CloudNets: Combining Clouds with Virtual Networking

Stefan Schmid

December, 2013

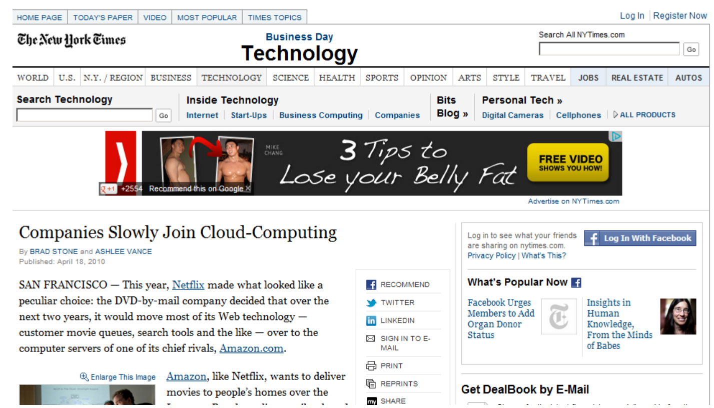
CloudNets: Combining Clouds with Virtual Networking

Stefan Schmid

December, 2013

Vision: Virtual Networking Cloud Resources.

Cloud computing is a big success!
But what is the point of clouds if they cannot be accessed?



Next Natural Step for Virtualization!

Success of Node Virtualization

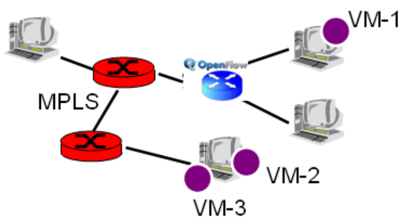
- a.k.a. end-host virtualization
- VMWare revamped server business
- OpenStack
- VM = flexible allocation, migration..
- «**Elastic computing**»

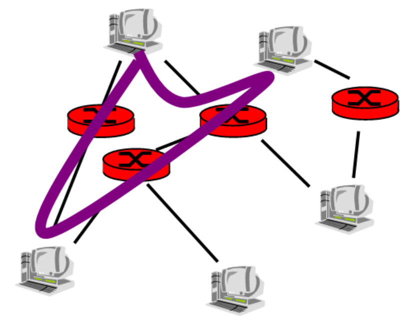
Trend of Link Virtualization

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

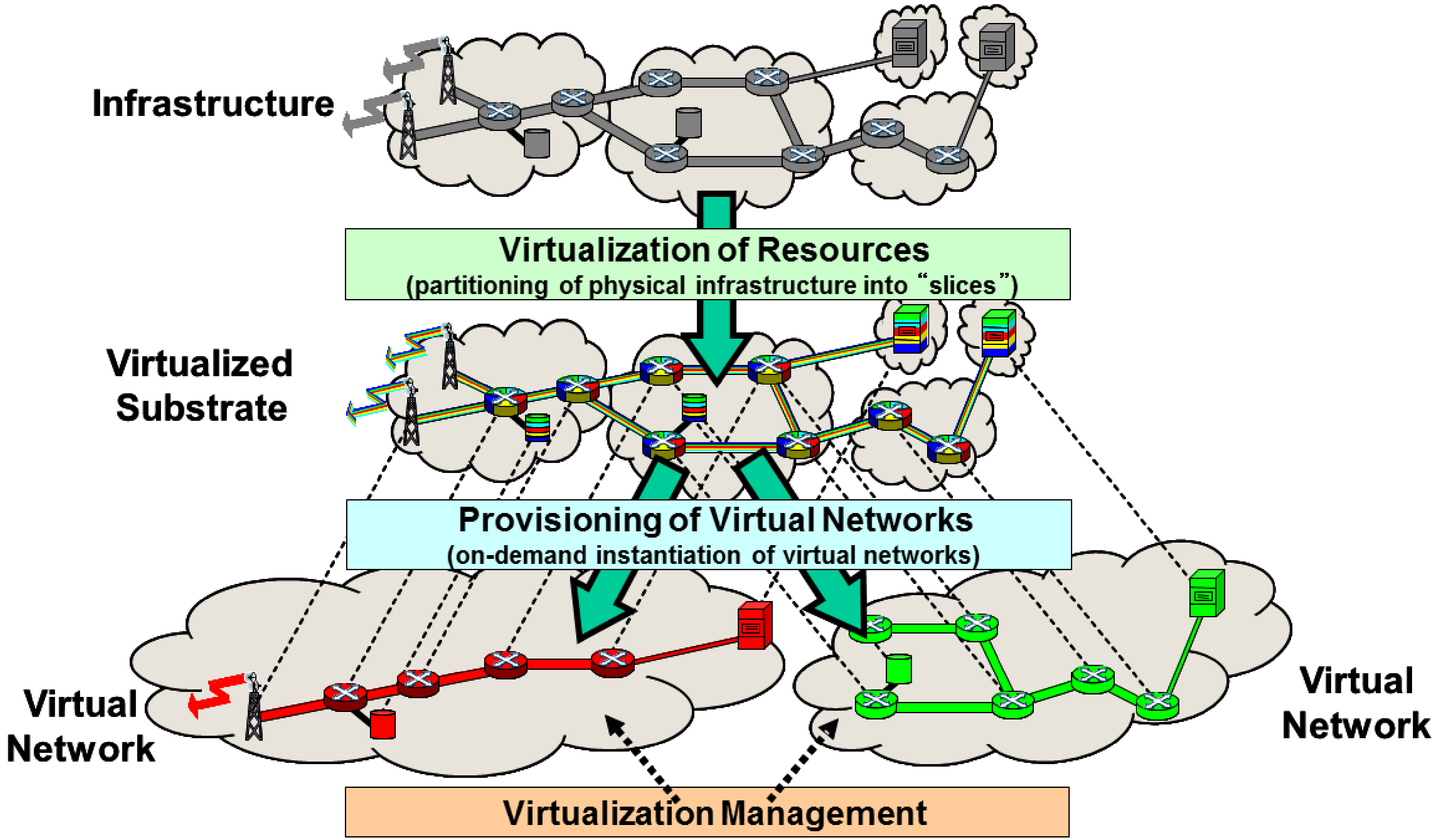
Unified, fully virtualized networks: **CloudNets**

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





The Vision: Sharing Resources.

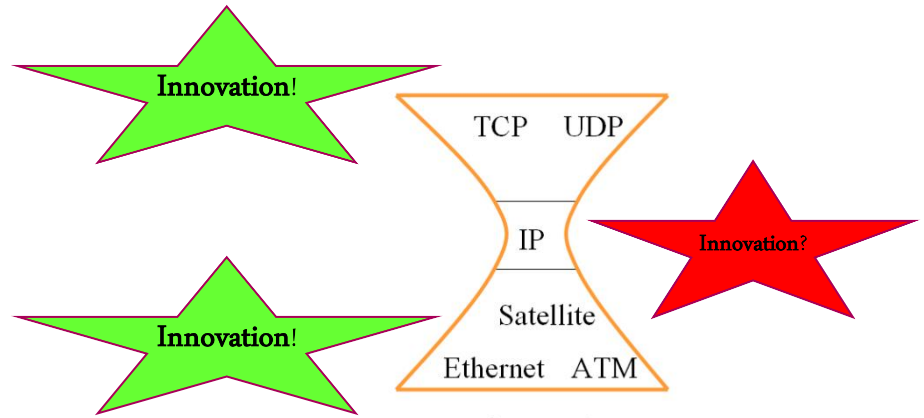


For Tomorrow's Internet…

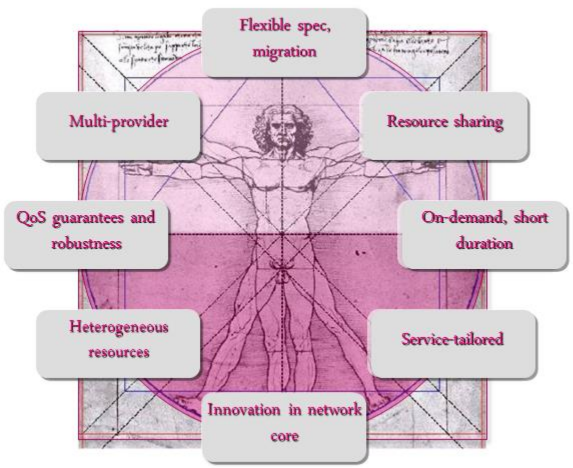
ERCIM News 2012

Today: Internet is one solution for everything!

CloudNets: Flexibly specifiable virtual networks, executing different **protocol stacks**, **cohabiting** the same substrate.



IP hourglass



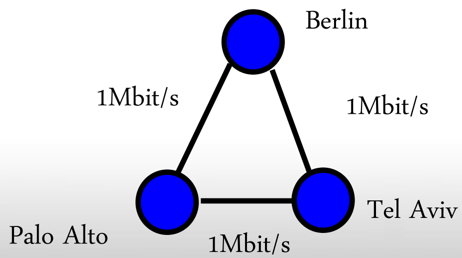
- Vision: **facilitate innovation** in network core
  - Make **network core programmable** (e.g., own intrusion detection system)
  - **Service-tailored** networks (for social networks, bulk data transfer, life streaming, etc.)
  - Co-existing **virtual networks** with QoS guarantees
  - No dependencies on IPv4, BGP, ...

Requests with Flexible Specification.

Optimization and Migration?

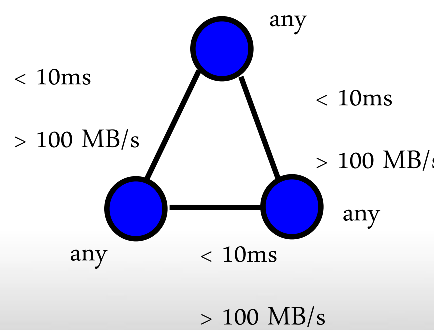
„VPN++“

Goal: *Fully specified CloudNet mapping constraints (e.g., end-points for a **telco**), but with **QoS guarantees** (e.g., bandwidth) along links*



„November 22, 1pm-2pm!“

Datacenters



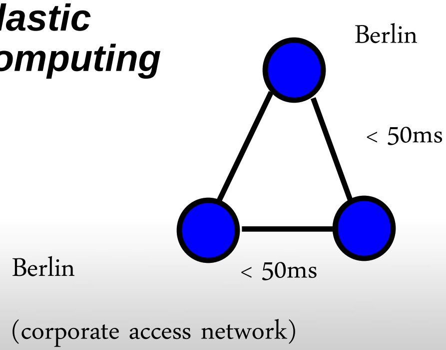
„Guaranteed resources, job deadlines met, no overhead!“

“Network may delay execution: costly for per hour priced VM!”

See, e.g., Octopus system (SIGCOMM 2011)

Spillover/Out-Sourcing

Elastic computing

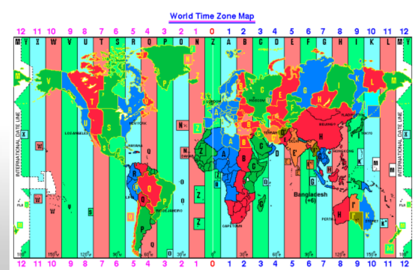


„50 TB storage, 10 Tflops computation!“

„any European cloud provider (e.g. due to legal issues?)“

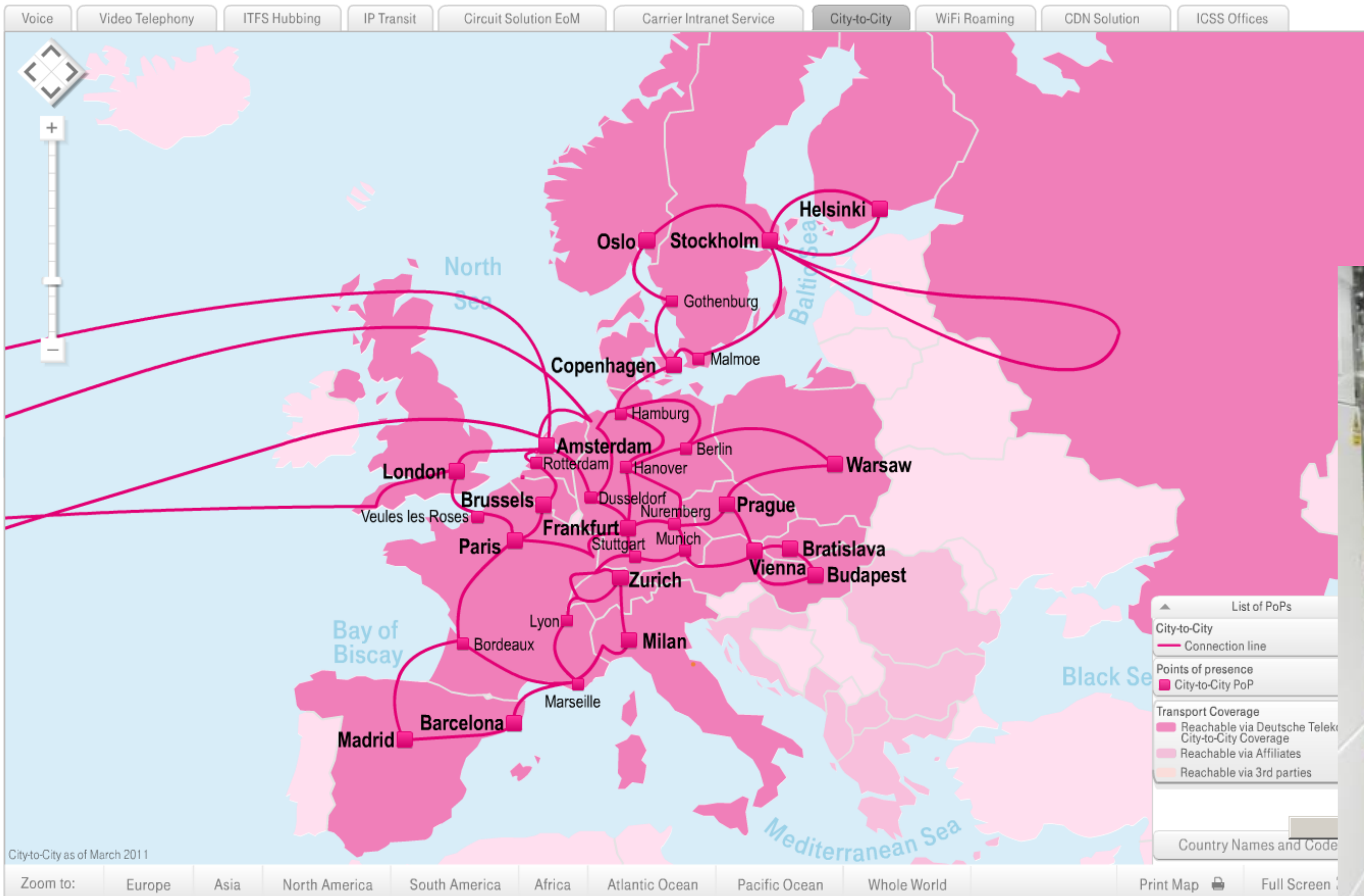
Migration / Service Deployment

Goal: *Move with the sun, with the commuters, (QoS) allow for **maintenance**, avoid roaming costs…: e.g., **SAP/game/translator server, small CDN server…***



Vision: Not only in data centers, but WAN

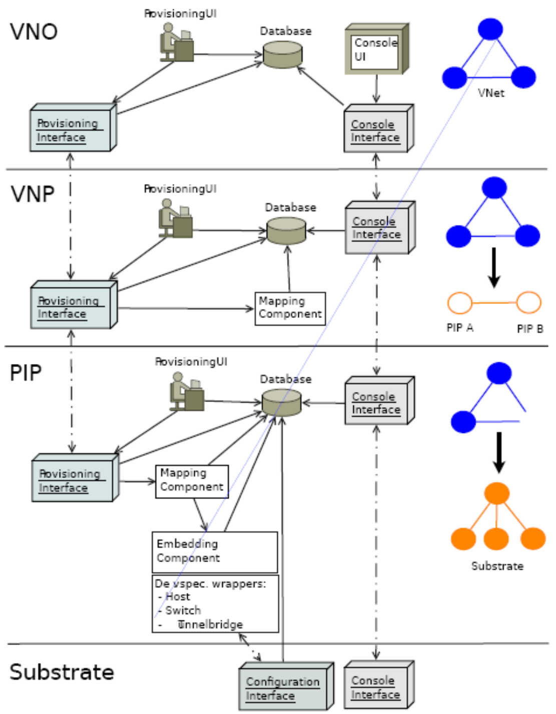
ISP network with resources at Points-of-Presence («nano datacenters»):
For service deployment!



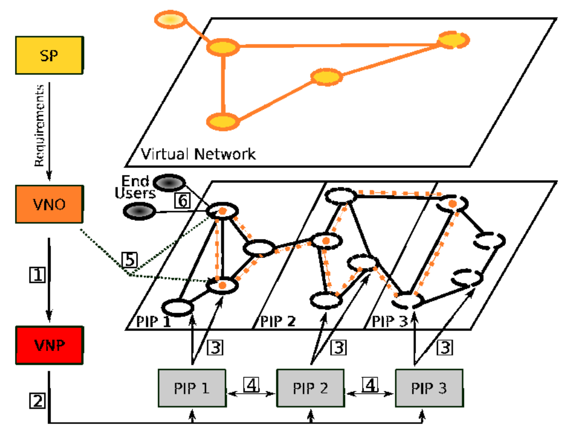
Opens New Business Roles.

Focus of our work architecture (unlike, e.g., single authority in GINI testbed), on multitude of players, providers, ...! (Of course, cross-layer infos if all same company...)

Roles



- Service Provider (SP): uses CloudNets to offer its services (streaming, OSN, CDN, ...): e.g., value-added application CloudNet, or transport CloudNet
- Virtual Network Operator (VNO): Installs and operates CloudNet over topology provided by VNP, offers tailored connectivity service, triggers cross-provider migration (by setting requirements), ...
- Virtual Network Provider (VNP): „Broker"/reseller that assembles virtual resources from different PIPs to provide virtual topology (no need to know PIP, can be recursive, ...)
- Physical Infrastructure Provider (PIP): Owns and manages physical infrastructure



QoS from PIP up to VNO or service provider: accounting via complete **set of contracts!** (unlike „sending party pays")

Federated CloudNet Architecture.

New Business Roles.

As in Internet today:

Netflix, Google, World of Warcraft…

As in Internet today: Telekom, AT&T, …

+ resource control interface (bootstrapping etc.)

Roles in CloudNet Arch.

Service Provider (SP)

(offers services over the top)

Virtual Network Operator (VNO)

(operates CloudNet, Layer 3+, triggers migration)

Virtual Network Provider (VNP)

(resource broker, compiles resources)

Physical Infrastructure Provider (PIP)

(resource and bit pipe provider)

Federated CloudNet Architecture.

SIGCOMM VISA 2009

New Business Roles.

As in Internet today:

Netflix, Google, World of Warcraft…

As in Internet today: Telekom, AT&T, …

+ resource control interface (bootstrapping etc.)

Roles in C udNet Arch.

(offe he top)

knows application

Virtual Network Operator (VNO)

(operates CloudNet, Layer 3+, triggers migration)

Virtual Networ Provider (VNP)

(resource broke iles resources)

Physical Infra e Provider (PIP)

knows network (uses resources at PoPs!)

Federated CloudNet Architecture.

New Business Roles.

Roles in CloudNet Arch.

Service Provider (SP)

(offers services over the top)

Virtual Network Operator (VNO)

(operates CloudNet, Layer 3+, triggers migration)

Virtual Network Provider (VNP)

(resource broker, compiles resources)

Physical Infrastructure Provider (PIP)

(resource and bit pipe provider)

Provide layer 2: assembles CloudNets, resource and management interfaces, provides indirection layer, across PIPs!

View: PIP graph

Resource broker… (recursive)

A virtualized infrastructure opens new roles for the allocation of resources and the operation of networks!

Federated CloudNet Architecture.

SIGCOMM VISA 2009

New Business Roles.

re
provid ... s!
View: PIP gr
Resource

PIP graph view: cross-provider QoS contracts

Roles in CloudNet Arch.

Service Provider (SP)
(offers services over the top)

Virtual Network Operator (VNO)
(operates CloudNet, Layer 3+, triggers migration)

Virtual Network Provider (VNP)
(resource broker, compiles resources)

Physical Infrastructure Provider (PIP)
(resource and bit pipe provider)

A virtualized infrastructure opens new roles for the allocation of resources and the operation of networks!

Federated CloudNet Architecture.

New Business Roles.

Build upon layer 2 (op on virt IDs): clean slate! (OSN, …)

Routing, addressing, multi-path/redundancy… (view inside CloudNet!)

Trigger migration (use provisioning interface of SP or live)

Roles in CloudNet Arch.

Service Provider (SP)

(offers services over the top)

Virtual Network Operator (VNO)

(operates CloudNet, Layer 3+, triggers migration)

Virtual Network Provider (VNP)

(resource broker, compiles resources)

Physical Infrastructure Provider (PIP)

(resource and bit pipe provider)

A virtualized infrastructure opens new roles for the allocation of resources and the operation of networks!

Federated CloudNet Architecture.

SIGCOMM VISA 2009

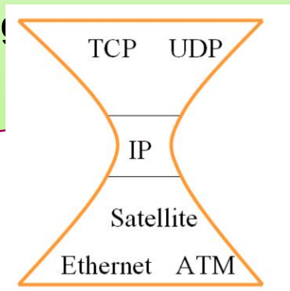
New Business Roles.

Bu… …, clean slate!
(OSN, …)

Innovation!

Routing, … path/redundancy…
(view in… CloudNet!)

Trigger mi… visioning interface of
SP or live)



IP hourglass

**IP hour glass:
innovation only on
low and high layers**

Roles in CloudNet Arch.

Service Provider (SP)

(offers services over the top)

Virtual Network Operator (VNO)

(operates CloudNet, Layer 3+, triggers migration)

Virtual Network Provider (VNP)

(resource broker, compiles resources)

Physical Infrastructure Provider (PIP)

(resource and bit pipe provider)

A virtualized infrastructure opens new roles for the allocation of resources and the operation of networks!

Federated CloudNet Architecture.

SIGCOMM VISA 2009

New Business Roles.

Roles in CloudNet Arch.

Service Provider (SP)
(offers services over the top)

Virtual Network Operator (VNO)
(operates CloudNet, Layer 3+, triggers migration)

Virtual Network Provider (VNP)
(resource broker, compiles resources)

Physical Infrastructure Provider (PIP)
(resource and bit pipe provider)

Contract based:
Communicate requirements
/specs down the hierarchy

API

API

API

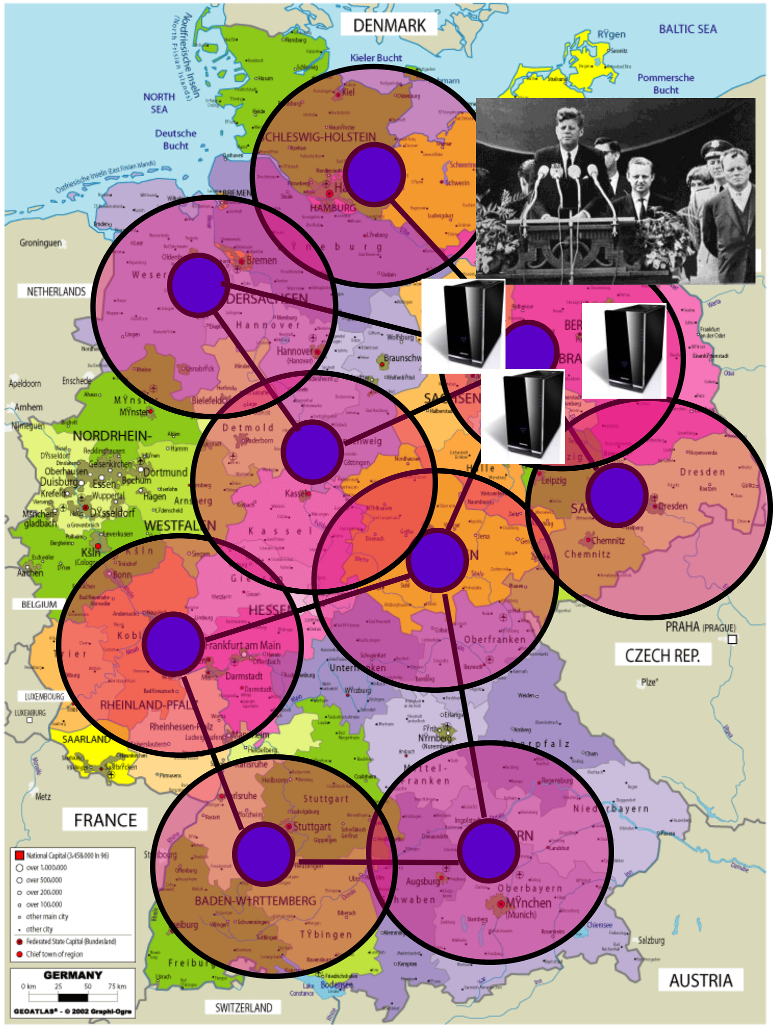
APIs: e.g., provisioning
interfaces (migration)

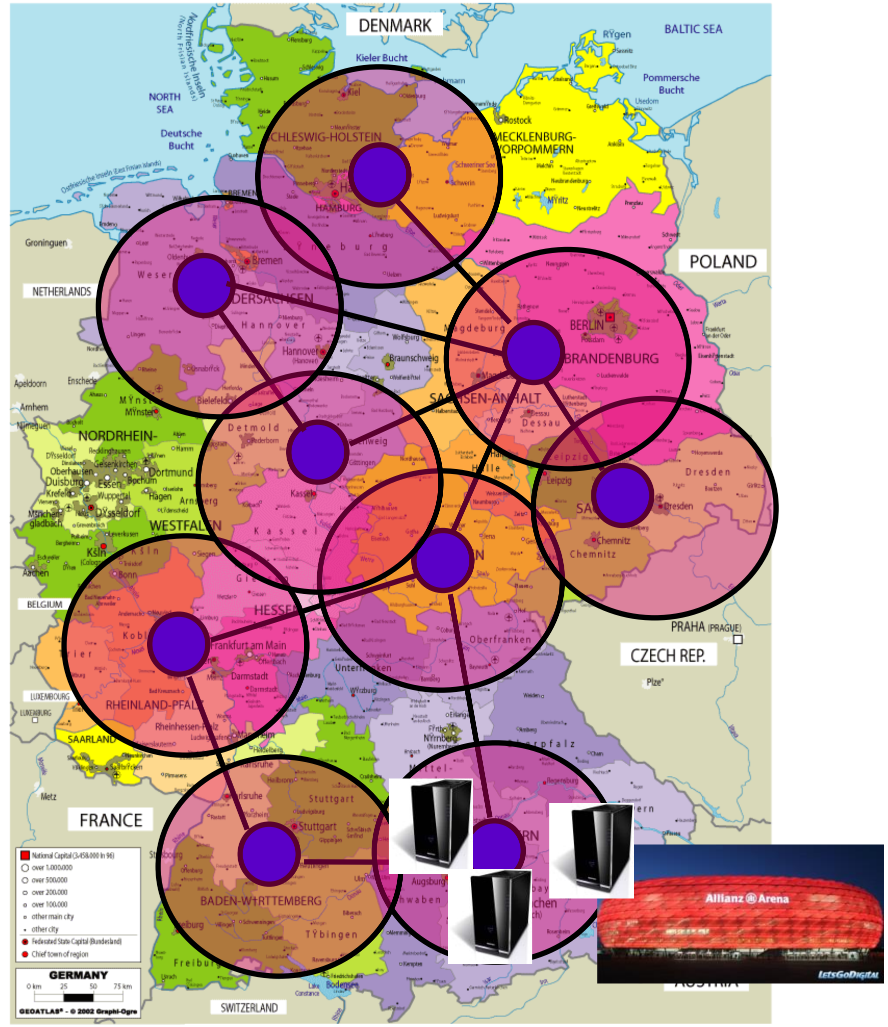
Roles: Zoom In.

- **Service Provider (SP):**
  - Netflix, Akamai, startup, ..
  - Specifies resources abstractly (latency, push content to customer...)
  - If services migratable: offer provisioning interface (API that VNO can call)
- **Virtual Network Operator (VNO):**
  - Implements layer 3 on top of VNP layer 2: addressing, routing, ...
  - Decides how to realize SP specification: use redundant physical paths => additional specs
  - To fulfill specs, calls API of SP to migrate (if application migratable), or makes live migration itself (not to violate specs)
  - Clean slate architecture possible!
- **Virtual Network Provider (VNP):**
  - Builds layer 2
  - „Broker"/reseller that assembles virtual resources from different PIPs to provide virtual topology (no need to know PIP, can be recursive, ...)
- **Physical Infrastructure Provider (PIP):**
  - Bit-pipe provider, owns and manages physical infrastructure

Use Cases (1): Migrate Resources.

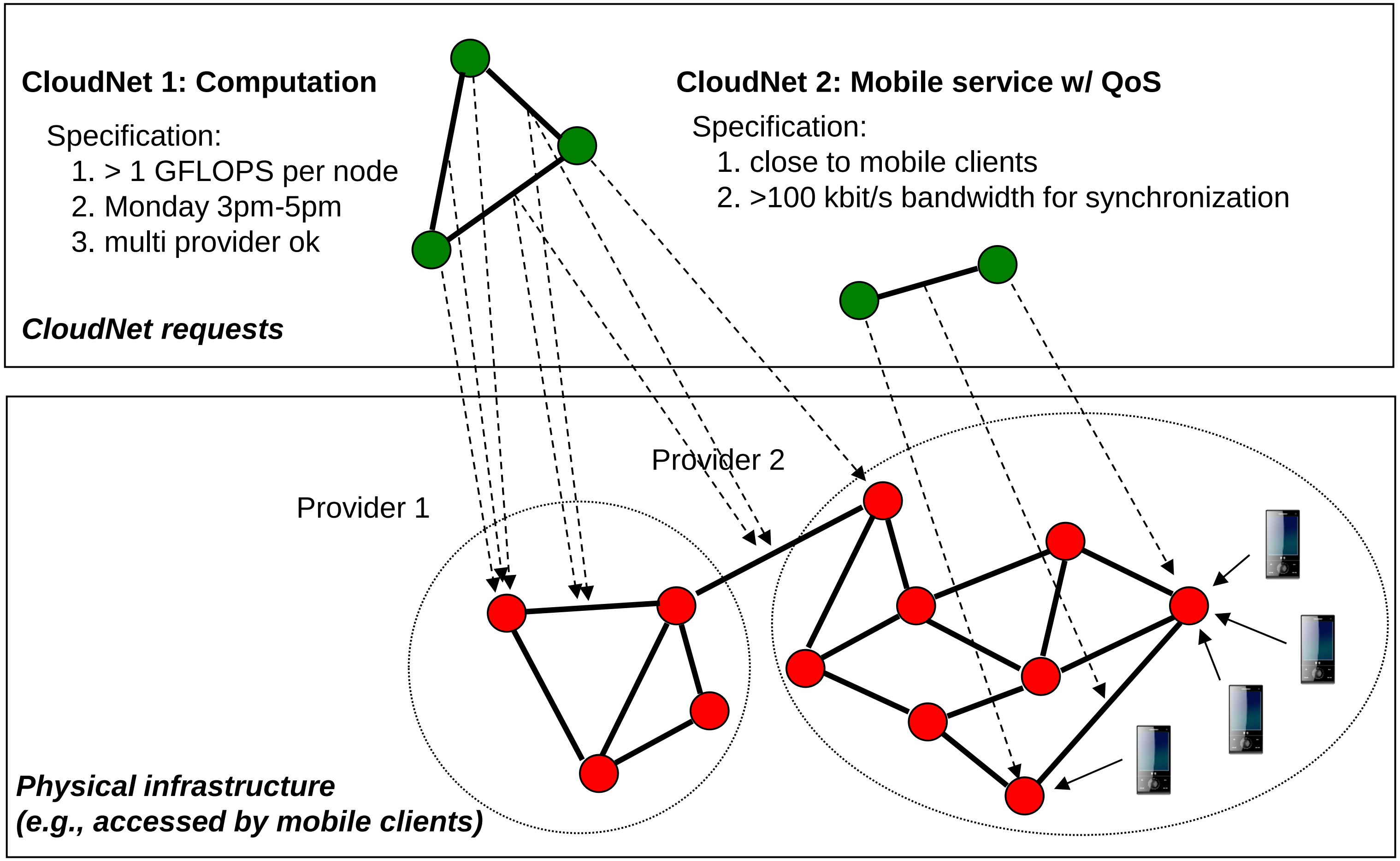
Resource allocation and migration where needed (energy saving otherwise!).





Use Cases (2).

Connecting Providers (Geographic Footprint).



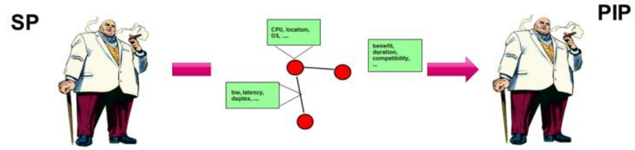
Research Overview.

ICCCN 2012

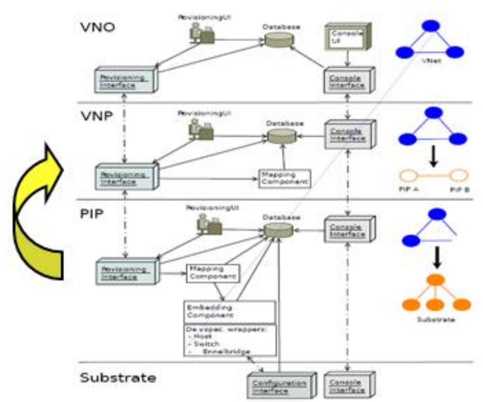
Flexible Specification

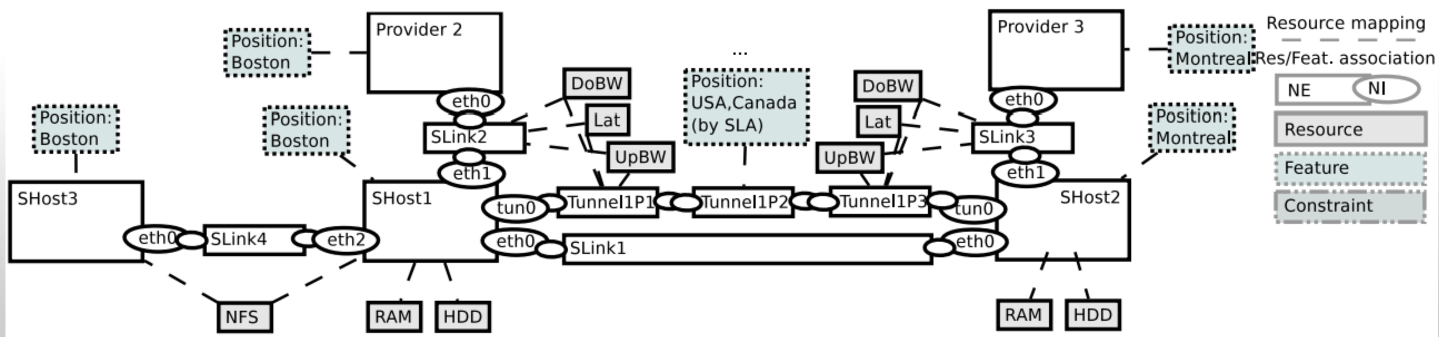
How to store and communicate CloudNets?

- Specify without losing flexibility
- Communicate non-topological requirements: consistency across multiple roles?
- Allow for aggregation and abstraction



aggregate
resources



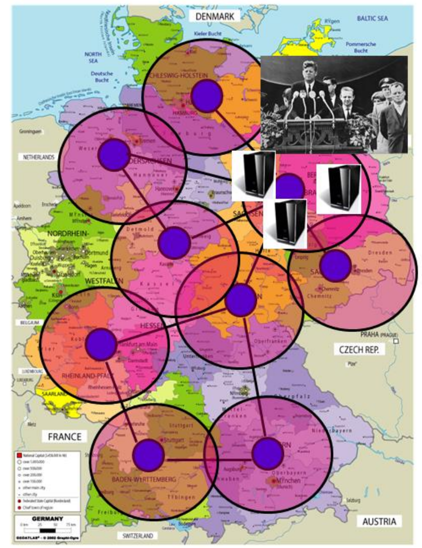


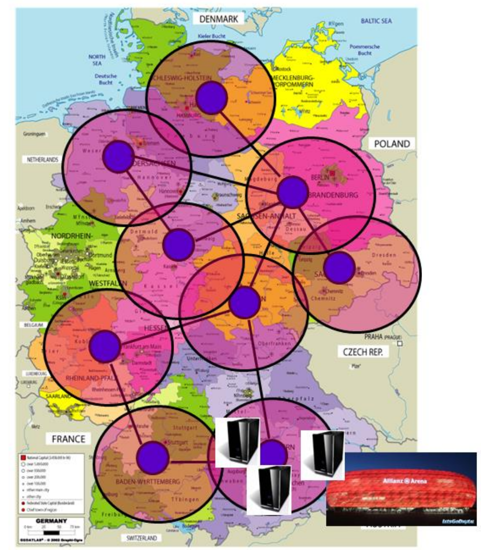
Research Overview.

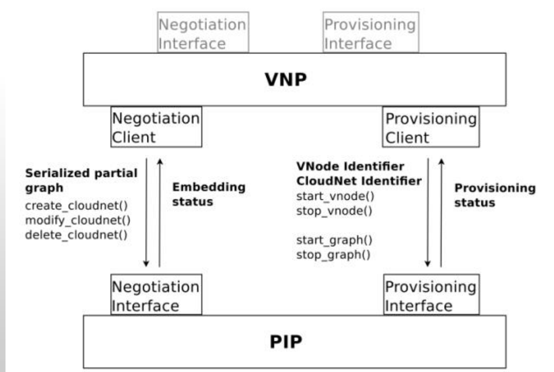
Prototype

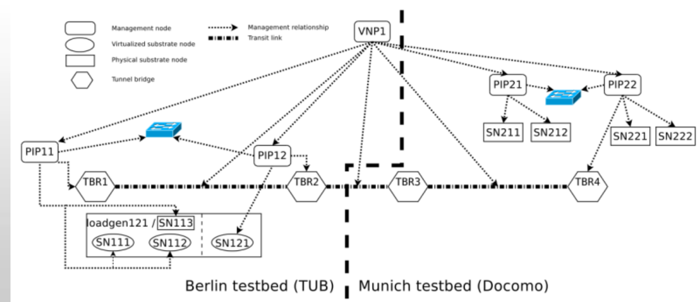
Plugin architecture

- Own cloud operating system
- Currently: VLAN based, but OpenFlow plugin started
- Provisioning interfaces, negotiation interfaces
- PIP and VNP role implemented
- Seamless migration, e.g., of streaming service
- Wide-area: OpenVPN tunnels
- Wide-area testbed: Munich (NTT Docomo) and Berlin







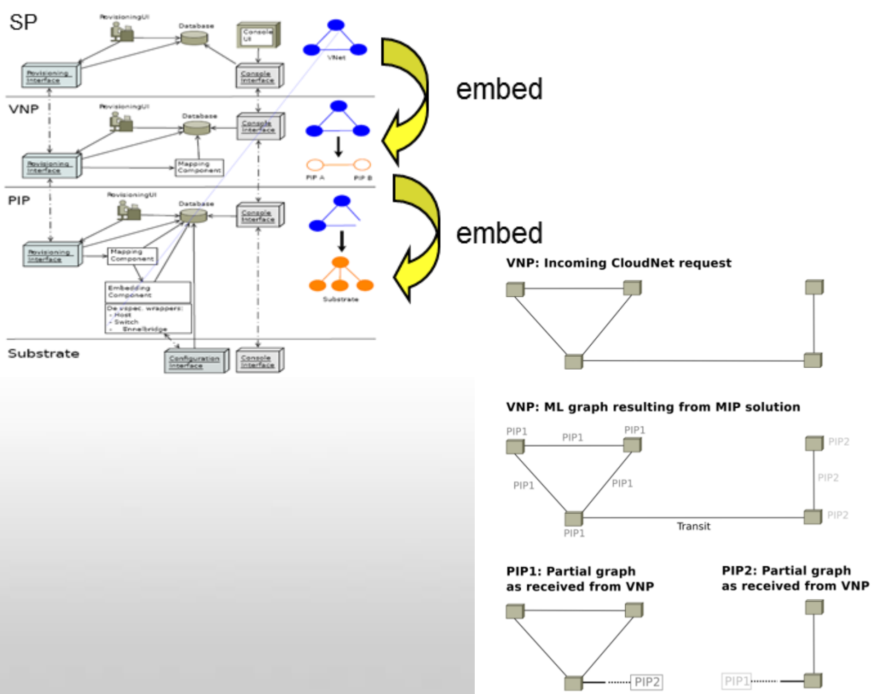


Research Overview.

Embedding

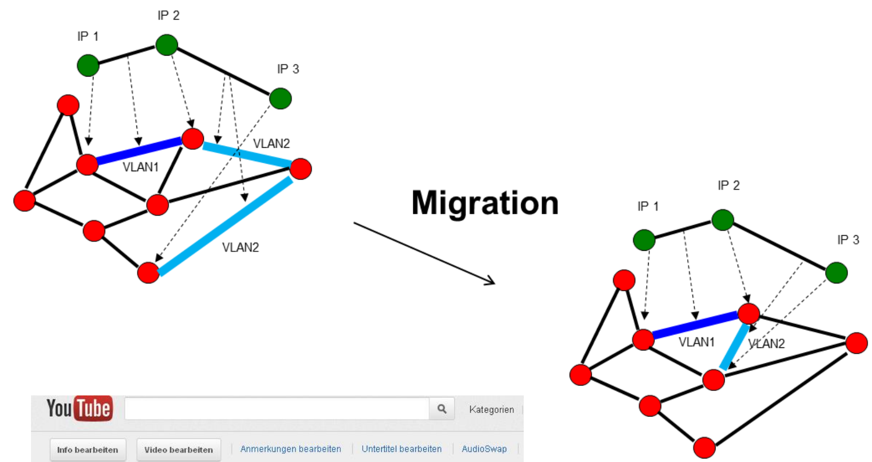
Two steps:

- Quick heuristic (spec => greedy)
- Optimizing “long-lived” or “heavy hitter” CloudNets only (mixed integer program)



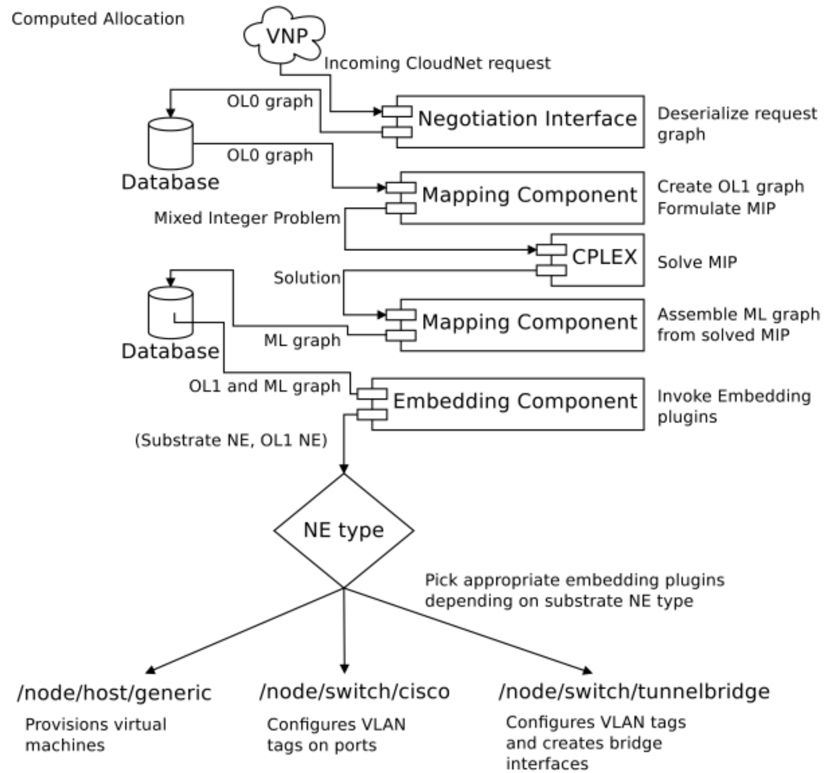
Migration

E.g., move VM by reconfiguring VLANs



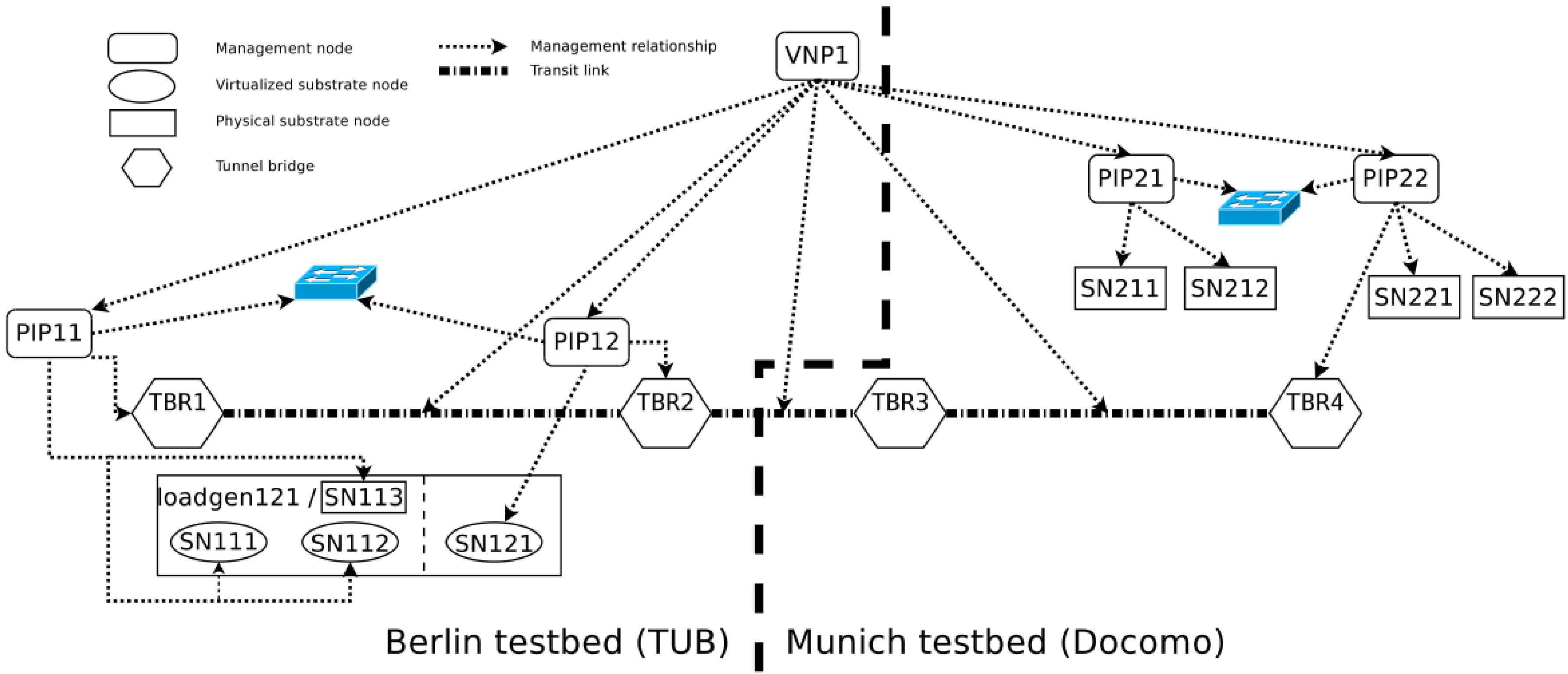
Architecture / Anatomy of Prototype.

- **Plugin based**
  - VLink technology (VLAN, OpenFlow, MPLS, ...)
  - Embedding algorithm (two stage!)
  - Cloud operating system
- **Prototype: proof-of-concept (flexibility&generality rather than speed)**
  - VLink plugin: VLANs (VPN tunnel for WAN)
  - Embedding: MIP
  - Cloud OS: own implementation
- **Plan: other plugins, OpenStack, ...**



Testbed.

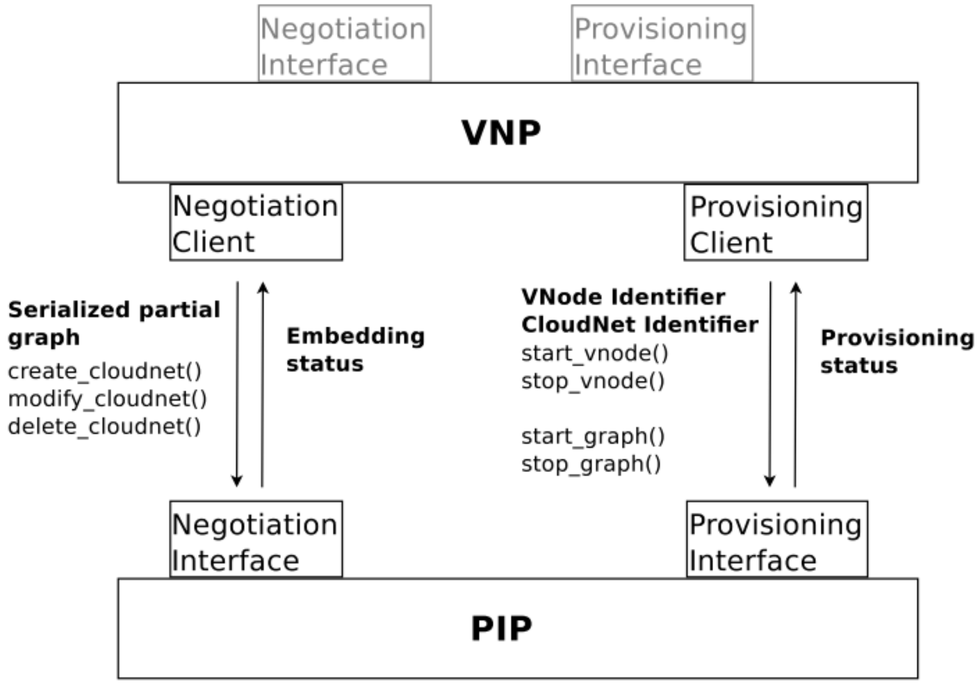
Two sites: TU Berlin and NTT Docomo Eurolabs Munich



The Prototype (1).

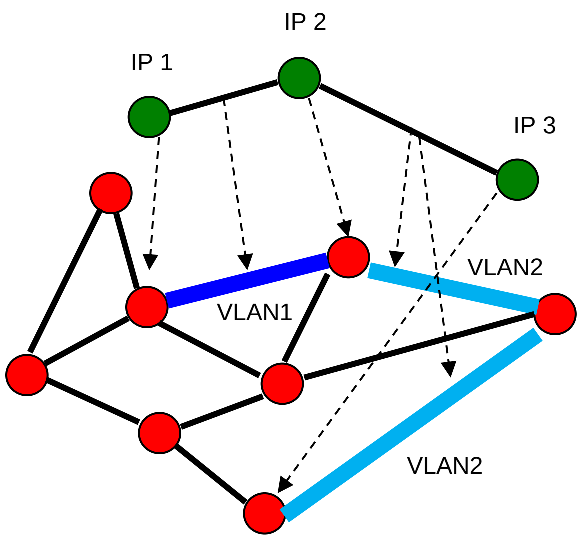
State of the Art

- Prototype based on KVM+Xen (nodes) and VLANs (links)
- Layer 2 comform: no need for **end-to-end** / routing (routing inside CloudNet only)
- Different VNets may have same internal virtual node addresses
  - VLAN ensures isolation
- PIP and VNP implemented

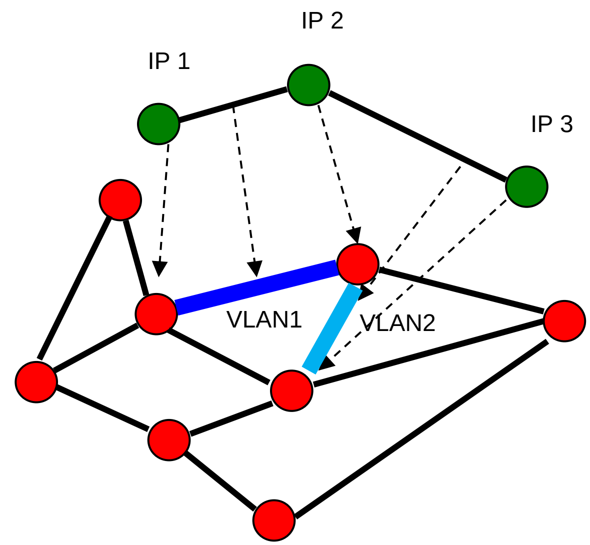


The Prototype (2).

- Each virtual link is a VLAN (broadcast domain)
- Migration: reconfigure VLANs, not addresses of virtual nodes!
- Transparent for users...



Migration

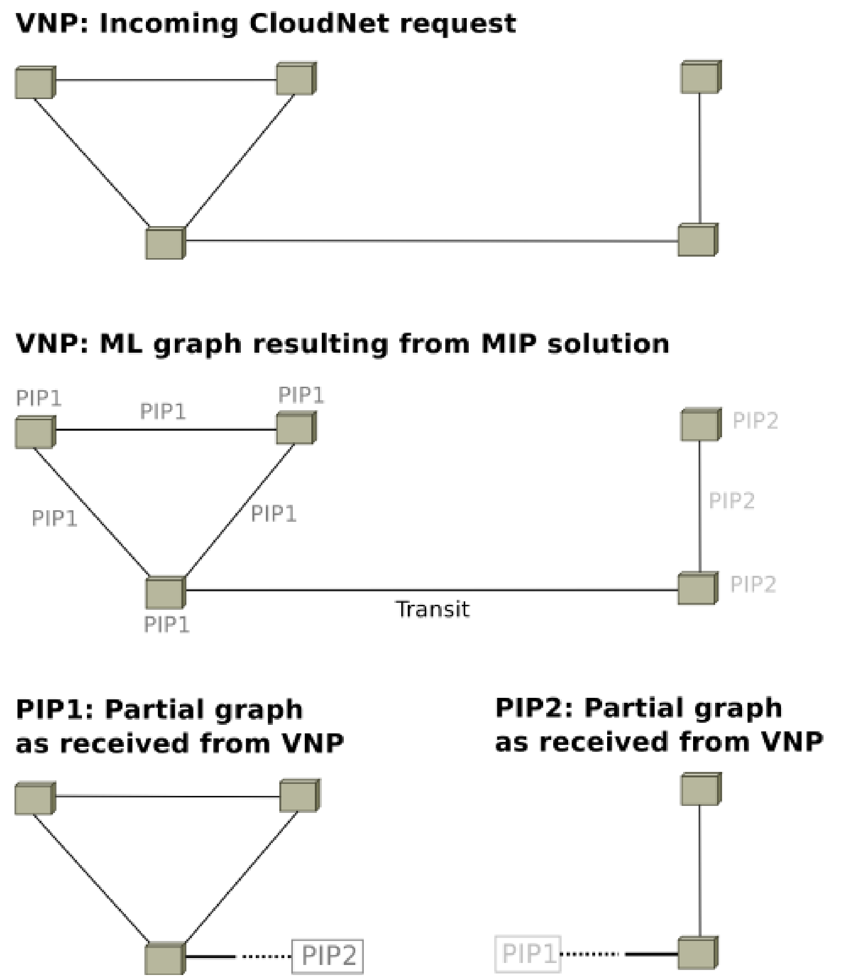


- Open vSwitch supports VLAN bridging
- Demultiplexing eth1 plus VLAN-tag to port in kernel
- To VM looks like Ethernet (no VLAN)

The Prototype (3).

Life of a CloudNet request:

- Topology broken down for PIPs
- Transit link (tunnel bridge and OpenVPN)
    - One VPN tunnel for control plane
    - One VPN tunnel for data plane
- To VM looks like Ethernet (no VLAN)

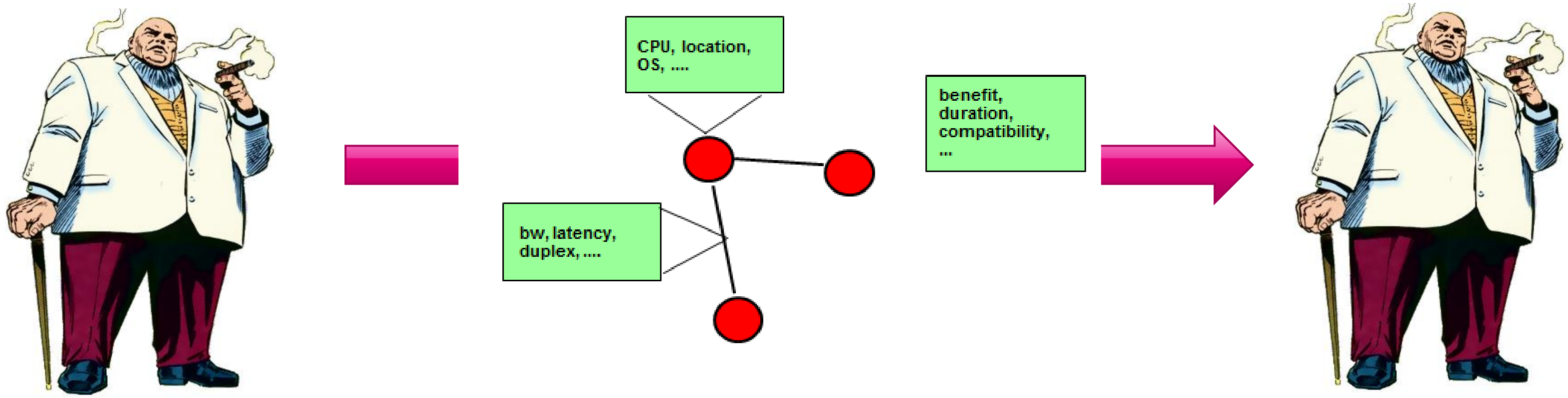


Need for a Language.

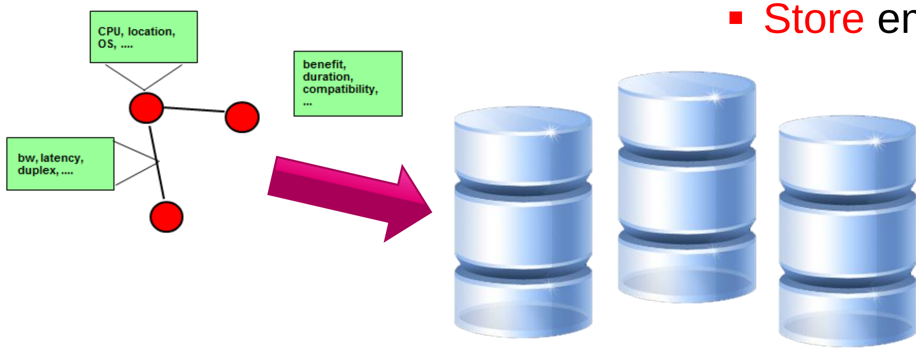
- Communicate CloudNets, substrate resources and embeddings to business partners or customers:

SP

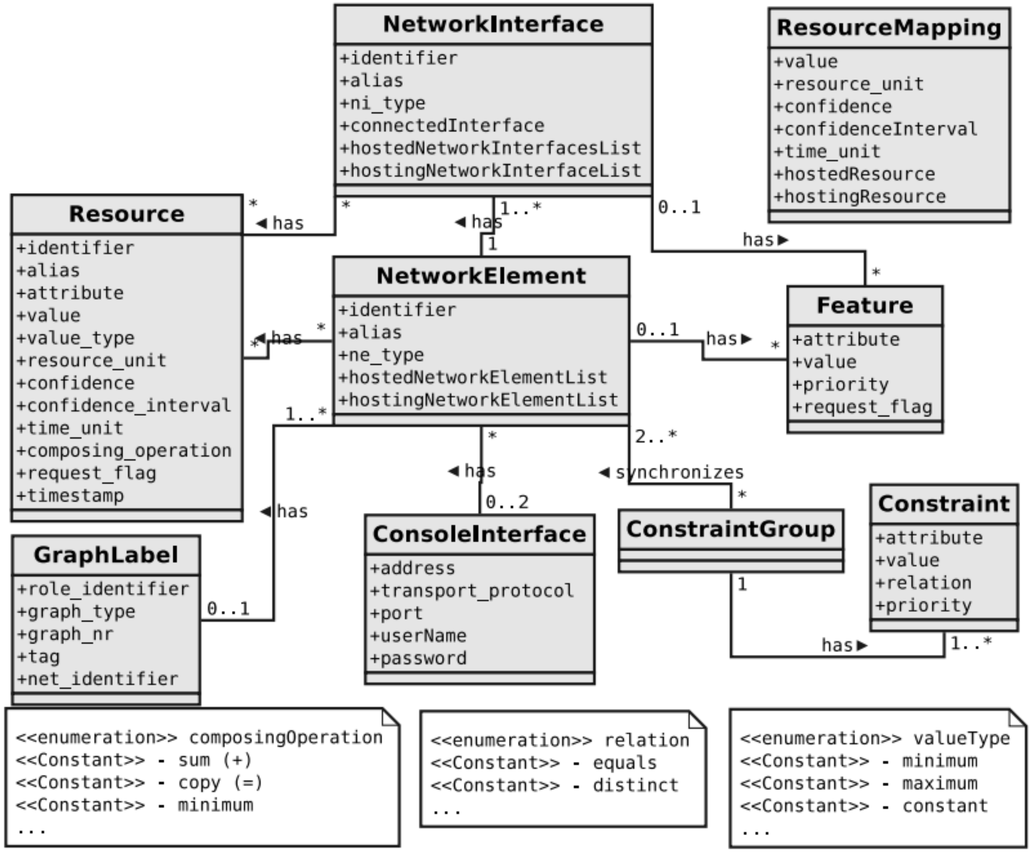
PIP



- Store embedding state internally:



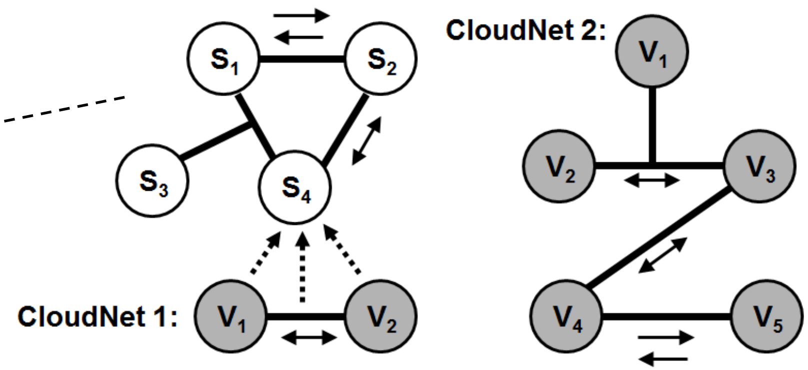
Exploiting Flexibilities: Resource Description Language.

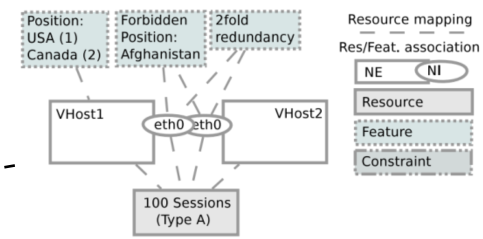


- Network Elements = nodes or links!
  - Connected via Network Interfaces
- Support for omission
- Support for multicast links
- Support for white and black lists...

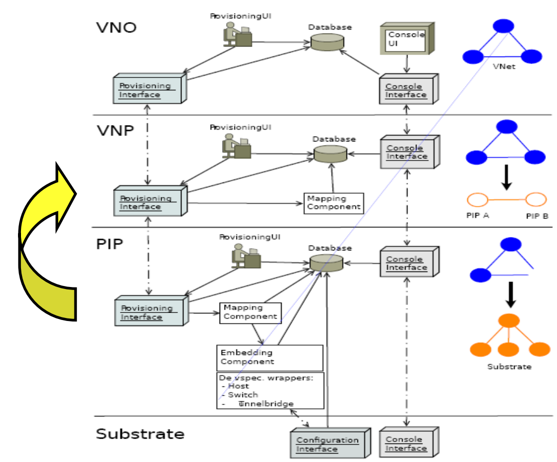
FleRD Requirements.

- Support all kinds of node (storage, computation, ...) and link (latency, bw, full-duplex/asymmetric, ...) resources (heterogeneity)
- Extensible, allow for syntactic changes over time, no need for global agreement on semantic values
- Facilitate resource leasing and allow PIPs to open abstract views on their substrate
- Allow for vagueness and omission: customers are unlikely to specifiy each CloudNet detail (e.g., KVM or Xen is fine, outsource to any European cloud provider): this opens ways for optimization (exploiting flexibilities)!
- Allow for aggregation of resources (business secret?)
- Non-topological requirements (e.g., wordsize compatibility)



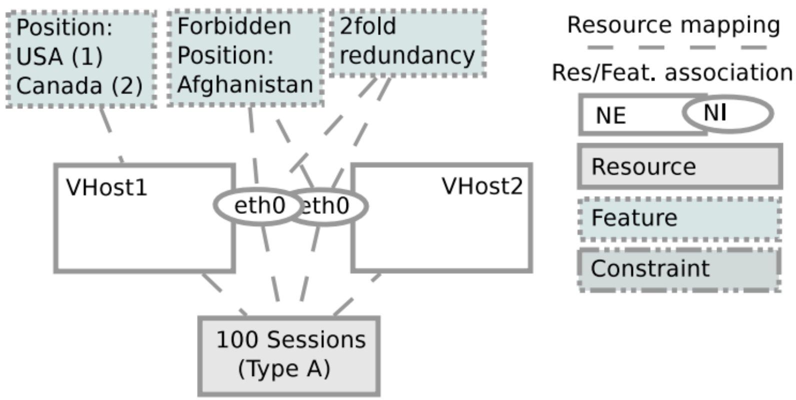


aggregate resources



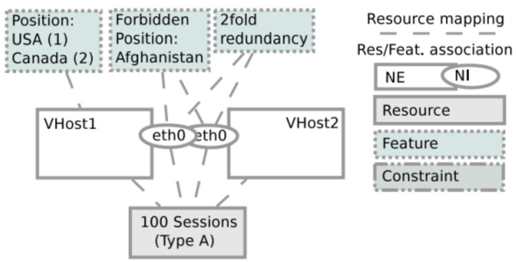
Exploiting Flexibilities: Resource Description Language.

Use Case: Web Service

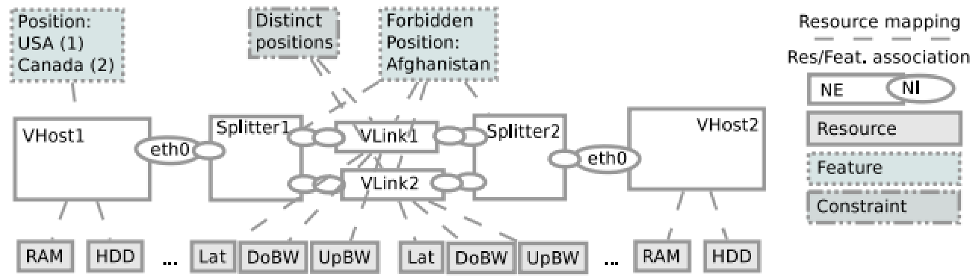


Exploiting Flexibilities: Resource Description Language.

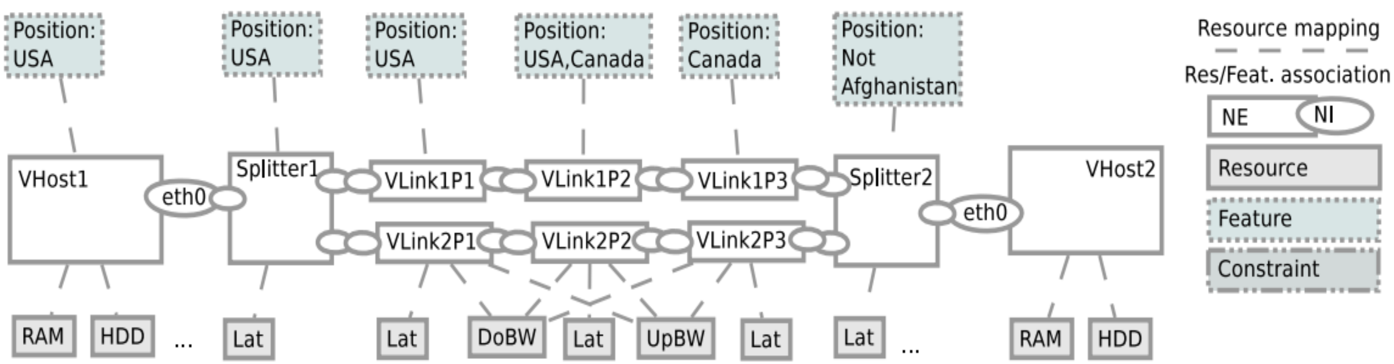
Web Service / Overlay 0:



Overlay 1: with splitters, two virtual links...

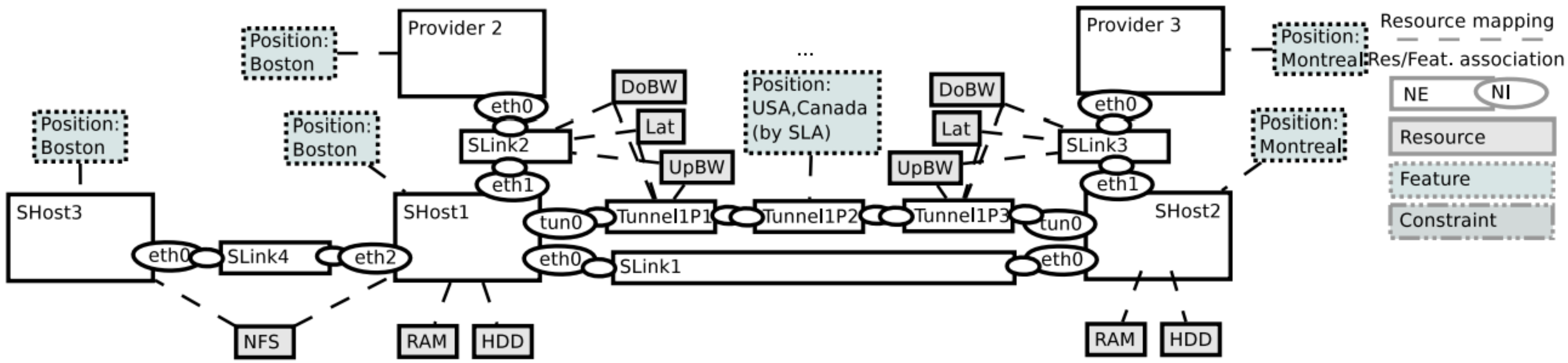


Mapping Layer: an virtual element per substrate element (n:m mapping)

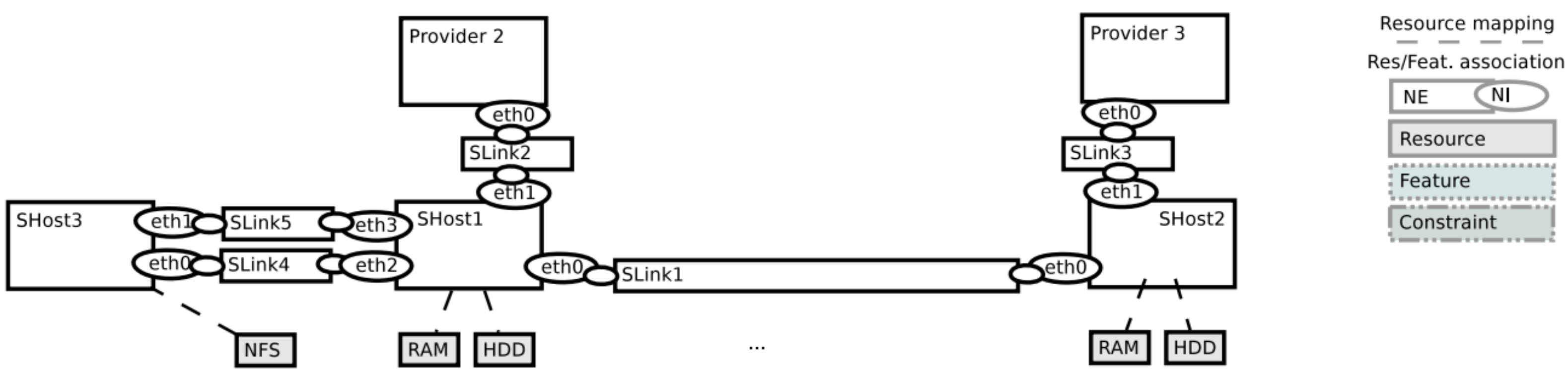


Exploiting Flexibilities: Resource Description Language.

Underlay 1: all-provider view (splitter once collocated and once seperate)



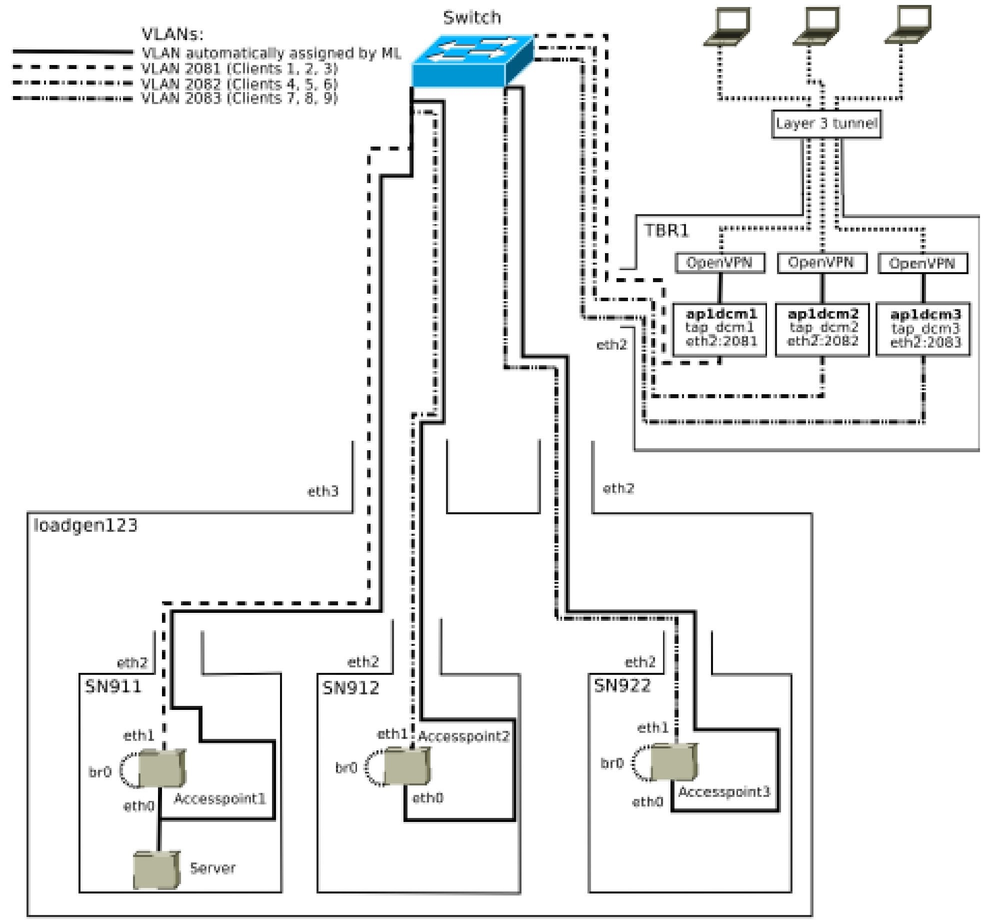
Underlay 0: provider 1 view (NFS at SHost3)



Demo.

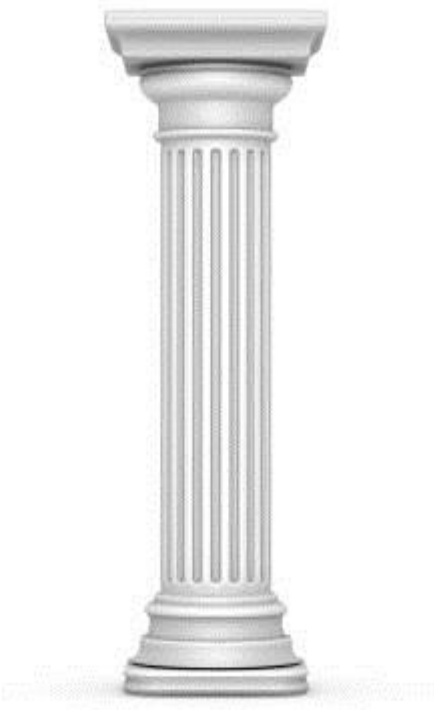
YouTube Migration Demo

http://www.youtube.com/watch?v=llJce0F1zHQ

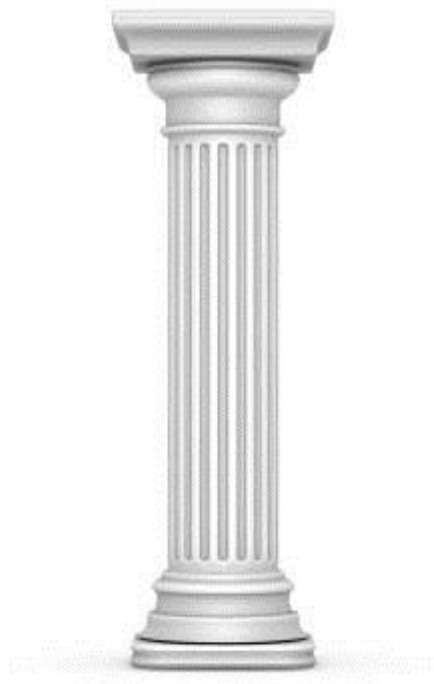


(Theoretical) Research Overview.

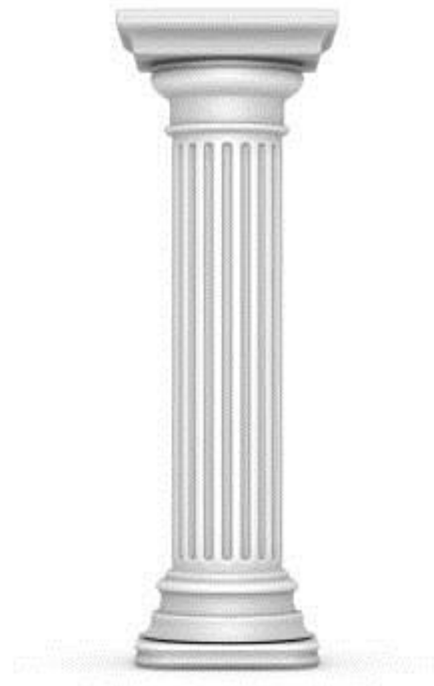
Access Control and Embedding



Service Migration

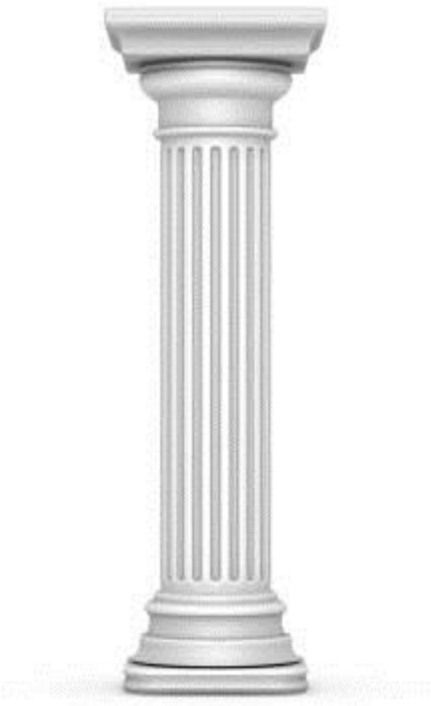


Security Issues

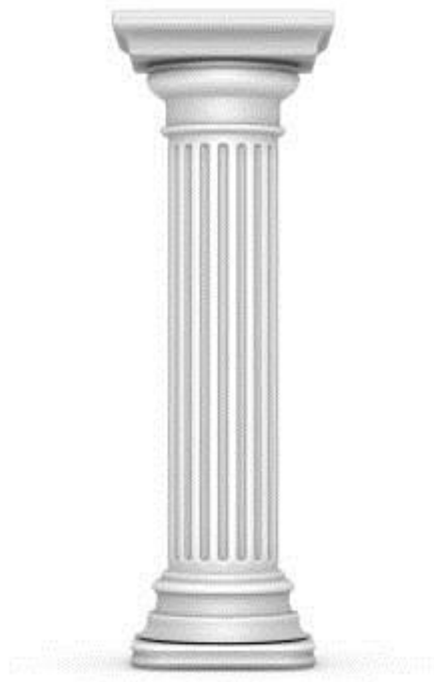


Offline Embedding.

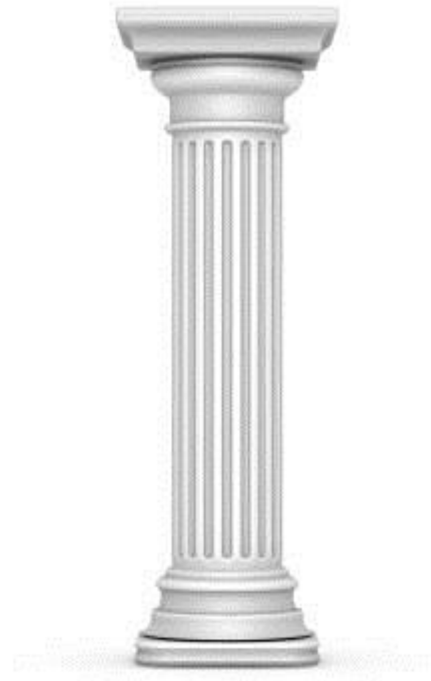
Access Control and Embedding



Service Migration



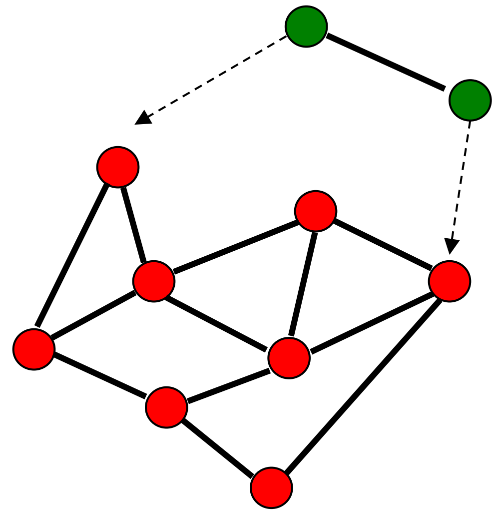
Security Issues

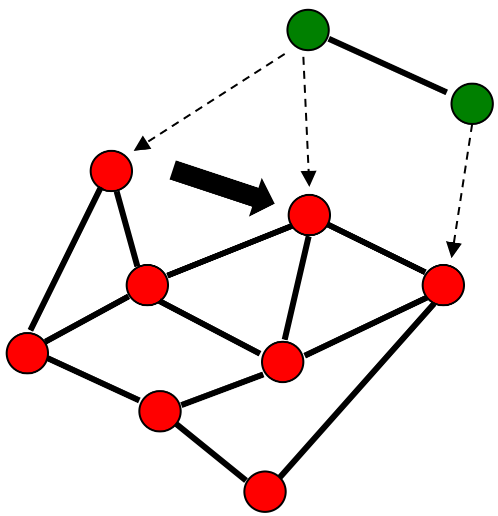


How to Embed CloudNets Efficiently?

Computationally hard...
Our 2-stage approach:

Stage 1: Map quickly and heuristically (dedicated resources)





Stage 2: Migrate long-lived CloudNets to «better» locations (min max load, max free resources, ...)
Typically: heavy-tailed durations, so old CloudNets will stay longer!

Flexible Embeddings.

General Mathematical Program (MIP)

Nodes:

`map_node:` $\sum_{v\in NE_S} new(u,v) = 1 \quad \forall u \in NE_{VN}$
`set_new:` $alloc_{r_S}(u,v,r_V) \le cap_{r_S}(v)new(u,v) \quad \forall u \in NE_{VN}, v \in NE_S, r_V \in R_V, r_S \in R_S$
`req_min:` $alloc_{r_V}(u,v) \ge new(u,v)req(u,r_V,s) \quad \forall u \in NE_{VN}, r_V \in R_V, r_S \in R_S, s = \texttt{minimum}$
`req_max:` $alloc_{r_V}(u,v) \le new(u,v)req(u,r_V,s) \quad \forall u \in NE_{VN}, r_V \in R_V, r_S \in R_S, s = \texttt{maximum}$
`req_con:` $alloc_{r_V}(u,v) = new(u,v)req(u,r_V,s) \quad \forall u \in NE_{VN}, r_V \in R_V, r_S \in R_S, s = \texttt{constant}$

Mapping:

`relate_V:` $alloc_{r_V}(u,v) \ge min\_alloc_{r_V} \cdot new(u,v) \quad \forall u \in NE_V, v \in NE_S, r_V \in R_V$
`allowed:` $suit(u,v) \ge new(u,v) \quad \forall u \in NE_V, v \in NE_S$
`ne_capacity:` $\sum_{u\in NE_V} \sum_{r_V\in R_V} alloc_{r_S}(u,v,r_V) \le cap_{r_S}(v) \quad \forall v \in NE_S, r_S \in R_S$
`capacity:` $\sum_{v\in NE_S} \sum_{u\in NE_V} \sum_{r_V\in R_V} alloc_{r_S}(u,v,r_V) \le cap(r_S) \quad \forall r_S \in R_S$
`load:` $weight_{r_S}/cap(r_S) \cdot \sum_{v\in NE_S} \sum_{u\in NE_V} \sum_{r_V\in R_V} alloc_{r_S}(u,v,r_V) \le load(r_S) \quad \forall r_S \in R_S$
`max_load:` $load(r_S) \le max\_load \quad \forall r_S \in R_S$

Resource-Variable Relation:

`resource:` $\sum_{r_S\in R_S} prop(r_V,r_S)alloc_{r_S}(u,v,r_V) = alloc_{r_V}(u,v) \quad \forall u \in NE_V, v \in NE_S, r_V \in R_V$
`flow_res:` $\sum_{r_S\in R_S} prop(r_V,r_S)flow_{r_S}(f,v,w,r_V) = flow_{r_V}(f,v,w) \quad \forall f \in Fl(u), (v,w) \in NE_S^2, r_V \in R_f, \forall u \in NE_{VL}$

Links:

`map_link:` $\sum_{v\in NE_S} new(u,v) \ge 1 \quad \forall u \in NE_{VL}$
`map_flow:` $new(f,v) \le new(u,v) \quad \forall f \in Fl(u), v \in NE_S, \forall u \in NE_{VL}$
`map_src:` $new(f,v) \ge new(q_f,v) \quad \forall f \in Fl(u), v \in NE_S, q_f$ source of $f; \forall u \in NE_{VL}$
`map_sink:` $new(f,v) \ge new(d_f,v) \quad \forall f \in Fl(u), v \in NE_S, d_f$ sink of $f; \forall u \in NE_{VL}$
`req_min:` $\sum_{w\in NE_S}(flow_{r_V}(f,v,w) - flow_{r_V}(f,w,v)) \ge new(q_f,v)req(u,r_V,s) - new(d_f,v)\infty$
$\forall f \in Fl(u), v \in NE_S, r_V \in R_f; \forall u \in NE_{VL}, s = \texttt{minimum}$
`req_max:` $\sum_{w\in NE_S}(flow_{r_V}(f,v,w) - flow_{r_V}(f,w,v)) \le new(q_f,v)req(u,r_V,s) + new(d_f,v)\infty$
$\forall f \in Fl(u), v \in NE_S, r_V \in R_f; \forall u \in NE_{VL}, s = \texttt{maximum}$
`req_const:` $\sum_{w\in NE_S}(flow_{r_V}(f,v,w) - flow_{r_V}(f,w,v)) = new(q_f,v)req(u,r_V,s) - new(d_f,v)req(u,r_V,s)$
$\forall f \in Fl(u), v \in NE_S, r_V \in R_f; \forall u \in NE_{VL}, s = \texttt{constant}$

Link Allocation:

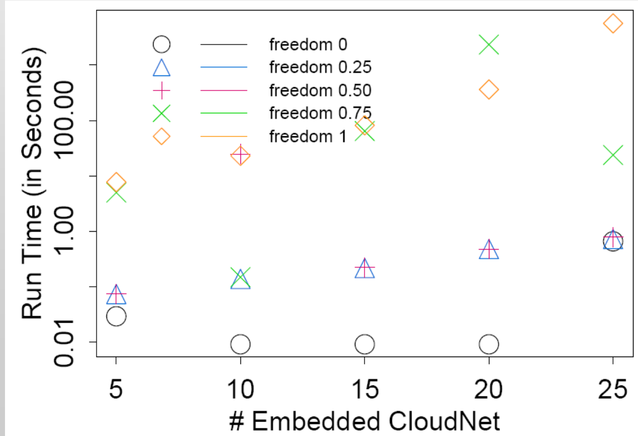
`exp_out:` $\sum_{w\in NE_S} flow_{r_S}(f,v,w,r_V) \le alloc_{r_S}(u,v,r_V) \quad \forall f \in Fl(u), v \in NE_S, r_V \in R_f, r_S \in R_S, \forall u \in NE_{VL}$
`exp_in:` $\sum_{w\in NE_S} flow_{r_S}(f,w,v,r_V) \le alloc_{r_S}(u,v,r_V) \quad \forall f \in Fl(u), v \in NE_S, r_V \in R_f, r_S \in R_S, \forall u \in NE_{VL}$
`direction:` $flow_{r_S}(f,v,w,r_V) \le new(u,v)cap_{r_S}(v,w) \quad \forall f \in Fl(u), (v,w) \in NE_S^2, r_V \in R_f, r_S \in R_S, \forall u \in NE_{VL}$
`relate_f:` $\sum_{w\in NE_S} flow_{r_S}(f,v,w,r_V) + flow_{r_S}(f,w,v,r_V) \ge new(f,v) \quad \forall f \in Fl(u), \forall u \in NE_{VL}, v \in NE_S, r_V \in R_f, r_S \in R_S$

Migration:

`new:` $\sum_{v\in NE_S} old(u,v) \ge mig(u) \quad \forall u \in NE_V$
`migrated:` $old(u,v) - new(u,v) \le mig(u) \quad \forall u \in NE_V, v \in NE_S$

Advantages:
1. Generic (backbone vs datacenter) and allows for migration
2. Allows for different objective functions
3. Optimal embedding: for backgound optimization of heavy-tailed (i.e., long-lived) or «heavy hitter» CloudNets, quick placement e.g., by clustering

But: slow...



The Solution.

General Mathematical Program (MIP)

Nodes:
map_node
set_new:
req_min:
req_max:
req_con:
Mapping:
relate_V
allowed:
ne_capac
capacity
load:
max_load
Resource-
resource
flow_res
Links:
map_link
map_flow
map_src:
map_sink
req_min:
req_max:
req_cons
Link Allo
exp_out:
exp_in:
directio
relate_f
Migration
new:
migrated

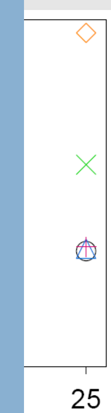
Advantages:

nter)

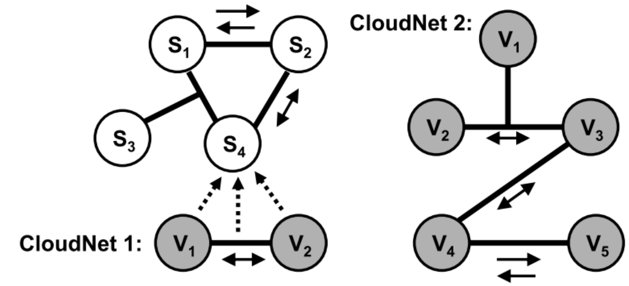
ound
long-
nt e.g.,

Advantages of MIP:

- Very general
- Supports easy replacement of objective functions
- Can use standard, optimized software tools such as CPLEX, Gorubi, etc.



Generality of the MIP.



Objective functions:

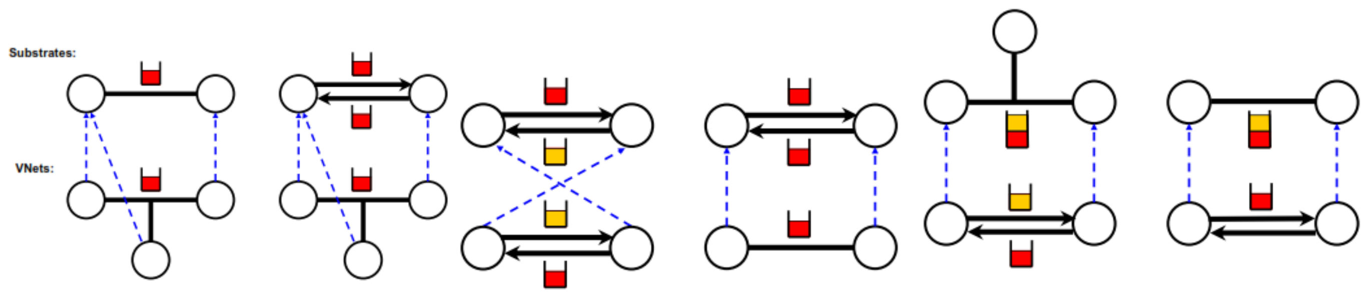
- minimize maximum load (= load balance)
- maximize free resources (= compress as much as possible), ...

Migration support:

- costs for migration: per element, may depend on destination, etc.
- answer questions such as «what is cost/benefit if I migrate now?»

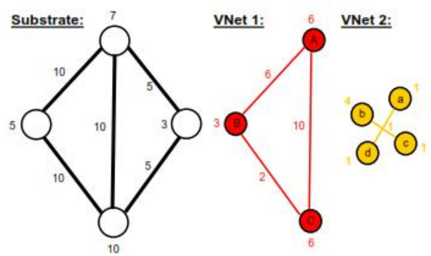
Embedding:

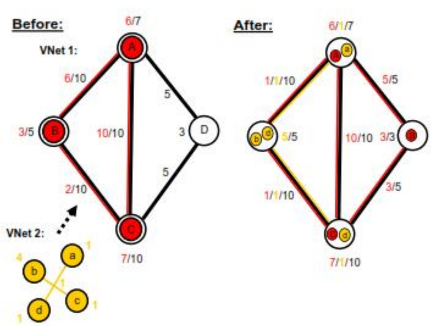
- embedding full-duplex on full-duplex links
- full-duplex on half-duplex links
- or even multiple endpoint links (e.g., wireless) supported!



On the Use of Migration.

Res.	w/o	w/ Link	w/ Link&Node	Opt
1	3	3	4	**4**
2	5	5	9	**9**
3	7	8	13	**13**
4	1	1	17	**17**
5	17	22	24	**24**
6	2	2	27	**27**
7	31	32	32	**32**
8	37	37	37	**37**





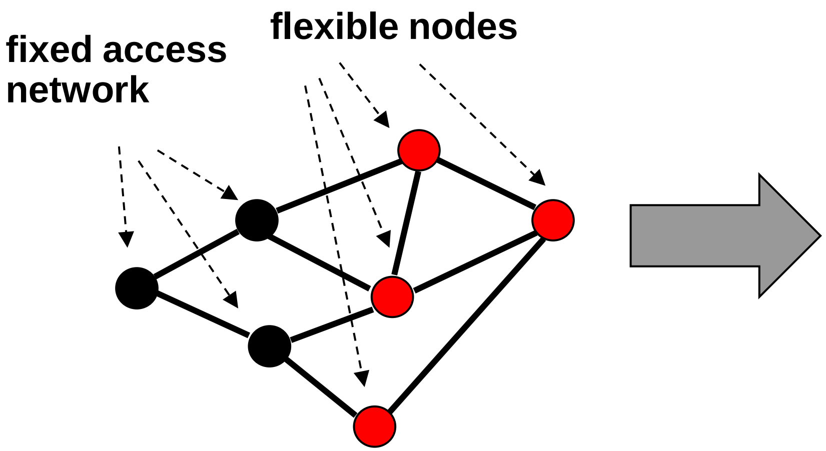
Migration: Useful to increase the number of embeddable CloudNets, especially in under-provisioned scenarios

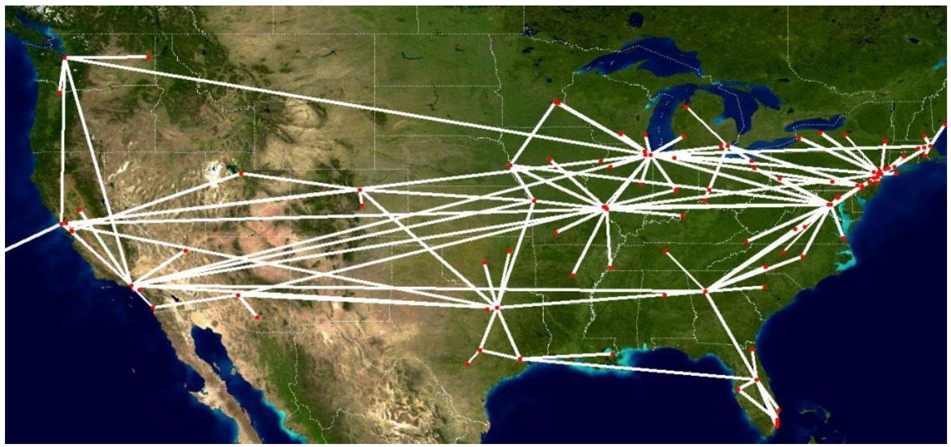
Performance of the MIP: Setup.

Substrate: Rocketfuel ISP topologies (with 25 nodes)

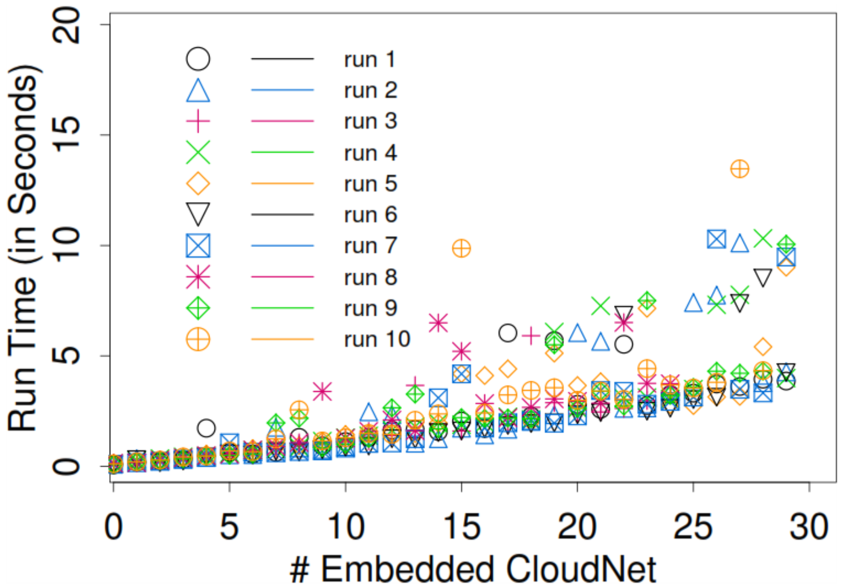
CloudNets: Out-sourcing scenario, CloudNets with up to ten nodes, subset of nodes fixed (access points) and subset flexible (cloud resources)

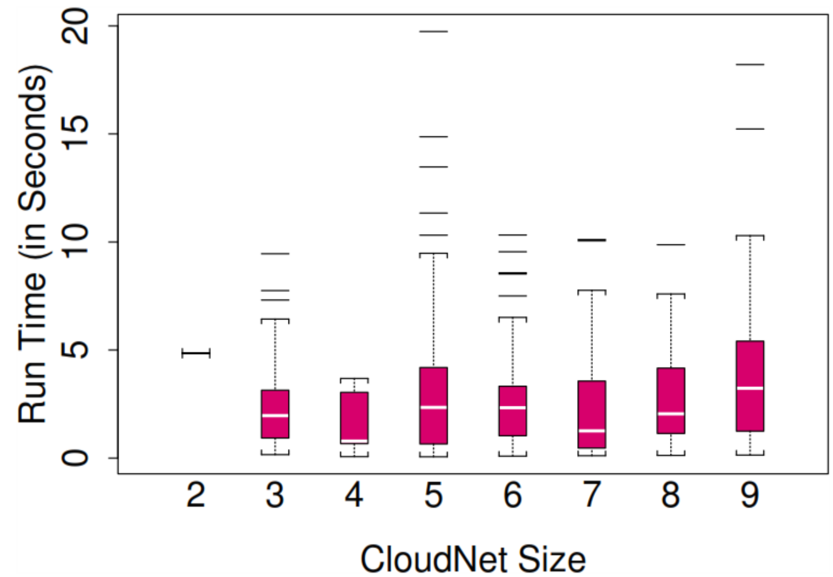
Solver: CPLEX on 8-core Xeon (2.5GHz)





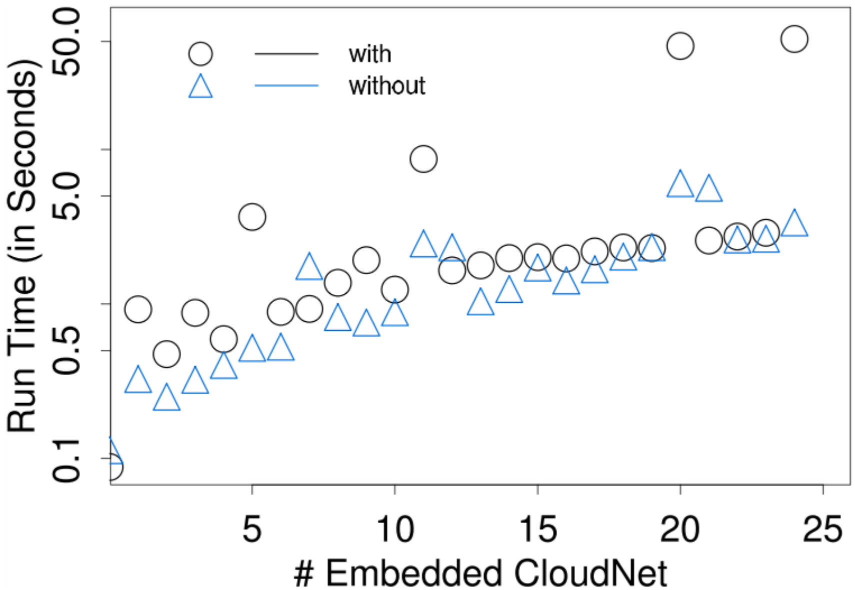
Performance of the MIP.

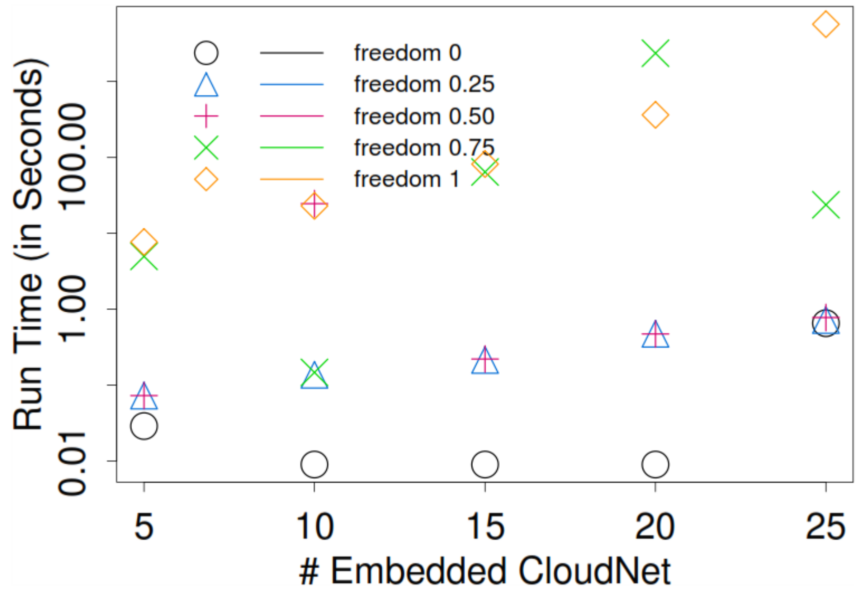




- Runtime below 1 minute per CloudNet, slightly increasing under load
- Impact of CloudNet size relatively small

Performance of the MIP.

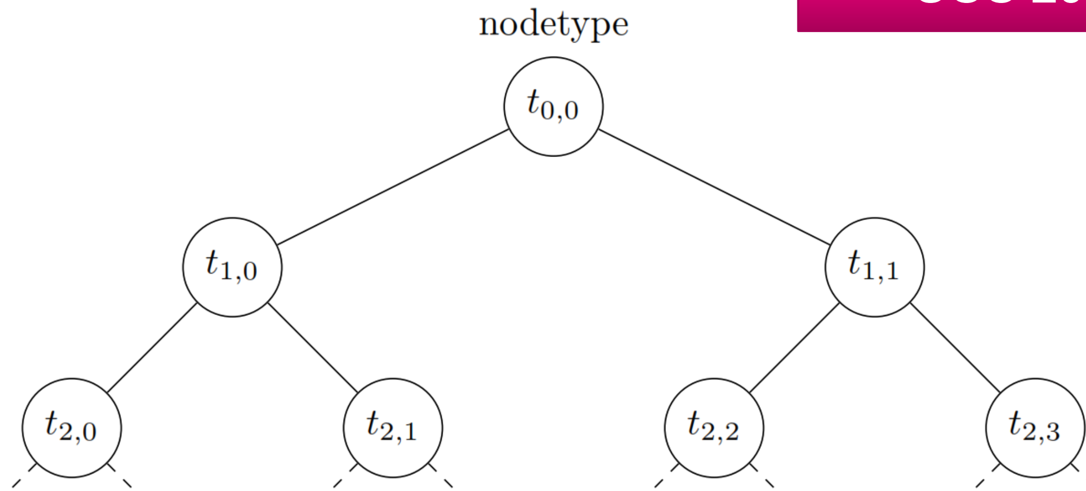




- Enabling option to migrate can increase execution time significantly (log scale!)
- Also number of flexible CloudNet components is important

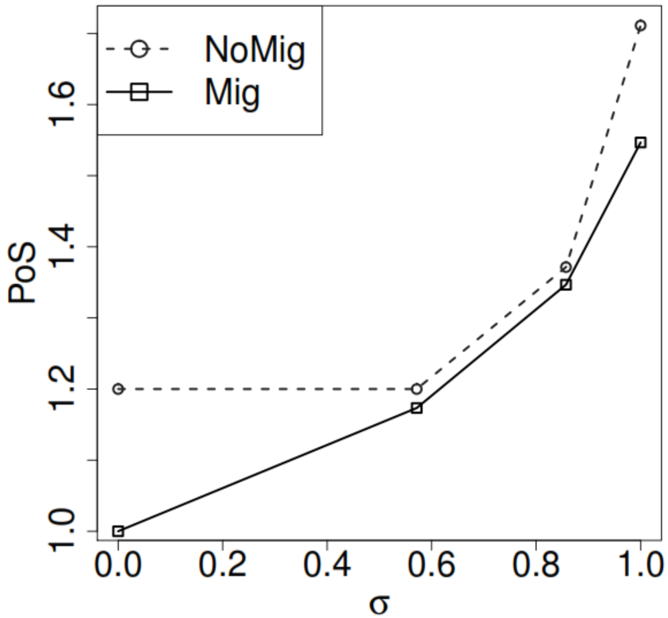
Use of Flexibility.

UCC 2012



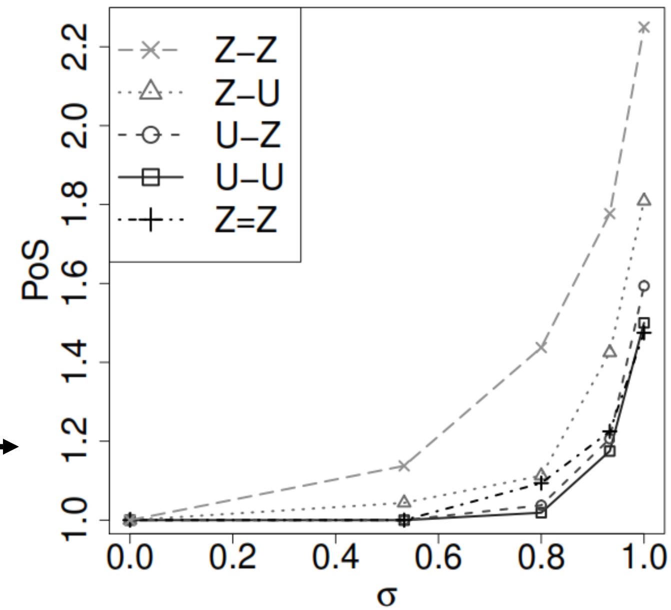
PoS

How much link resources are needed to embed a CloudNet with specificity s%?



Up to 60%, even a little bit more if no migrations are possible!

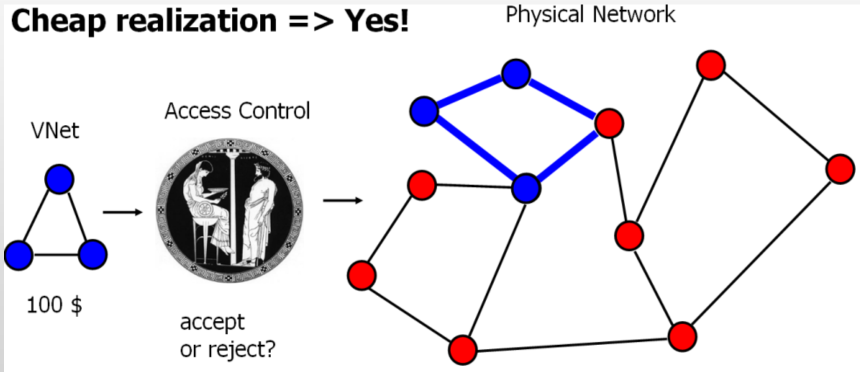
Skewed (Zipf) distributions worst when not matching.



CloudNet Embedding.

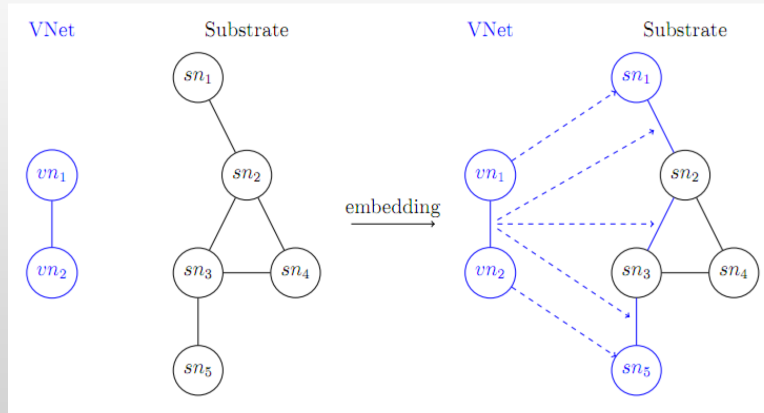
Online Access Control

Goal:
*Decide online which VNet requests to accept, such that **profit** is maximized*



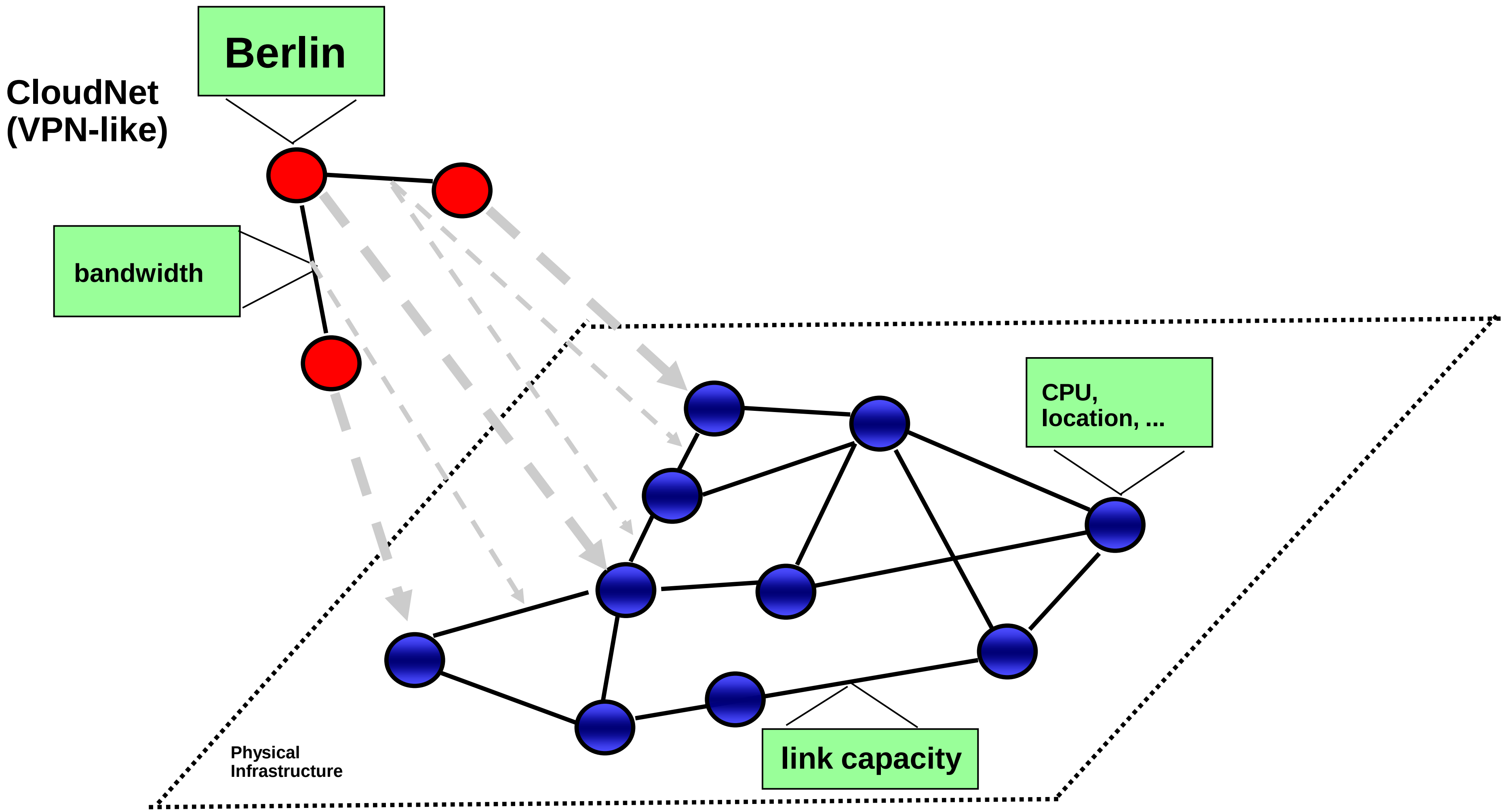
Mapping and Allocation

Goal:
*Where to realize CloudNet such that spec is met? Objective, e.g.: minimize allocation **resources**, minimize **max load**, **save energy**, …*



*Currently focus on optimizing existing CloudNets (**heavy-tailed lifetime** assumption): but we are also working on quick embeddings (**clustering**, iterative, ...)*

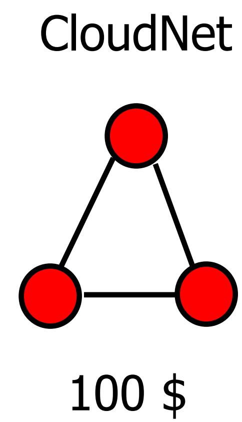
Competitive Access Control: Model (1)



Competitive Access Control: Model (2)

Specification of CloudNet request:

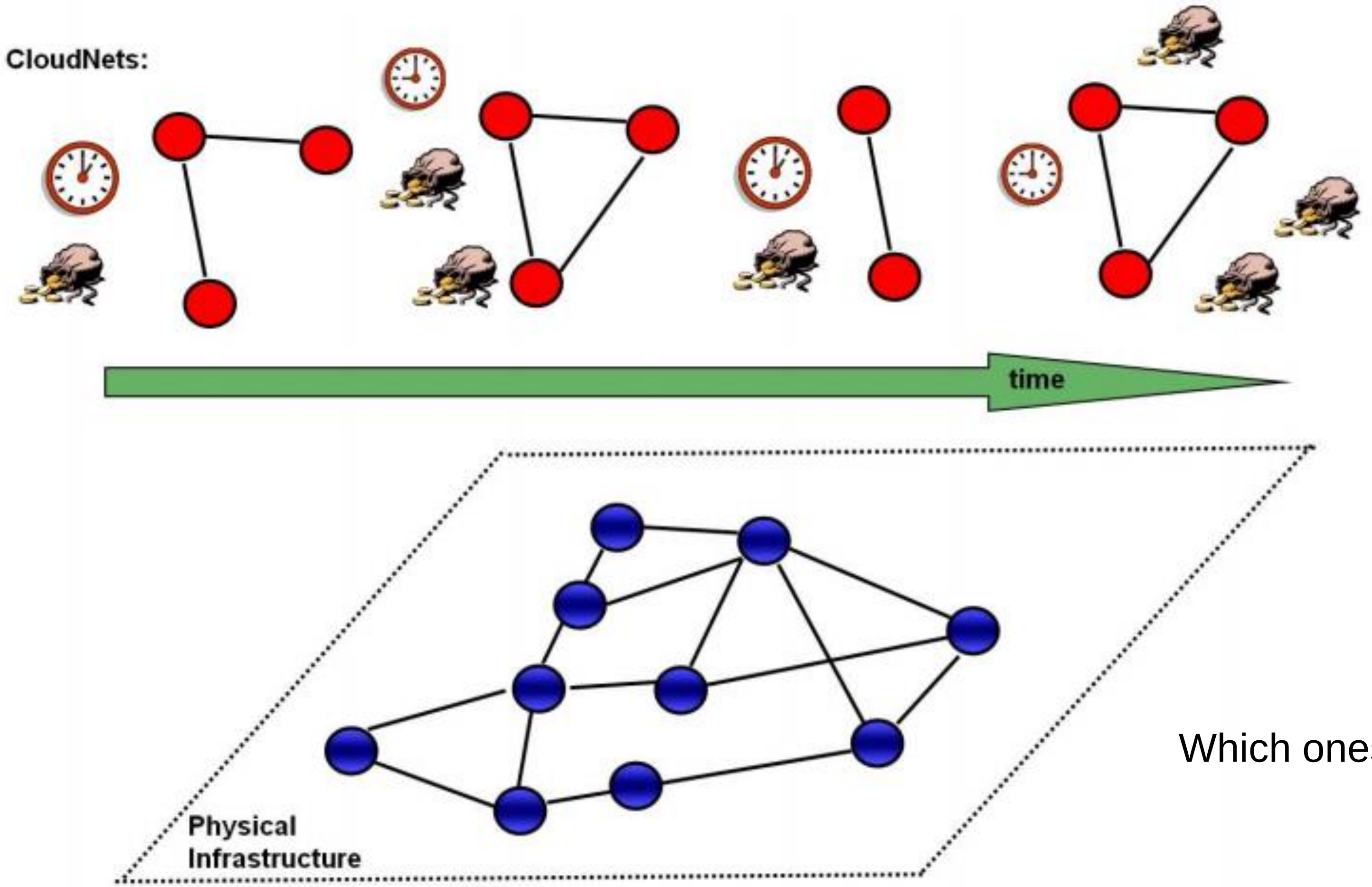
- terminal locations to be connected
- benefit if CloudNet accepted (all-or-nothing, no preemption)
- desired bandwidth and allowed traffic patterns
- a routing model
- duration (from when until when?)



If CloudNets with these specifications arrive over time, which ones to accept online?

Objective: maximize **sum of benefits** of accepted CloudNets

Competitive Access Control: Model (3)

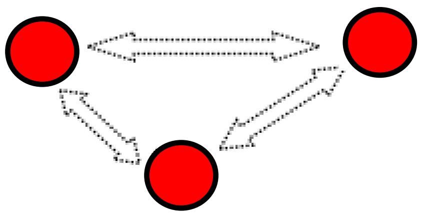


Which ones to accept?

CloudNet Specifications (1): Traffic Model.

Customer Pipe

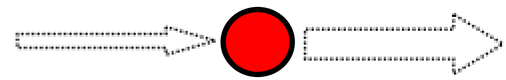
Every pair (u,v) of nodes requires a certain bandwidth.



Detailed constraints, only this traffic matrix needs to be fulfilled!

Hose Model

Each node v has max ingress and max egress bandwidth: each traffic matrix fulfilling them must be served.

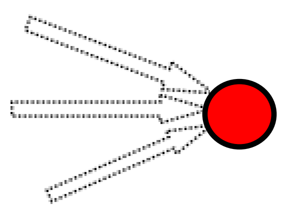


More flexible, must support many traffic matrices!

«virtual switch»

Aggregate Ingress Model

Sum of ingress bandwidths must be at most a parameter I.

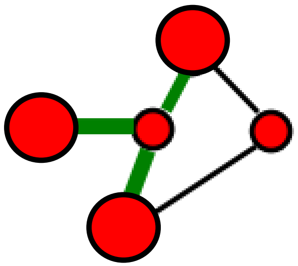


Simple and flexible! Good for multicasts etc.: no overhead, duplicate packets for output links, not input links already!

CloudNet Specifications (2): Routing Model.

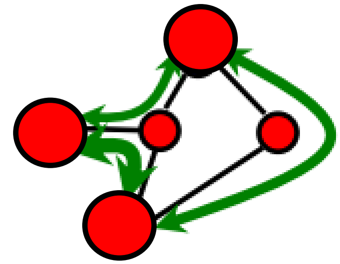
Tree

VNet is embedded as Steiner tree:



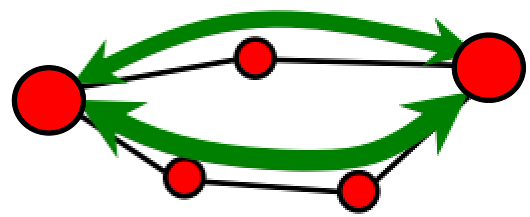
Single Path

Each pair of nodes communicates along a single path.



Multi Path

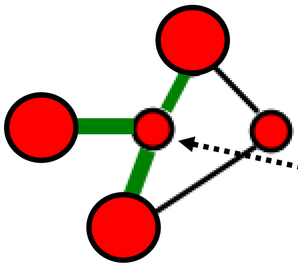
A linear combination specifies split of traffic between two nodes.



CloudNet Specifications (2): Routing Model.

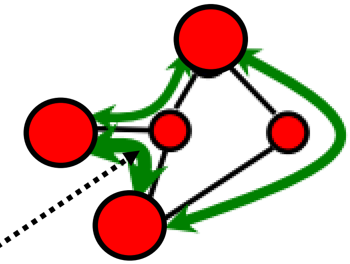
Tree

VNet is embedded as Steiner tree:



Single Path

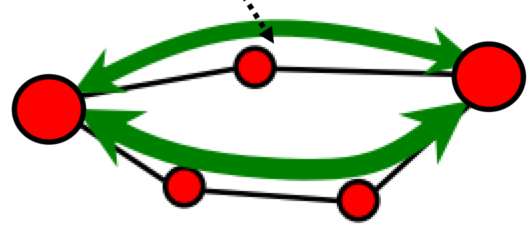
Each pair of nodes communicates along a single path.



Relay nodes may add to embedding costs!
(resources depend, e.g., on packet rate)

Multi Path

A linear combination specifies split of traffic between two nodes.



Competitive Embeddings.

Competitive analysis framework:

Online Algorithm

Online algorithms make decisions at time t without any knowledge of inputs / requests at times t'>t.

Competitive Ratio

Competitive ratio r,

r = Cost(ALG) / cost(OPT)

The price of not knowing the future!

Competitive Analysis

An *r-competitive online algorithm* ALG gives a worst-case performance guarantee: the performance is at most a factor r worse than an optimal offline algorithm OPT!

No need for complex predictions but still good!

Buchbinder&Naor: Primal-Dual Approach.

Algorithm design and analysis follows online primal-dual approach recently invented by Buchbinder&Naor!
(Application to general VNet embeddings, traffic&routing models, router loads, duration, approx oracles, ...)

1. Formulate dynamic primal and dual LP

(I)	(II)
$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

2. Derive GIPO algorithm which always produces feasible primal solutions and where Primal >= 2*Dual

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do
   $$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$
   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Result.

Theorem

The presented online algorithm GIPO is log-competitive in the amount of resources in the physical network!
If capacities can be exceeded by a log factor, it is even constant competitive.

However, competitive ratio also depends on max benefit!

Algorithm and Proof Sketch (1).

Embedding oracle: GIPO invokes an oracle procedure to determine cost of CloudNet embedding!

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Algorithm and Proof Sketch (1).

If resource cost lower than benefit: accept!

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Algorithm and Proof Sketch (1).

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

update allocations for accepted CloudNet...

Algorithm and Proof Sketch (1).

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

otherwise reject
(no change in substrate)

Algorithm and Proof Sketch (1).

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

otherwise reject
(no change in substrate)

Algorithm efficient... except for oracle (static, optimal embedding)!
What if we only use a suboptimal embedding here?

Algorithm and Proof Sketch (2).

Problem: computation of optimal embeddings NP-hard!
Thus: use approximate embeddings! (E.g., Steiner tree)

GIPO:

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Competitive ratio ρ

Embedding approx.:

<insert your favorite approx algo>

Approx ratio r

Lemma

The approximation does not reduce the overall competitive ratio by much: we get ρ*r ratio!

Proof Sketch (1): Simplified LP.

maximize benefit!

... while ensuring capacity and no more than demand!

realization of i-th request (will be integer, accept fully or not at all)

$$
\begin{aligned}
&\min \sum_{e\in E} x_e \cdot c(e) + \sum_{v\in V} x_v \cdot c(v) + \sum_i z_i \cdot d_i \quad s.t.\\
&\text{(Covering Const.)} \forall i \forall \Delta \in \Delta_i z_i + \alpha(i,\Delta) \quad \geq b_i\\
&\forall i \forall \Delta \in \Delta_i x_e, x_v, z_i \quad \geq 0
\end{aligned}
$$

(I)

$$
\begin{aligned}
&\max \sum_i b_i \cdot \sum_{\Delta_{ij}\in\Delta_i} f_{ij} \quad s.t.\\
&\text{(Vertex Capacity Const.)} \quad \forall v \in V \;\; flow(v) \leq c(v)\\
&\text{(Edge Capacity Const.)} \quad \forall e \in E \;\; flow(e) \leq c(e)\\
&\text{(Demand Const.)} \quad \forall i \sum_{\Delta_{ij}\in\Delta_i} f_{ij} \leq d_i\\
&f \geq 0
\end{aligned}
$$

(II)

Fig. 1: (I) The Primal linear embedding program. (II) The Dual linear embedding program.

Proof Sketch (2): Simplified LP.

essentially, exponential load...

$$\min \sum_{e \in E} x_e \cdot c(e) + \sum_{v \in V} x_v \cdot c(v) + \sum_i z_i \cdot d_i \quad s.t.$$

$$\begin{aligned} \text{(Covering Const.)} \forall i \forall \Delta \in \Delta_i z_i + \alpha(i, \Delta) &\geq b_i \\ \forall i \forall \Delta \in \Delta_i x_e, x_v, z_i &\geq 0 \end{aligned}$$

(I)

$$\max \sum_i b_i \cdot \sum_{\Delta_{ij} \in \Delta_i} f_{ij} \quad s.t.$$

$$\begin{aligned} &\text{(Vertex Capacity Const.)} && \forall v \in V \;\; \mathit{flow}(v) \leq c(v) \\ &\text{(Edge Capacity Const.)} && \forall e \in E \;\; \mathit{flow}(e) \leq c(e) \\ &\text{(Demand Const.)} && \forall i \sum_{\Delta_{ij} \in \Delta_i} f_{ij} \leq d_i \\ & && f \geq 0 \end{aligned}$$

(II)

Fig. 1: (I) The Primal linear embedding program. (II) The Dual linear embedding program.

Proof Sketch (3): Simplified LP.

Algorithm 1 The ISTP Algorithm.

Input: $G = (V, E)$ (possibly infinite), sequence of requests $\{r_i\}_{i=1}^{\infty}$ where $r_i \triangleq (U_i, c_i, d_i, b_i)$.

Upon arrival of request r_i:

1) $j \leftarrow \text{argmin}\{\alpha(i,j) : \Delta_{ij} \in \Delta_i\}$ (find a lightest realization over the terminal set U_i using an oracle).
2) If $\alpha(i,j) < b_i$ then, (accept r_i)
   a) $f_{ij} \leftarrow d_i$.
   b) For each $e \in E(\Delta_{ij})$ do
   $$x_e \leftarrow x_e \cdot 2^{d_i/c(e)} + \frac{1}{|V(\Delta_{ij})|} \cdot (2^{d_i/c(e)} - 1).$$
   c) For each $v \in V(\Delta_{ij})$ do
   $$x_v \leftarrow x_v \cdot 2^{c_i/c(v)} + \frac{d_i/c_i}{|V(\Delta_{ij})|} \cdot (2^{c_i/c(v)} - 1).$$
   d) $z_i \leftarrow b_i - \alpha(i,j)$.
3) Else, (reject r_i)
   a) $z_i \leftarrow 0$.

oracle
(triangle only)

update primal
variables if accepted

Proof Sketch (4): Simplified LP.

Step (2b) increases the cost $\sum_e x_e \cdot c(e)$ as follows (change $\Delta(x_e) = \sum_e (x_e^t - x_e^{t-1}) \cdot c(e)$):

$$
\begin{aligned}
\Delta(x_e) &\leq \sum_{e\in\Delta} \left[x_e \cdot (2^{d_i/c(e)} - 1) + \frac{1}{|V(\Delta_{ij})|} \cdot (2^{d_i/c(e)} - 1) \right] \cdot c(e) \\
&= \sum_{e\in\Delta} \left(x_e + \frac{1}{|V(\Delta_{ij})|} \right) \cdot (2^{d_i/c(e)} - 1) \cdot c(e) \\
&\leq c_{\min}(e) \cdot (2^{d_i/c_{\min}(e)} - 1) \sum_{e\in\Delta} \left(x_e + \frac{1}{|V(\Delta_{ij})|} \right) \\
&\leq d_i \cdot (2^1 - 1) \sum_{e\in\Delta} \left(x_e + \frac{1}{|V(\Delta_{ij})|} \right) \\
&\leq d_i \cdot \sum_{e\in\Delta} x_e + d_i \cdot \sum_{e\in\Delta} \frac{1}{|V(\Delta_{ij})|} \\
&\leq d_i \cdot \sum_{e\in\Delta} x_e + d_i \; . \qquad (1)
\end{aligned}
$$

Step (2c) increases the cost $\sum_v x_v \cdot c(v)$ as follows (change $\Delta(x_v) = \sum_v (x_v^t - x_v^{t-1}) \cdot c(v)$):

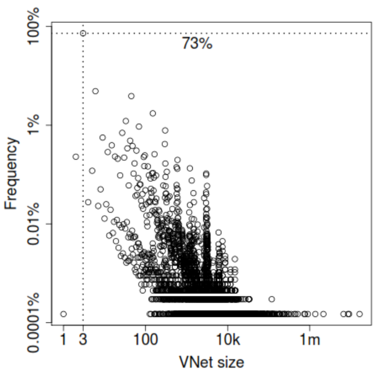
$$
\begin{aligned}
\delta(x_v) &\leq \sum_{v\in\Delta} \left[x_v \cdot (2^{c_i/c(v)} - 1) + \frac{d_i/c_i}{|V(\Delta_{ij})|} \cdot (2^{c_i/c(v)} - 1) \right] \cdot c(v) \\
&= \sum_{v\in\Delta} \left(x_v + \frac{d_i/c_i}{|V(\Delta_{ij})|} \right) \cdot (2^{c_i/c(v)} - 1) \cdot c(v) \\
&\leq c_{\min}(v) \cdot (2^{c_i/c_{\min}(v)} - 1) \sum_{v\in\Delta} \left(x_v + \frac{d_i/c_i}{|V(\Delta_{ij})|} \right) \\
&\leq c_i \cdot (2^1 - 1) \sum_{v\in\Delta} \left(x_v + \frac{d_i/c_i}{|V(\Delta_{ij})|} \right) \\
&\leq c_i \cdot \sum_{v\in\Delta} x_v + c_i \cdot \sum_{v\in\Delta} \frac{d_i/c_i}{|V(\Delta_{ij})|} \\
&\leq c_i \cdot \sum_{v\in\Delta} x_v + d_i \; . \qquad (2)
\end{aligned}
$$

after each request, primal variables constitute feasible solutions...

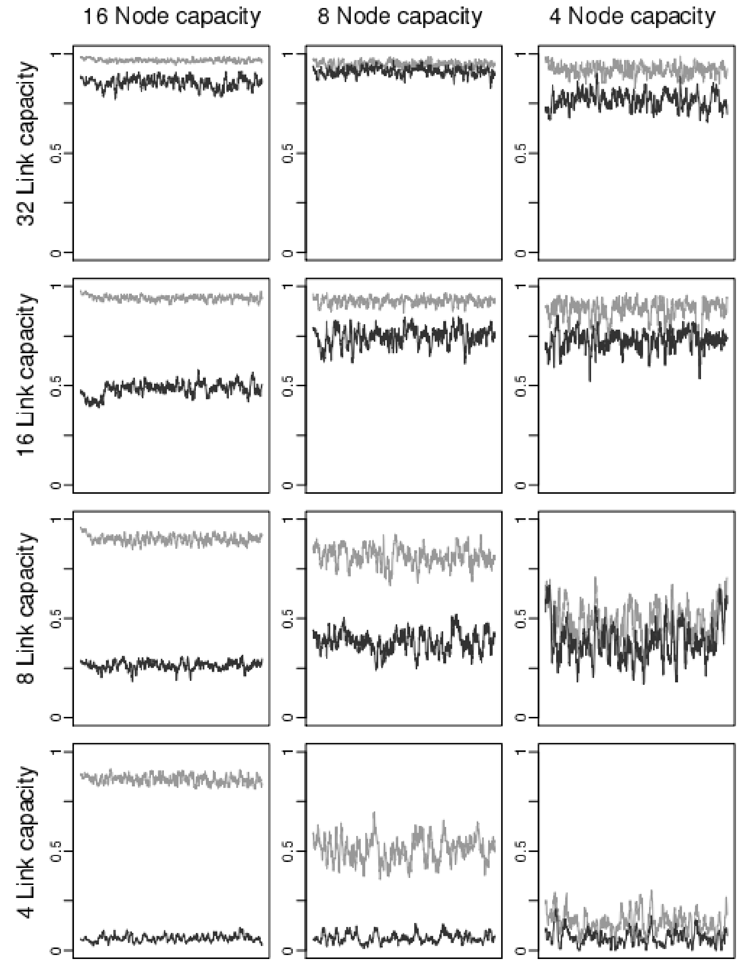
On the Benefit of Collocation.

Fuerst et al.: CLOUDNETS 2012

Google cluster: many small networks, over 90% allow for collocation



Greedy vs SecondNet vs ViNE:
Greedy collocation algorithm beats them all...!

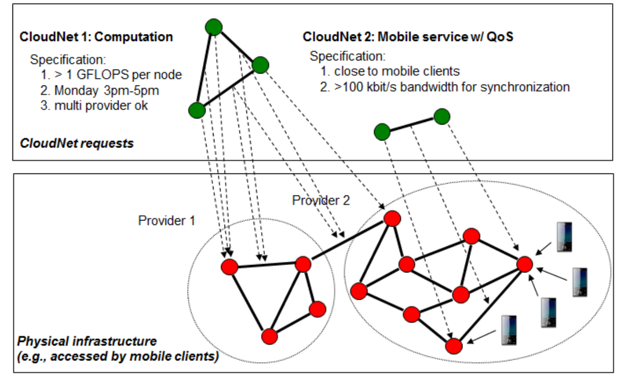


Algorithm 1 The LoCo Algorithm

Require: VNet $G = (V, E)$, $M = \{s\}$ for some $s \in V(G)$, $P = (\Gamma(s))$
  while $|P| > 0$ **do**
    sort P (* decreasing link capacities *)
    choose $u = P[0]$ (* next node to map *)
    map u (* forward checking *)
    map $\{u, v\}$ $\forall$ $v \in M$, **where** $\{u, v\} \in E(G)$
    $M = M \cup \{u\}$ **and** $P = P \setminus \{u\}$
  end while
  if (embedding failed), **backtrack** on s

Mixed Integer Programs

Problem 1: Classic VNet Embedding (VNEP).



- Map virtual nodes to substrate nodes
  - Collocation possible
  - But not splitting of virtual nodes

- Map virtual links
  - One
  - Linear combination
  - Hose

- Mixed Integer Model
  - VINO ☺

- Open Problems
  - Everything ☺

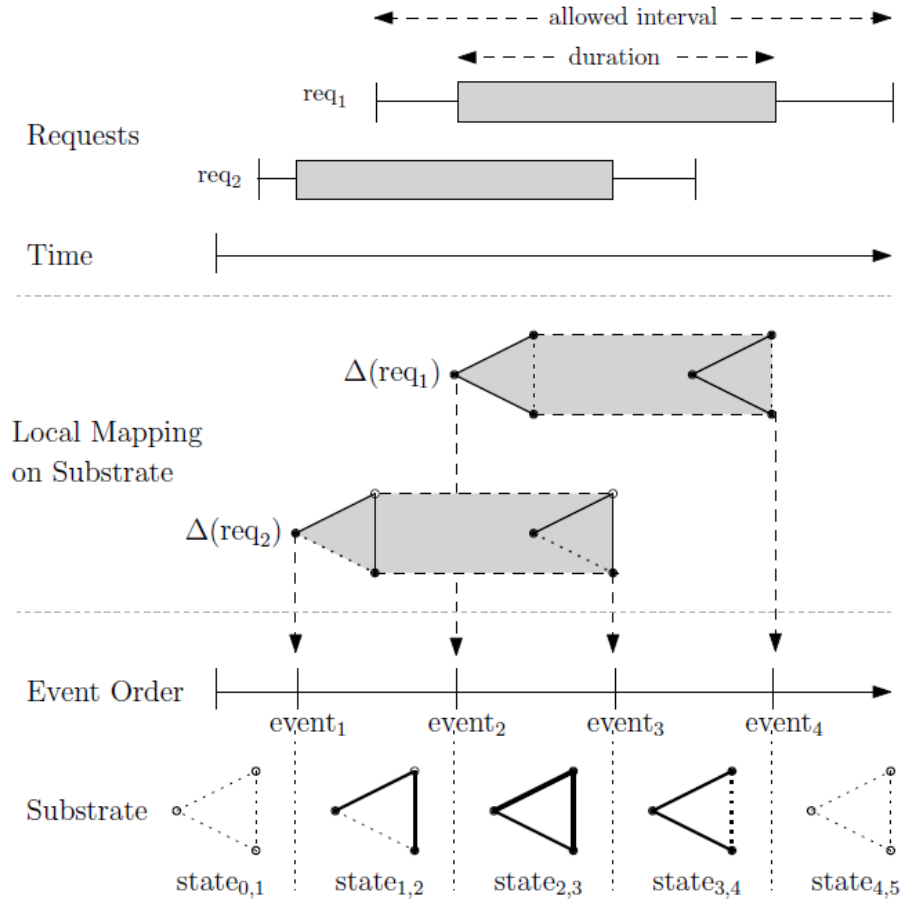
The Informal Guide to the
Virtual Network Embedding MIP Creator
TU Berlin, Germany

Matthias Rost
mrost@inet.tu-berlin.de

Rost et al.:
Tech Report

Problem 2: Embedding with Time Flexibilities (TVNEP).

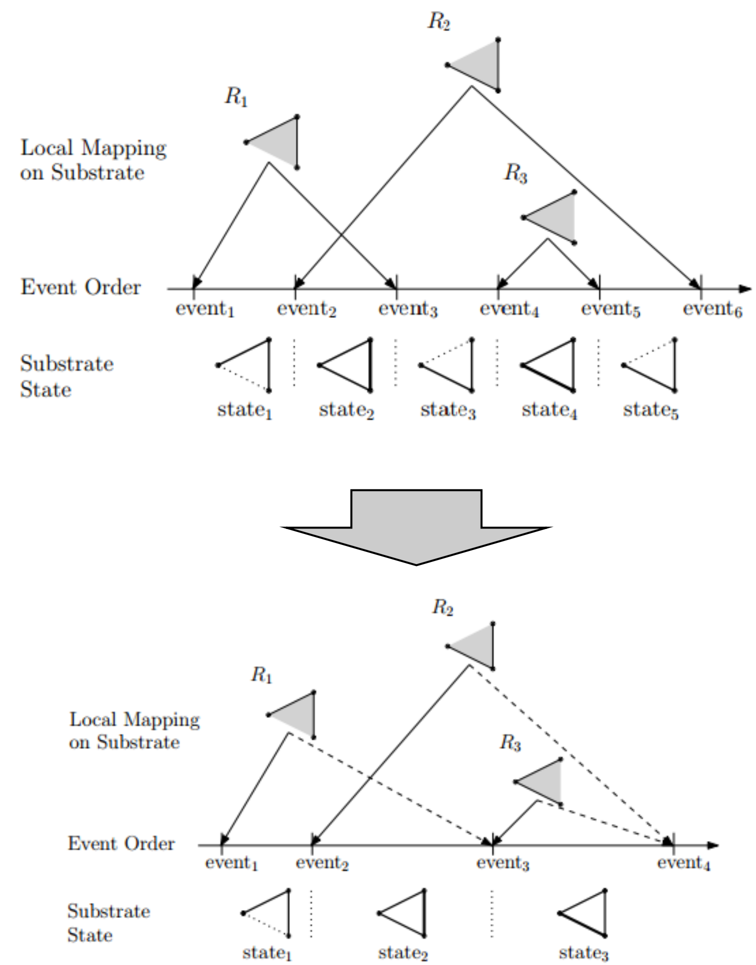
- VNets come with time flexibilities
- Example: delay-tolerant computations, bulk data transfers, etc.
- Where to embed and when to schedule VNets?



Problem 2: Embedding with Time Flexibilities (TVNEP).

- Continuous time (less binary variables): state model (explicit states) and delta model (only differences)
  - Delta model yields bad relaxations
  - State model has more variables, but still better

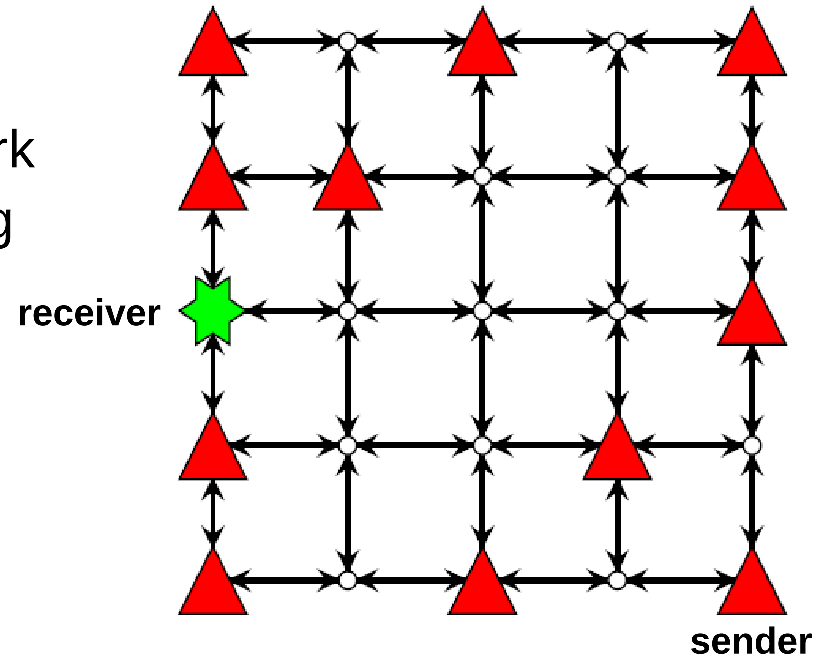
- Compact variant:
  - State reduction: feasibility check only at start of request sufficient (when finishes less resources)
  - Minimize smear-out: Distribute start and end to as few event points as possible
  - «Merge» multiple endpoints with start points
  - Compute temporal dependencies graph cuts



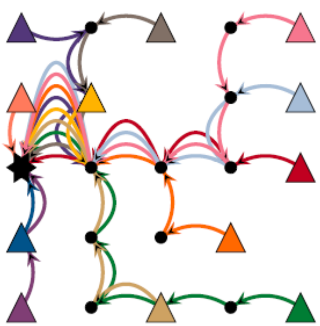
Problem 3: VirtuCast / In-Network Processing (CVSAP).

- Network Function Virtualization
  - Can aggregate / split streams in network
  - E.g., streaming or wide-area monitoring

- Universal nodes need to be activated: Joint optimization of processing and communication?
  - Note: DAG may not be optimal!



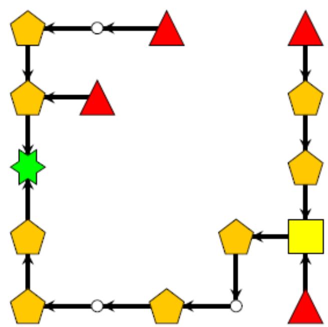
N unicasts



Communication costs!

vs.

Steiner



Processing costs!

Generalization?

Rost et al.: OPODIS 2013

Problem 3: VirtuCast / In-Network Processing (CVSAP).

- Multi-Commodity flow bad: for 200 Steiner nodes and 6800 edges, 1.3 mio binary variables!
- Single-Commodity approach: solvable!
- Idea: single commodity and then path decomposition
- Open question: can we even relax path variables and optimally round it afterwards?

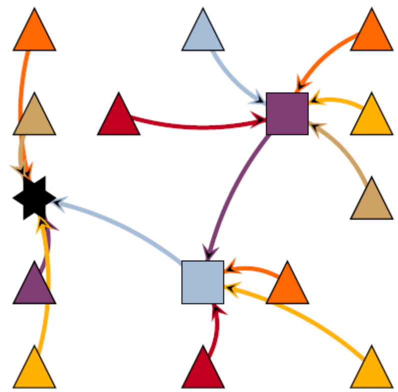


Figure: Virtual Arborescence

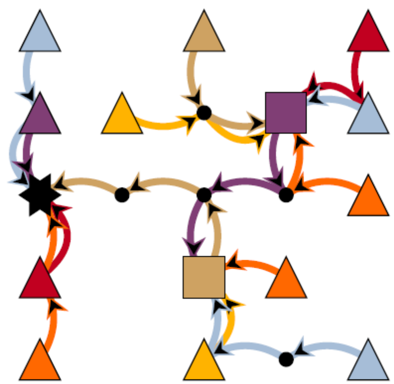
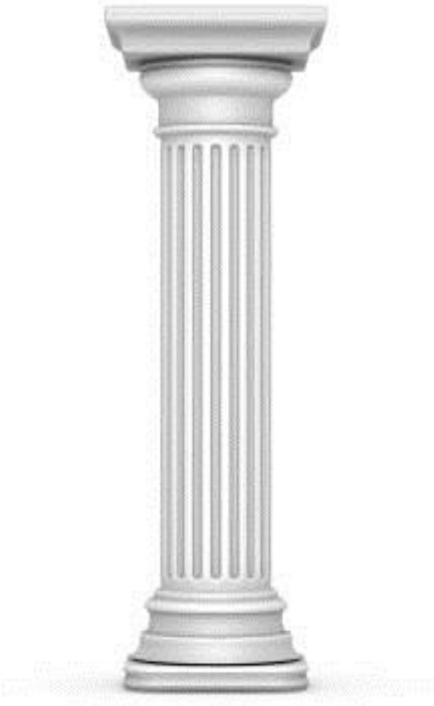


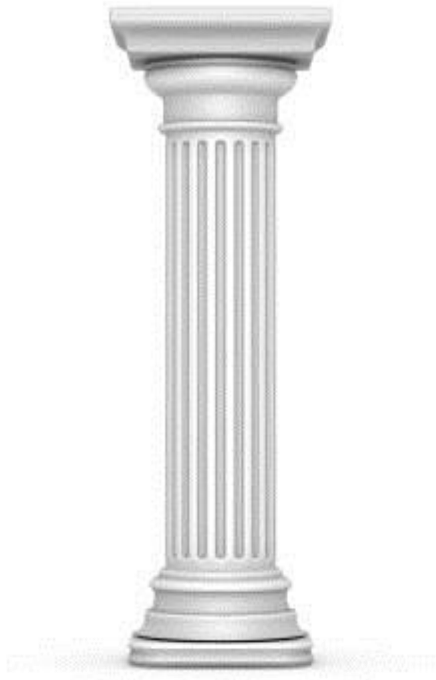
Figure: Flow in original graph

Migration.

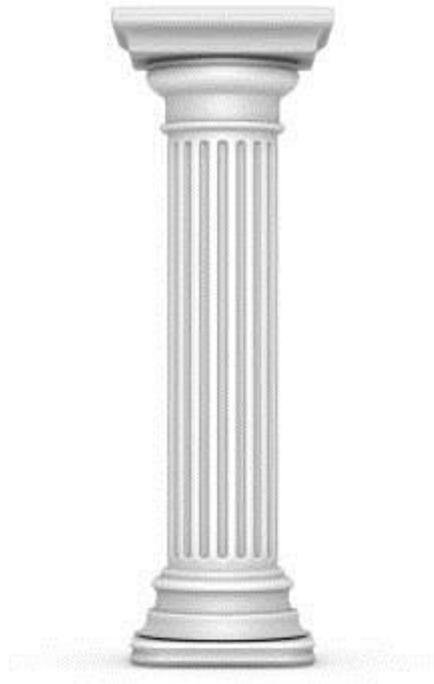
Access Control and Embedding



Service Migration



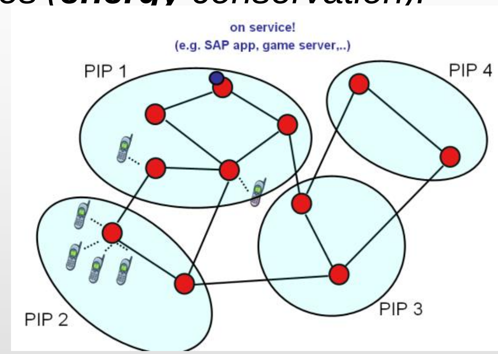
Security Issues

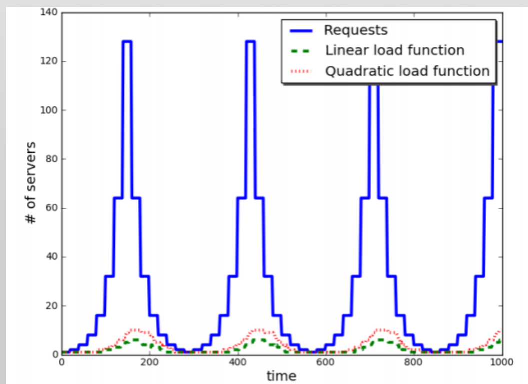


Online Service Migration.

Online Migration and Allocation

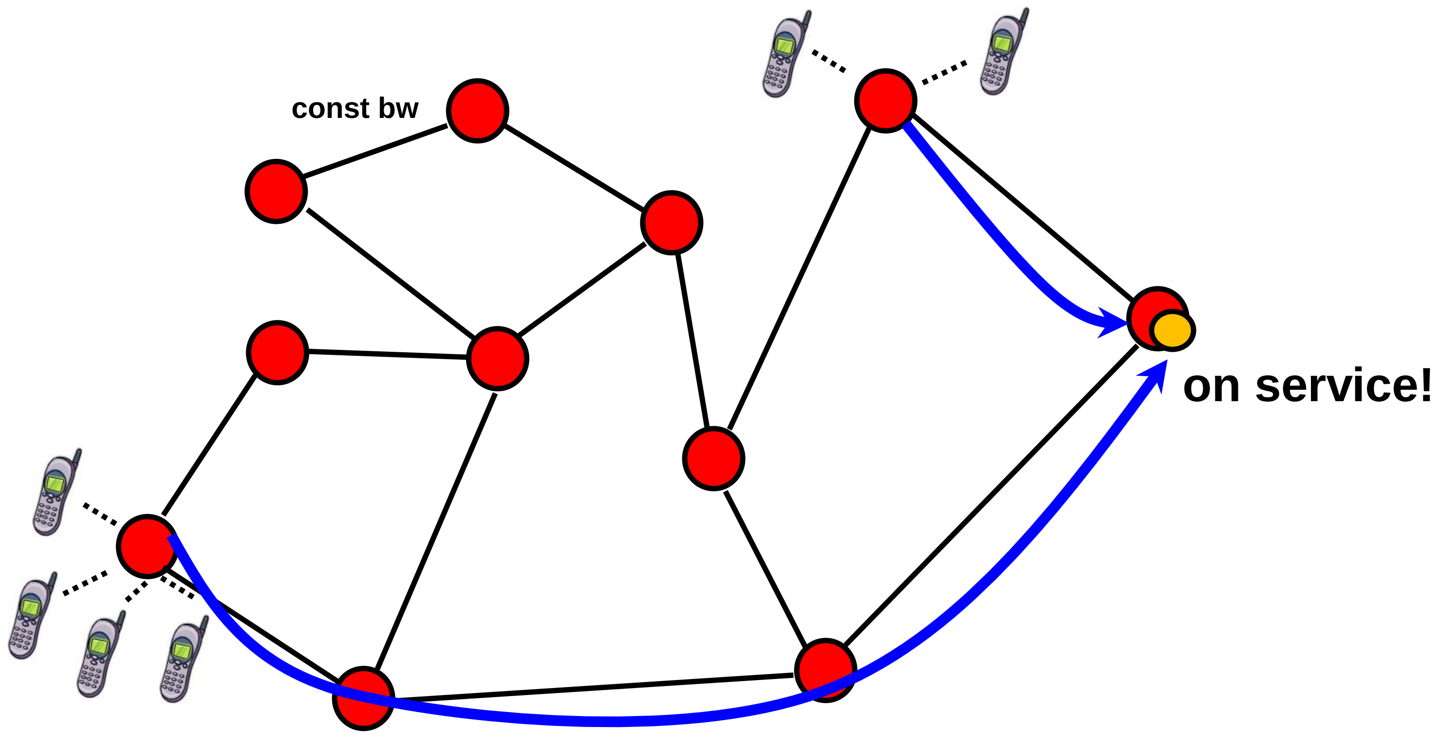
Goal:
*E.g., **QoS** (="move with the sun", or with commuters); or for **maintenance** or to turn off resources (**energy** conservation).*





The Virtual Service Migration Problem.

Bienkowski et al.: SIGCOMM VISA 2010



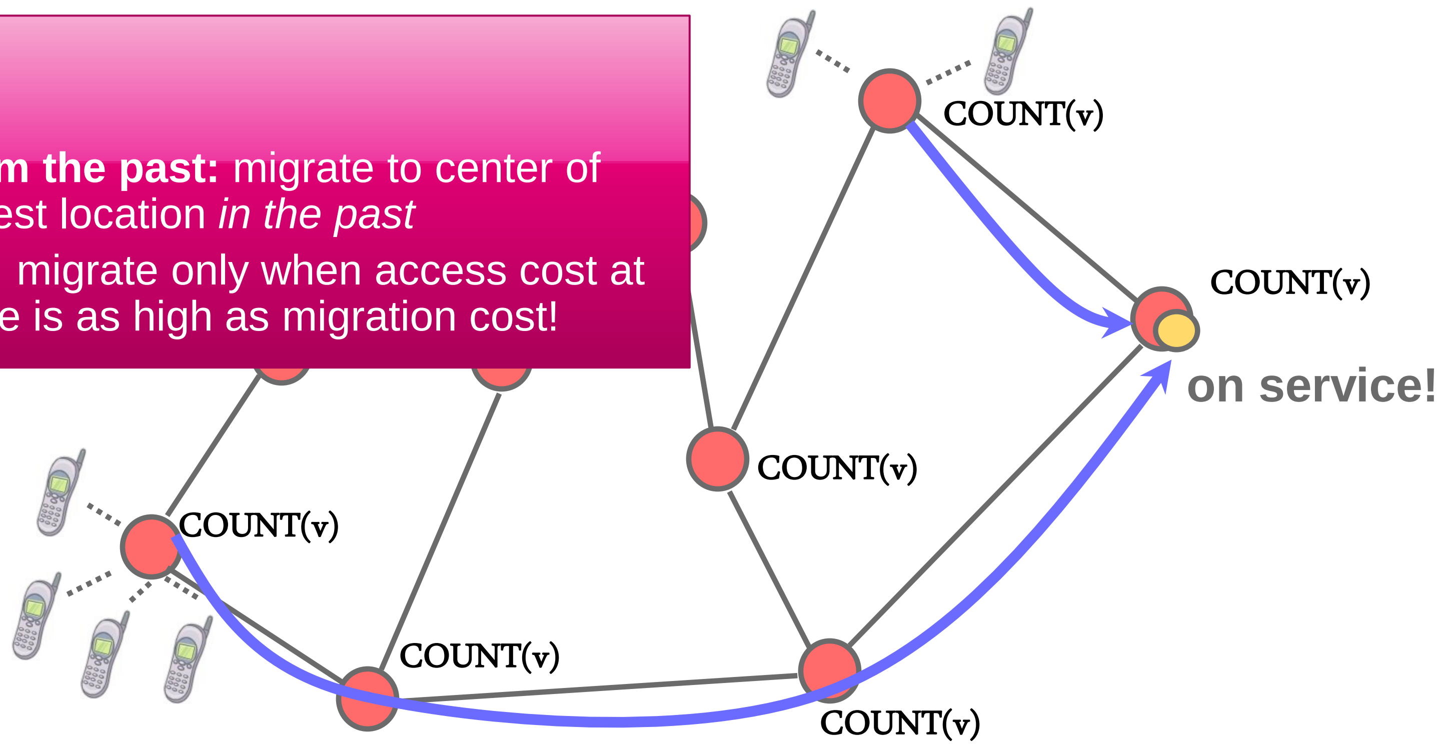
Given a virtual network with guaranteed bandwidth: where to migrate service?

Simple model: one service, constant migration cost (interruption), access along graph.

The Virtual Service Migration Problem.

Idea:

- **Learn from the past:** migrate to center of gravity of best location *in the past*
- **Amortize:** migrate only when access cost at current node is as high as migration cost!



Given a virtual network with guaranteed ban

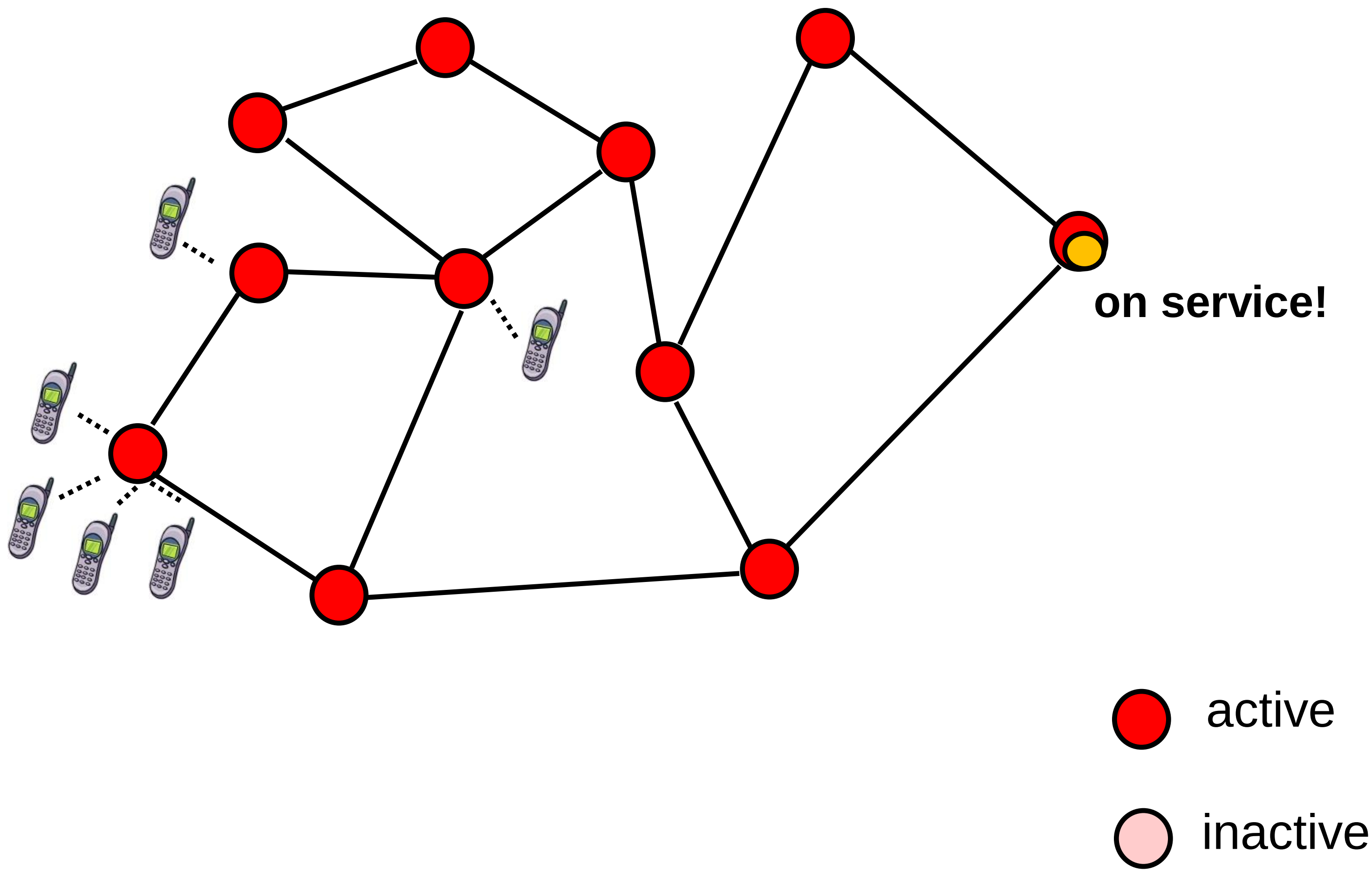
Simple model: one service, constant migration cost (i

Center of Gravity Migration

1. Each node *v*: COUNT(v) = access cost ***epoch***
2. Call nodes v with COUNT(v) < *m* ***active***.
3. If service is at node *w*, a ***phase*** ends when COUNT(w)$\geq$m
4. The service is migrated to the ***center of gravity*** of the remaining active nodes
5. If no such node is left, the epoch ends.

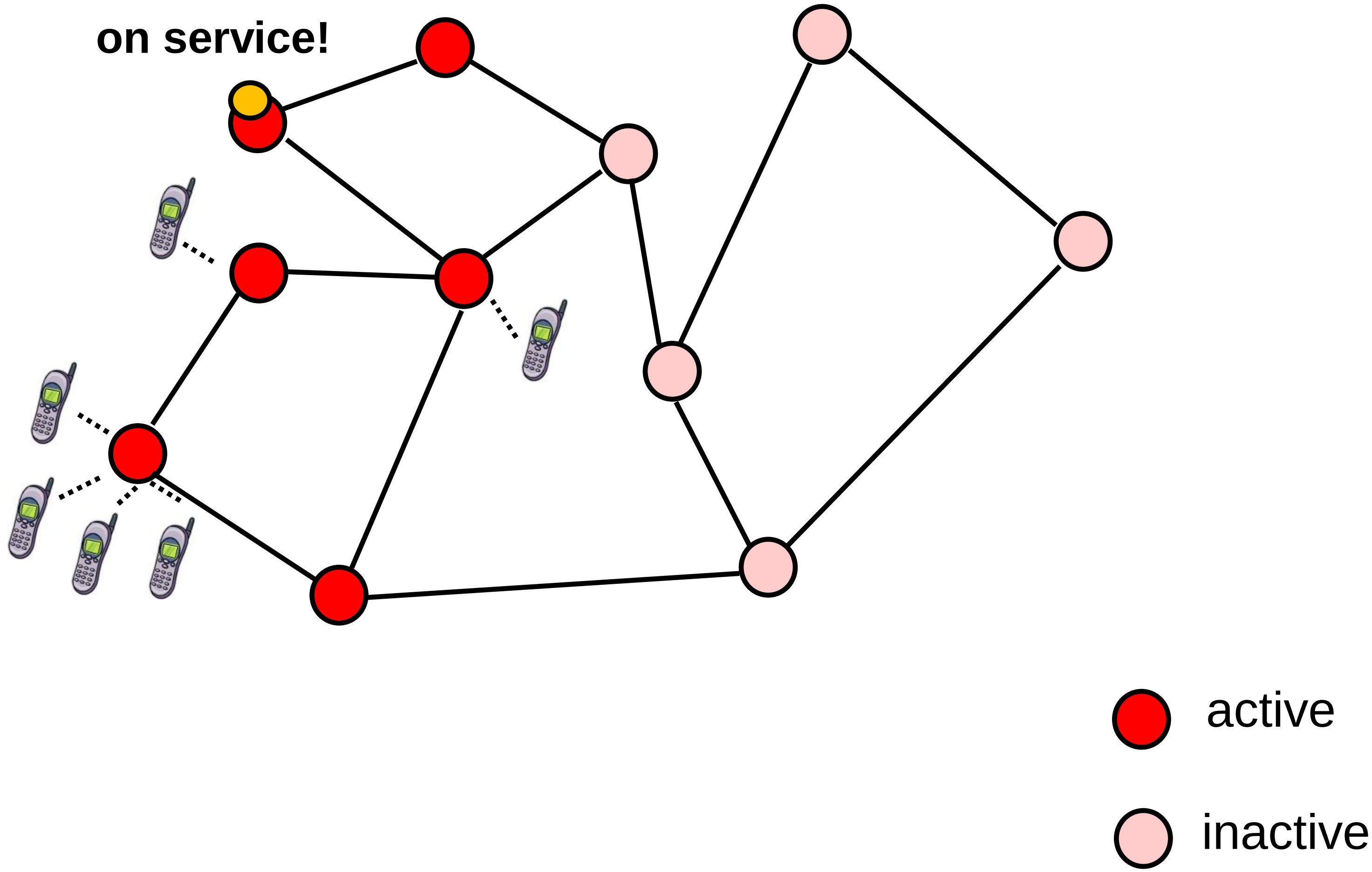
Center-of-Gravity Algo: Example.

Before phase 1:



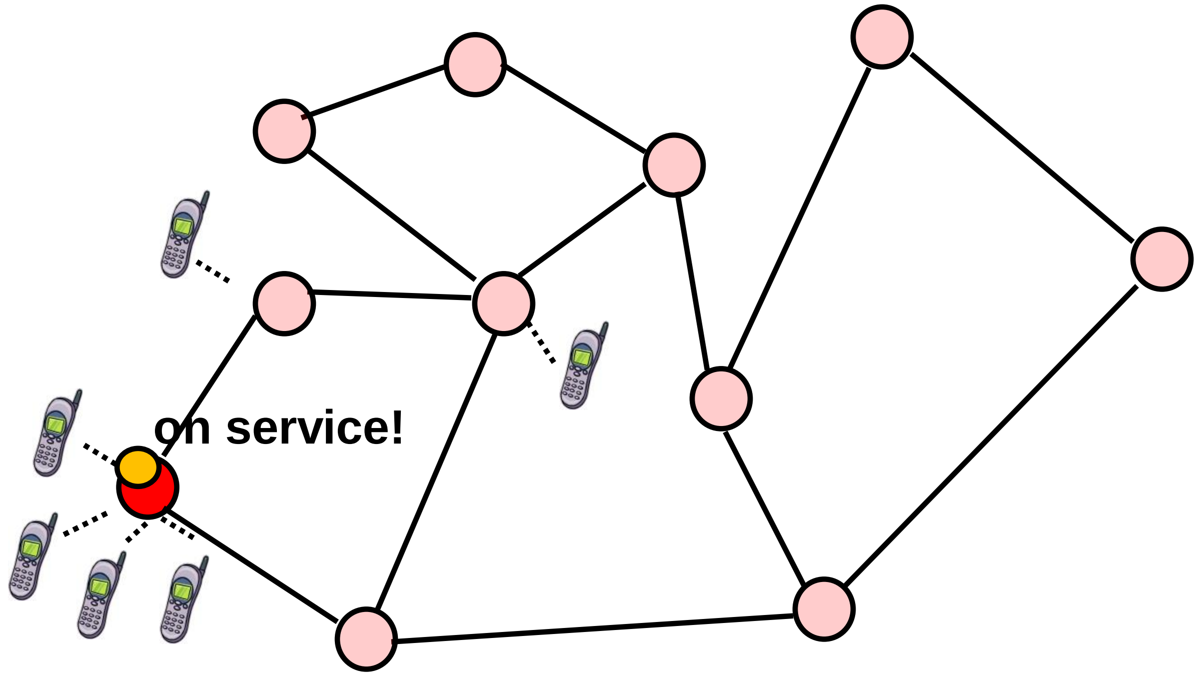
Center-of-Gravity Algo: Example.

Before phase 2:

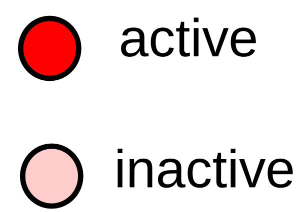


Center-of-Gravity Algo: Example.

End of epoch:



Of course, not converging if demand is dynamic!
(Simplified example.)



Center-of-Gravity Algo: Result.

Competitive analysis? Assume constant bandwidths!

r = ALG / OPT ?

Lower bound cost of OPT:

In an **epoch**, each node has at least access cost ***m***, or there was a migration of cost ***m***.

Upper bound cost of ALG:

We can show that each **phase** has cost at most ***2m*** (access plus migration), and there are at most ***log(n)*** many phases per **epoch**!

Theorem

ALG is log(n) competitive!

A special uniform metrical task system (graph metric for access)!

Optimality?

Schneider et al.: INFOCOM 2013

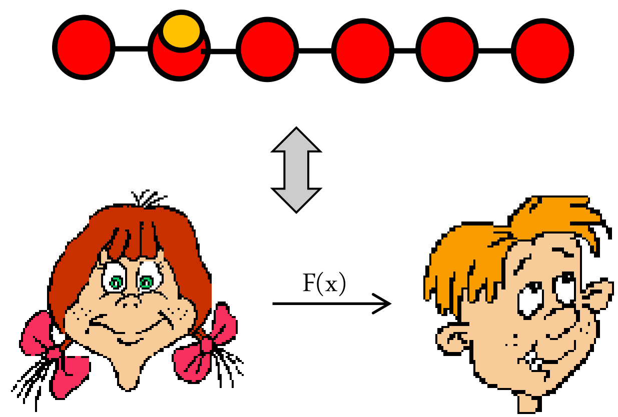
Theorem

«Center of Gravity» algorithm is log(n) competitive!

Also a much simpler randomized algorithm achieves this!

log(n)/loglog(n) lower bound follows from online function tracking reduction!

on service!



Optimality?

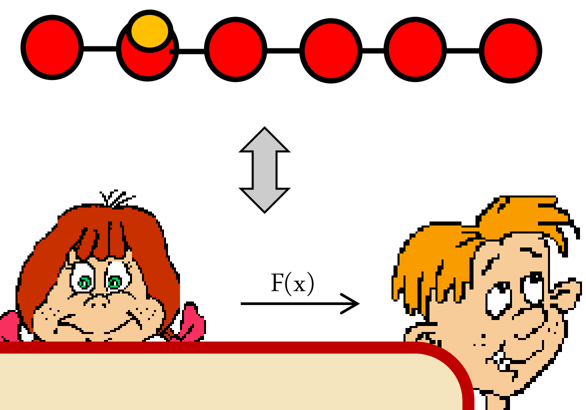
Schneider et al.: INFOCOM 2013

Theorem

«Center of Gravity» algorithm is log(n) competitive!

Also a much simpler randomized algorithm achieves this!

on service!



log(n)/loglog(n) lower bound follows from online function tracking reduction!

There is an asymptotically optimal called FOLLOWER!

The Online Algorithm FOLLOWER.

Concepts:

- **Learn from the past:** migrate to center of gravity of best location *in the past*
- **Amortize:** migrate only when access cost at current node is as high as migration cost!

Simplified Follower

1. F_i are requests handled while service at f_i
2. to compute f_{i+1} (new pos), Follower only takes into account requests during f_i: F_i
3. migrate to center of gravity of F_i, as soon as migration costs there are amortized (and «reset counters» immediately)!

Algorithm *Follower*

$i := 0$; $k_0 := 0$ $\forall j$: $F_j = \{\}$ {The server starts at an arbitrary node f_0}

Upon a new request r do:

Serve request r with server at f_i

$F_i := F_i \cup r$

$f' :=$ arbitrary $u \in CG(F_i)$

$x' := d(f_i, f')$ {for `co.di.`, and $x' := 1$ for `co.nb.m.`}

if $C(f_i, F_i) \geq g(x'|k_i)$ **then**

$f_{i+1} := f'$; $x_i := x'$

$y(w) := d(f_i, w) + d(w, f_{i+1})$ {for `co.di.`, and for `co.nb.m.` $y(w) := 2$ for $w \neq f_{i+1}$ and $y(w) := 1$ otherwise }

$slack(w \in V) := g(y(w)|k_i) - C(f_i, F_i)$

$w_i :=$ Node w with minimum $slack(w)$ such that $slack(w) \geq 0$

Move server to w_i and if $w_i \neq f_{i+1}$ onto f_{i+1}

$k_{i+1} := k_i + y(w_i)$

$i := i + 1$

end if

The Online Algorithm FOLLOWER.

Concepts:

- **Learn from the past:** migrate to center of gravity of best location *in the past*
- **Amortize:** migrate only when access cost at current node is as high as migration cost!

Algorithm *Follower*

$i := 0$; $k_0 := 0$ $\forall j$: $F_j = \{\}$ {The server starts at an arbitrary node f_0}

Upon a new request r do:

Serve request r with server at f_i

$F_i := F_i \cup r$

$f' :=$ arbitrary $u \in CG(F_i)$

$x' := d(f_i, f')$ {for `co.di.`, and $x' := 1$ for `co.nb.m.`}

if $C(f_i, F_i) \geq g(x'|k_i)$ **then**

  $f_{i+1} := f'$; $x_i := x'$

  $y(w) := d(f_i, w) + d(w, f_{i+1})$ {for `co.di.`, and for `co.nb.m.` $y(w) := 2$ for $w \neq f_{i+1}$ and $y(w) := 1$ otherwise }

  $slack(w \in V) := g(y(w)|k_i) - C(f_i, F_i)$

  $w_i :=$ Node w with minimum $slack(w)$ such that $slack(w) \geq 0$

  Move server to w_i and if $w_i \neq f_{i+1}$ onto f_{i+1}

  $k_{i+1} := k_i + y(w_i)$

  $i := i + 1$

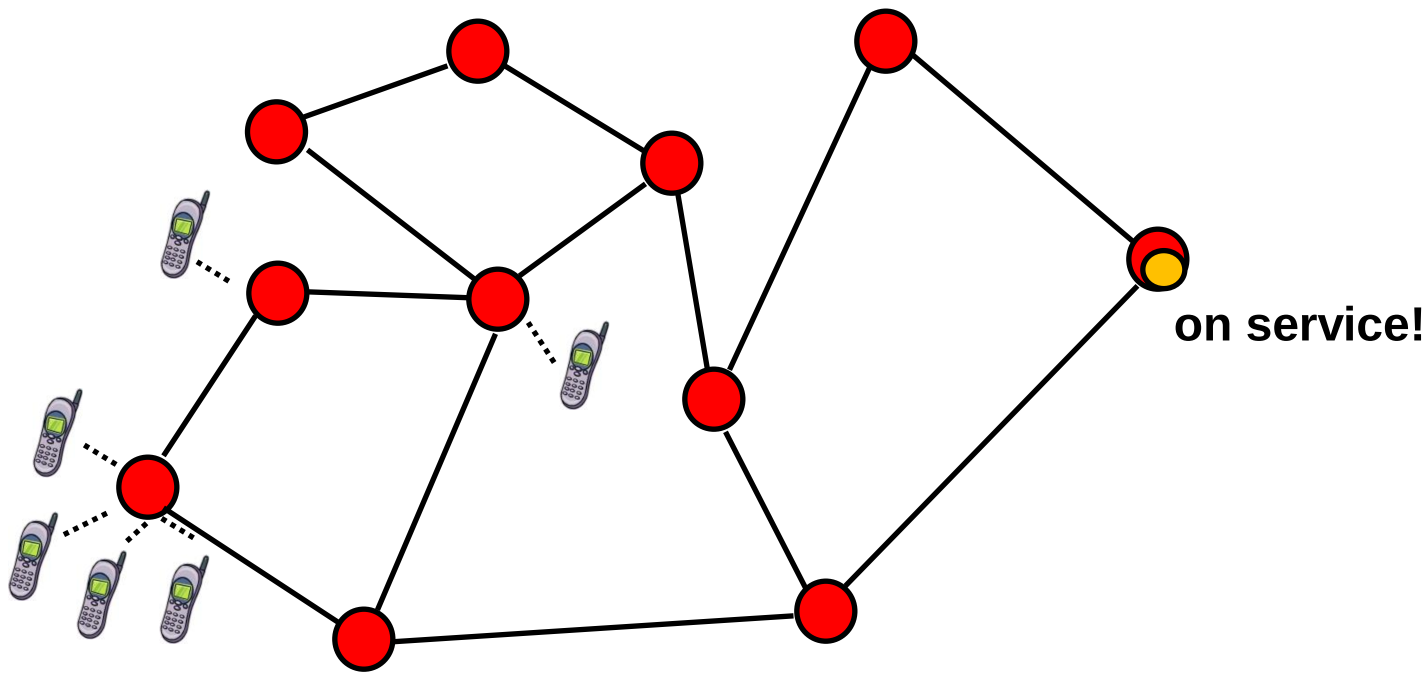
end if

Simplified Follower

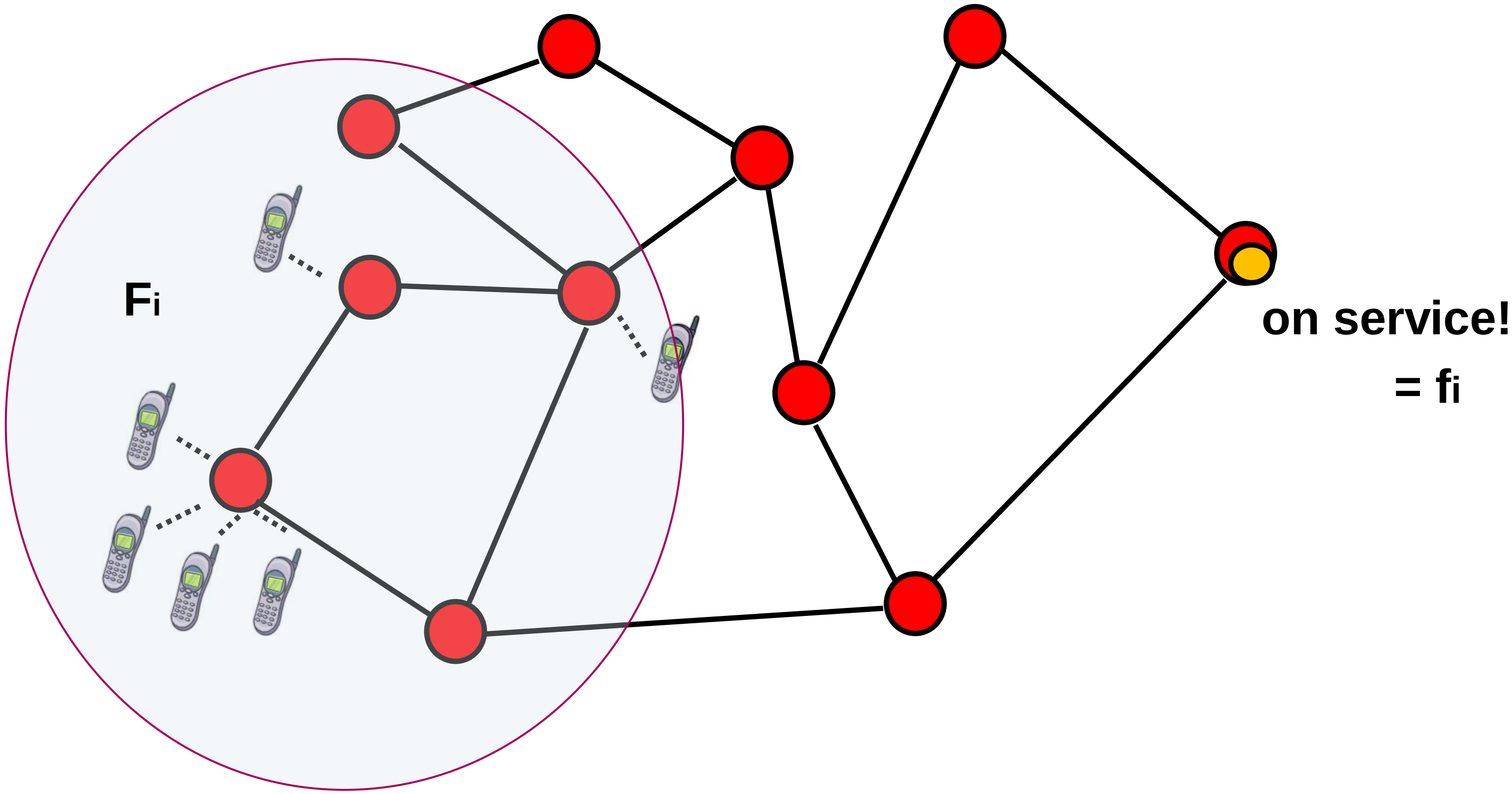
1. F_i are requests handled while service at f_i
2. to compute f_{i+1} (new pos), Follower only takes into account requests during f_i: F_i
3. migrate to center of gravity of F_i, as soon as migration costs there are amortized (and «reset counters» immediately)!

Also works for migrations with discount!
(Reseller/broker gives discount!)

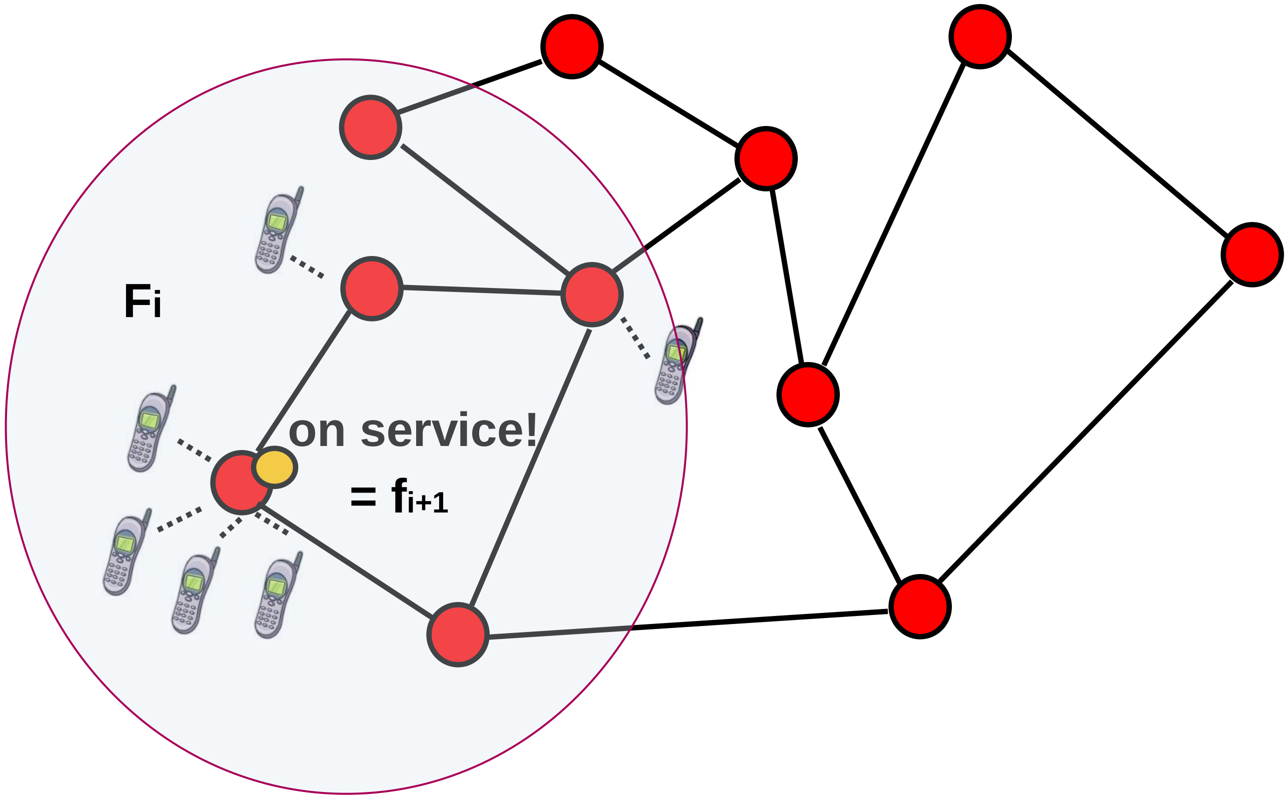
Intuition.



Intuition.



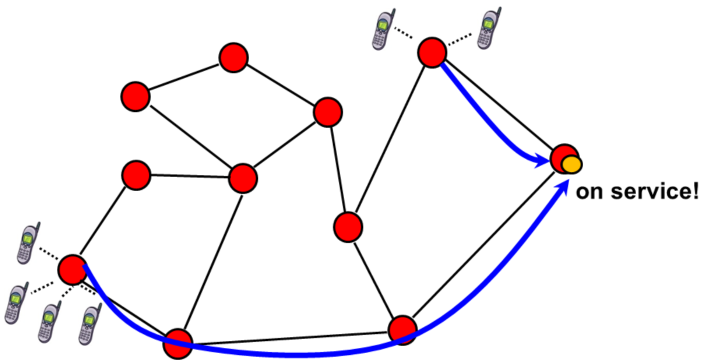
Intuition.



Modeling Access and Migration Costs.

Access Costs

Latency along shortest path in graph.
(Graph distances, and in particular: metric!)



Migration Costs

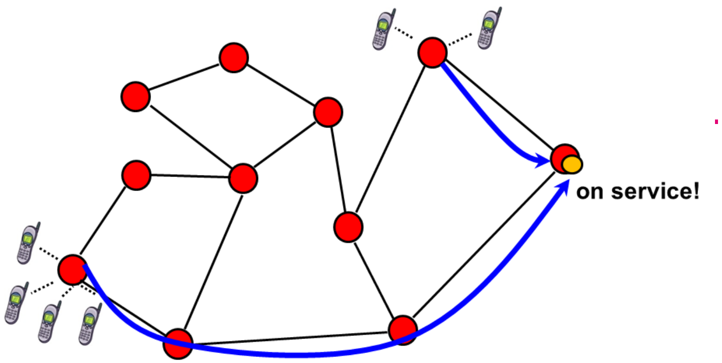
Generalized models:

- E.g., depends on bandwidth along path (duration of service interruption)
- E.g., depends on distance travelled (latency)
- Discount: e.g., VNP (number of migrations, distance travelled, ...)

Modeling Access and Migration Costs.

Access Costs

Latency along shortest path in graph.
(Graph distances, and in particular: metric!)



Migration Co

Generalized mo

- E.g.

(d

r

elled, ...)

General cost function g(x|y): cost of migrating distance x given already travelled y

Or g(1|y): cost of migration given we already migrated y times

Competitive Ratio of FOLLOWER.

Competitive analysis? **FOLLOWER / OPT?**

Theorem

**If no discounts are given,
Follower is log(n)/loglog(n) competitive!**

Simple model with *migration costs = bandwidth*, and *homogeneous*

Page migration model with *migration costs = distance,* but discounts

Theorem

If migration costs depend on travelled distance (page migration), competitive ratio is O(1), even with discounts.

Related Work.

- Metrical Task Systems:
  - Classical online problem where server at certain location («state») serves requests at certain costs; state transitions also come at certain costs («migration»)
  - Depending on migration cost function more general (we have graph access costs) and less general (we allow for migration discounts)
  - E.g., uniform space metrical task system: migration costs constant, but access costs more general than graph distances! Lower bound of log(n) vs log(n)/loglog(n) upper bound in our case.

- Online Page Migration
  - Classical online problem from the 80ies; we generalize cost function to distance discounts, while keeping O(1)-competitive

Our work lies between!

Simulation.

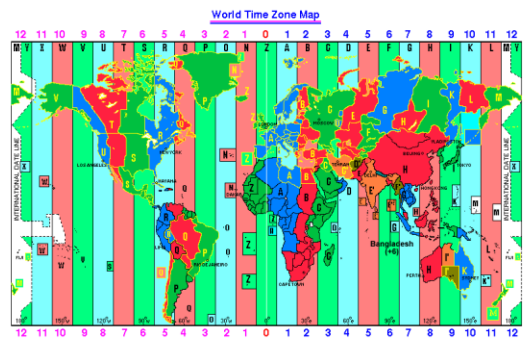
Commuter Scenario

Dynamics due to mobility: requests cycle through a 24h pattern: in the morning, requests distributed widely (people in suburbs), then focus in city centers; in the evening, reverse.

Time Zone Scenario

Dynamics due to time zone effects: request originate in China first, then more requests come from European countries, and finally from the U.S.

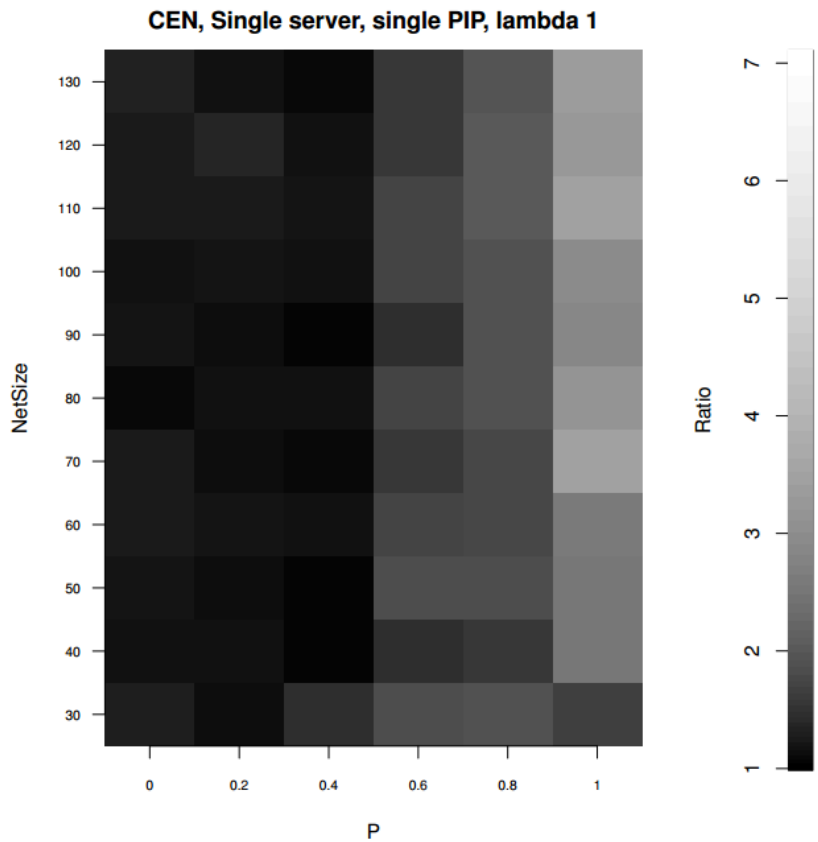
Predictable scenarios, but we do not exploit that. Reality less predictable!

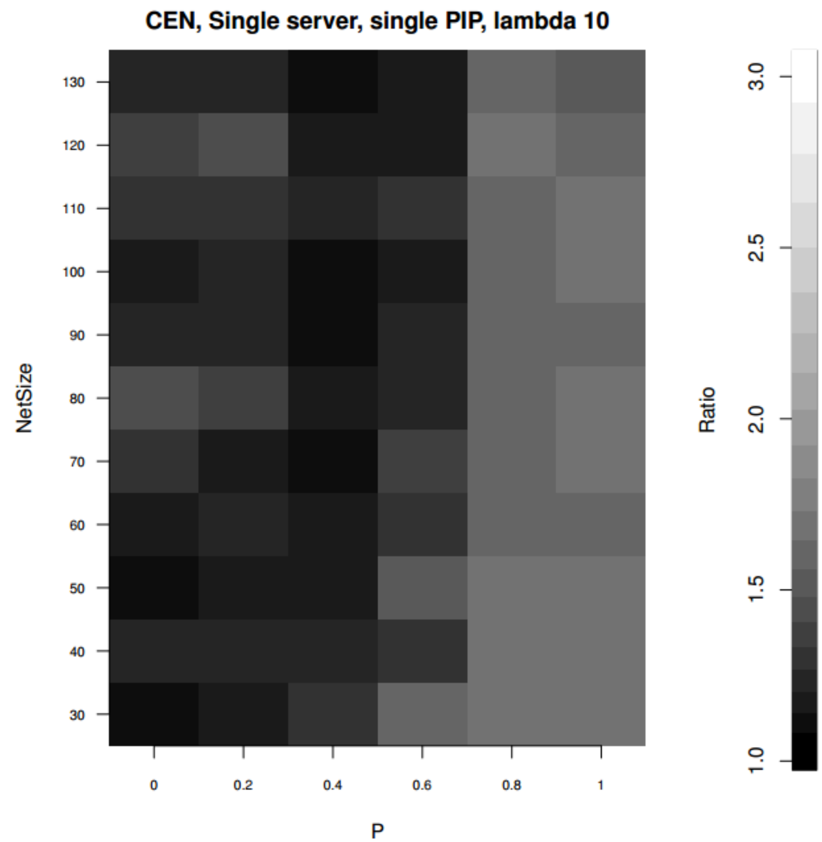


Static Algorithm

Algorithm which uses optimal static server placements for a given request seq.

Results.





Competitive ratio generally relatively low. Increases for more correlated requests and more dynamics.

Related Work.

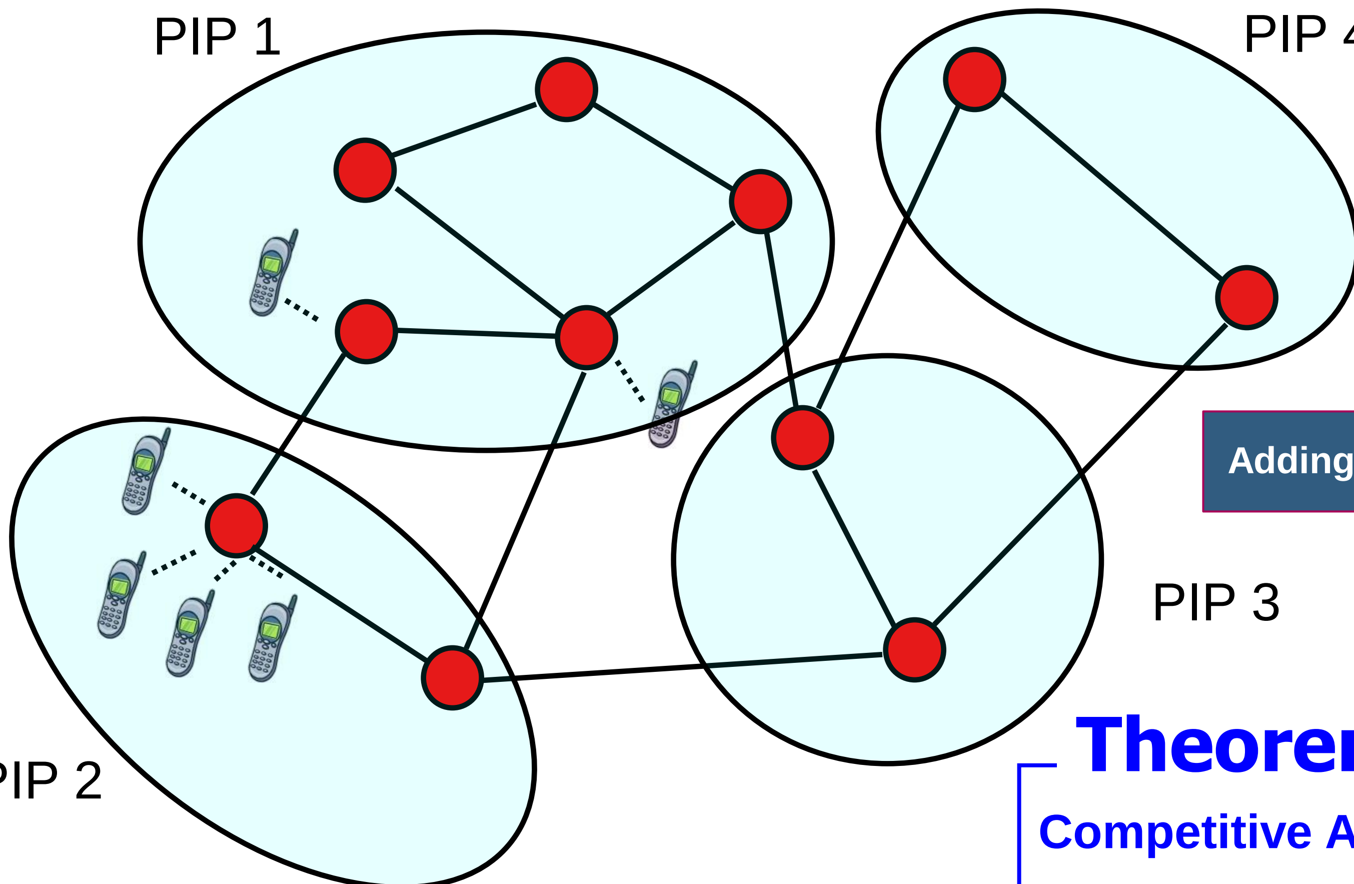
- Metrical Task Systems:
  - Classical online problem where server at certain location («state») serves requests at certain costs; state transitions also come at certain costs («migration»)
  - Depending on migration cost function more general (we have graph access costs) and less general (we allow for migration discounts)
  - E.g., uniform space metrical task system: migration costs constant, but access costs more general than graph distances! Lower bound of log(n) vs log(n)/loglog(n) upper bound in our case.

- Online Page Migration
  - Classical online problem from the 80ies; we generalize cost function to distance discounts, while keeping O(1)-competitive

Our work lies between!

Extension: Inter-Provider Migration.

Migration across provider boundary costs transit/roaming costs (# transit providers), detailed topology not known, etc.



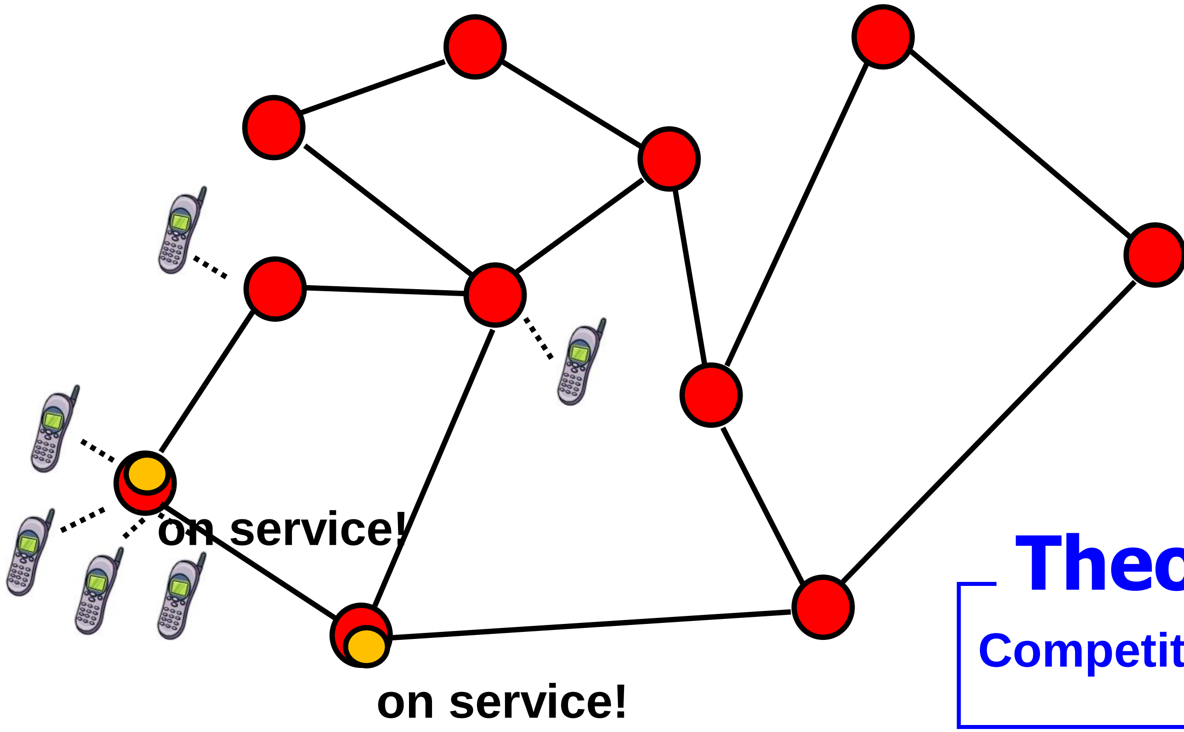
Adding page migration aspects...

Theorem

Competitive ALGs still exist! ☺

Extension: Multiple Servers.

Multiple servers allocated and migrated dynamically depending on demand and load, servers have running costs, etc.



Theorem

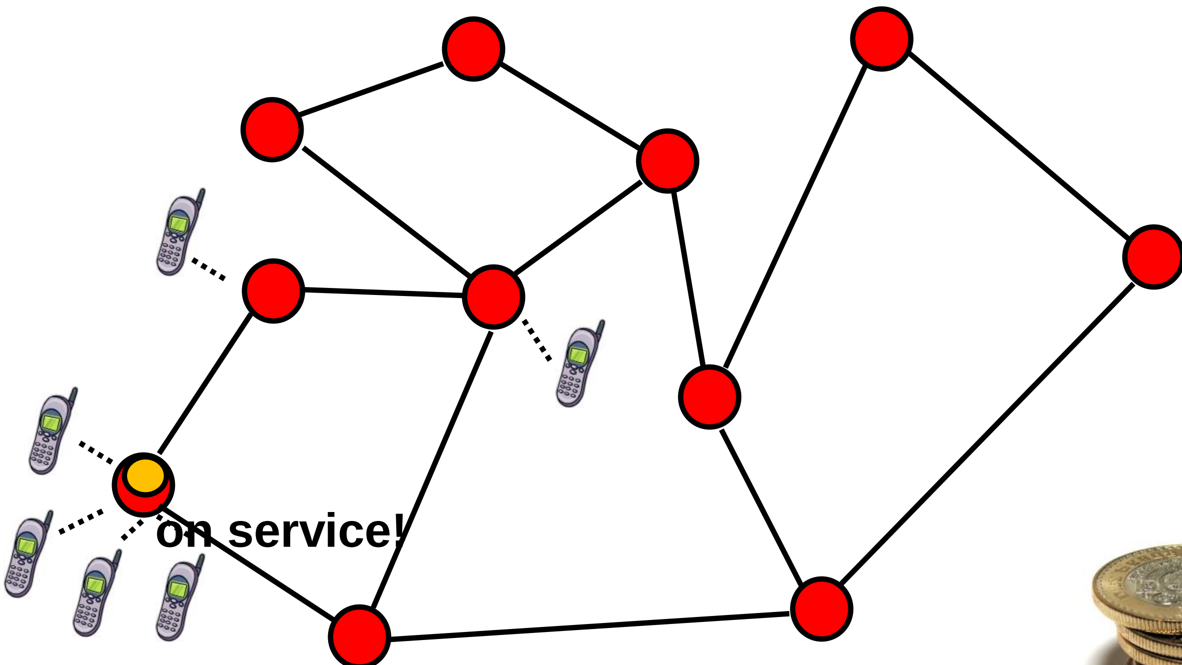
Competitive ALGs still exist! ☺

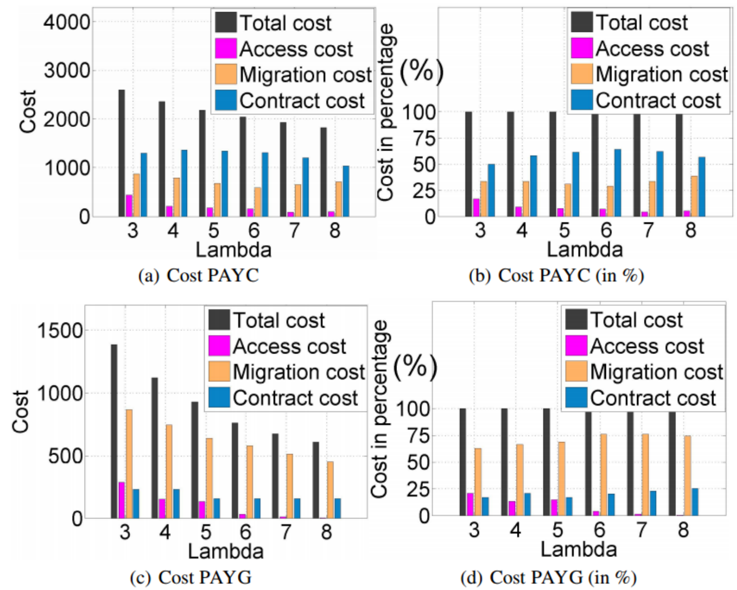
Extension: Economical Aspects.

Hu et al.: ICDCN 2013

How to price resources?

Pay-as-you-go vs Pay-as-you-come?

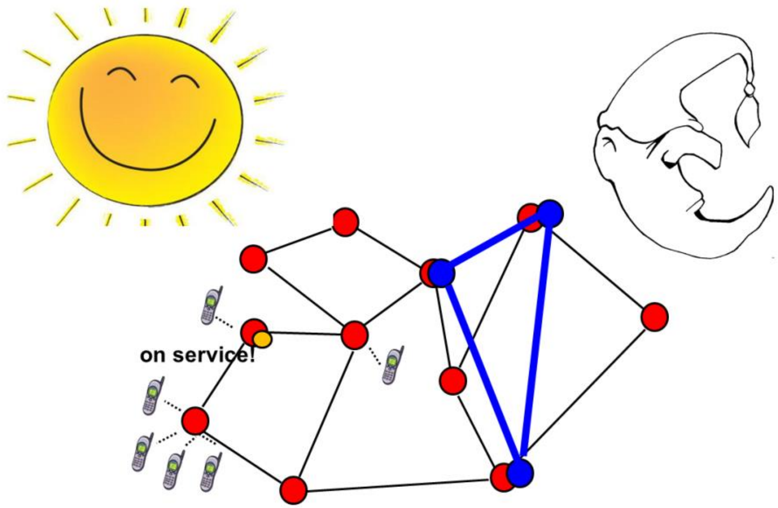




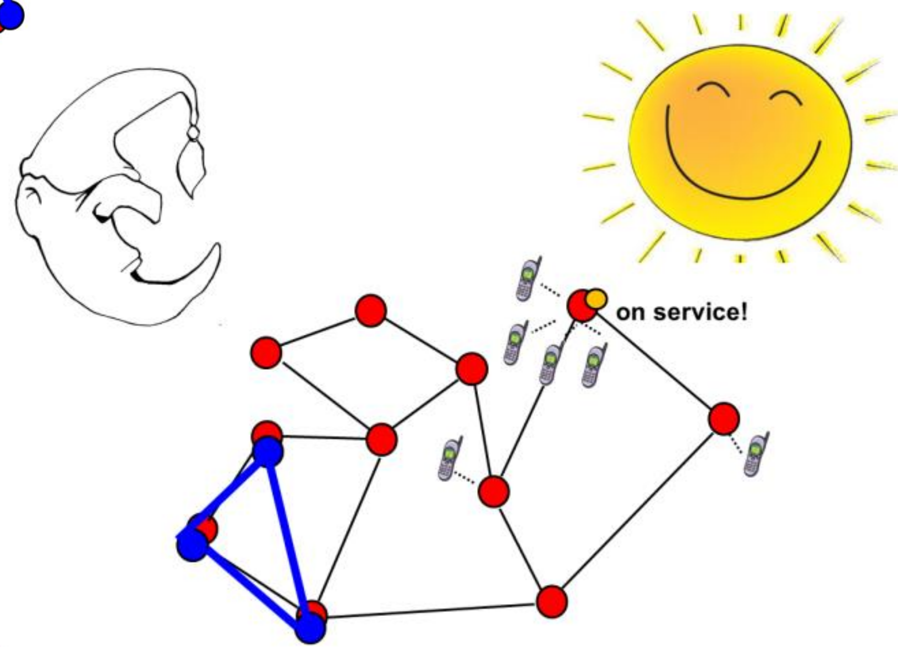
(a) Cost PAYC
(b) Cost PAYC (in %)
(c) Cost PAYG
(d) Cost PAYG (in %)

Migration of Entire CloudNets.

INFOCOM Demo 2013



2 pm in Europe

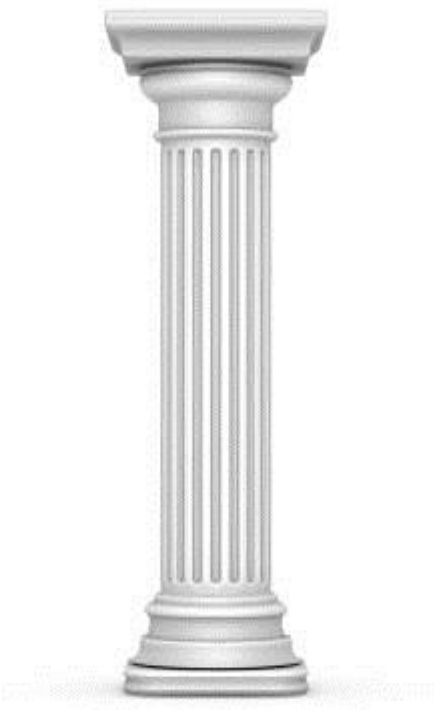


2 pm in Japan

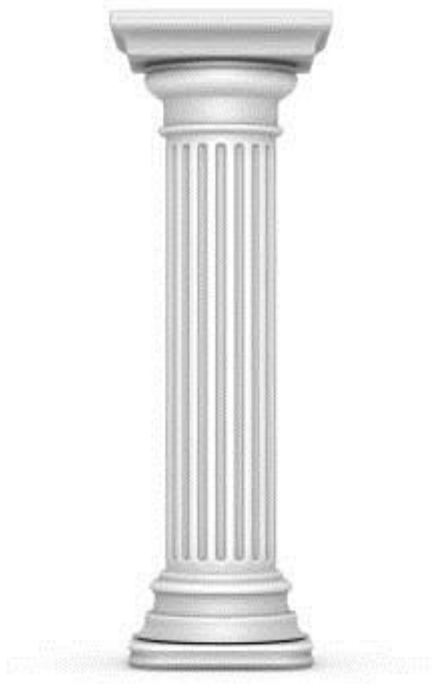
Latency-resource tradeoffs: move-with-the-sun vs move-with-the-moon?

Security of Embedding.

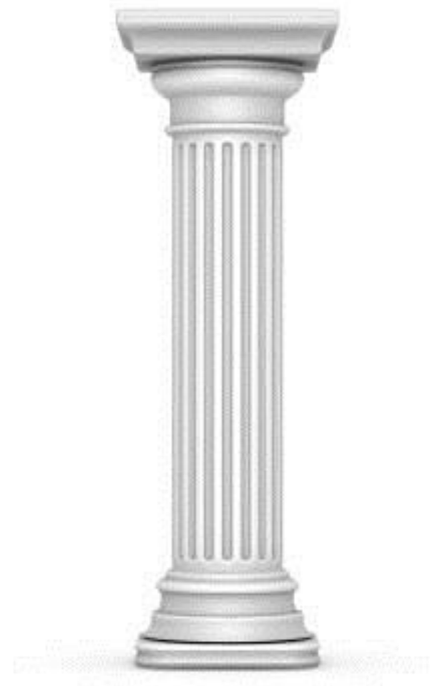
Access Control and Embedding



Service Migration

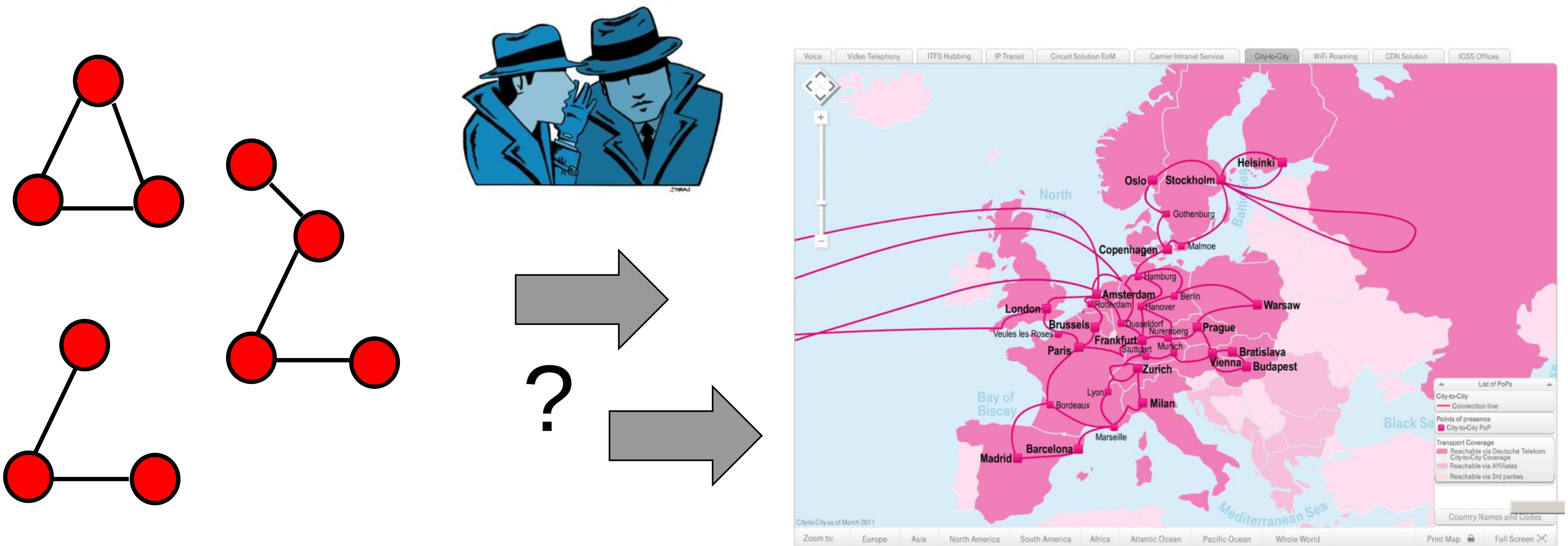


Security Issues



Security Issues.

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

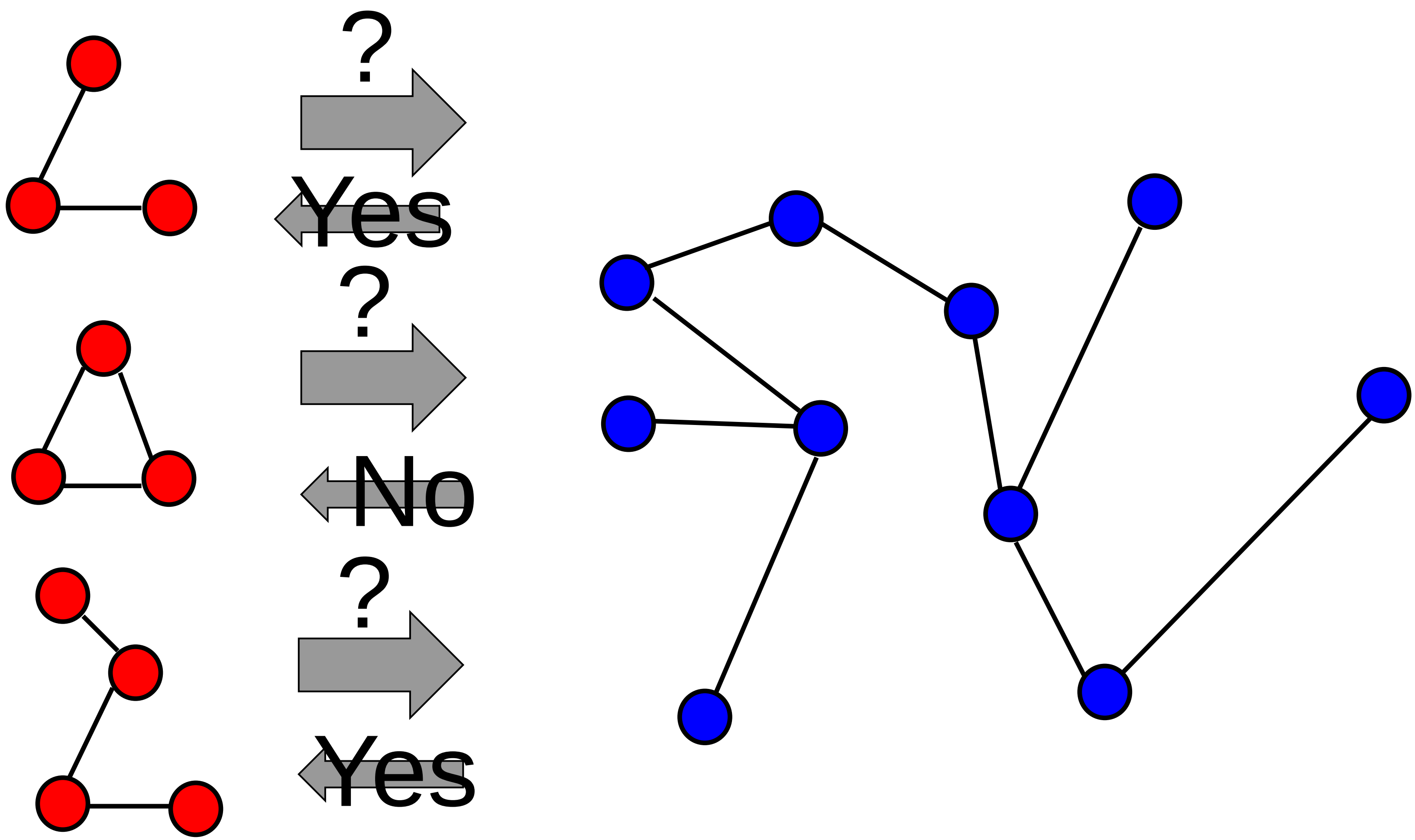


Request Complexity.

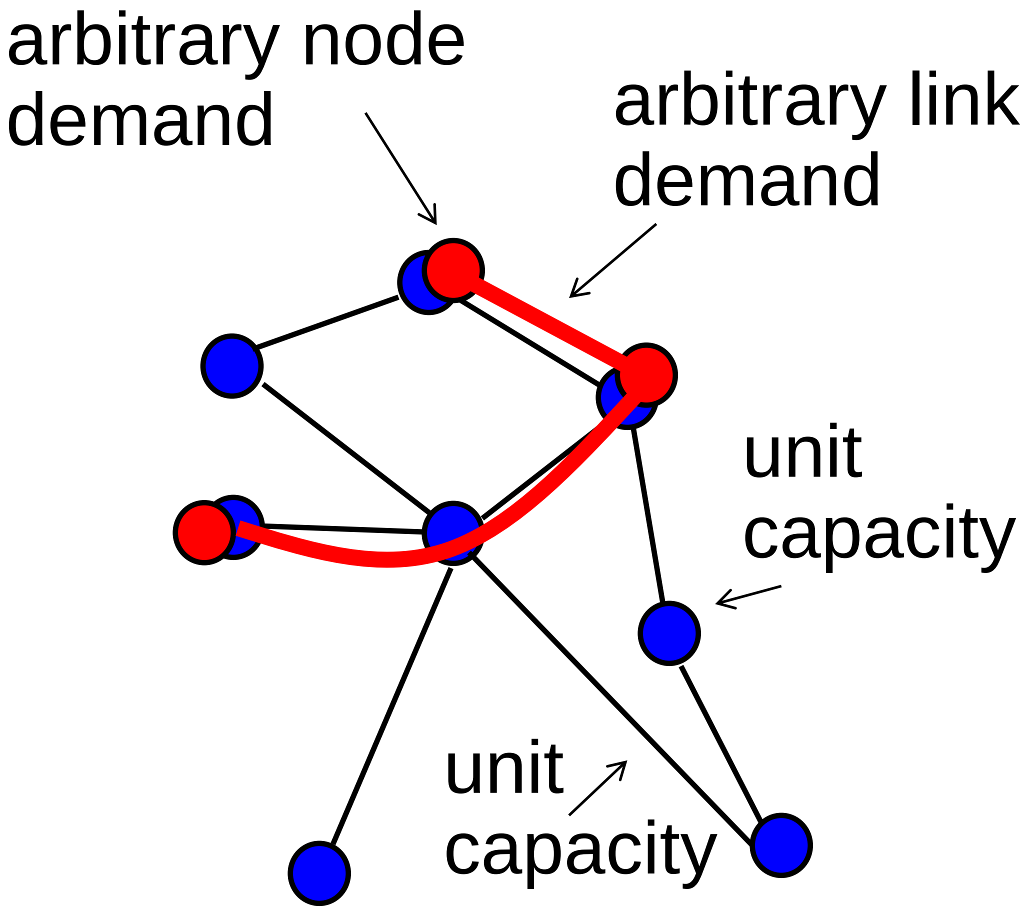
Are CloudNet embeddings a threat for ISPs?

Request Complexity

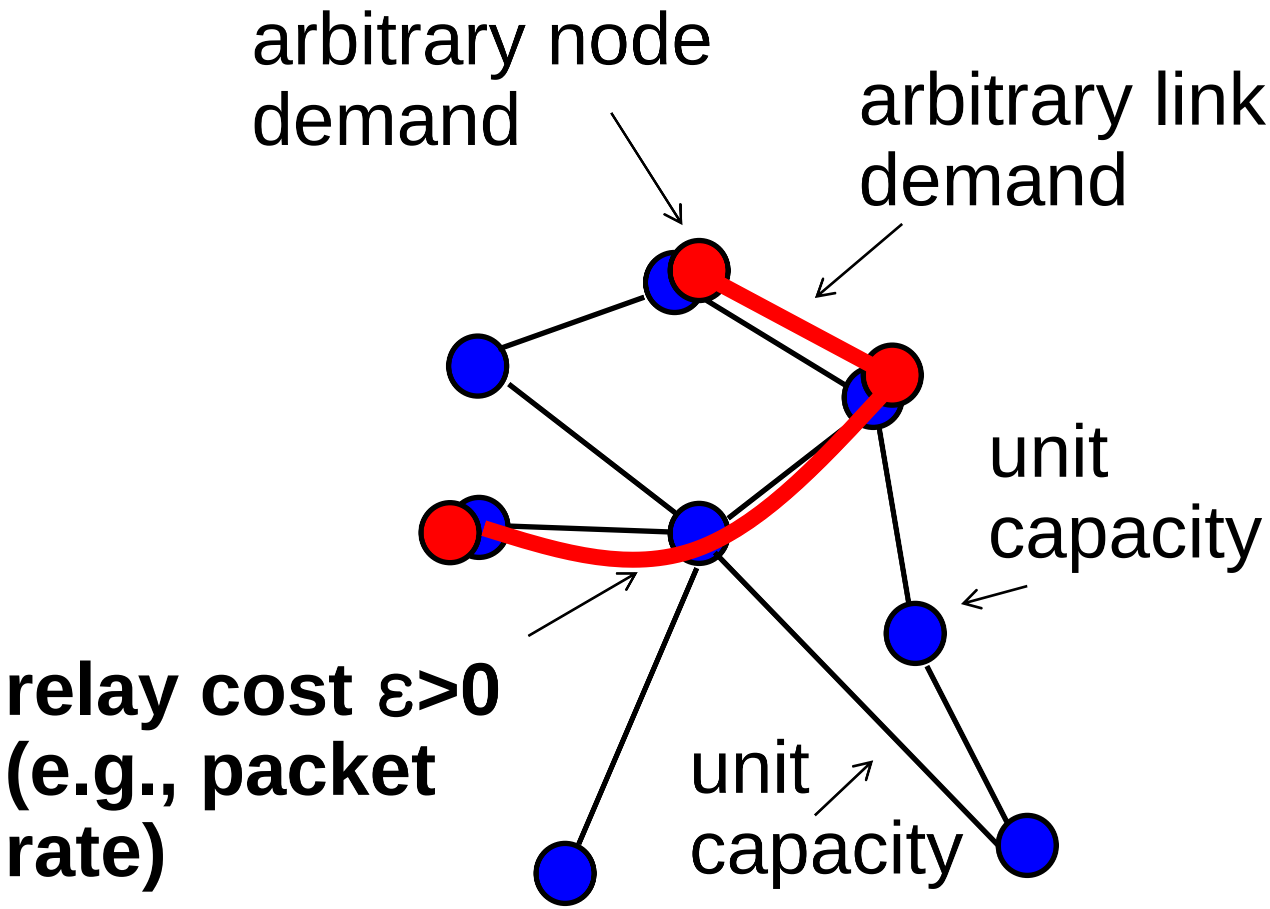
How many embeddings needed to fully reveal topology?



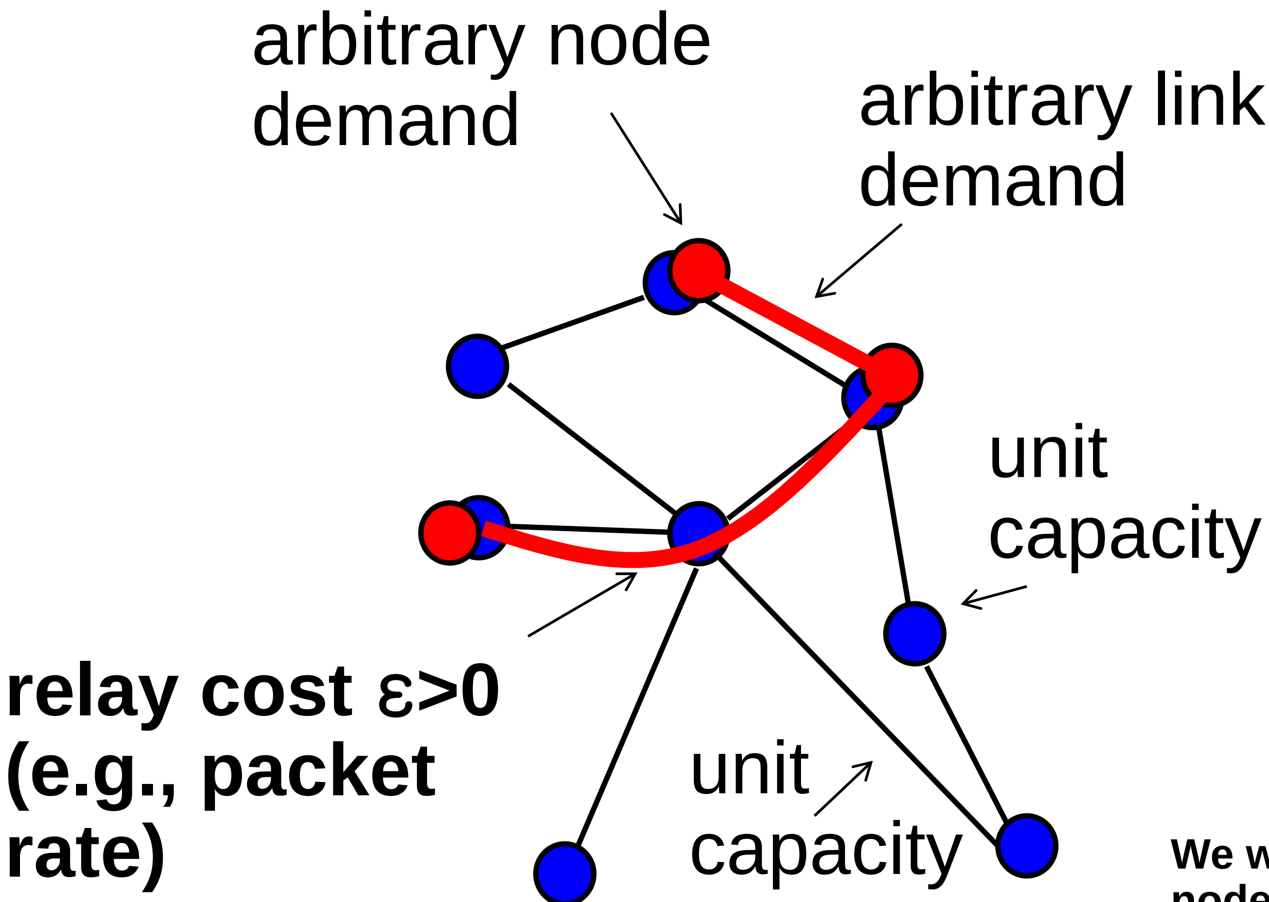
Embedding Model.



Embedding Model.



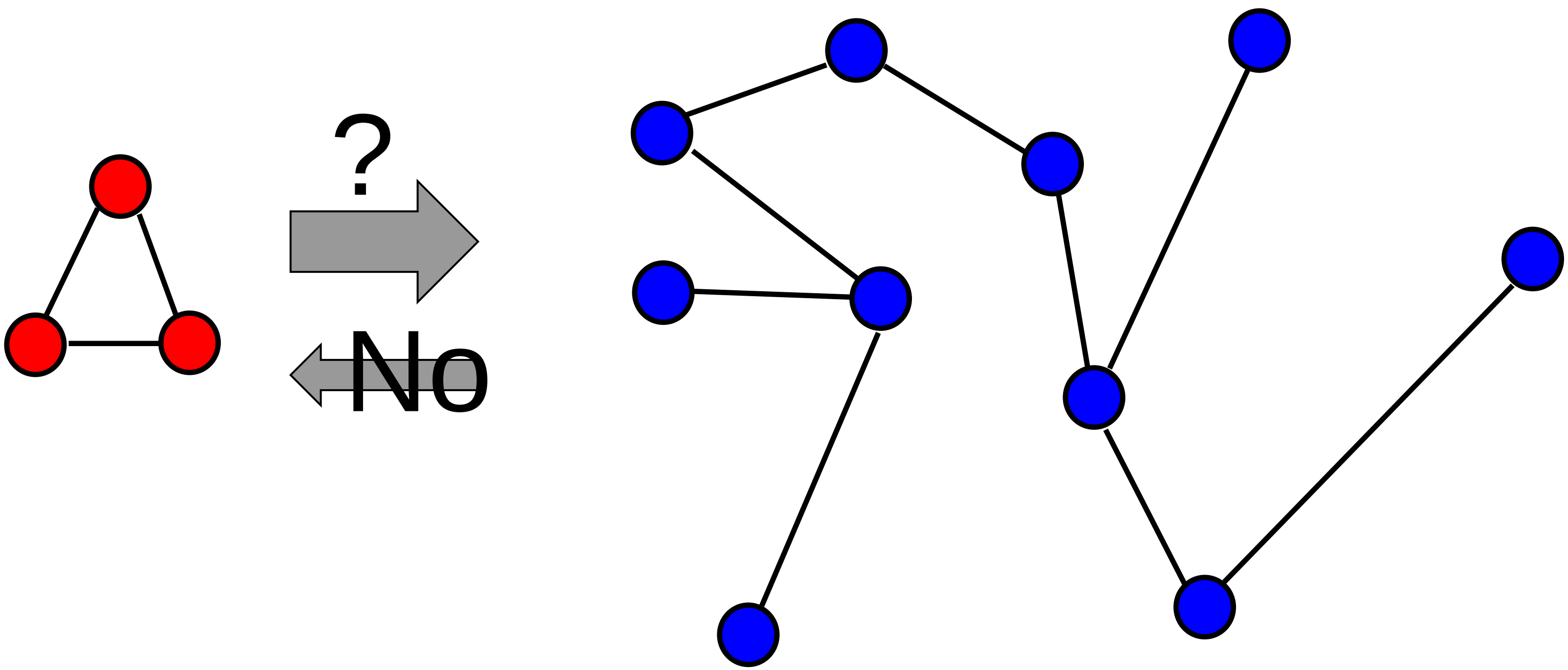
Embedding Model.



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?»

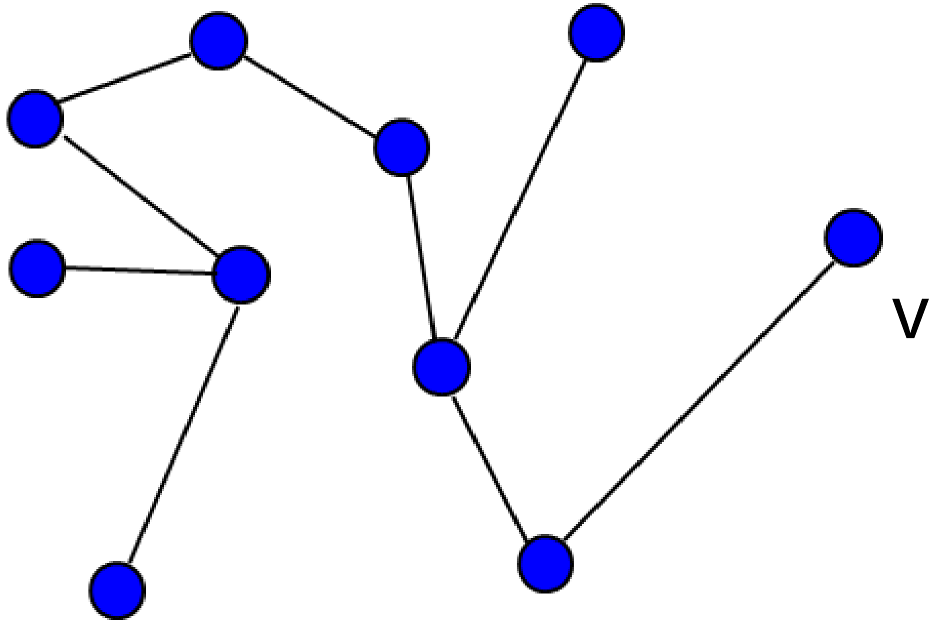


Example: Tree.

How to discover a tree?

Graph growing:

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

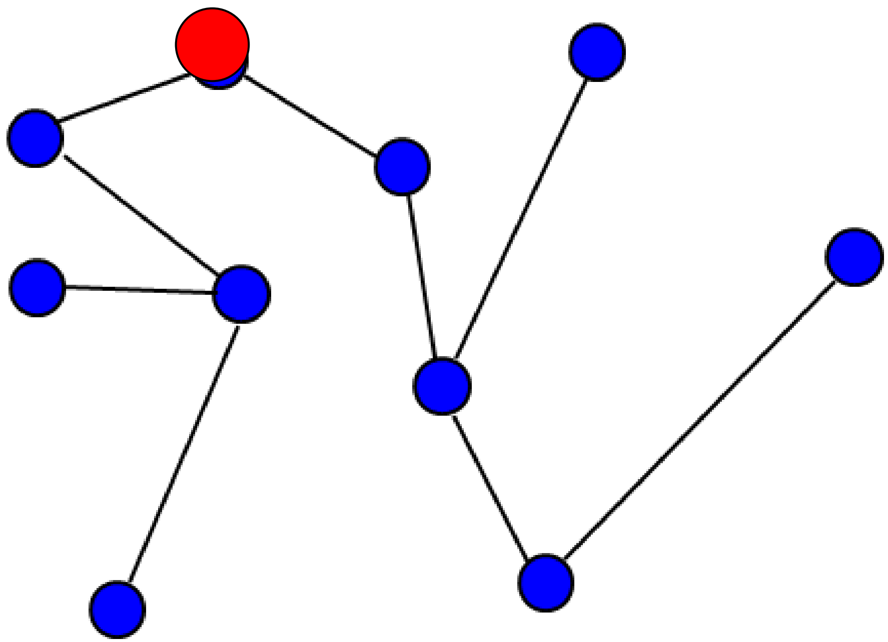


Example: Tree.

How to discover a tree?

Graph growing:

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

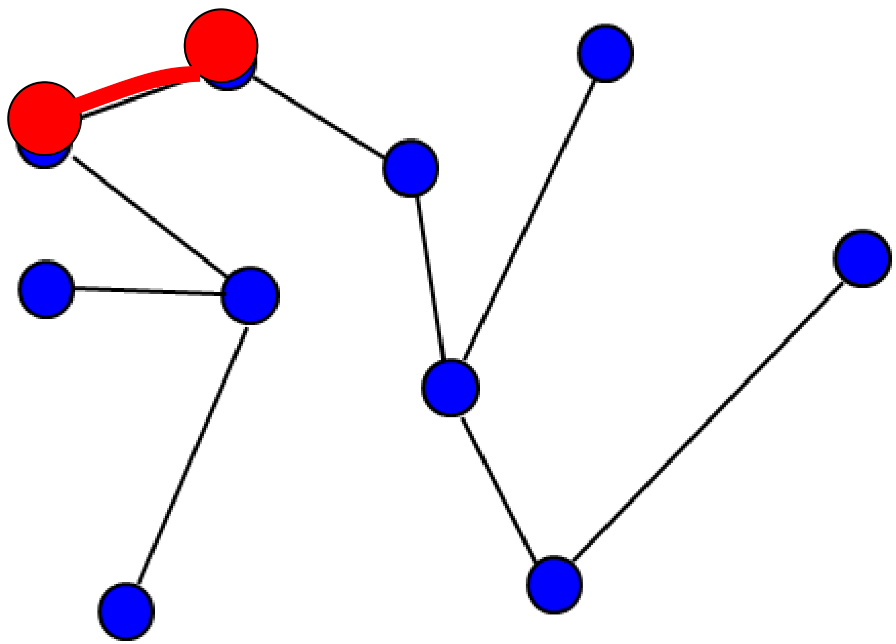


Example: Tree.

How to discover a tree?

Graph growing:

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

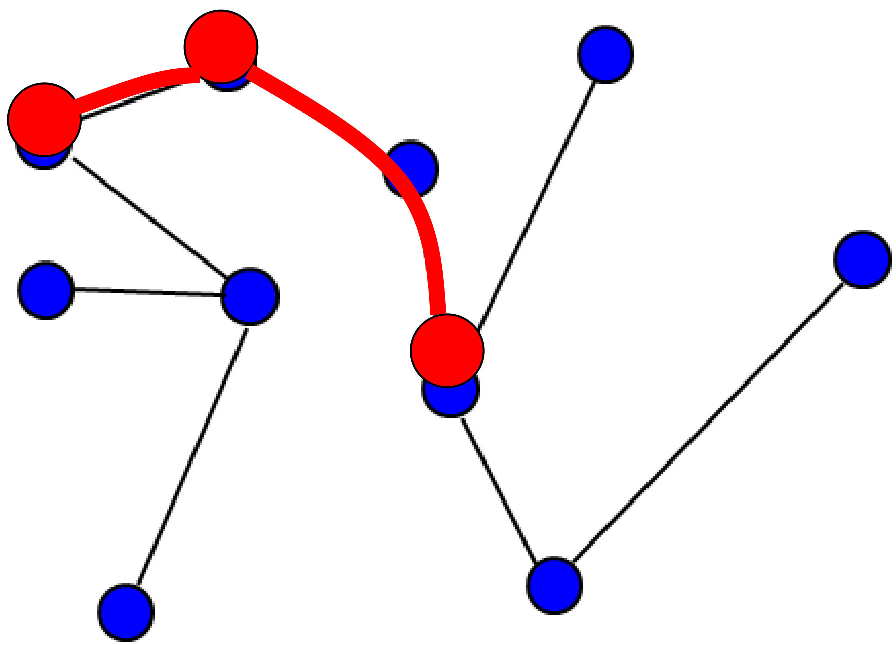


Example: Tree.

How to discover a tree?

Graph growing:

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

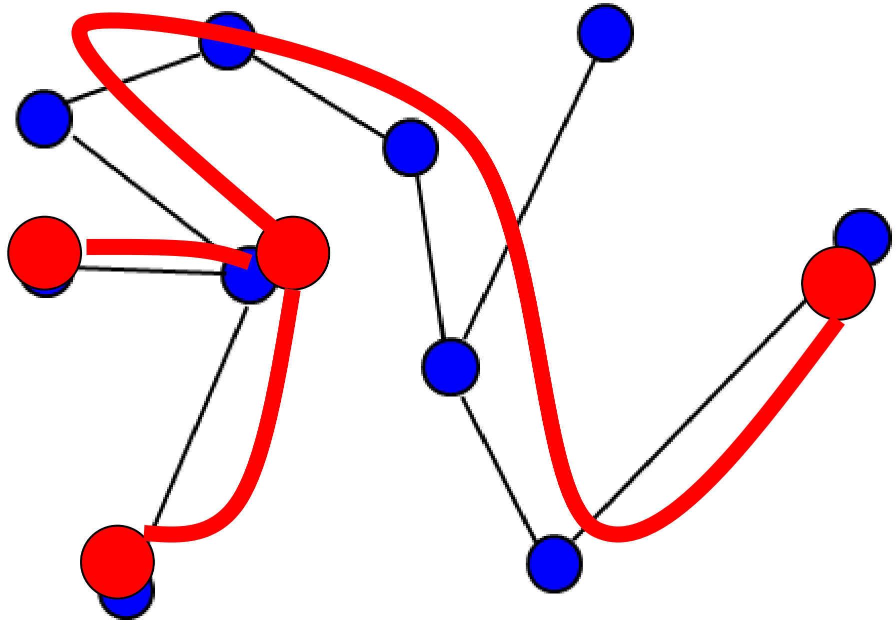


Example: Tree.

How to discover a tree?

Graph growing:

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

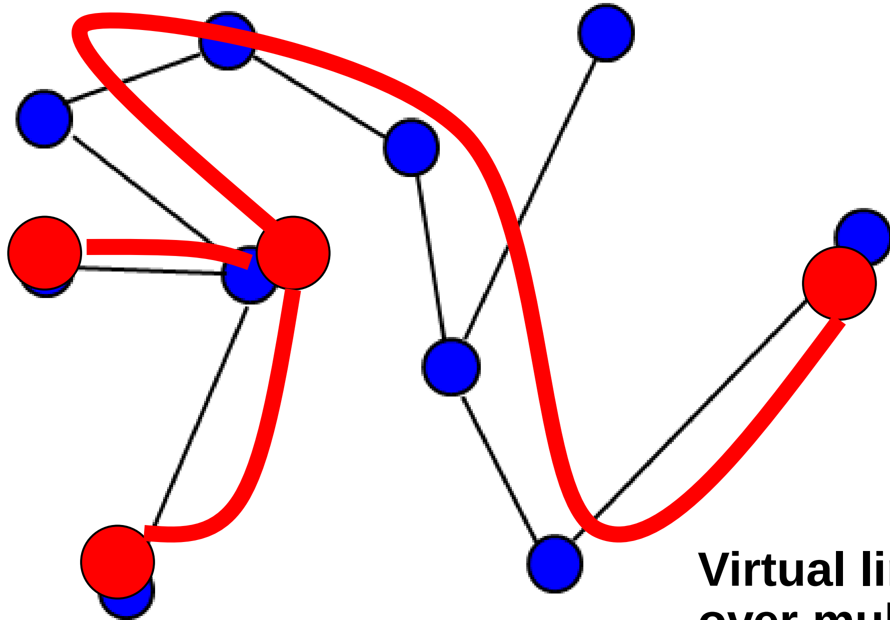


Example: Tree.

How to discover a tree?

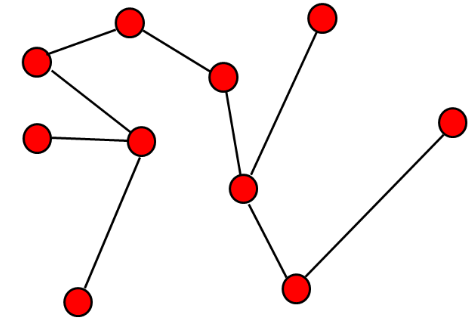
Graph growing:

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



Virtual links may be embedded over multiple physical links!

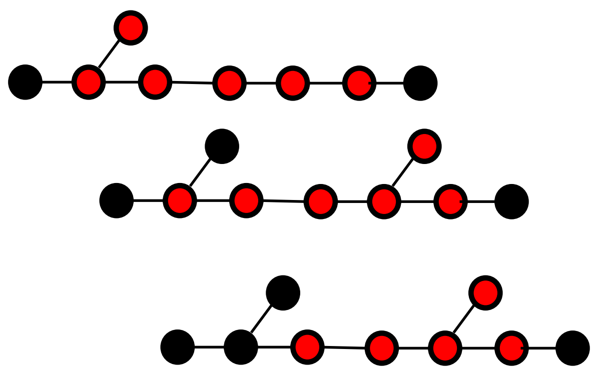
Tree Solution: Graph Growing.



TREE ALGORITHM: line strategy

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

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



Analysis: Amortized analysis on links: Per discovered physical link at most one query, plus at most one per physical node (no incident links).

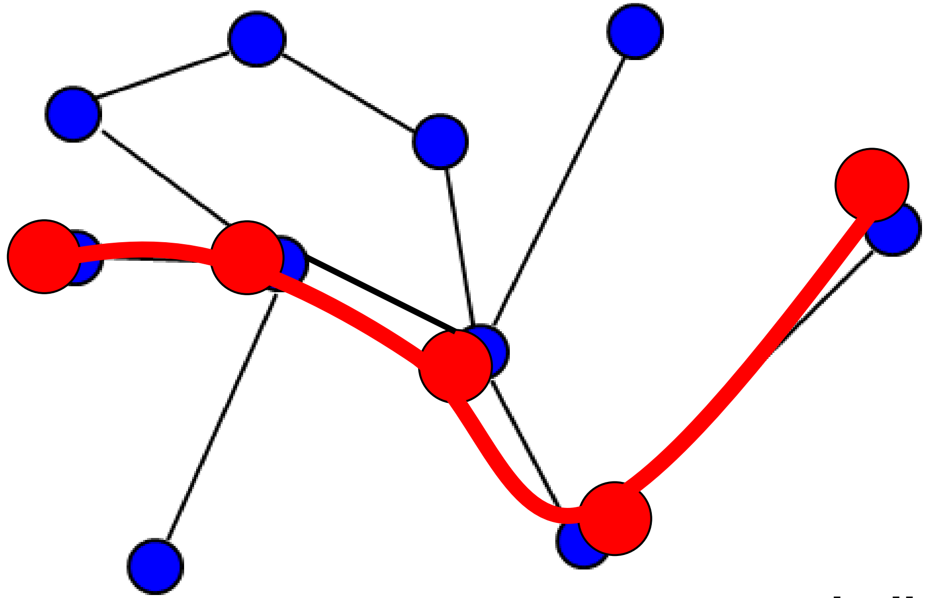
Algorithm 1 Tree Discovery: TREE

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

exploreSequence(v)

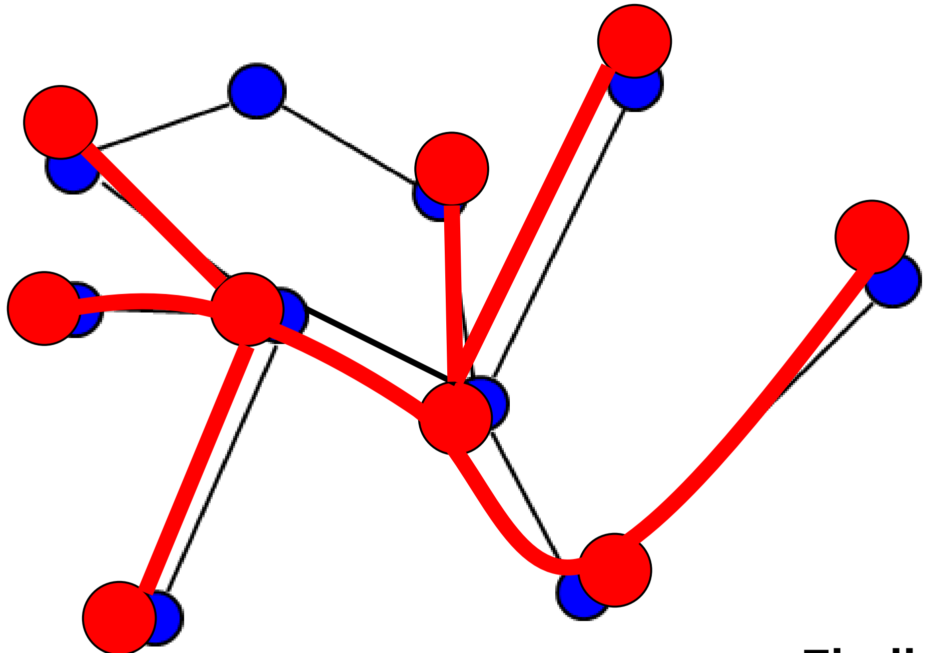
$S := \emptyset$
if `request`(GvC, H) **then**
    find max j s.t. $GvC^j \mapsto H$ (binary search)
    $S := C^j$
return S

Greedy Graph Growing on General Graphs? (1)



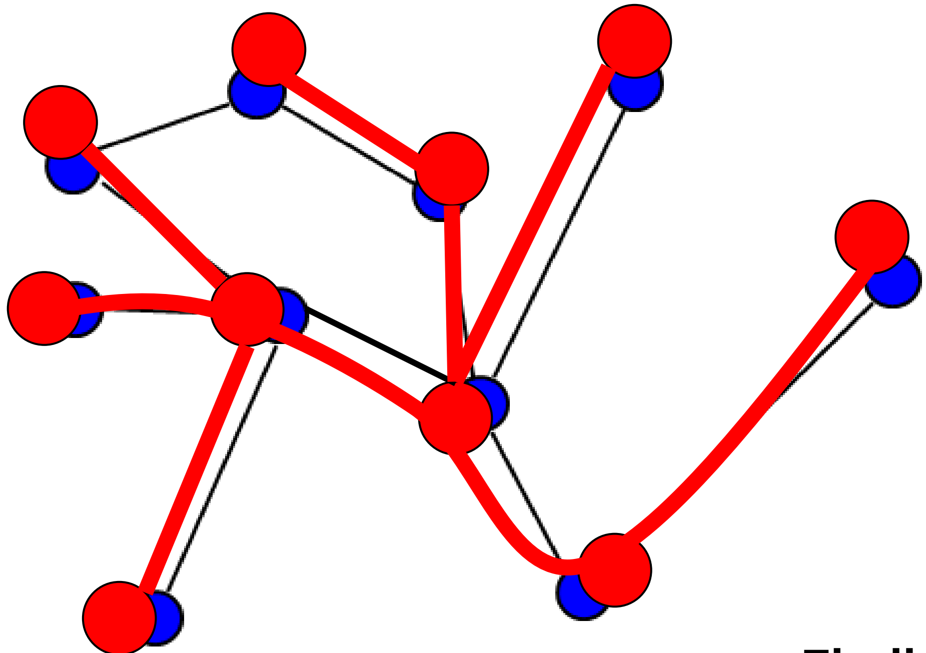
Finding path...

Greedy Graph Growing on General Graphs? (1)



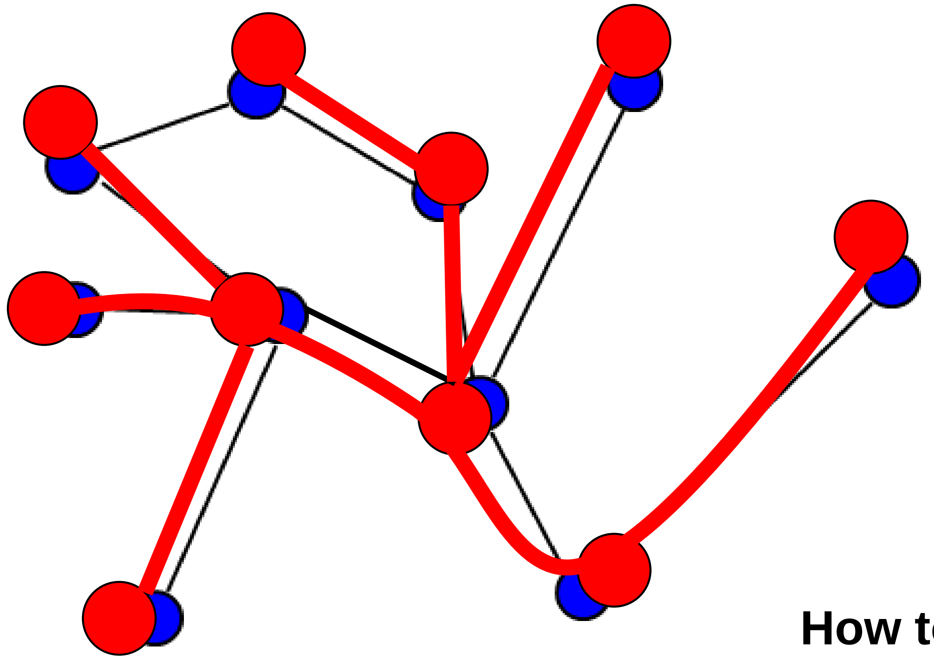
Finding neighbors...

Greedy Graph Growing on General Graphs? (1)



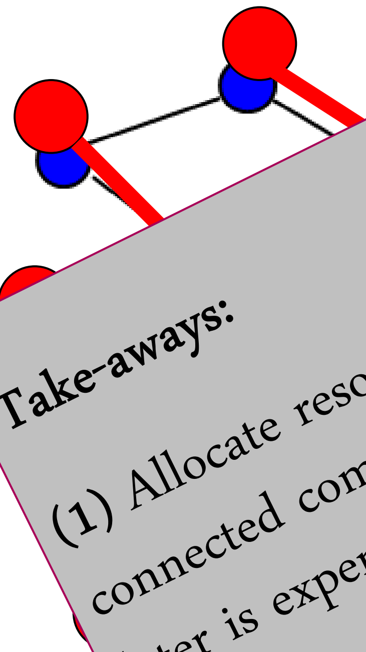
Finding more neighbors...

Greedy Graph Growing on General Graphs? (1)



How to close the gap?
Adding connections between existing CloudNet nodes is expensive: try all pairs!

Greedy Graph Growing on General Graphs? (1)



Take-aways:

(1) Allocate resources on all links of highly connected components first: finding these links later is expensive.

(2) In particular, if graph X can be embedded on Y, try to embed Y first!

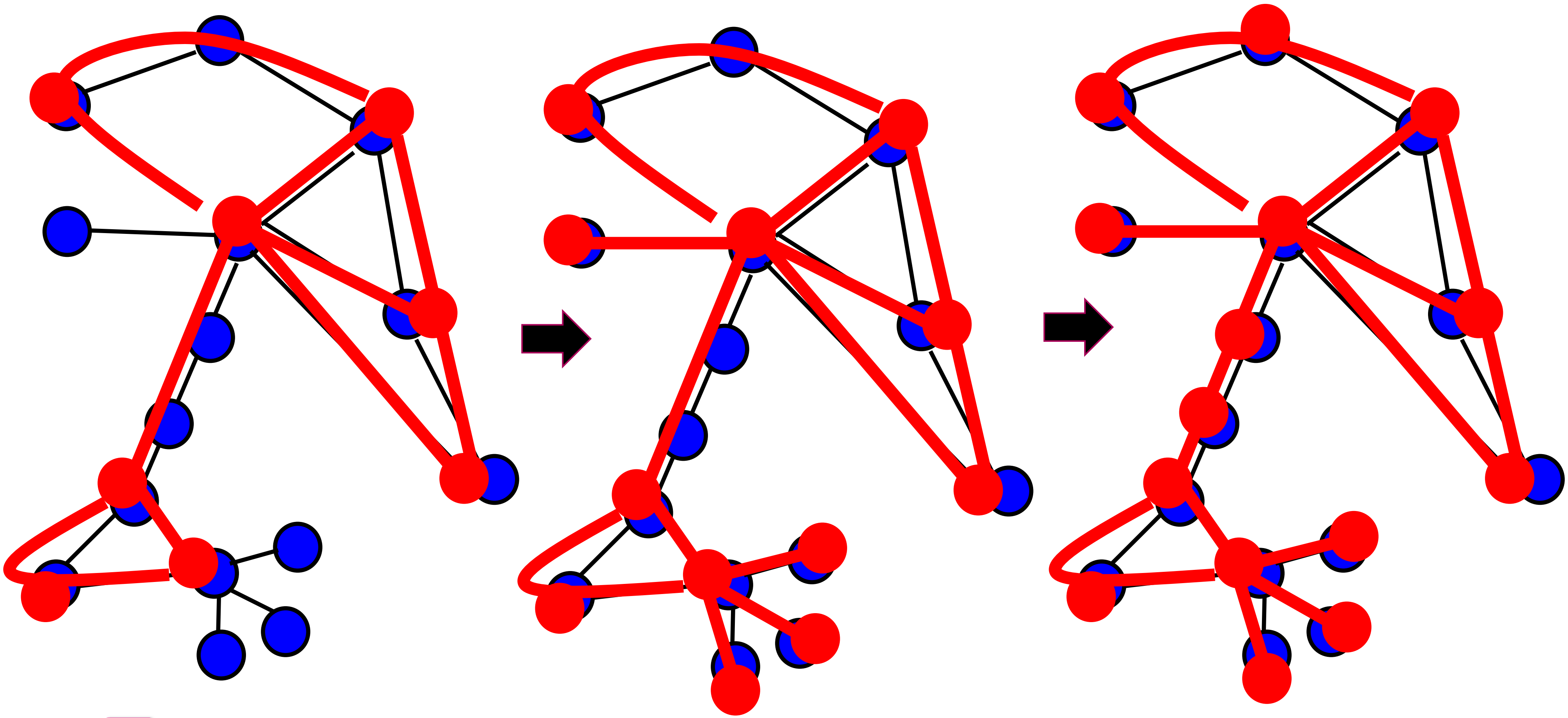
...ose the gap?
...ng connections between existing CloudNet nodes is expensive: try all pairs!

Greedy Graph Growing on General Graphs? (2)

Simple solution: First try to find the «knitting»!

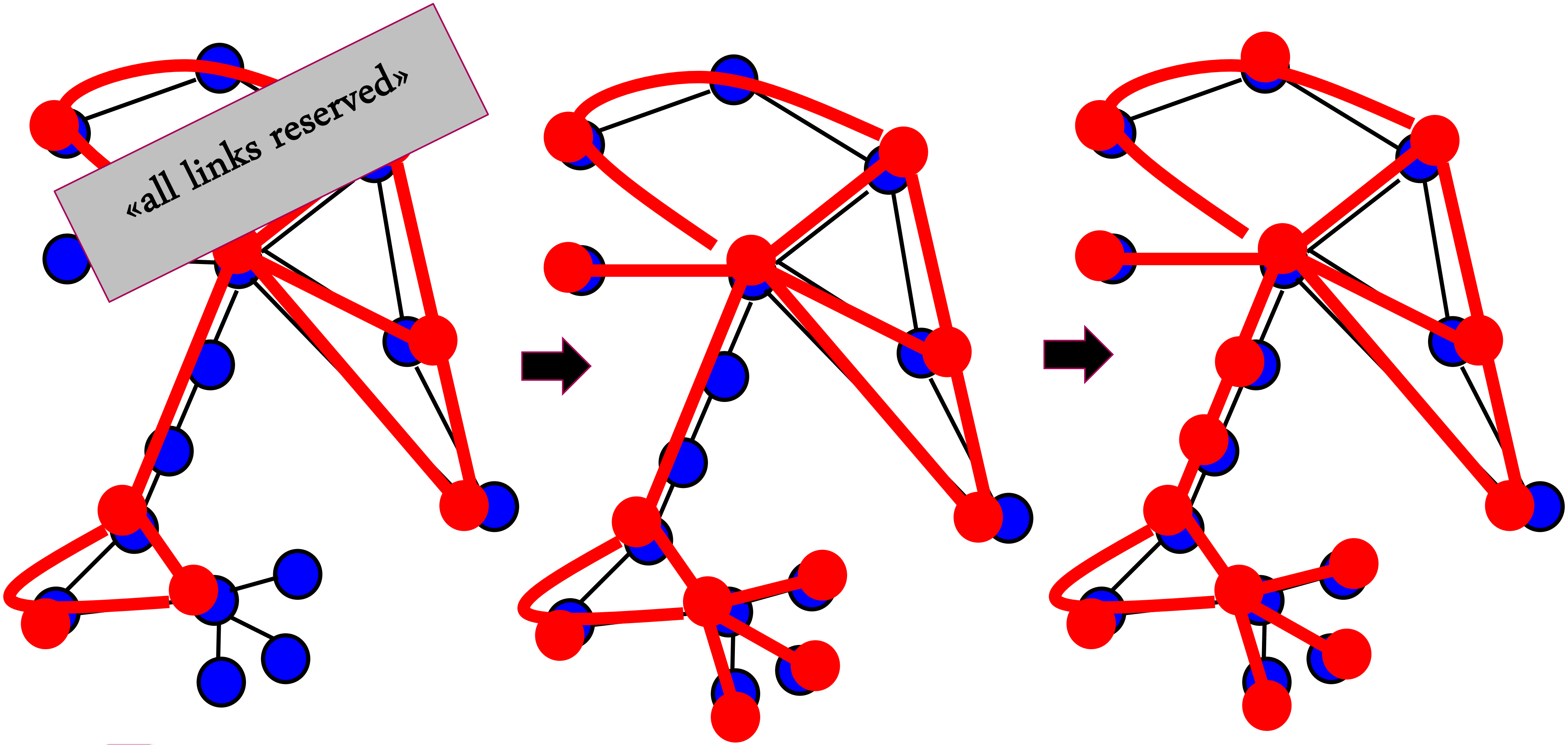
- The «two-or-more» connected components

- Later «expand nodes» and «expand edges»



Greedy Graph Growing on General Graphs? (2)

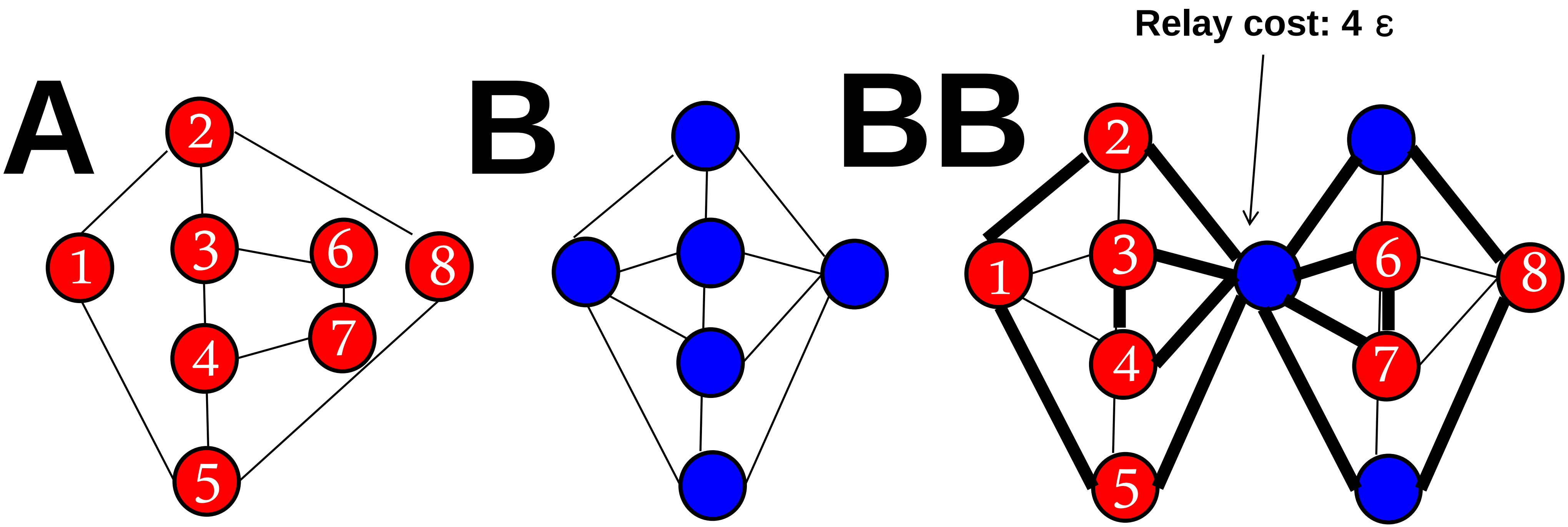
Simple solution: First try to find the «knitting»!
- The «two-or-more» connected components
- Later «expand nodes» and «expand edges»



Greedy Graph Growing on General Graphs? (3)

Idea: Ask graph «motif» only if it's guaranteed that it cannot be embedded over a more highly connected subgraph! (And connectivity has to be added later.)

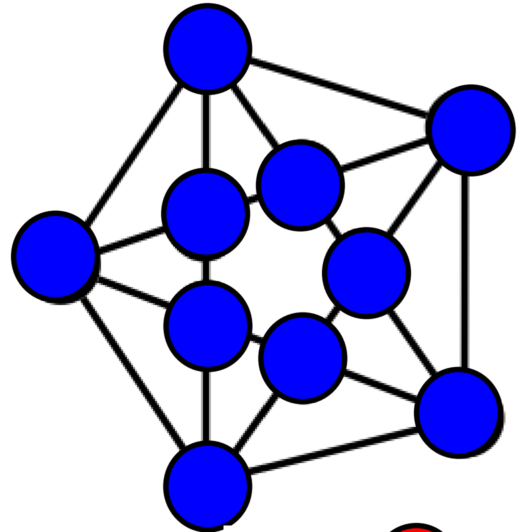
Careful: What goes first also depends on entire motif sequences!
- **A cannot be embedded into B and B cannot be embedded into A**
- **But A can be embedded into BB!**

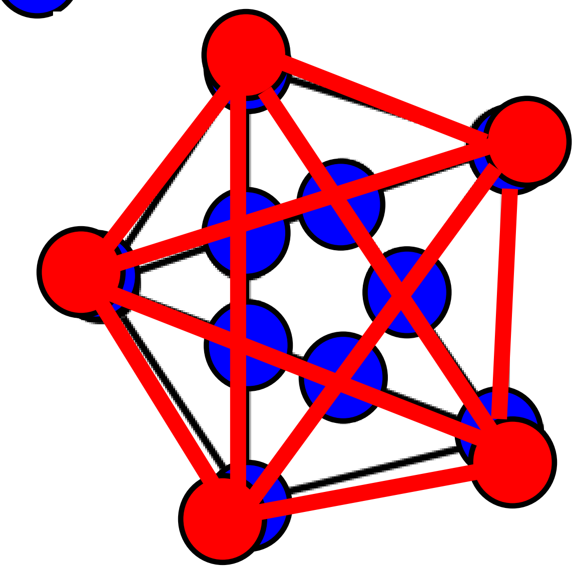


Remark.

Minor vs embedding:

Even with unit link capacity, for small epsilon, graph A may be embeddable (→) into graph B although A is not a minor of B!





Graph Minor

Graph A is a minor of B if A can be obtained from B by (1) deleting nodes, (2) deleting edges, or (3) contracting two nodes along edges.

Planar graph (and hence K5-minor free): But K5 can be embedded here!

Dictionary Attack: Expansion Framework.

Motif

Basic “knittings” of the graph.

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.)

Poset

Poset = partially ordered set
(1) Reflexive: $G \rightarrow G$
(2) Transitive: $G \rightarrow G'$ and $G' \rightarrow G''$, then $G \rightarrow G''$
(3) Antisymmetric: $G \rightarrow G'$ and $G' \rightarrow G$ implies $G=G'$ (isomorphic)

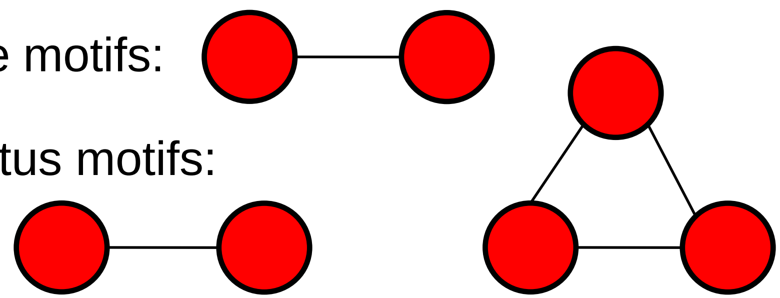
Framework

Explore branches according to dictionary order, exploiting poset property.

Examples

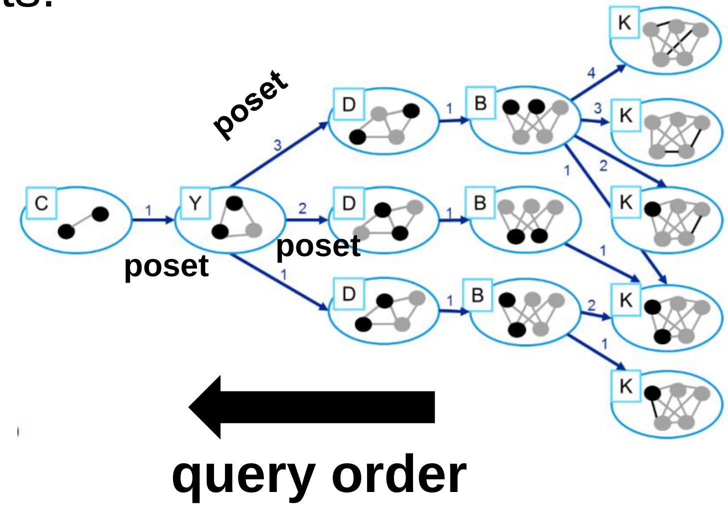
Tree motifs:

Cactus motifs:

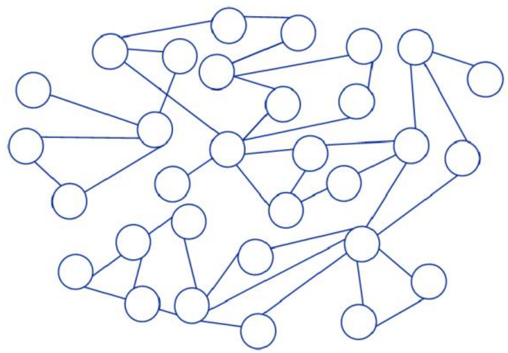


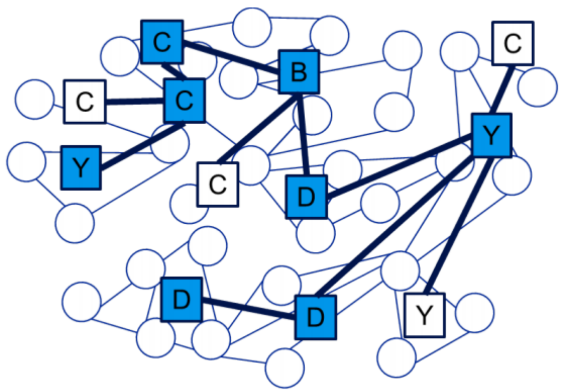
Dictionary Attack: Expansion Framework.

Dictionary dag (for chain C, cycle Y, diamond D, ...) with attachment points:



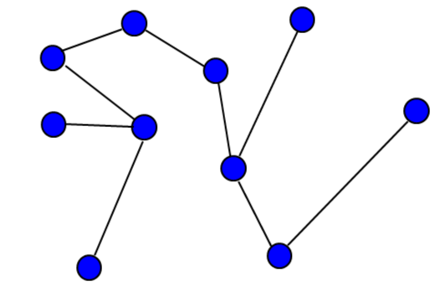
solves, e.g.,





Complexity:
Depends on dictionary depth and number of attachment points

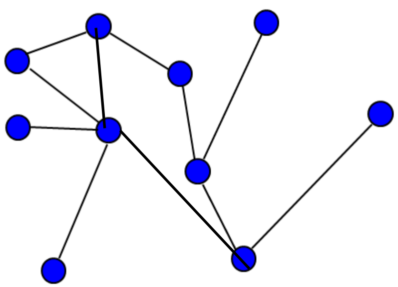
Overview of Results.



Tree

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

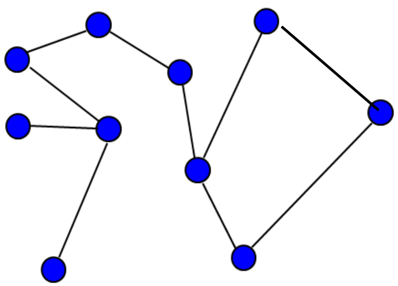
Lower bound: via number of possible trees and binary information.



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!)

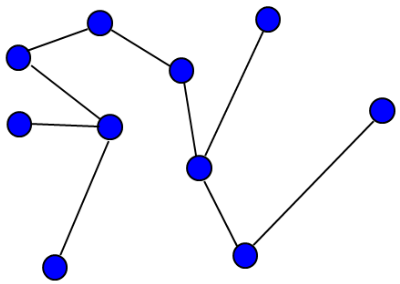


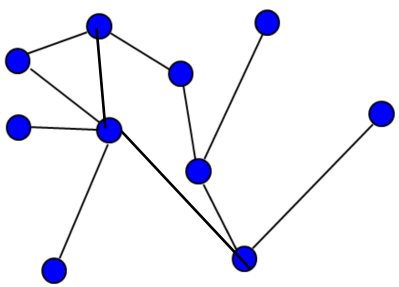
Cactus Graph

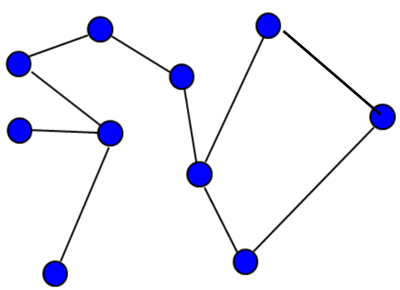
Can be explored in O(n) requests. This is optimal!

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

Overview of Results.







Algorithm 2 Cactus Discovery: CAC

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

exploreSequence(v)

$S := \emptyset$
if $GvYCY \mapsto H$ **then**
  find max j s.t. $GvY^jCY \mapsto H$
  $S := Y^jCY$, $P' := \{C\}$
  while $P' \neq \emptyset$ **do**
    for all $C_i \in P'$ **do**
      $A := prefix(C_i, S)$, $B := postfix(C_i, S)$;
      if $GvACYCB \mapsto H$ **then**
        find max j, k s.t. $GvAC(Y^jC)^kB \mapsto H$
        for $l := 1, \ldots, k$ **do**
          $P'' := P'' \cup \{C_l\}$
        $S := AC(Y^jC)^kB$
    $P' := P''$, $P'' := \emptyset$
if $\texttt{request}(GvSY, H)$ **then**
  find max j s.t. $GvSY^j \mapsto H$
  $S := SY^j$
if $\texttt{request}(GvSC, H)$ **then**
  $S := SC$
return S

edgeExpansion(e)

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

ower bound: via number of ossible trees and binary ıformation.

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

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

Dictionary Attacks: Expand Framework.

Motif

Basic “knitti

Poset

Partially orde
relation fulfills
symmetry, tra

Framew

Explore bra
to dictiona
exploiting

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

Algorithm 5 Motif Graph Discovery DICT

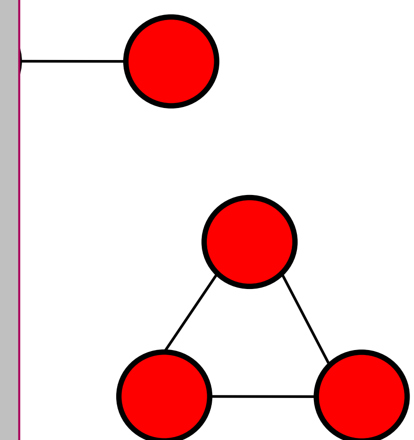
$H' := \{\{v\}, \emptyset\}$ /* current request graph */
$\mathcal{P} := \{v\}$ /* pending set of unexplored nodes*/
while $\mathcal{P} \neq \emptyset$ **do**
  choose $v \in \mathcal{P}$, $T := \mathit{find\_motif\_sequence}(v, \emptyset, \emptyset)$
  if $T \neq \emptyset$ **then**
    $H' := H'vT$,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

$\mathit{find\_motif\_sequence}(v, T^{<}, T^{>})$

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

$\mathit{edge\_expansion}(e)$

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



Application: BigFoot EU Project.

- "BigData is the answer: what was the question?"
  - How to interact with fast growing data volumes?
  - Which technology to obtain insights?

- BigFoot in three points:
  - **Automatic and self-tuned deployments** through virtualization
  - Cross-layer **optimization**
  - Data **interaction made easy**

- Applications:
  - **SMART-GRID DATA**
    - Billing & revenue assurance
    - Customer segmentation for service personalization
    - Pattern analysis for infrastructure provisioning
  - **ICT SECURITY DATA**
    - Attack attribution
    - Multi-feature classification

http://www.bigfootproject.eu

EURECOM Sophia Antipolis

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

TU berlin Technische Universität Berlin

GRIDPOCKET PERSONAL SMARTGRID SOLUTIONS

Symantec.

Application: BigFoot EU Project.

- “BigData is the answer: what was the question?”
  - How to interact with fast growing data volumes?
  - Which technology to obtain **insights**?

- BigFoot in three points:
  - **Automatic and self-tuned deployments** through virtualization
  - Cross-layer **optimization**
  - Data **interaction made easy**

Our focus!

- Applications:
  - **SMART-GRID DATA**
    - Billing & revenue assurance
    - Customer segmentation for service personalization
    - Pattern analysis for infrastructure provisioning
  - **ICT SECURITY DATA**
    - Attack attribution
    - Multi-feature classification

http://www.bigfootproject.eu

EURECOM Sophia Antipolis

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

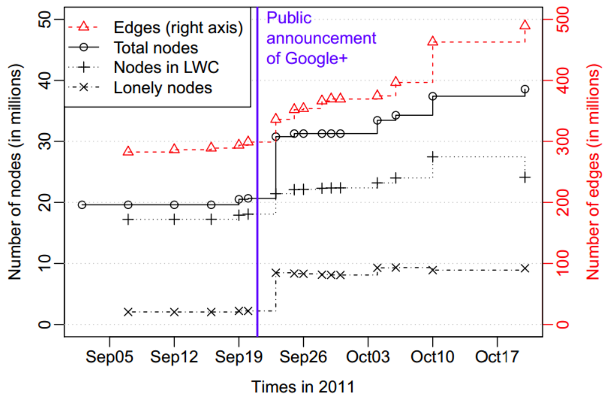
Technische Universität Berlin

GRIDPOCKET PERSONAL SMARTGRID SOLUTIONS

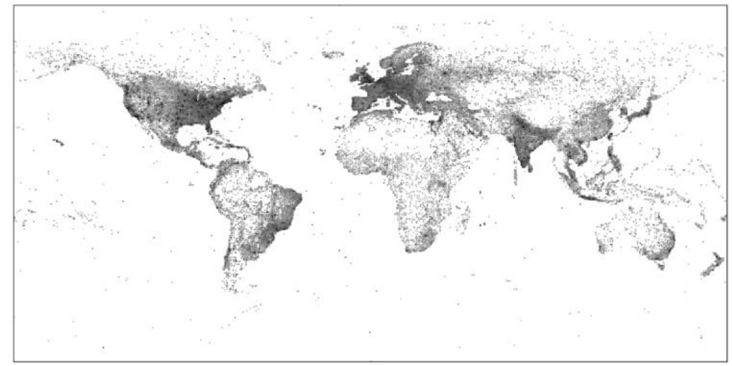
Symantec.

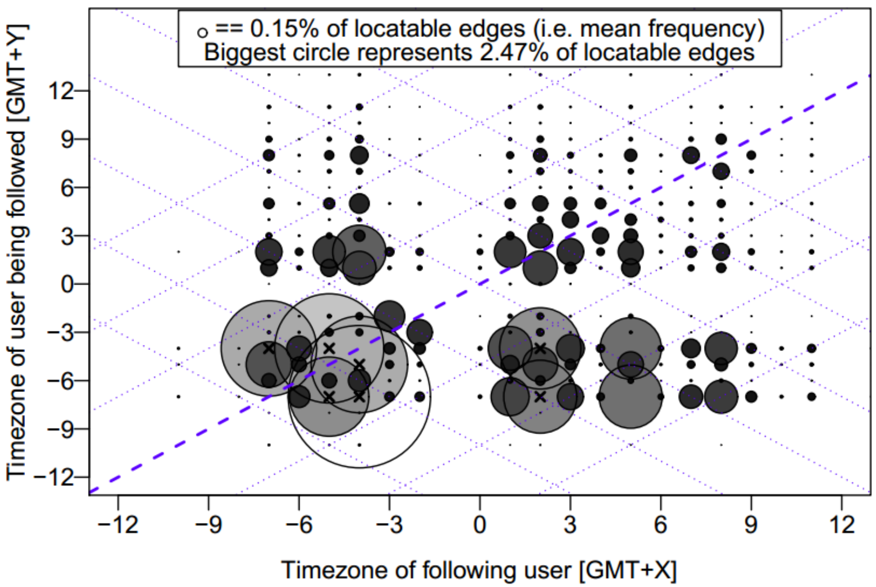
Big Data: OSN Analysis (Google+).

Schiöberg et al.: ACM WebSci 2012



Now 100M+ users...
... from all over the world!



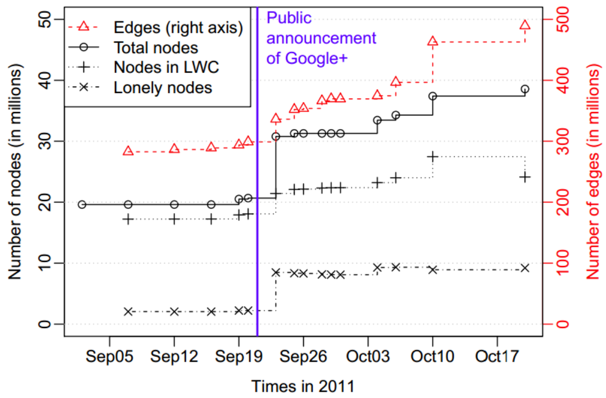


Research:

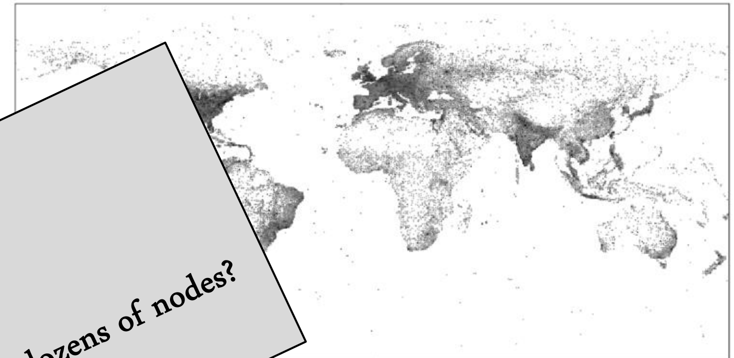
- # of «followers» (easy)
- Which time zones follow which time zones? (easy)
- K-cores, «stars», ... (easy)
- Diameter??
- Evolution of centralities??

Big Data: OSN Analysis (Google+).

Schiöberg et al.: ACM WebSci 2012



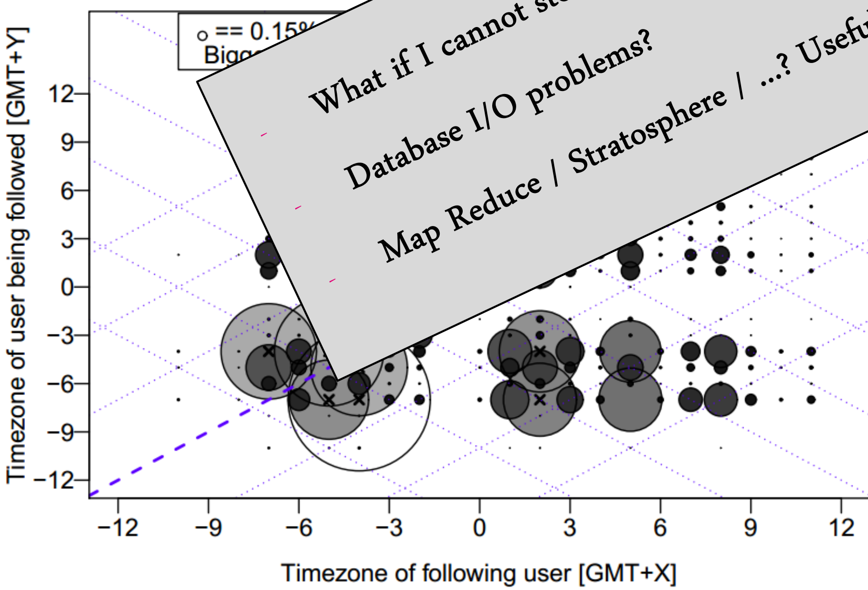
Now 100M+ users...
... from all over the ...



What if I cannot store full graph in memory

Database I/O problems?

Map Reduce / Stratosphere / ...? Useful with only dozens of nodes?

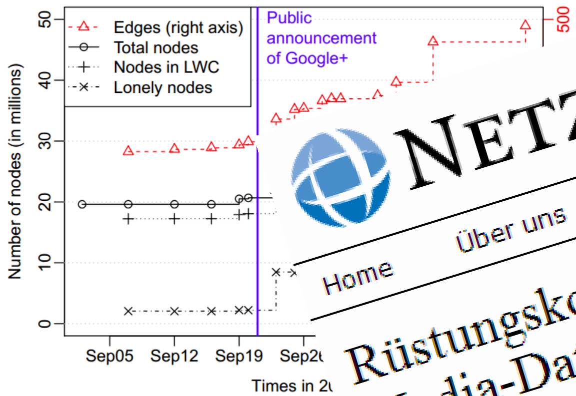


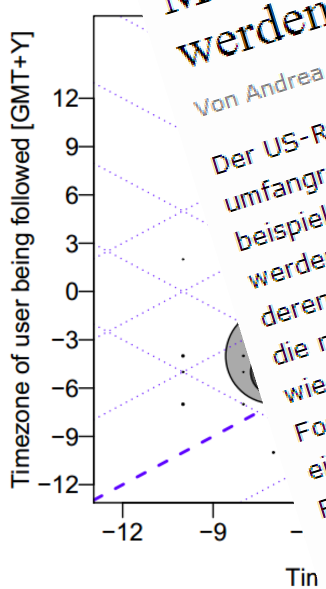
Research:

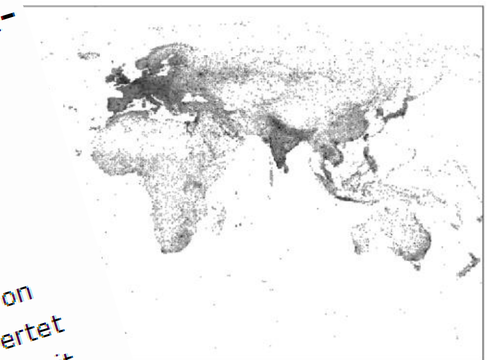
- # of «followers» (easy)
- Which time zones follow which time zones? (easy)
- K-cores, «stars», ... (easy)
- Diameter??
- Evolution of centralities??

Big Data: OSN Analysis (Googl

Schiöberg et al.: ACM WebSci 2012







NETZPOLITIK.ORG

Home | Über uns | Kontakt | Podcast | Netzpolitik TV | Facebook | Youtube

Rüstungskonzern entwickelt Software, mit der Social-Media-Daten zusammengeführt und durchsucht werden können, für die ‘nationale Sicherheit’

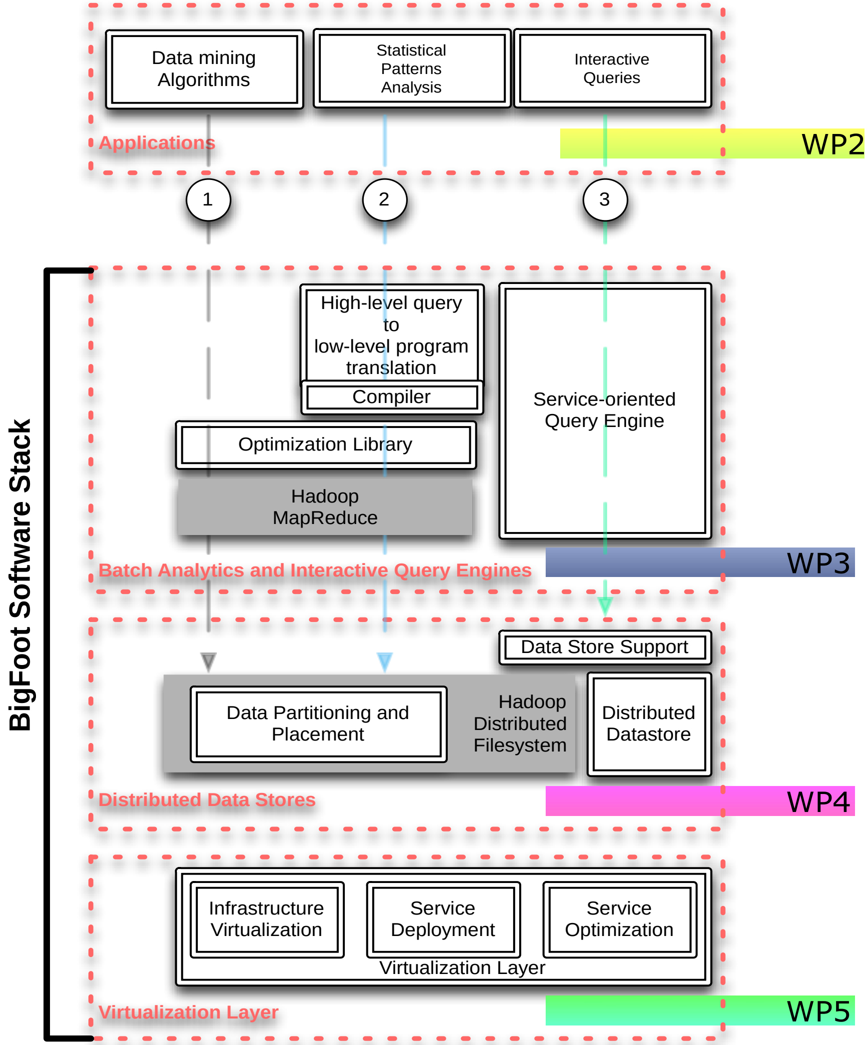
Von Andrea Jonjic | Veröffentlicht am: 11.02.2013 um 15:11h | 4 Responses

Der US-Rüstungs- und Elektronikkonzern Raytheon arbeitet laut Guardian an einer umfangreichen Software zur Durchleuchtung von Internetnutzern, indem Daten von beispielsweise Facebook, Twitter and Foursquare zusammengeführt und ausgewertet werden. RIOT – Rapid Information Overlay Technology – heißt die Suchmaschine, mit deren Hilfe aus Social-Media-Daten ermittelt werden soll, ob eine Person ein Risiko für die nationale Sicherheit darstellt. Der Guardian zeigt in einem Firmenvideo von Raytheon, wie der Konzern an einem Mitarbeiter die Fähigkeiten von RIOT demonstriert. Aus dessen Foursquare Check-ins wird eine Grafik der 10 Orte erstellt, an denen er am häufigsten eincheckt und schnell lässt sich erkennen, dass er meist montags um 6:00 Uhr im Fitnessstudio ist. Wer ihn oder seinen Laptop also mal erwischen wolle, könne dann ja dort vorbeischauen, so der ‘principal investigator’ von Raytheon, Brian Urch.

Es gibt ebenfalls eine "Personensuche", die Kontakte auf Twitter und anderen Plattformen in einer Netzwerkgrafik anzeigt und so Vernetzungen von Personen sichtbar macht. Bei all diesen Daten handelt es sich zwar um öffentlich einsehbare Informationen – problematisch ist allerdings die einfache Möglichkeit, verschiedene Quellen

y)
ow
sy)
asy)
...alities??

BigFoot Architecture.



Testbed:

- Hadoop (maybe Stratosphere)
- OpenStack resource management (+ own cloud operating system)
- OpenFlow (link virtualization)

Conclusion.

- CloudNets:
  - Elastic computing and networking
  - Federated architecture

- Competitive analysis: a framework to design and prove performance of online algorithms

- Good when:
  - No reliable prediction models exist
  - No data available
  - Worst case guarantees matter

- Examples: online embedding and service migration

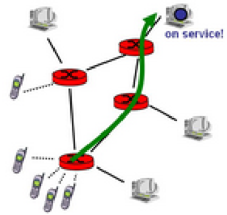
- Fully incorporated in prototype

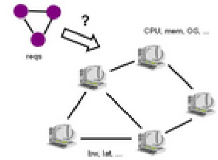
The Project Website.

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:** IPDPS 2014, OPODIS 2013, CLOUDNETS 2013 (*Best Paper*), 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