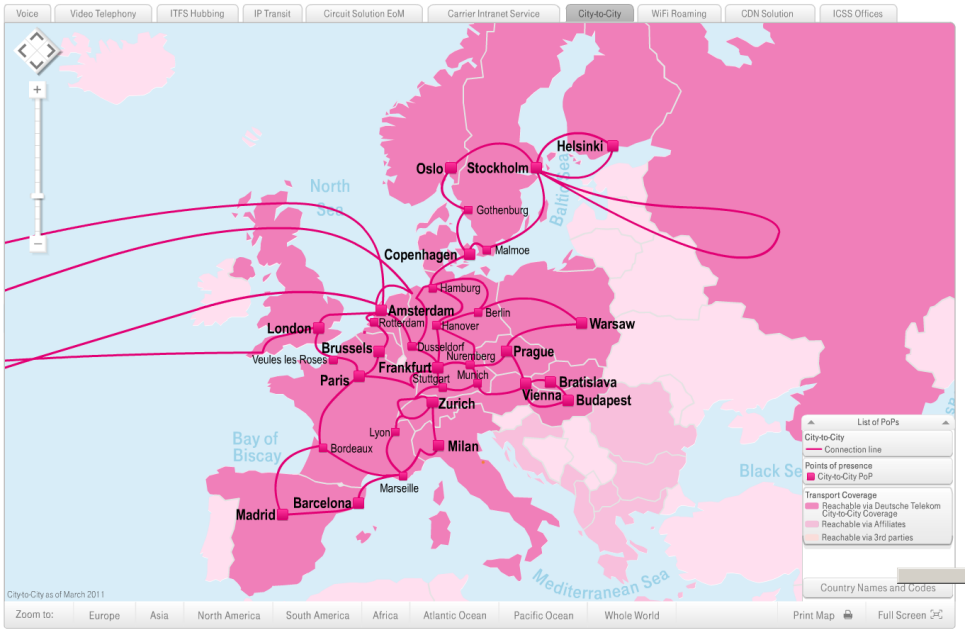
Embedding: From Service Provider via Broker to Infrastructure Provider.



Security Issues.

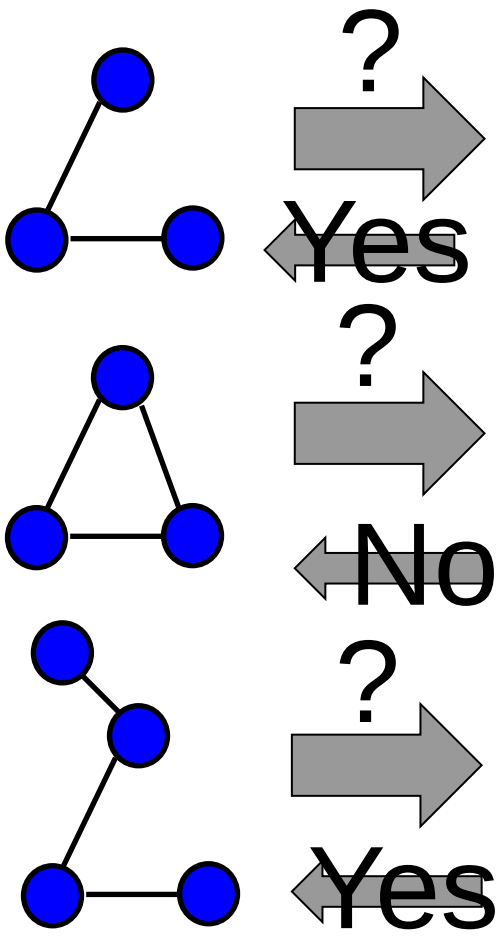
- Are CloudNet embeddings a threat for ISPs?
- Do embeddings leak information about infrastructure?





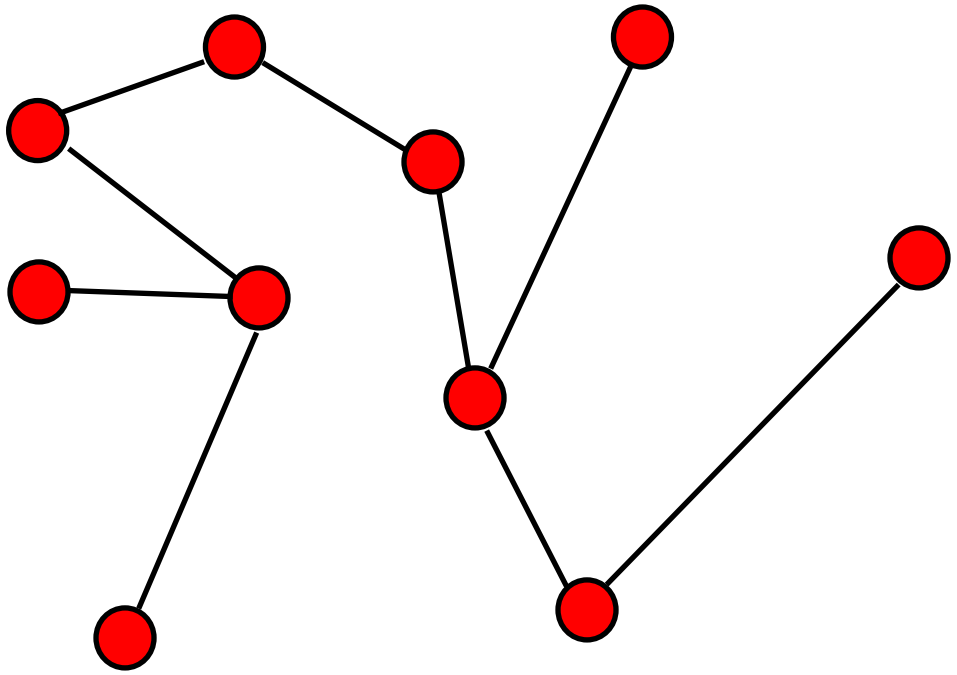
Request Complexity.

Are CloudNet embeddings a threat for ISPs?

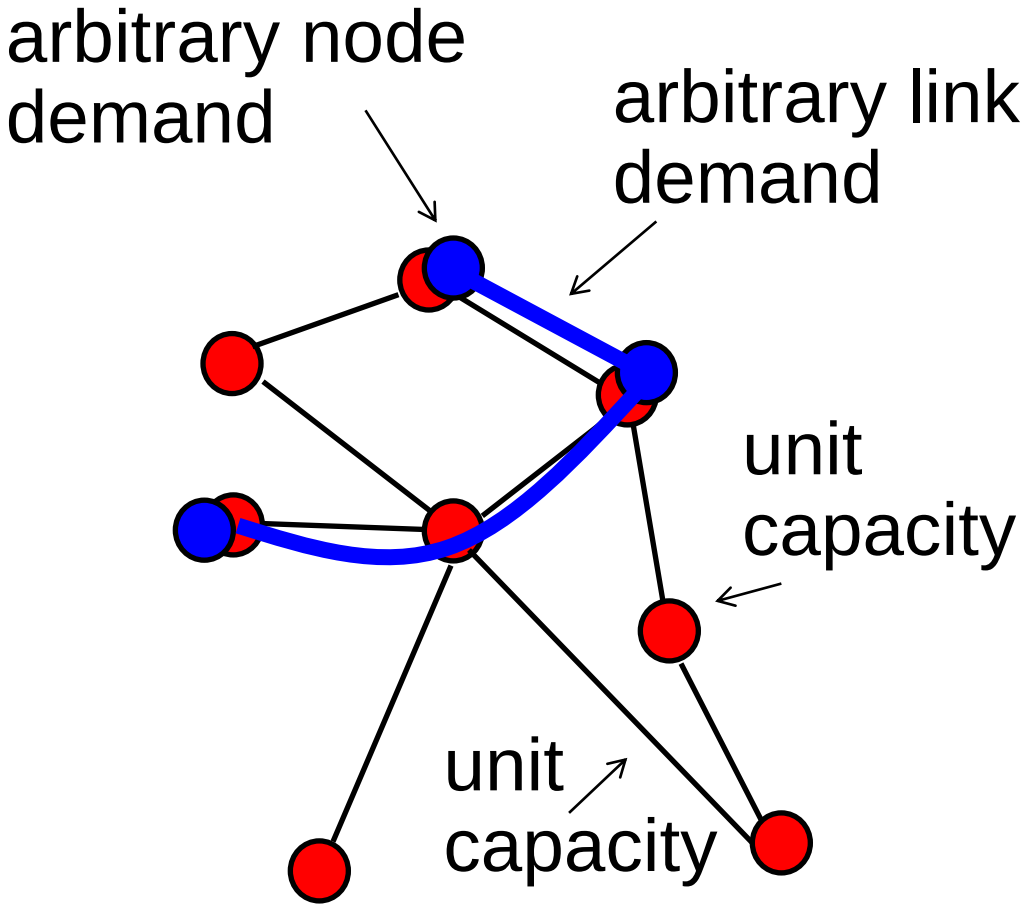


Request Complexity

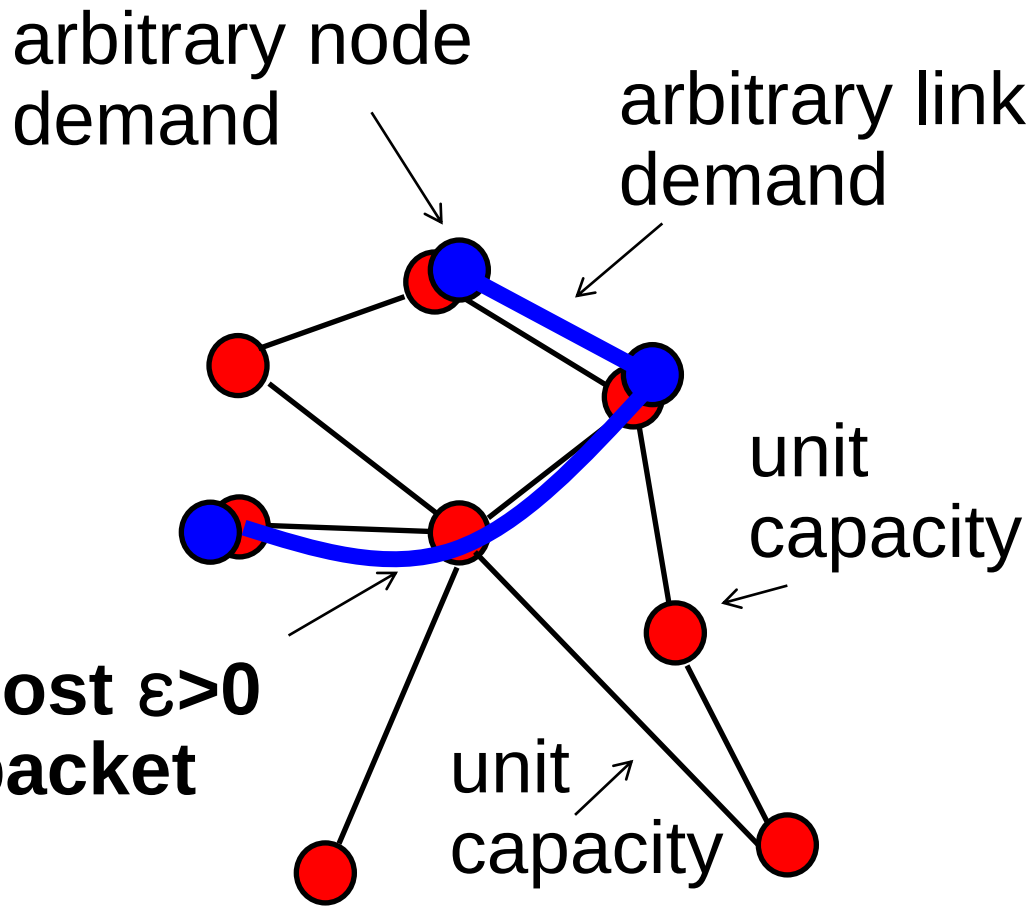
How many embeddings needed to fully reveal topology?



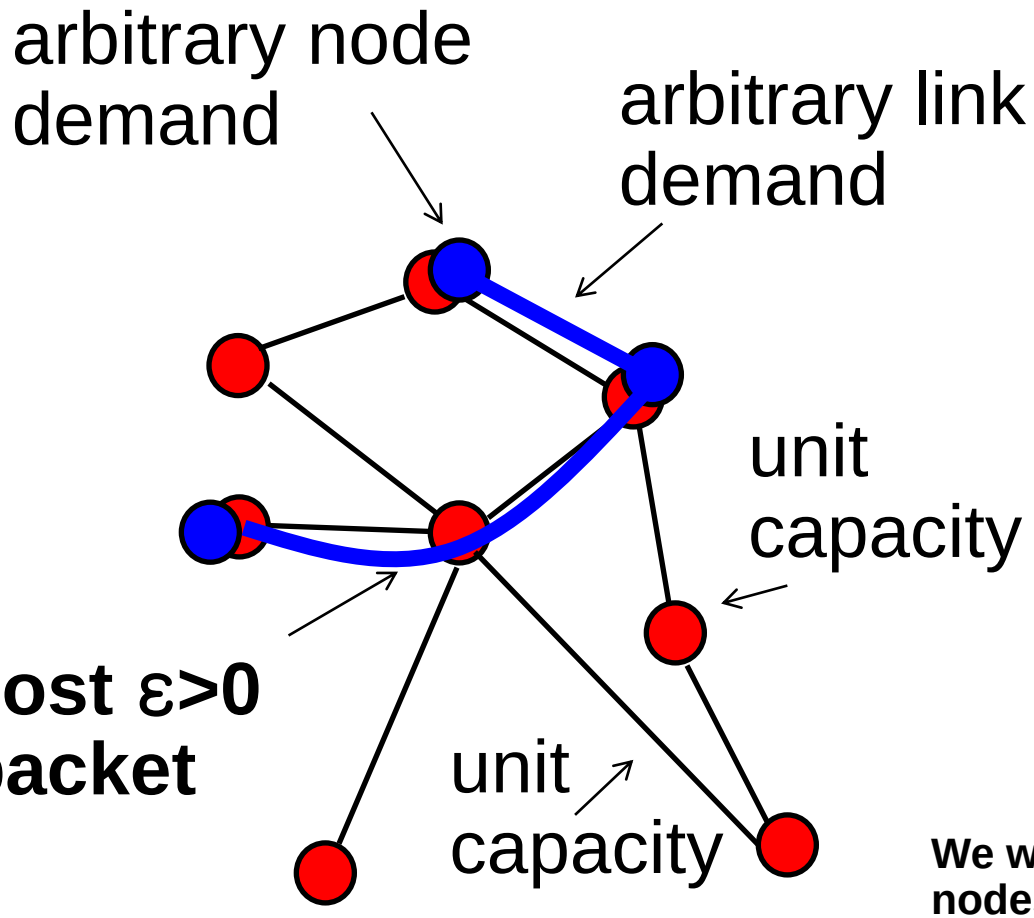
Embedding Model.



Embedding Model.



Embedding Model.

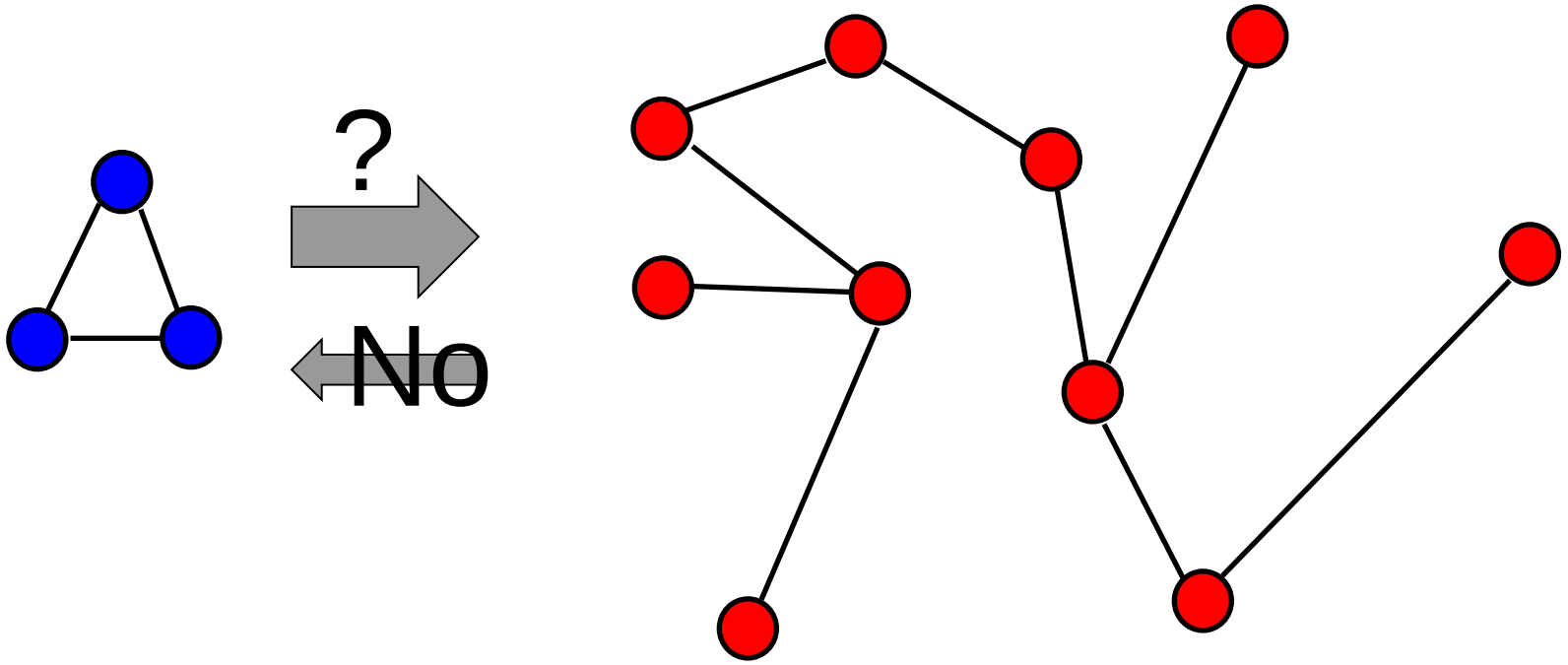


relay cost ε>0 (e.g., packet rate)

We will ask for unit capacities on nodes and links!
Essentially a graph immersion problem: disjoint paths for virtual links...

Some Properties Simple...

«Is the network 2-connected?»

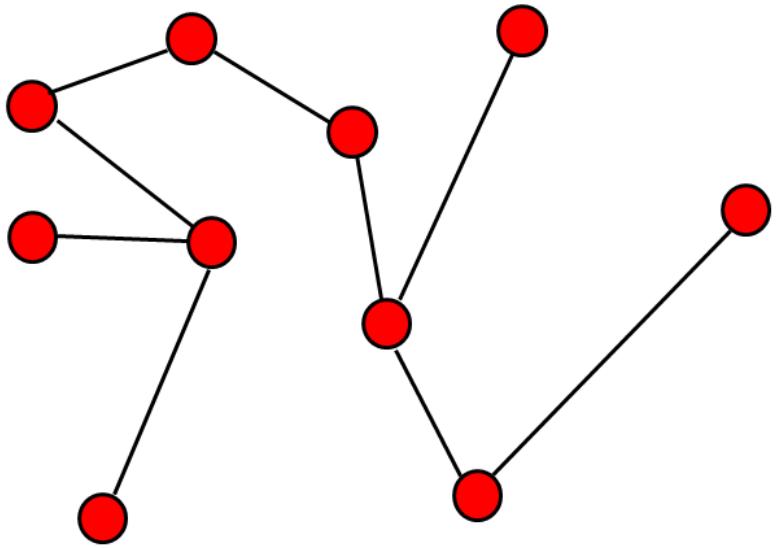


Example: Tree

How to discover a tree?

(Too) naive idea:

1. Test whether triangle fits? (loop-free)
2. Try to add neighbors to node until it works

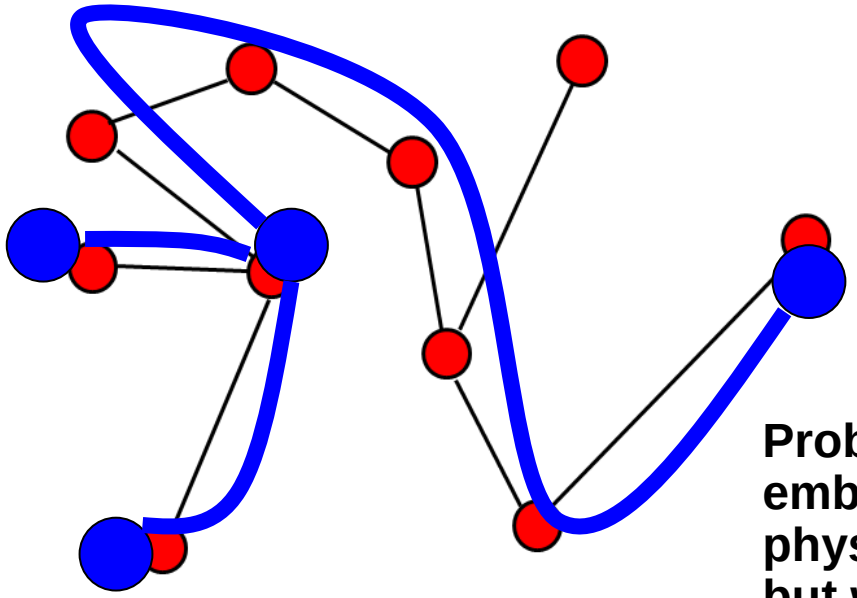


Example: Tree

How to discover a tree?

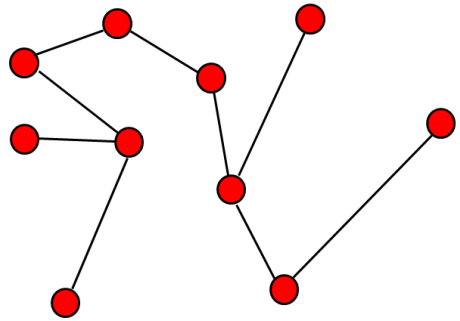
Naive idea:

1. Test whether triangle fits? (loop-free)
2. Try to add neighbors to node as long as possible, then continue with other node (degree strategy)



Problem: virtual links may be embedded over multiple physical links! Can still extend v, but will miss nodes along the way!

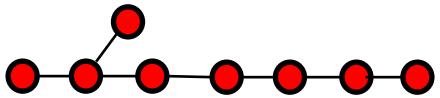
Solution: Tree



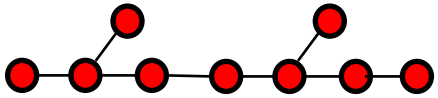
TREE ALGORITHM: line strategy

1. Binary search on longest path («anchor»):

2. Last and first node explored, explore «branches» at pending nodes



Then:



Analysis: Amortized analysis on links.

Algorithm 1 Tree Discovery: TREE

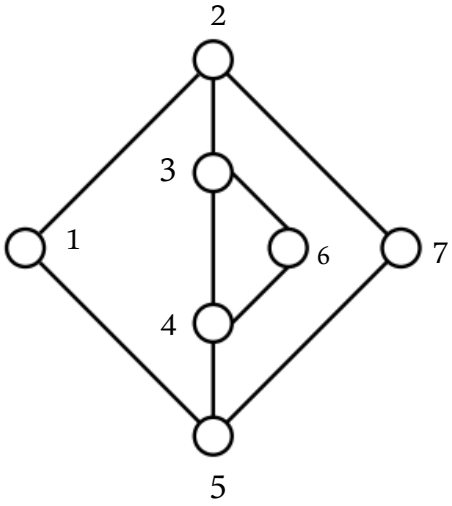
1: $G := \{\{v\}, \emptyset\}$ /* current request graph */
2: $\mathcal{P} := \{v\}$ /* pending set of unexplored nodes*/
3: **while** $\mathcal{P} \neq \emptyset$ **do**
4: choose $v \in \mathcal{P}$, $S :=$*exploreSequence*(v)
5: **if** $S \neq \emptyset$ **then**
6: $G := GvS$, add all nodes of S to $\mathcal{P}$
7: **else**
8: remove v from $\mathcal{P}$

exploreSequence(v)
1: $S := \emptyset$
2: **if** `request`(GvC, H) **then**
3: find max j s.t. $GvC^j \mapsto H$ (binary search)
4: $S := C^j$
5: **return** S

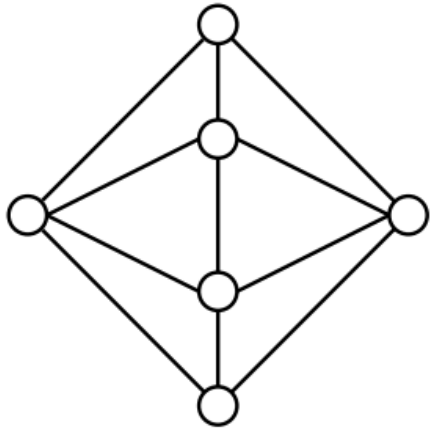
Be Careful with Embedding Relation!

Example:

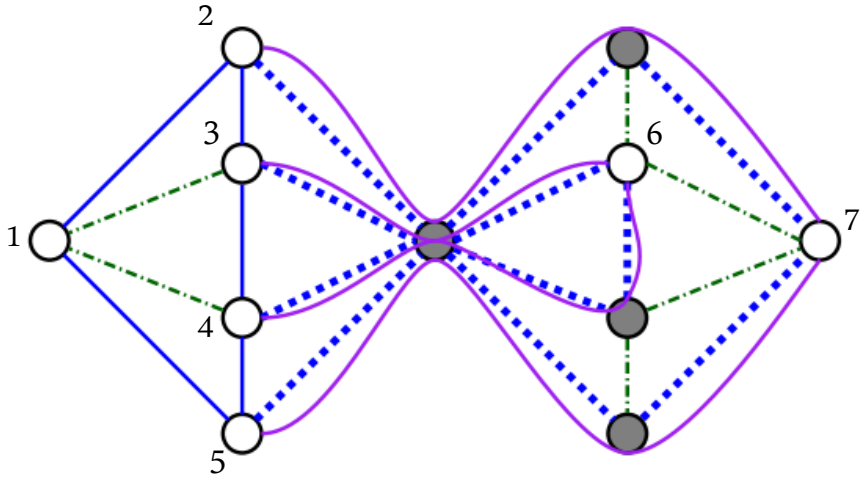
- **A cannot be embedded into B**
- **B cannot be embedded into A**
- **But A can be embedded into BB!**



A

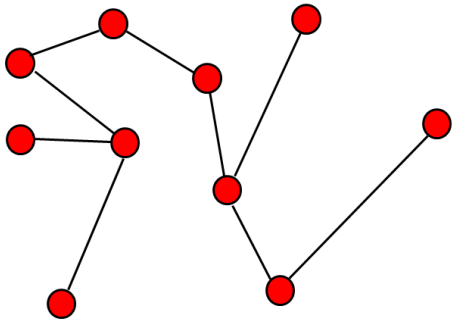


B



BB

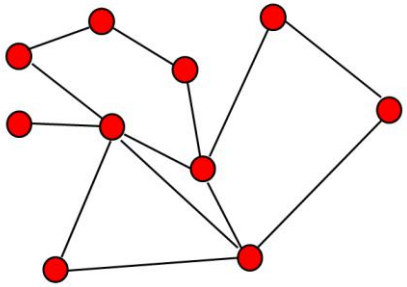
Overview of Results.



Tree

Can be explored in O(n) requests. This is optimal!

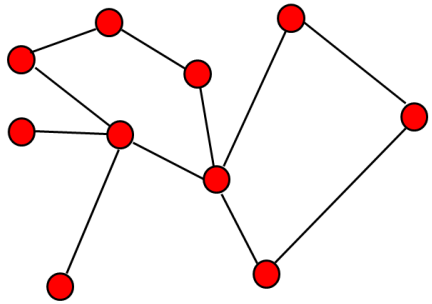
Lower bound: via number of possible trees and binary information.



General Graph

Can be explored in $O(n^2)$ requests. This is optimal!

Idea: Make spanning tree and then try all edges.
(Edges directly does not work!)

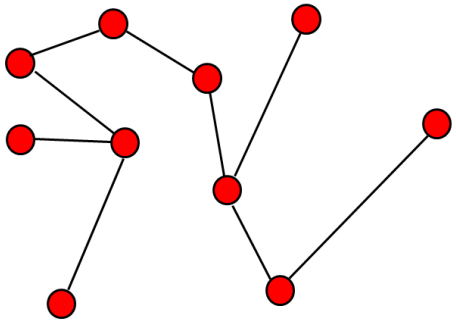


Cactus Graph

Can be explored in O(n) requests. This is optimal!

Via «graph motifs»!
A general framework exploiting poset relation.

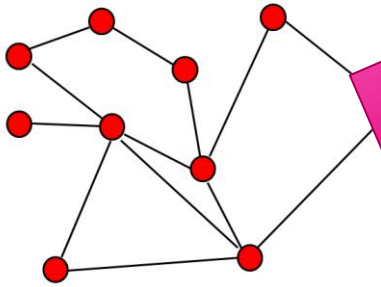
Overview of Results.



Tree

Can be explored in O(n) requests. This is o

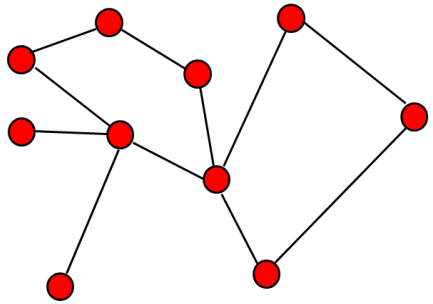
Lower bound: via number of possible trees and binary information.



G

ea: Make spanning tree and then try all edges.
(Edges directly does not work!)

All algorithms based on similar ideas:
1. grow request graph incrementally
2. find «knitting» first

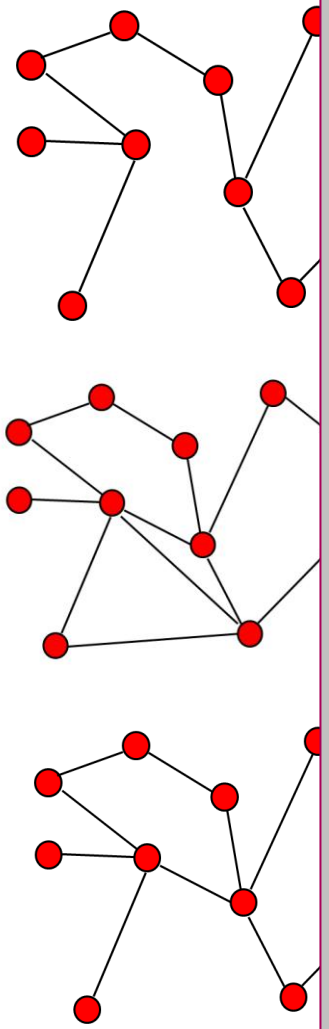


tus Graph

Can be explored in O(n) requests. This is optimal!

Via «graph motifs»!
A general framework exploiting poset relation.

Overview of Results.



Algorithm 2 Cactus Discovery: CAC

1: $G := \{\{v\}, \emptyset\}$ /* current request graph */
2: $\mathcal{P} := \{v\}$ /* pending set of unexplored nodes*/
3: **while** $\mathcal{P} \neq \emptyset$ **do**
4: choose $v \in \mathcal{P}$, $S :=$*exploreSequence*(v)
5: **if** $S \neq \emptyset$ **then**
6: $G := GvS$, add all nodes of S to $\mathcal{P}$
7: **for all** $e \in S$ **do** *edgeExpansion*(e)
8: **else**
9: remove v from $\mathcal{P}$

exploreSequence(v)

1: $S := \emptyset$
2: **if** $GvYCY \mapsto H$ **then**
3: find max j s.t. $GvY^jCY \mapsto H$
4: $S := Y^jCY$, $P' := \{C\}$
5: **while** $P' \neq \emptyset$ **do**
6: **for all** $C_i \in P'$ **do**
7: $A := prefix(C_i, S)$, $B := postfix(C_i, S)$;
8: **if** $GvACYCB \mapsto H$ **then**
9: find max j, k s.t. $GvAC(Y^jC)^kB \mapsto H$
10: **for** $l := 1, \ldots, k$ **do**
11: $P'' := P'' \cup \{C_l\}$
12: $S := AC(Y^jC)^kB$
13: $P' := P''$, $P'' := \emptyset$
14: **if** request$(GvSY, H)$ **then**
15: find max j s.t. $GvSY^j \mapsto H$
16: $S := SY^j$
17: **if** request$(GvSC, H)$ **then**
18: $S := SC$
19: **return** S

edgeExpansion(e)

1: let u, v be the endpoints of edge e, remove e from G
2: find max j s.t. $GvC^ju \mapsto H$
3: $G := GvC^ju$, add newly discovered nodes to $\mathcal{P}$

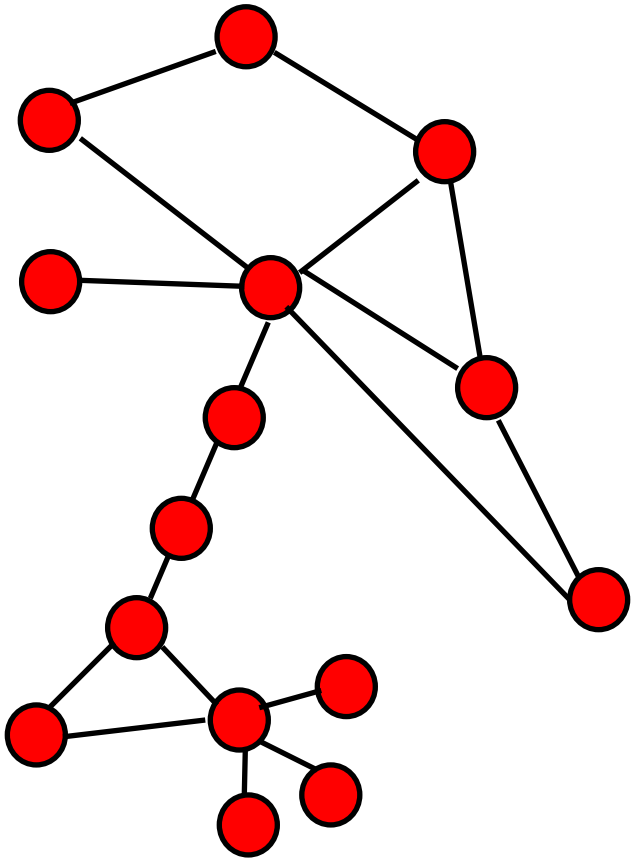
bound: via number of
le trees and binary
ation.

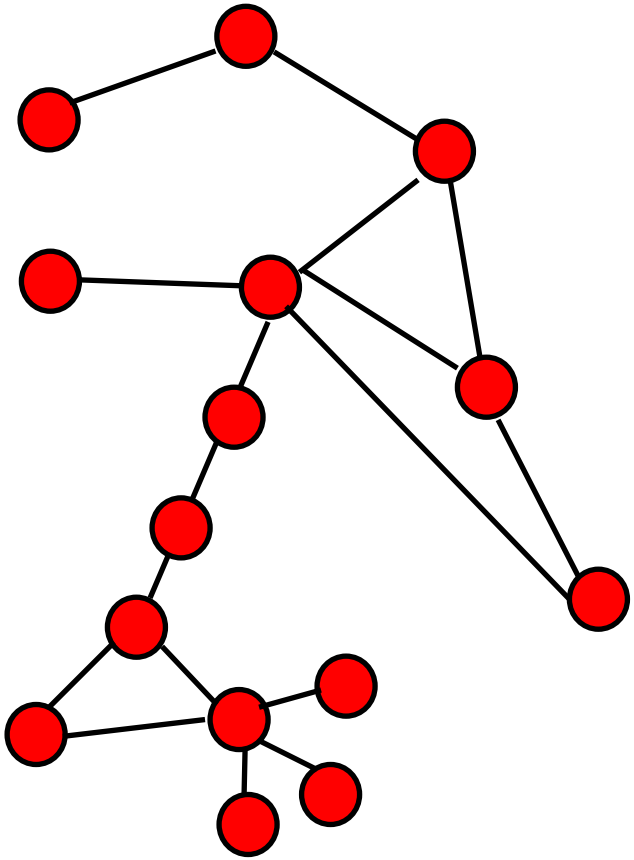
Make spanning tree and
ry all edges.
es directly does not work!)

a «graph motifs»!
general framework
xploiting poset relation.

General Recipe: Expand Request!

1. Find «knitting»:
- increasing connectivity too late comes at high cost!
- so first find graph structure of connected «motifs»
- motif = 2-connected components



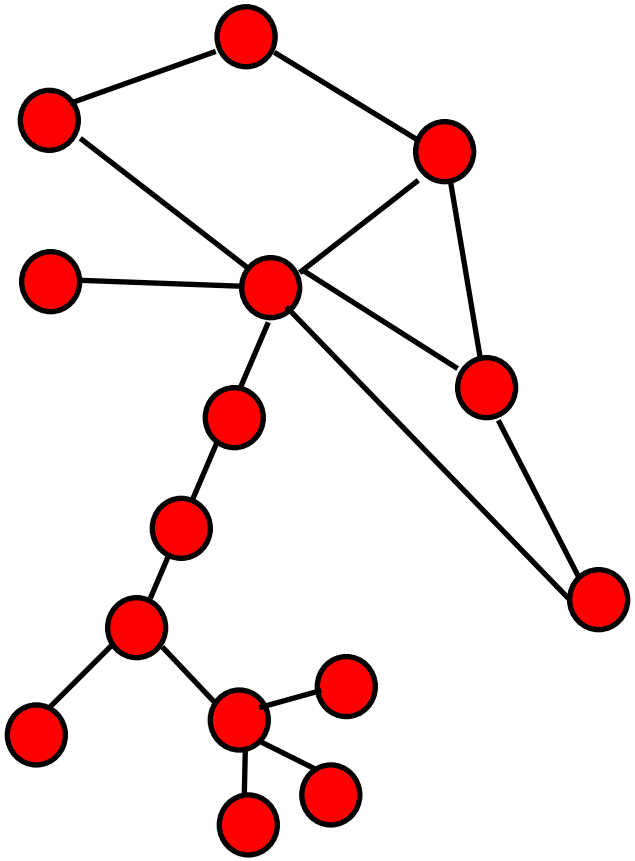


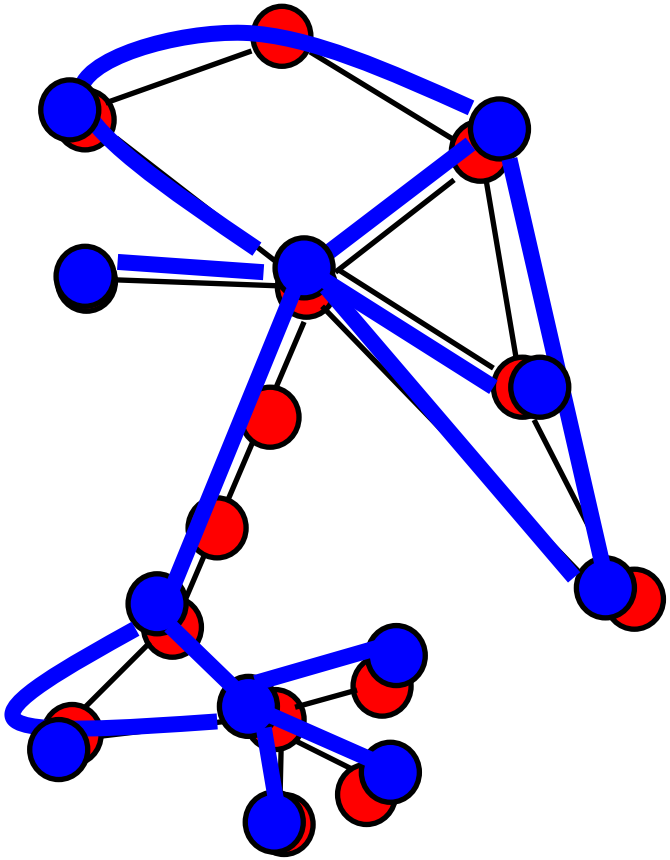
2. Expand knitting
- add nodes along virtual edges

General Recipe.

1. Find «knitting»:

- increasing connectivity too late comes at high cost!
- so first find graph structure of connected «motifs»
- motif = 2-connected components





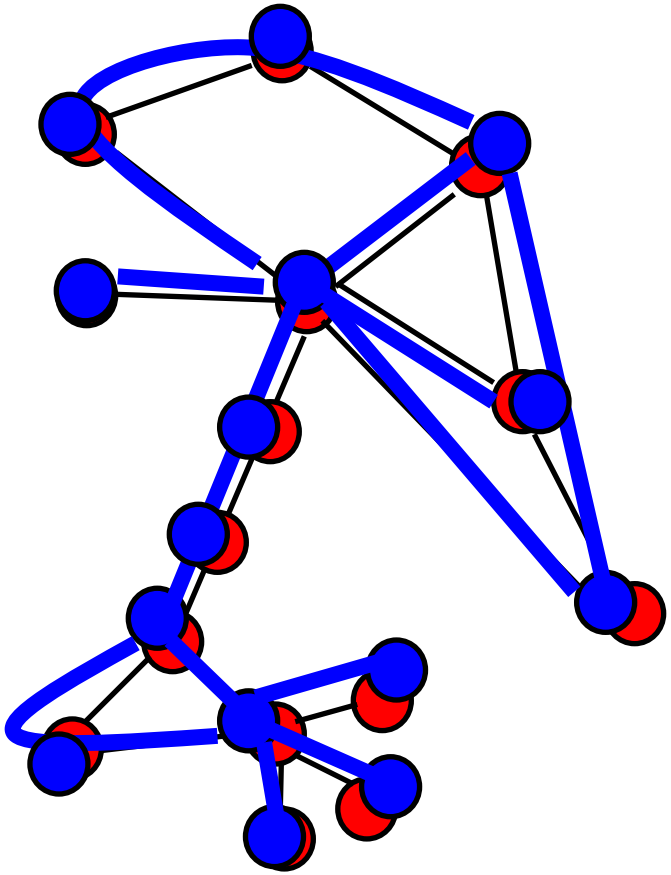
2. Expand knitting

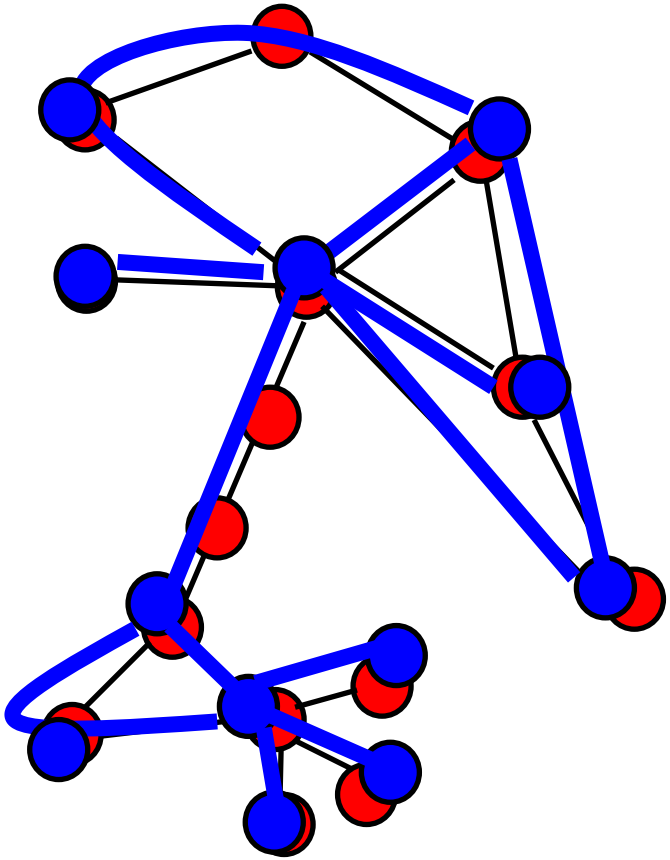
- add nodes along virtual edges

General Recipe.

1. Find «knitting»:

- increasing connectivity too late comes at high cost!
- so first find graph structure of connected «motifs»
- motif = 2-connected components





2. Expand knitting

- greedily add nodes along virtual edges

Dictionary Attack: Expansion Framework.

Motif

Basic "knittings" of the graph.

Poset

Partially ordered set: embedding relation fulfills reflexivity, anti-symmetry, transitivity.

Framework

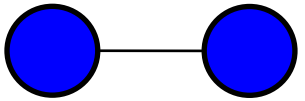
Explore branches according to dictionary order, exploiting poset property.

Dictionary

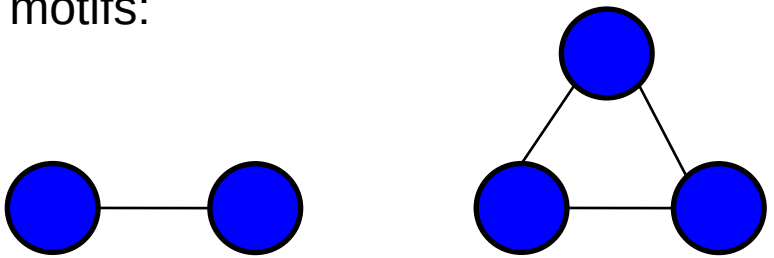
Define an order on motif sequences: Constraints on which sequence to ask first in order not to overlook a part of the topology. (E.g., by embedding links across multiple hops.)

Examples

Tree motifs:



Cactus motifs:



Dictionary Attacks: Expand Framework.

Motif

Basic "knitti

equences:
ence to ask first
part of the
ling links across

Poset

Partially orde
relation fulfills
symmetry, tra

Framev

**Explore bra
to dictiona
exploiting**

Algorithm 5 Motif Graph Discovery DICT

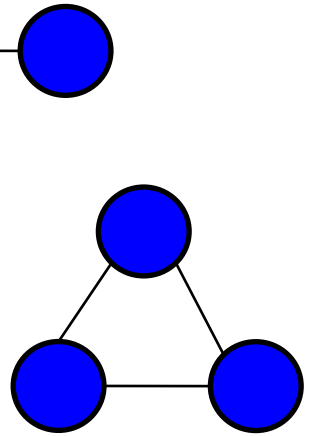
1: $H' := \{\{v\}, \emptyset\}$ /* current request graph */
2: $\mathcal{P} := \{v\}$ /* pending set of unexplored nodes*/
3: **while** $\mathcal{P} \neq \emptyset$ **do**
4: choose $v \in \mathcal{P}, T := find\_motif\_sequence(v, \emptyset, \emptyset)$
5: **if** $T \neq \emptyset$ **then**
6: $H' := H'vT$, add all nodes of T to $\mathcal{P}$
7: **for all** $e \in T$ **do** $edgeExpansion(e)$
8: **else**
9: remove v from $\mathcal{P}$

find_motif_sequence$(v, T^<, T^>)$

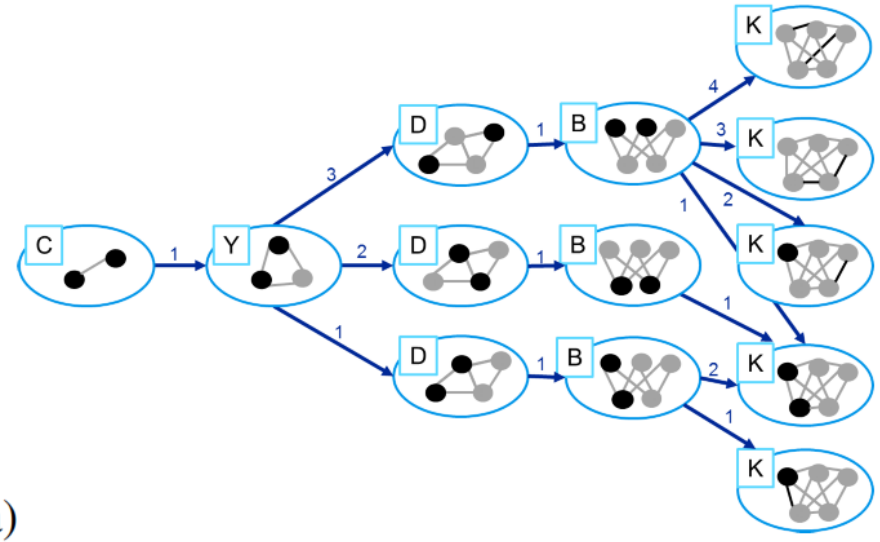
1: find max $i, j, \text{BF}, \text{AF}$ s.t.
$H'v\ (T^<)\ \text{BF}\ (D[i])^j\ \text{AF}\ (T^>) \mapsto H$ where
$\text{BF}, \text{AF} \in \{\emptyset, C\}^2$ /*issue requests*/
2: **if** $(i, j, \text{BF}, \text{AF}) = (0, 0, C, \emptyset)$ **then**
3: **return** $T^<CT^>$
4: **if** $\text{BF} = C$ **then**
5: $\text{BF} = find\_motif\_sequence(v, T^<, (D[i])^j\ \text{AF}\ T^>)$
6: **if** $\text{AF} = C$ **then**
7: $\text{AF} = find\_motif\_sequence(v, T^<\ \text{BF}\ (D[i])^j, T^>)$
8: **return** $\text{BF}\ (D[i])^j\ \text{AF}$

edge_expansion(e)

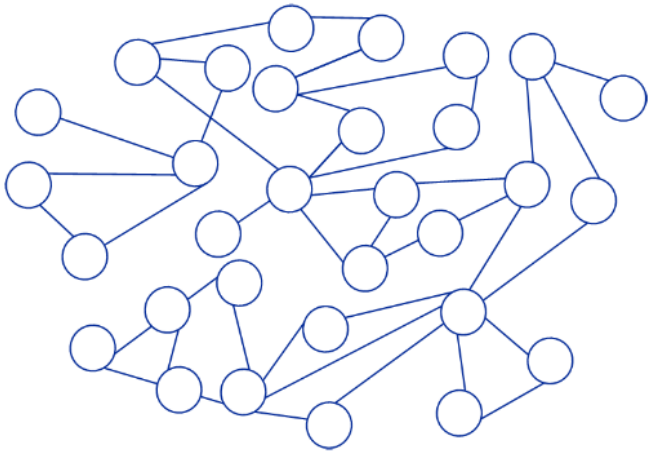
1: let u, v be the endpoints of edge e, remove e from H'
2: find max j s.t. $H'vC^ju \mapsto H$ /*issue requests*/
3: $H' := H'vC^ju$, add newly discovered nodes to $\mathcal{P}$



Example: Dictionary as a DAG.



a)

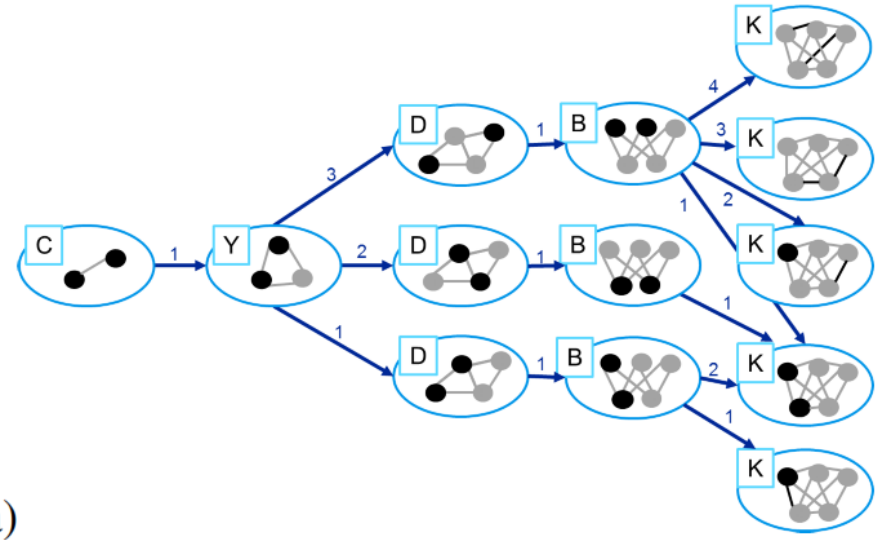


b)

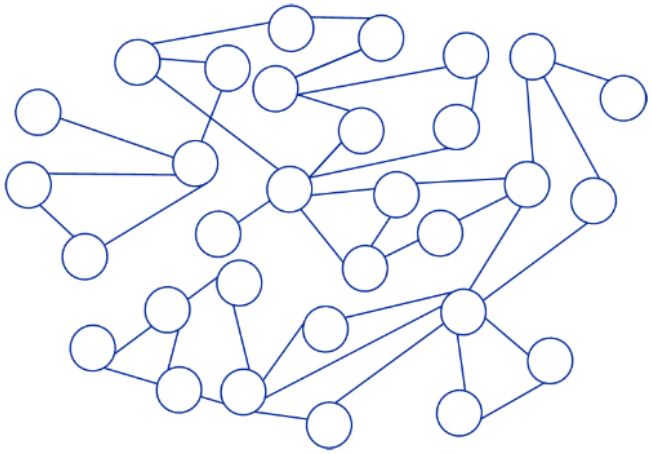
Dictionary is a directed acyclic graph: defines which motifs to ask first! (Ensure: no important part overlooked!)

Graph b) can be derived from dictionary a)!

Example: Dictionary as a DAG.



a)



b)

Request complexity depends on depth!
(E.g., number of motifs?)

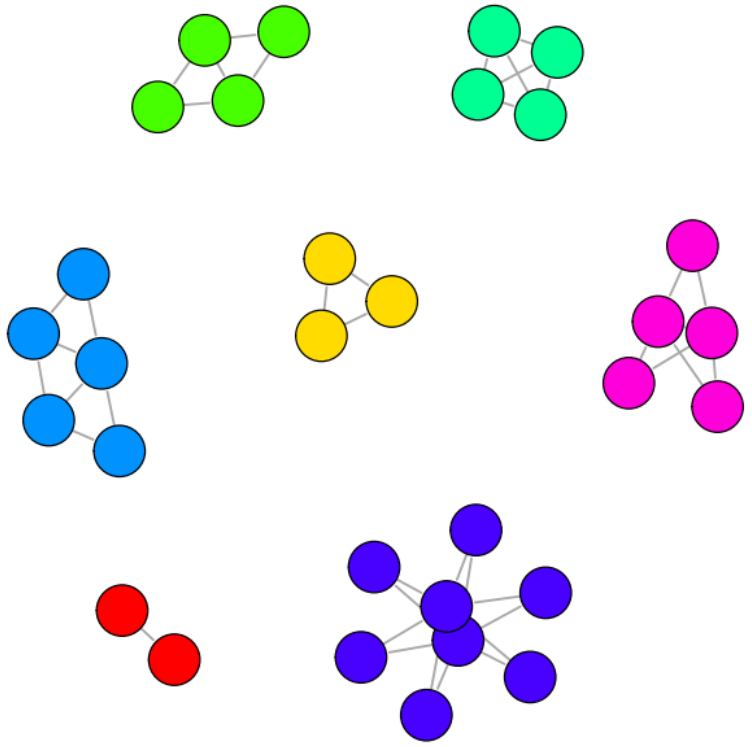
Dictionary is a directed acyclic graph: defines which motifs to ask first! (Ensure: no important part overlooked!)

Graph b) can be derived from dictionary a)!

Evaluation: Request Complexity in Real Graphs.

A small set of motifs is often sufficient to discover all or most of a topology! (Size of dictionary influences request complexity…)

E.g., motifs for autonomous system AS 1239 (Sprint) from Rocketfuel:



Conclusion.

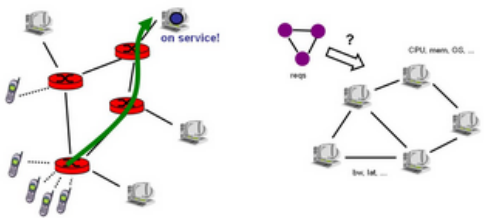
- **CloudNets:**
  - Elastic computing and networking
  - Federated architecture: embeddings
- **New security challenges**
- **With binary replies, graphs can not be exploited very efficiently (at least linear time)**
- **A first look at topology!**

The Project Website.

Combining Clouds with Virtual Networking

The CloudNet Project

Internet Network Architectures (INET)
TU Berlin / Telekom Innovation Labs (T-Labs)
Contact: Stefan Schmid



- News
- Overview
- People
- Magazines
- Publications
- Demo
- Talks/Posters

News

- Watch on YouTube: migration demonstrator video!
- We are looking for students and interns with good algorithmic background to contribute to Virtu! Contact us for more details or have a look at some open topics.

Project Overview

CloudNets are virtual networks (VNets) connecting cloud resources. The network virtualization paradigm allows to run multiple CloudNets on top of a shared physical infrastructure. These CloudNets can have different properties (provide different security or QoS guarantees, run different protocols, etc.) and can be managed independently of each other. Moreover, (parts of) a CloudNet can be migrated dynamically to locations where the service is most useful or most cost efficient (e.g., in terms of energy conservation). Depending on the circumstances and the technology, these migrations can be done live and without interrupting ongoing sessions. The flexibility of the paradigm and the decoupling of the services from the underlying resource networks has many advantages; for example, it facilitates a more efficient use of the given resources, it promises faster innovations by overcoming the ossification of today's Internet architecture, it simplifies the network management, and it can impove service performance.

We are currently developing a prototype system for this paradigm (currently based on VLANs), which raises many scientific challenges. For example, we address the problem of where to embed CloudNet requests (e.g., see [1] for online CloudNet embeddings and [2] for a general mathematical embedding program), or devise algorithms to migrate CloudNets to new locations (e.g., due to user mobility) taking into account the

Collaborators and Publications.

- **People**
  - **T-Labs / TU Berlin:** Anja Feldmann, Carlo Fürst, Johannes Grassler, Arne Ludwig, Matthias Rost, Gregor Schaffrath, Stefan Schmid
  - **Uni Wroclaw:** Marcin Bienkowski
  - **Uni Tel Aviv:** Guy Even, Moti Medina
  - **NTT DoCoMo Eurolabs:** Group around Wolfgang Kellerer
  - **LAAS:** Gilles Tredan
  - **ABB:** Yvonne Anne Pignolet
  - **IBM Research:** Johannes Schneider
  - **Arizona State Uni:** Xinhui Hu, Andrea Richa

- **Publications**
  - **Prototype:** J. Information Technology 2013, ICCCN 2012, ERCIM News 2012, SIGCOMM VISA 2009
  - **Migration:** ToN 2013, INFOCOM 2013, ICDCN 2013 + Elsevier TCS Journal, Hot-ICE 2011, IPTComm 2011, SIGCOMM VISA 2010
  - **Embedding:** INFOCOM 2013 (Mini-Conference), CLOUDNETS 2012, 2 x ACM UCC 2012, DISC 2012, ICDCN 2012 (*Best Paper Distributed Computing Track*)

Contact.

Dr. Stefan Schmid

Telekom Innovation Laboratories
Ernst-Reuter-Platz 7, D-10587 Berlin
E-mail: stefan@net.t-labs.tu-berlin.de
Project website:
http://www.net.t-labs.tu-berlin.de/~stefan/virtu.shtml