GIAN Course on Distributed Network Algorithms

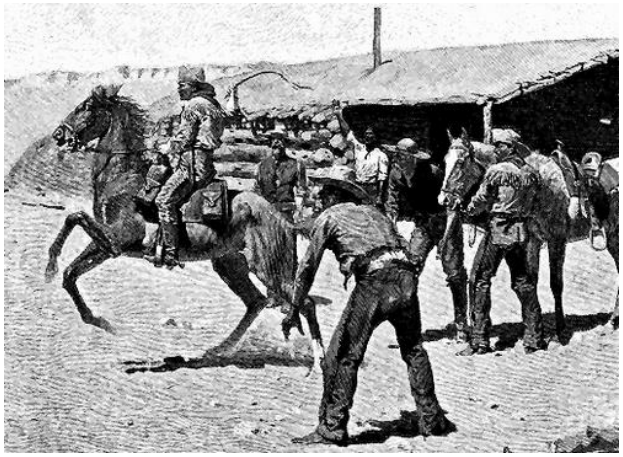
Network Topologies and Local Routing

Network Topologies and Local Routing

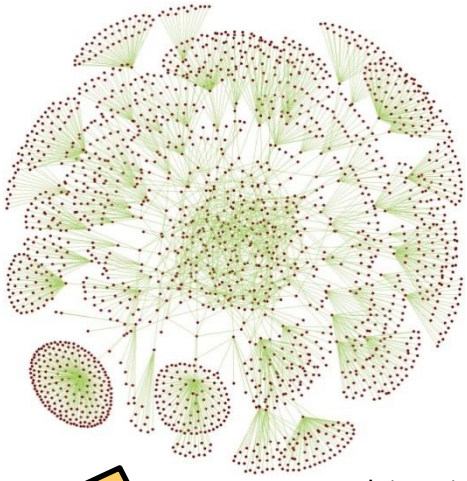
If you can, do exploit topology knowledge when designing LOCAL algorithms! E.g., distributed coloring algorithms for trees or planar graphs may perform better.

The Many Faces and Flavors of Network Topologies

Random??

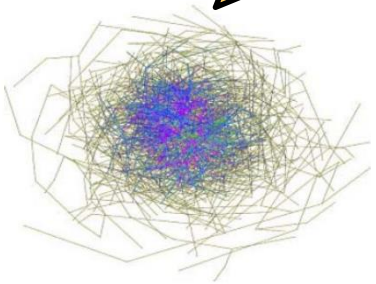


Social Networks.



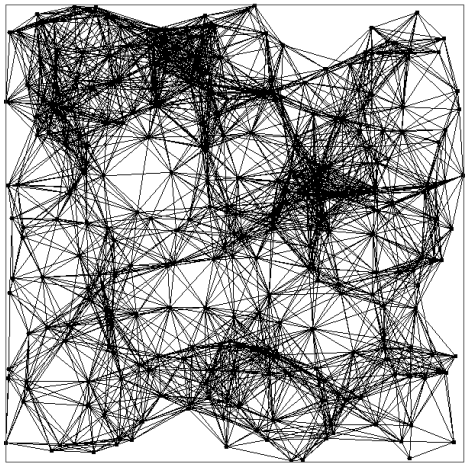
Internet.

Powerlaw?

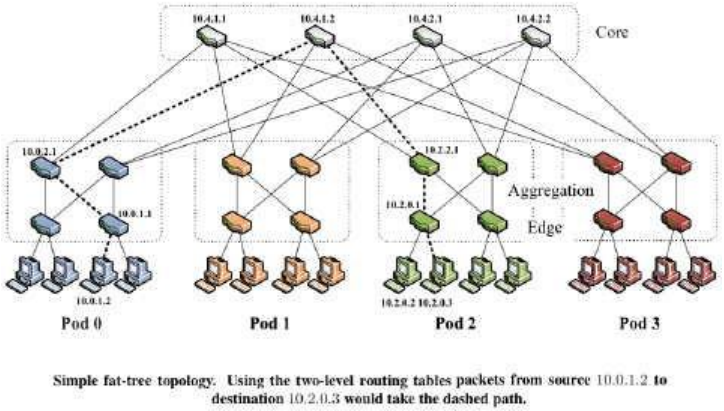


Gnutella P2P.

Smart grid.



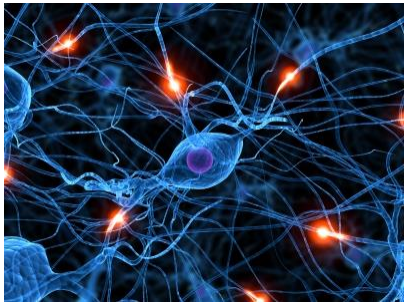
Wireless network before topology control



Simple fat-tree topology. Using the two-level routing tables packets from source 10.0.1.2 to destination 10.2.0.3 would take the dashed path.

Datacenter network

Regular?



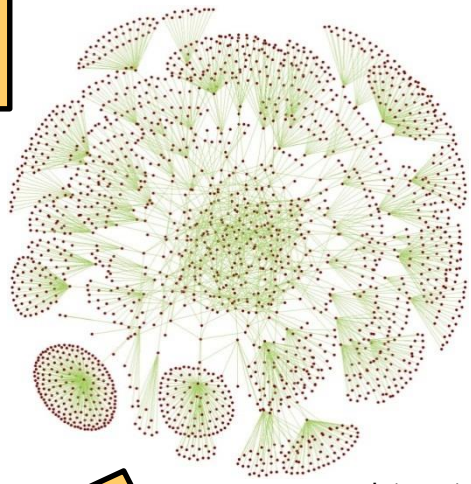
Nervous system.

Th…rs of Network Topologies

Topology also depends on the abstraction: router level topology vs Autonomous System topology?

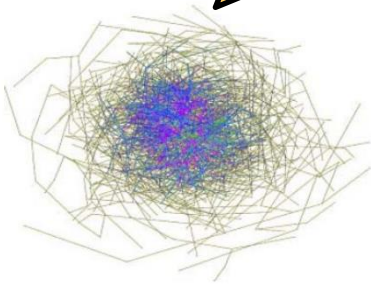
Random??

Social Networks.



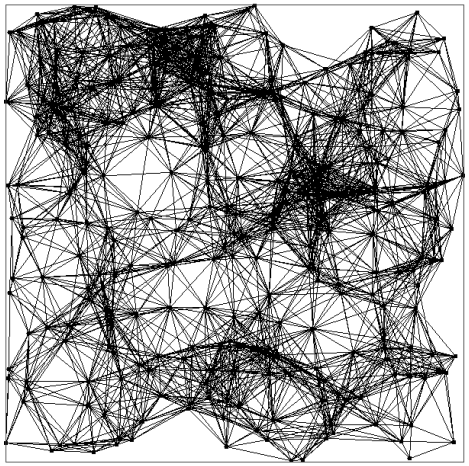
Internet.

Powerlaw?

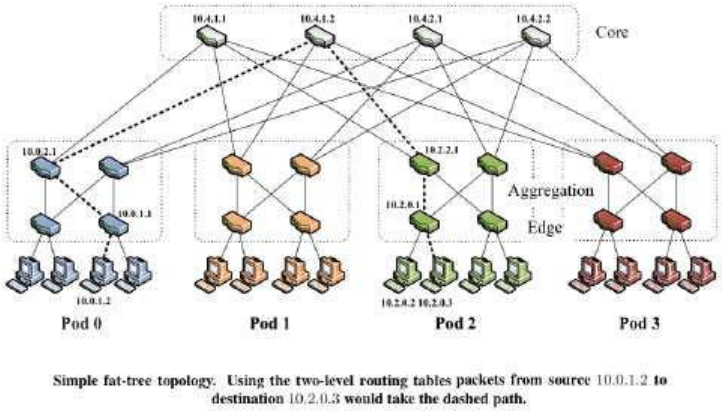


Gnutella P2P.

Smart grid.



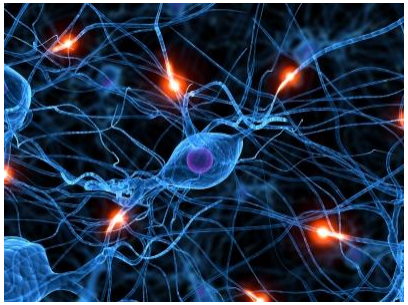
Wireless network before topology control



Simple fat-tree topology. Using the two-level routing tables packets from source 10.0.1.2 to destination 10.2.0.3 would take the dashed path.

Datacenter network

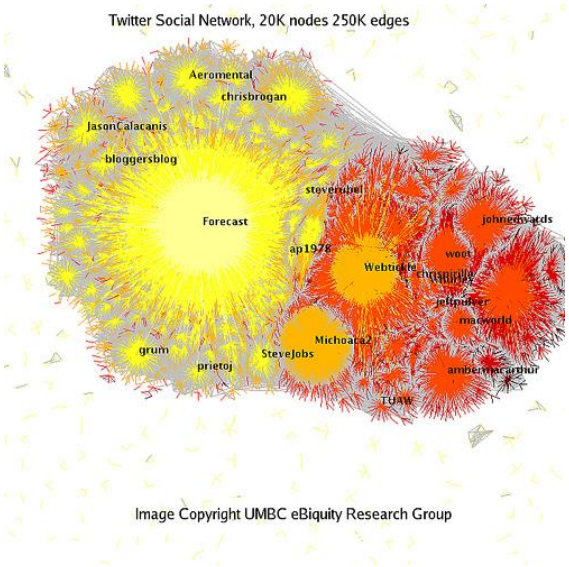
Regular?



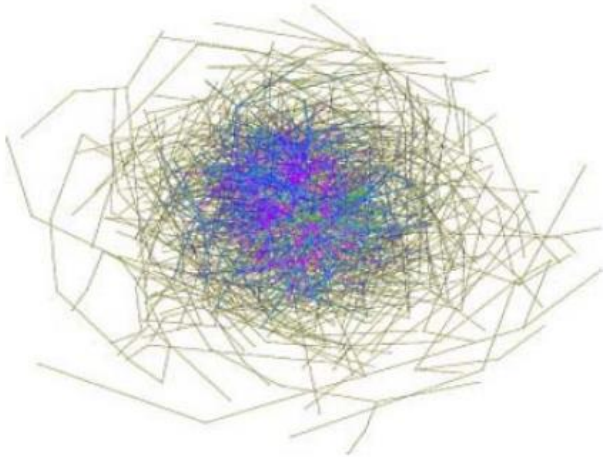
Nervous system.

The Many Faces and Flavors of Network Topologies

given
(social network)

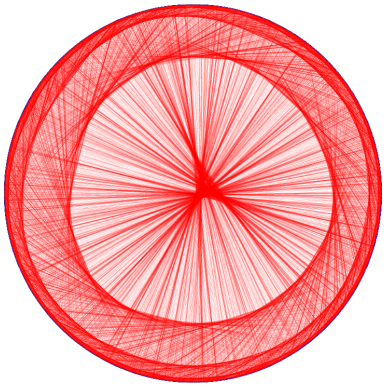


semi – or unstructured
(according to simple join protocol)



Gnutella 2001
(unstructured p2p system)

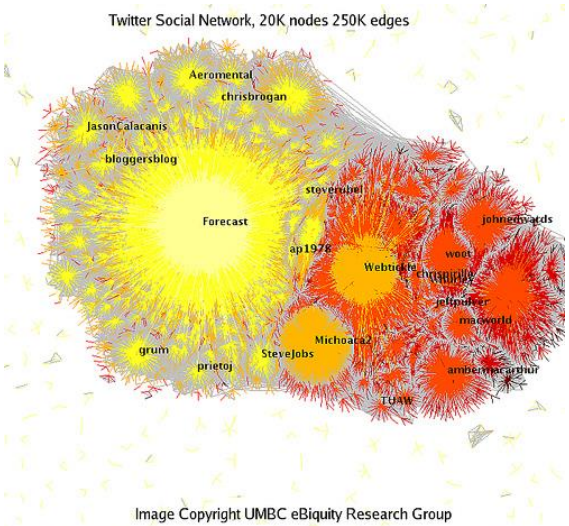
subject to optimization
(datacenter or structured p2p)



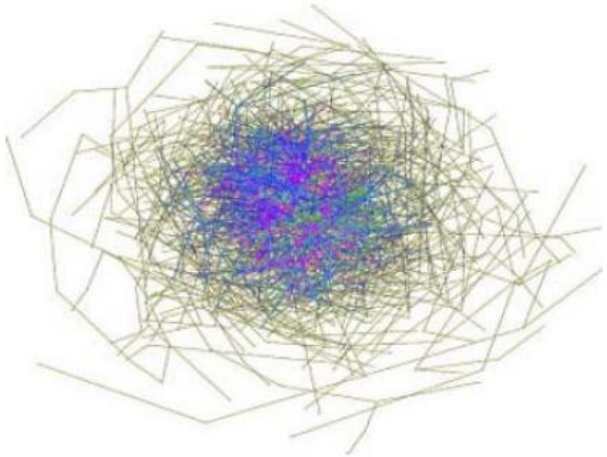
Chord DHT
(structured p2p system)

The Many Faces and Flavors of Network Topologies

given
(social network)

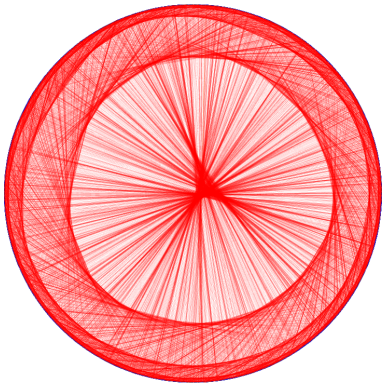


semi – or unstructured
(according to simple join protocol)



Gnutella 2001
(unstructured p2p system)

subject to optimization
(datacenter or structured p2p)



Chord DHT
(structured p2p system)

So what makes a good network topology?

Properties of a good network topology? It depends…!

E.g., support simple and efficient routing!

e.g., low diameter and short routes (wrt #hops, latency, energy, ...?), etc.

E.g., scalable!

e.g., small degree (number of neighbors to store and maintain?), little state needed at „routers“ (destination address defines next hop), high path diversity, bottleneck links, ...

E.g, robustness (random or worst-case failures?):

e.g., redundant paths, no single point of failure, ...

Etc.!

Properties of a good

Small diameter does not necessarily imply short routes, especially in distributed setting! Remember the maze… Need good algorithm to exploit the low diameter!

E.g., support simple and efficien[illegible]ting!

e.g., low diameter and short routes (wrt #hops, latency, energy, ...?), etc.

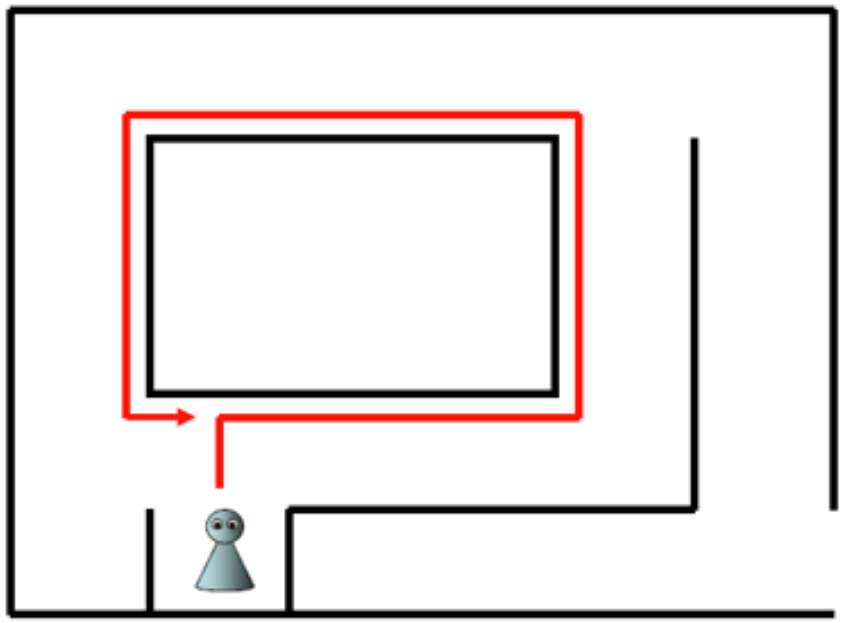
E.g., scalable!

e.g., small degree (number of n[illegible]te needed at „routers“ (destinat[illegible] diversity, bottleneck links, ...

E.g, robustness (random or w[illegible]

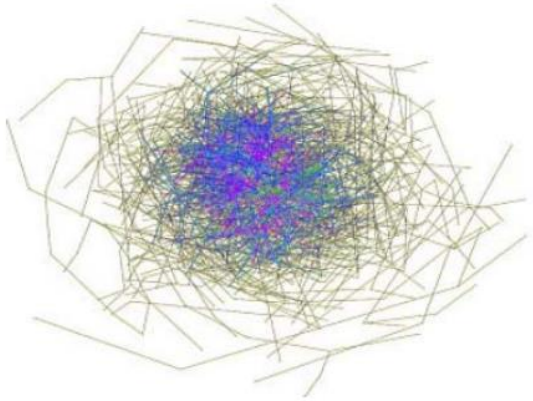
e.g., redundant paths, no single

Etc.!



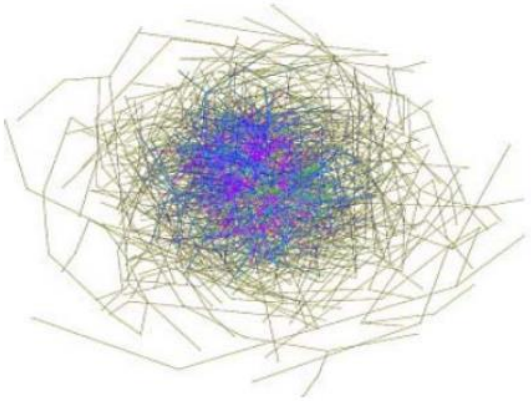
Argh.....

Example: Is the Gnutella P2P network robust?

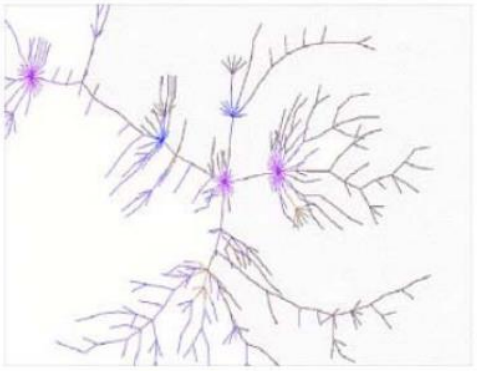


Example: Is the Gnutella P2P network robust?

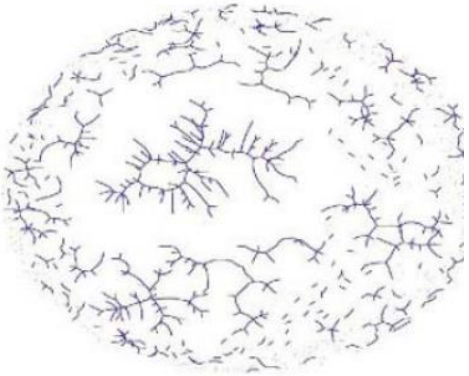
It depends…



all connections



30% random peers removed: still mostly connected („giant component“), robust to random failures / leaves



4% highest degree peers removed: many small disconnected components, not robust

Measurement study 2001 with ~2000 peers: [Saroiu et al. 2002]

Network Topologies = Graphs

Network topologies are often described as graphs!

Graph G=(V,E): V = set of nodes/peers/..., E= set of edges/links/...

?

d(.,.): distance between two nodes = shortest path, e.g. d(A,D)=?

D(G): diameter (D(G)=max length shortest path), e.g. D(G)=?

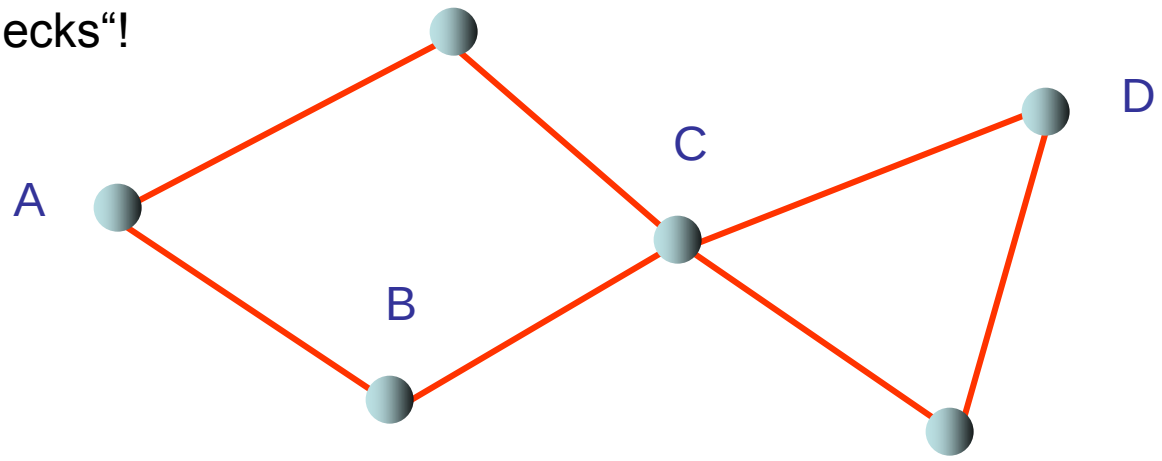
?

Γ(U): neighbor set of nodes U (not including nodes in U)

α**(U)** = $|\Gamma(U)| / |U|$ (size of neighbor set compared to size of U)

α**(G)** = $\min_{U,\ |U| < V/2} \alpha(U)$: expansion of G

Expansion captures „bottlenecks“!



Network Topologies = Graphs

Network topologies are often described as graphs!

Graph G=(V,E): V = set of nodes/peers/..., E= set of edges/links/...

3

d(.,.): distance between two nodes = shortest path, e.g. d(A,D)=?

D(G): diameter (D(G)=max length shortest path), e.g. D(G)=?

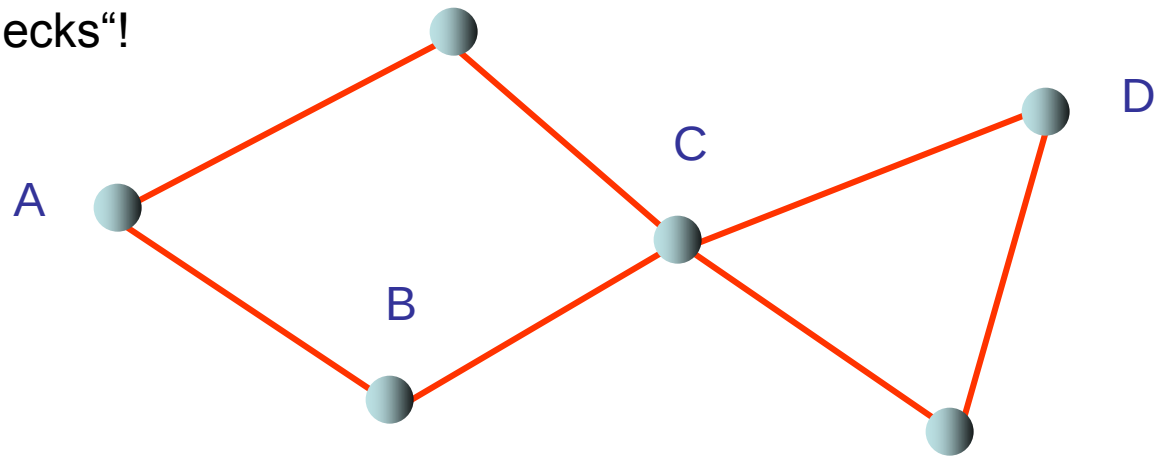
3

Γ(U): neighbor set of nodes U (not including nodes in U)

α**(U)** = $|\Gamma(U)| / |U|$ (size of neighbor set compared to size of U)

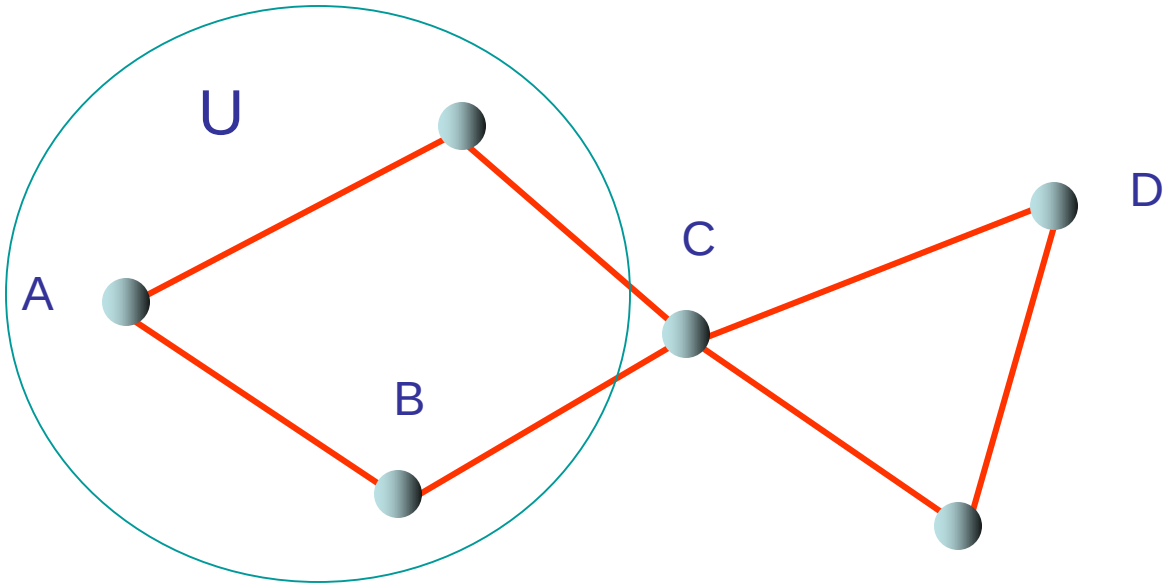
α**(G)** = $\min_{U,\ |U| < V/2}\ \alpha(U)$: expansion of G

Expansion captures „bottlenecks“!



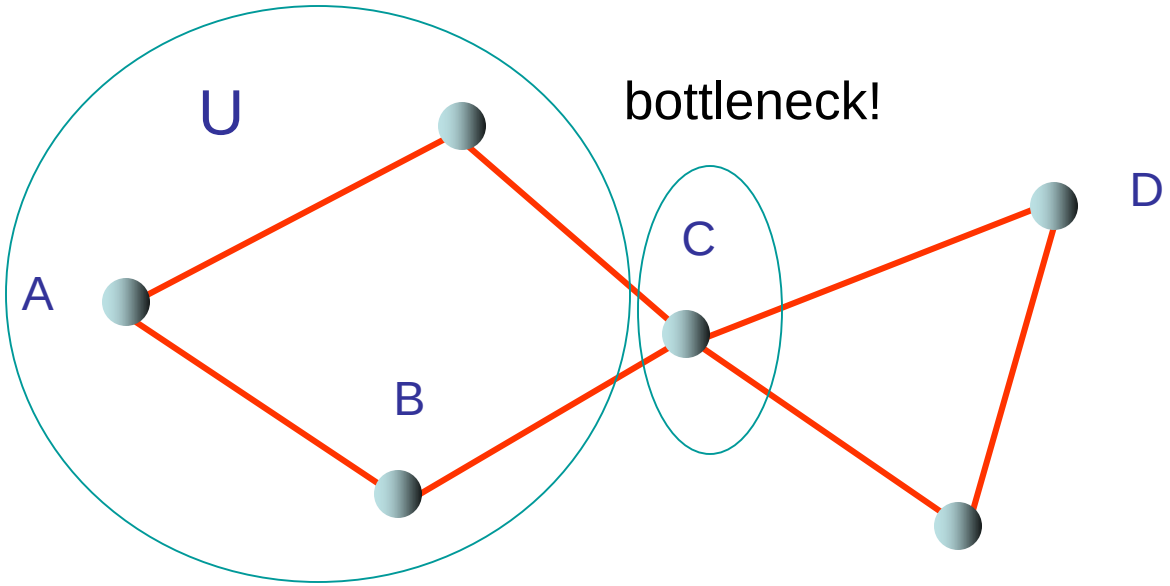
Example: Expansion

$\Gamma(U)$, $\alpha(U)$?



Example: Expansion

$\Gamma(U)$, $\alpha(U)$?

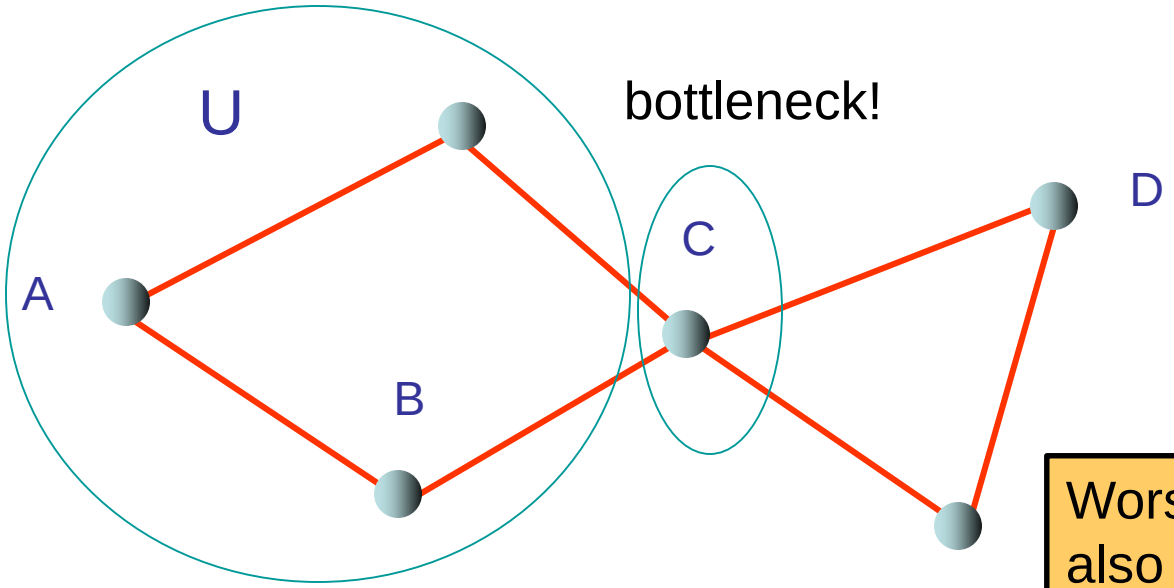


$\Gamma(U) = \{C\}$

Therefore: $\alpha(U)=1/3$

Example: Expansion

$\Gamma(U)$, $\alpha(U)$?



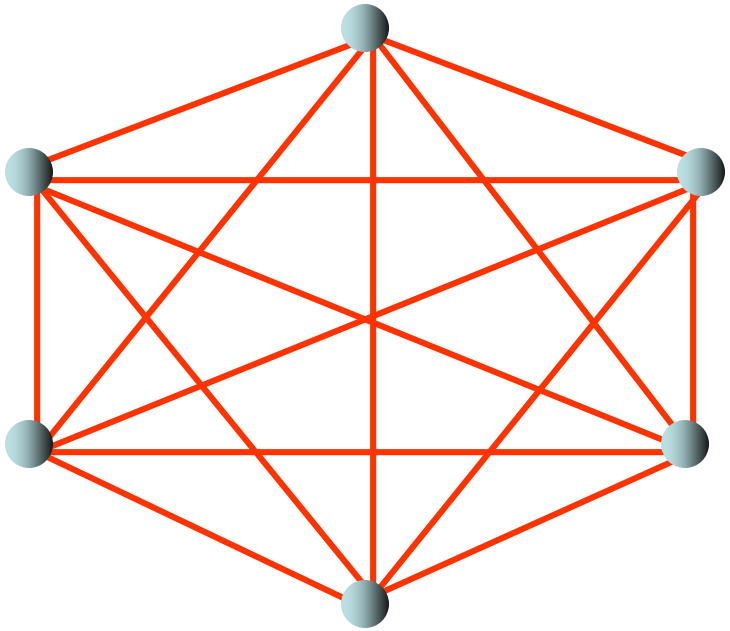
Worst possible: so also expansion of the whole graph!

$\Gamma(U) = \{C\}$

Therefore: $\alpha(U)=1/3$

Some Examples: The Clique

Complete network: pro and cons?



: robust, simple and fast routing, small diameter...

: does not scale! (degree?, number of edges?, ...)

Some Examples: The Line

Line network: pro and cons?

Degree? Diameter? Expansion?

 : simple and fast routing (tree = unique paths!), small degree (2)...

 : does not scale! (diameter = n-1)

Expansion?

Some Examples: The Line

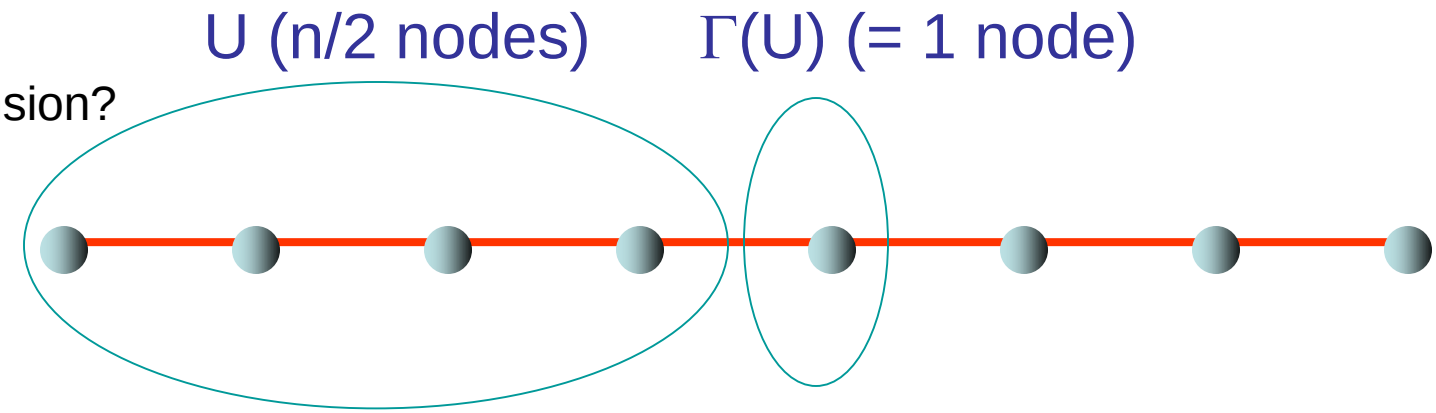
Line network: pro and cons?

Degree? Diameter? Expansion?

 : simple and fast routing (tree = unique paths!), small degree (2)...

 : does not scale! (diameter = n-1, expansion = 2/n, ...)

Expansion?



Some Examples: The Line

Line network: pro and cons?

Degree? Diameter? Expansion?

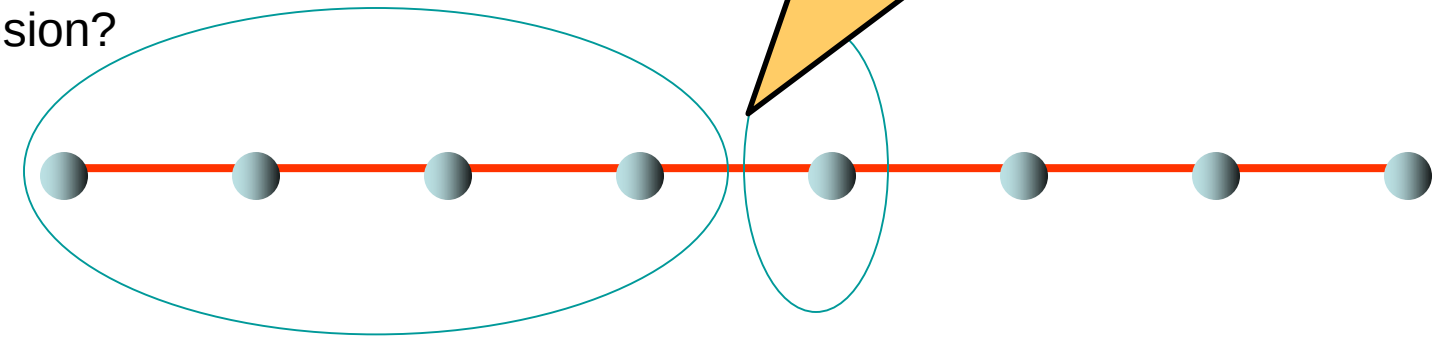
: simple and fast routing (tree = unique paths!), small degree (2)...

: does not scale! (diameter = n-1,

Can we reduce diameter without increasing degree by much?

U (n/2 nodes) Γ(U 1 node)

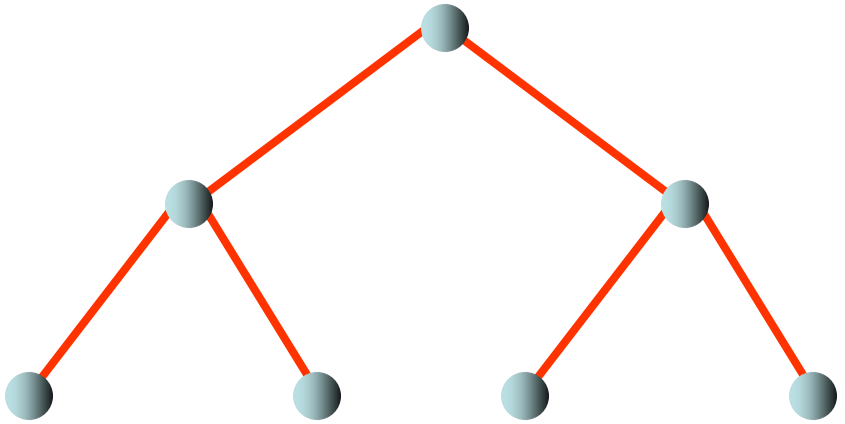
Expansion?



Some Examples: The Tree

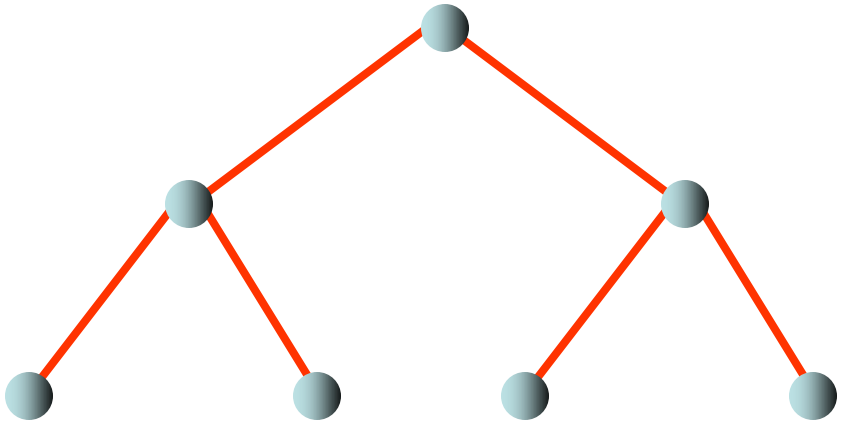
Binary tree network: pro and cons?

Degree? Diameter? Expansion?



Some Examples: The Tree

Binary tree network: pro and cons?



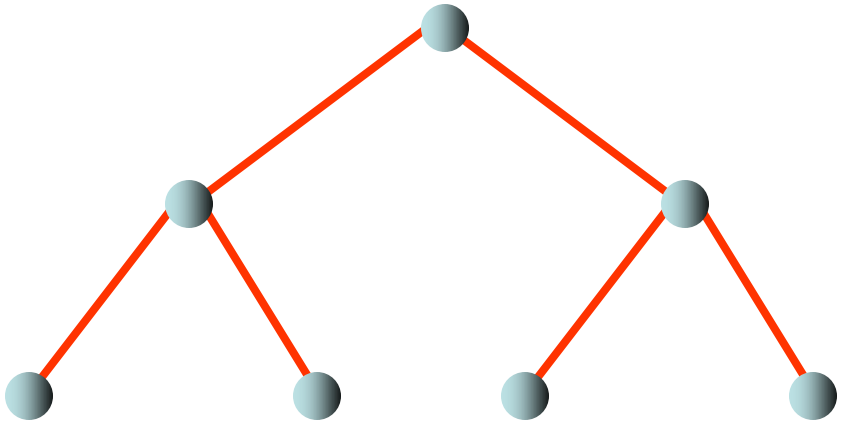
Degree? Diameter? Expansion?

: easy and fast routing (tree = unique paths!), small degree (3), log diameter...

: bad expansion?

Some Examples: The Tree

Binary tree network: pro and cons?

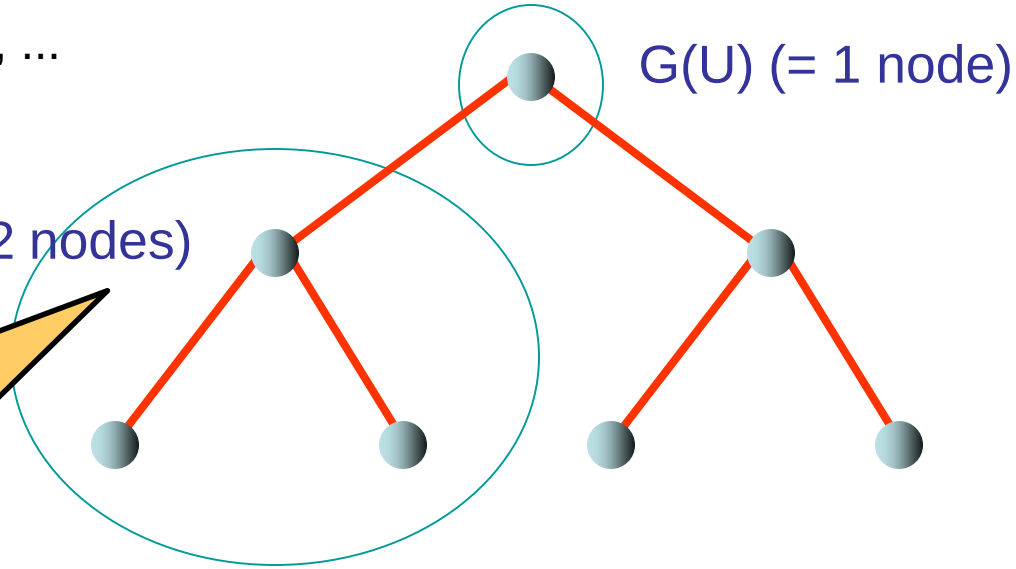


Degree? Diameter? Expansion?

: easy and fast routing (tree = unique paths!), small degree (3), log diameter...

: bad expansion = 2/n, ...

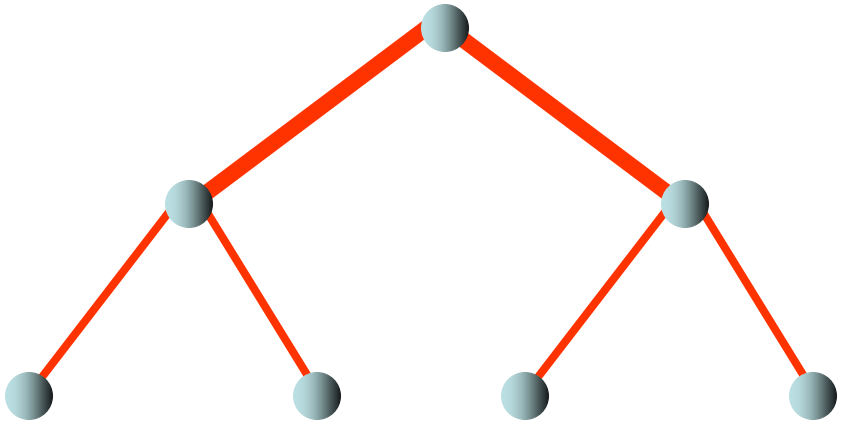
Expansion:



All communication from left to right tree goes through root! ☹ (no «bisection bandwidth»)

Remark: Datacenter Interconnects

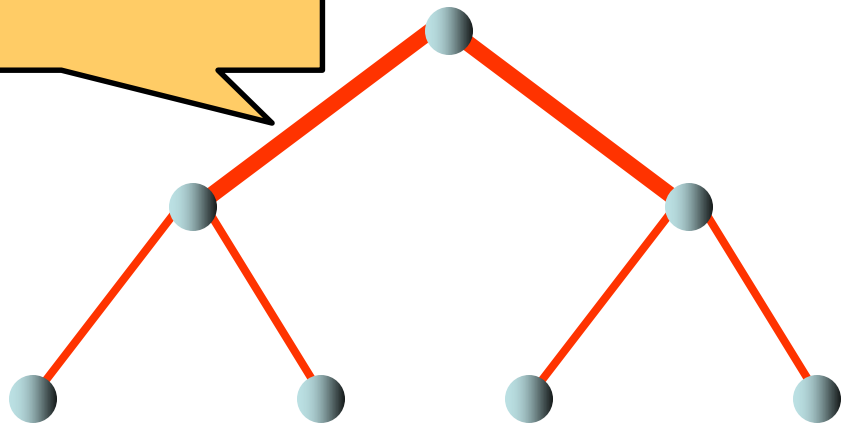
For this reason, datacenter inter connects form fat-trees: more bandwidth at higher layers which interconnect more nodes.



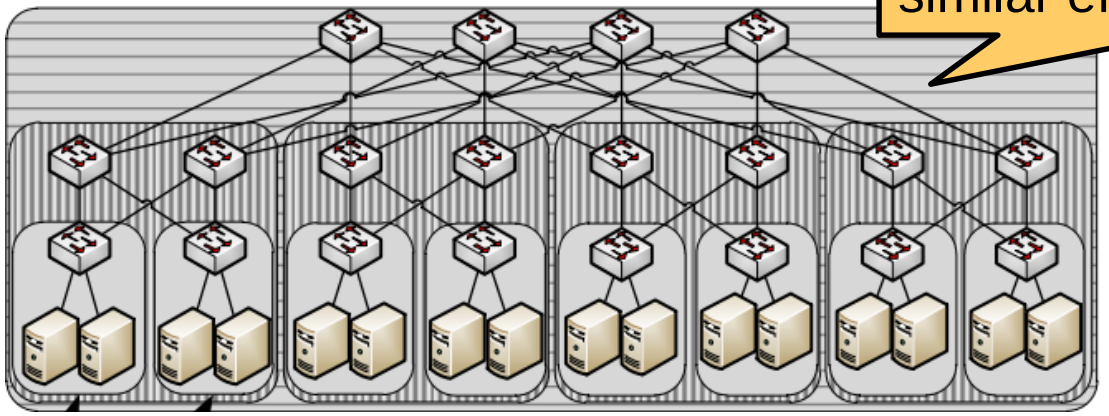
Remark: Datacen

More bandwidth for constant bisection bandwidth!

For this reason, datacenter inter connects form fat-trees: more bandwidth at higher layers which interconnect more nodes.



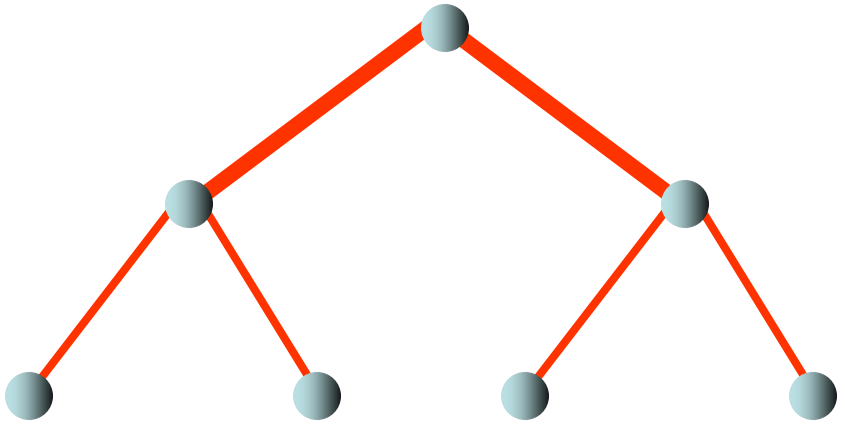
Note: more wires, not thicker wires! (Due to ECMP similar effect though?)



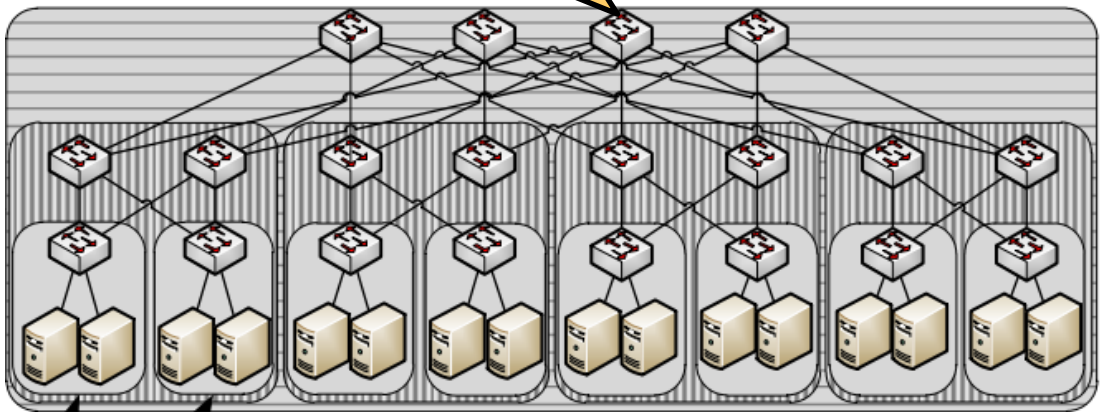
Example **Clos topology**: one of the most common datacenter inter-connects today.

Remark: Datacenter Interconnects

For this reason, datacenter inter
connects form fat-trees: more
which



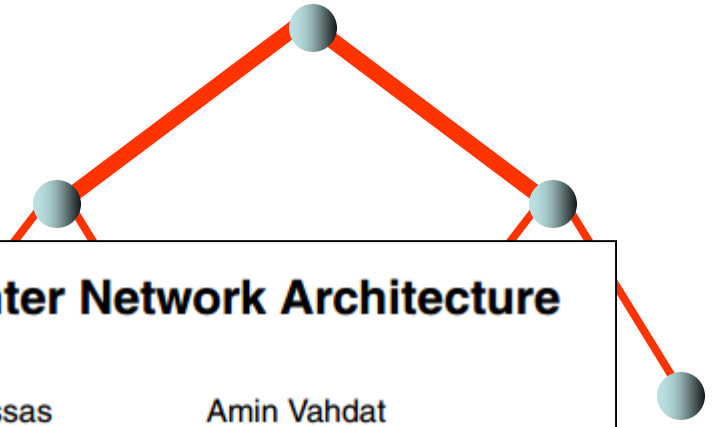
Servers organized into racks, racks organized into pods. The core routers usually connect to the Internet.



Example **Clos topology**:
one of the most common
datacenter inter-connects today.

Remark: Datacenter Interconnects

For this reason, datacenter inter
connects form fat-trees: more



Servers
racks, r
into poc
routers
to the Ir

A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares
malfares@cs.ucsd.edu

Alexander Loukissas
aloukiss@cs.ucsd.edu

Amin Vahdat
vahdat@cs.ucsd.edu

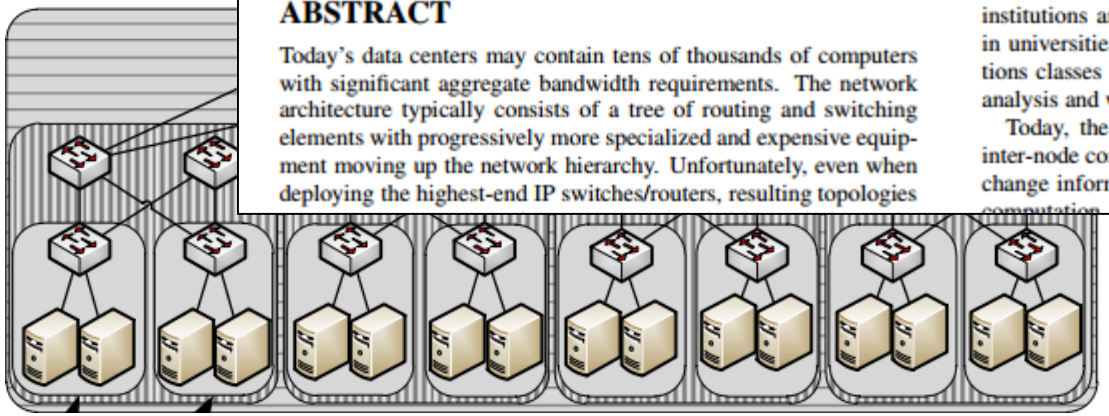
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404

ABSTRACT

Today's data centers may contain tens of thousands of computers with significant aggregate bandwidth requirements. The network architecture typically consists of a tree of routing and switching elements with progressively more specialized and expensive equipment moving up the network hierarchy. Unfortunately, even when deploying the highest-end IP switches/routers, resulting topologies

institutions and thousand-node clusters are increasingly common in universities, research labs, and companies. Important applications classes include scientific computing, financial analysis, data analysis and warehousing, and large-scale network services.

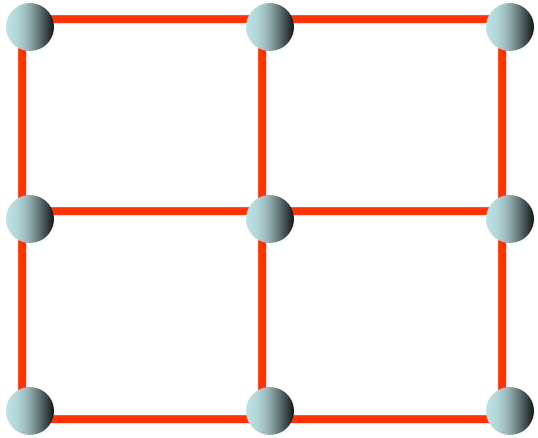
Today, the principle bottleneck in large-scale clusters is often inter-node communication bandwidth. Many applications must exchange information with remote nodes to proceed with their local computation. For example, MapReduce [12] must perform signif



y:
one of the most common
datacenter inter-connects today.

The Mesh

2d Mesh: pro and cons?



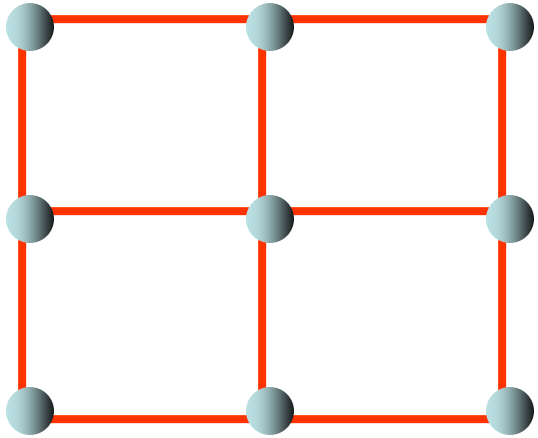
Degree? Diameter? Expansion?

 : easy and fast routing (**coordinates**!), small degree (4)…

 : diameter = ?, expansion = ?

The Mesh

2d Mesh: pro and cons?



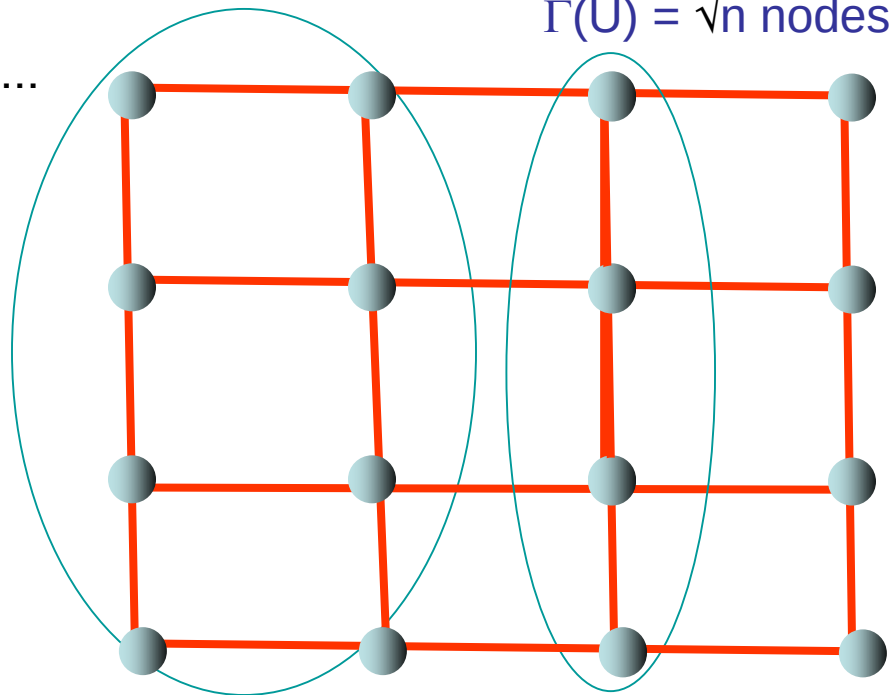
Degree? Diameter? Expansion?

: easy and fast routing (coordinates!), small degree (4)…

: diameter = √n, expansion = ~2/√n, ...

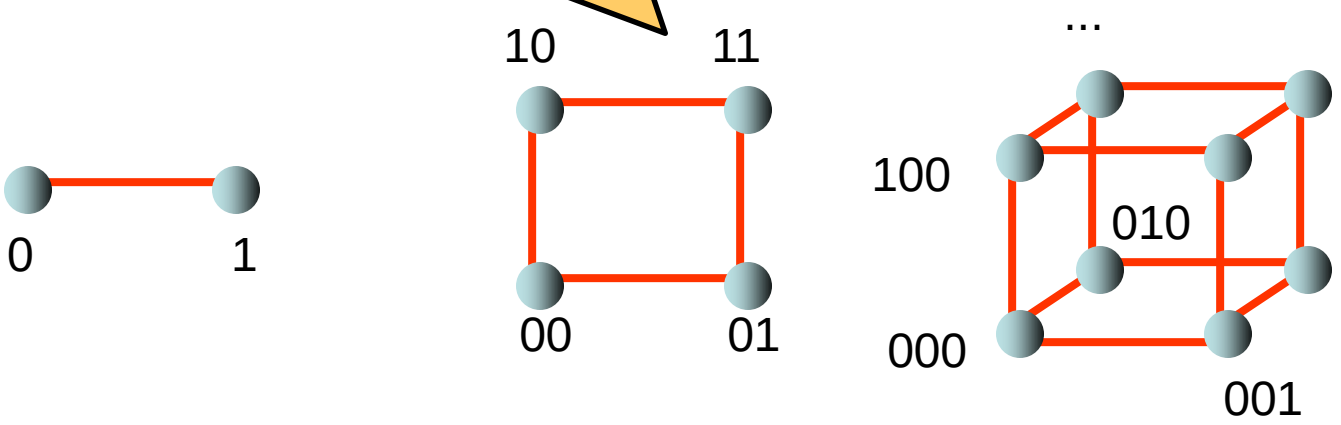
U ~ n/2 nodes

Γ(U) = √n nodes



The Hypercube

General approach to describe topologies and do routing: use identifier manipulation! Here: binary strings. Hypercube of d-dimensions: d bits.



Connected iff Hamming distance = 1: flip exactly one bit.

The Hypercube

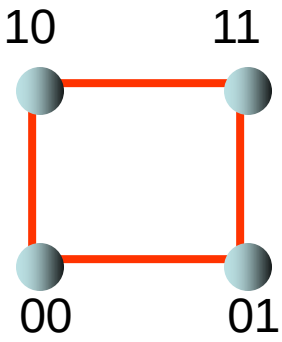
Formally:

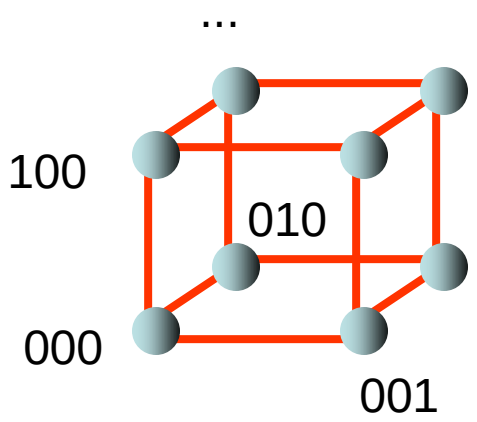
d-dim Hypercube:

Nodes V = {(b_1,...,b_d), b_i binary} (nodes are bitstrings!)

Edges E = for all i: (b_1,..., b_i, ..., b_d)

connected to (b_1, ..., 1-b_i, ..., b_d)





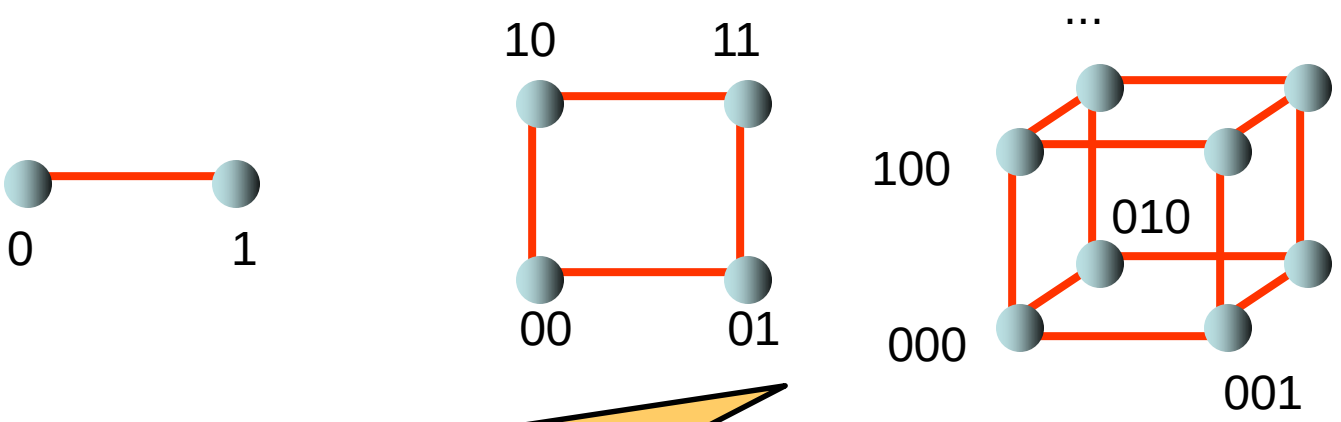
The Hypercube

d-dim Hypercube:

Nodes V = {(b_1,...,b_d), b_i binary} (nodes are bitstrings!)

Edges E = for all i: (b_1,..., b_i, ..., b_d)

connected to (b_1, ..., 1-b_i, ..., b_d)



Degree? Diameter? Expansion? How to get from (100101) to (011110)?

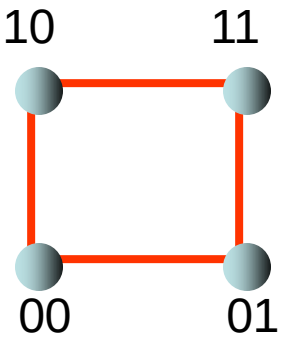
The Hypercube

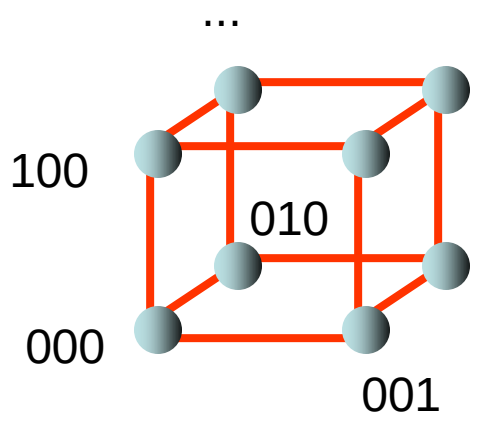
d-dim Hypercube:

Nodes V = {(b_1,...,b_d), b_i binary} (nodes are bitstrings!)

Edges E = for all i: (b_1,..., b_i, ..., b_d)

connected to (b_1, ..., 1-b_i, ..., b_d)





2^d = n nodes, hence d = log(n): degree

Diameter: fix one bit after another, so log(n) as well

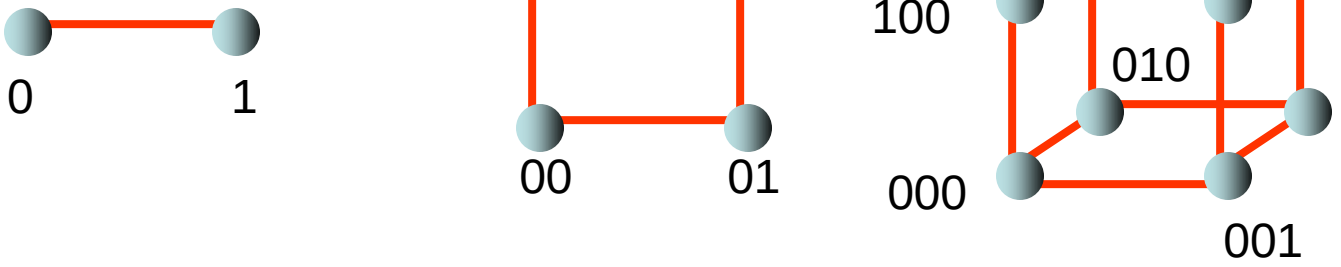
The Hypercube

d-dim Hypercube:

Nodes V = {(b_1,...,b_d), b_i binary} (nodes are bitstrings!)

Edges E = for all i: (b_1,..., b_i, ..., b_d)

connected to (b_1, ..., 1-b_i, ..., b_d)



Simply the maximal Hamming distance!

2^d = n nodes, hence d = log(n): degree

Diameter: fix one bit after another, so log(n) as well

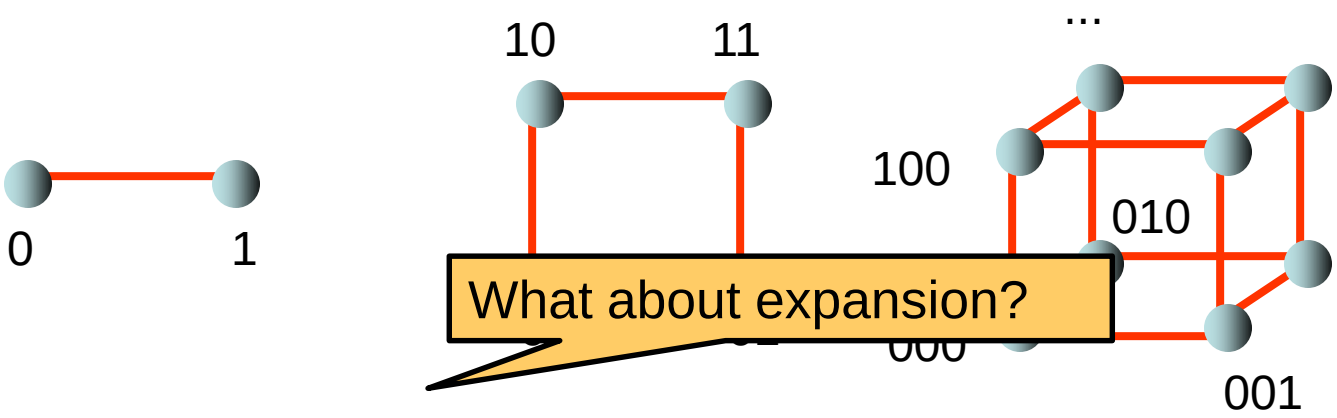
The Hypercube

d-dim Hypercube:

Nodes V = {(b_1,...,b_d), b_i binary} (nodes are bitstrings!)

Edges E = for all i: (b_1,..., b_i, ..., b_d)

connected to (b_1, ..., 1-b_i, ..., b_d)



2^d = n nodes, hence d = log(n): degree

Diameter: fix one bit after another, so log(n) as well

Expansion of Hypercube

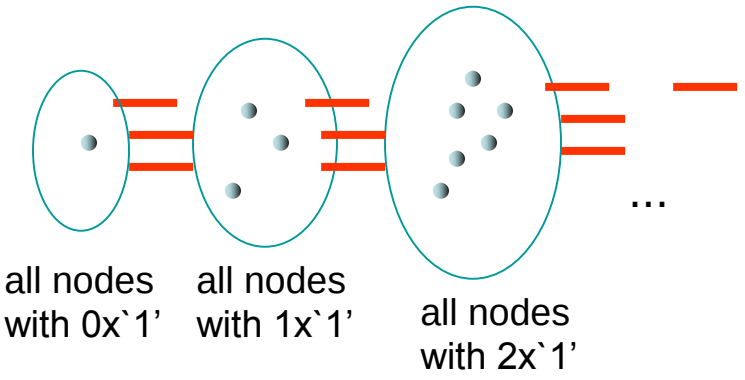
d-dim Hypercube:

Nodes V = {(b_d,...,b_1), b_i binary}

Edges E = for all i: (b_d,..., b_i, ..., b_1)

connected to (b_d, ..., 1-b_i, ..., b_1)

Idea: let's grow a ball around 00…00 node!



Note: nodes with i x`1' are connected nodes with (i-1) x`1' and (i+1) x`1'...:

Expansion of Hypercube

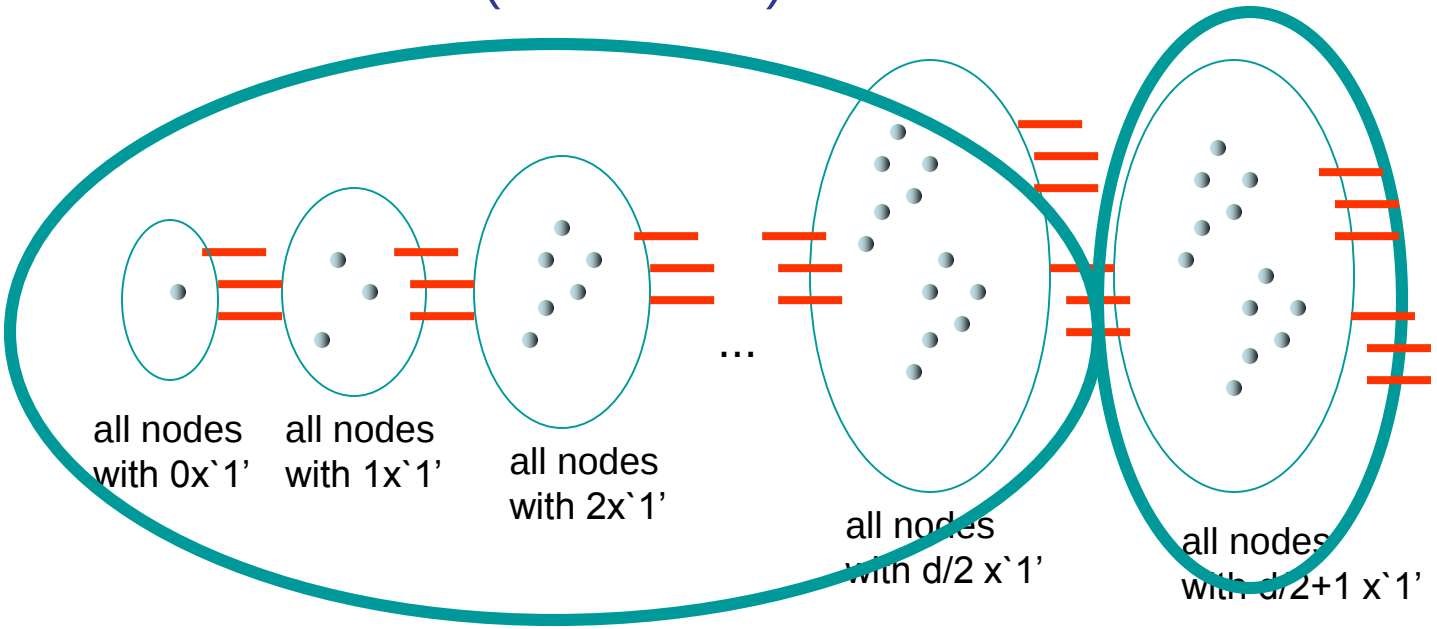
Size of neighborhood = number of possibilities to place d/2+1 `1‘s at d positions.

Idea:

Total number of nodes with between 0 and d/2 ,1-bits?

U (~n/2 nodes)

$\Gamma(U)$ = binomial(d,d/2+1)



Expansion of Hypercube

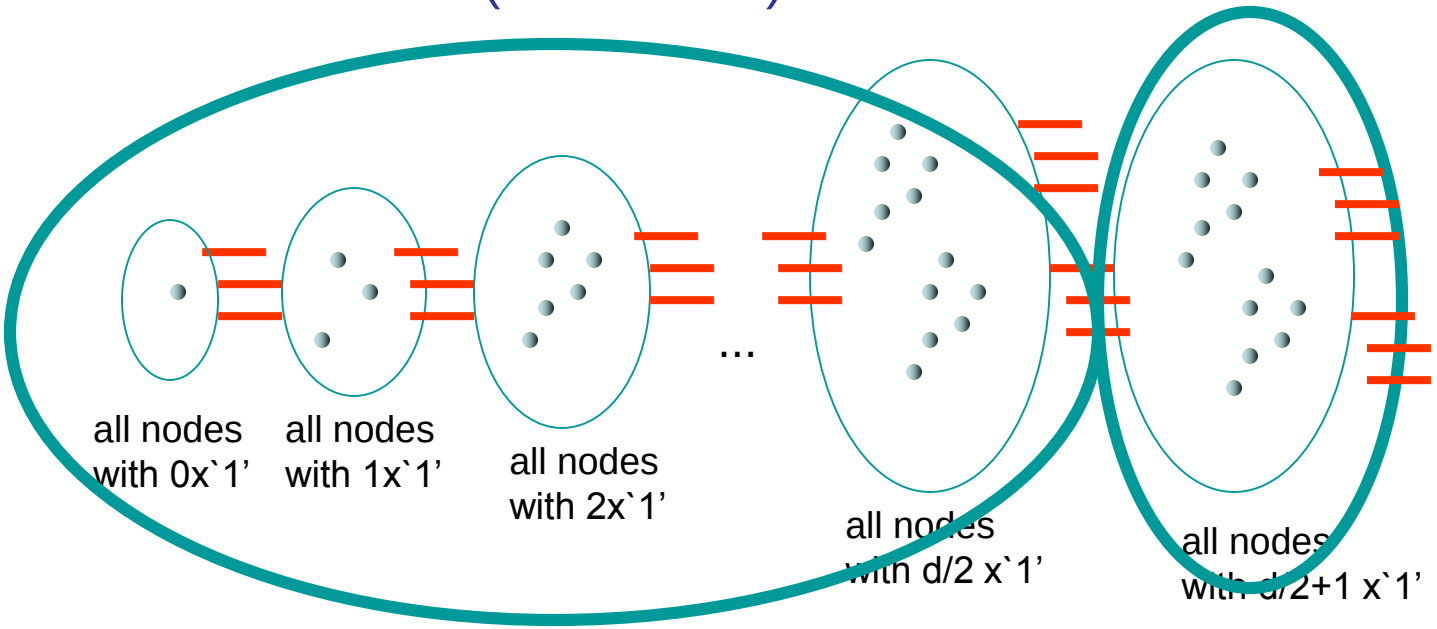
Size of neighborhood = number of possibilities to place d/2+1 `1's at d positions.

Idea:

Total number of nodes with between 0 and d/2 ,1-bits?

U (~n/2 nodes)

$\Gamma(U)$ = binomial(d,d/2+1)



Expansion $1/\sqrt{(\log n)}$ then follows from computing the ratio...

Expansion of Hypercube

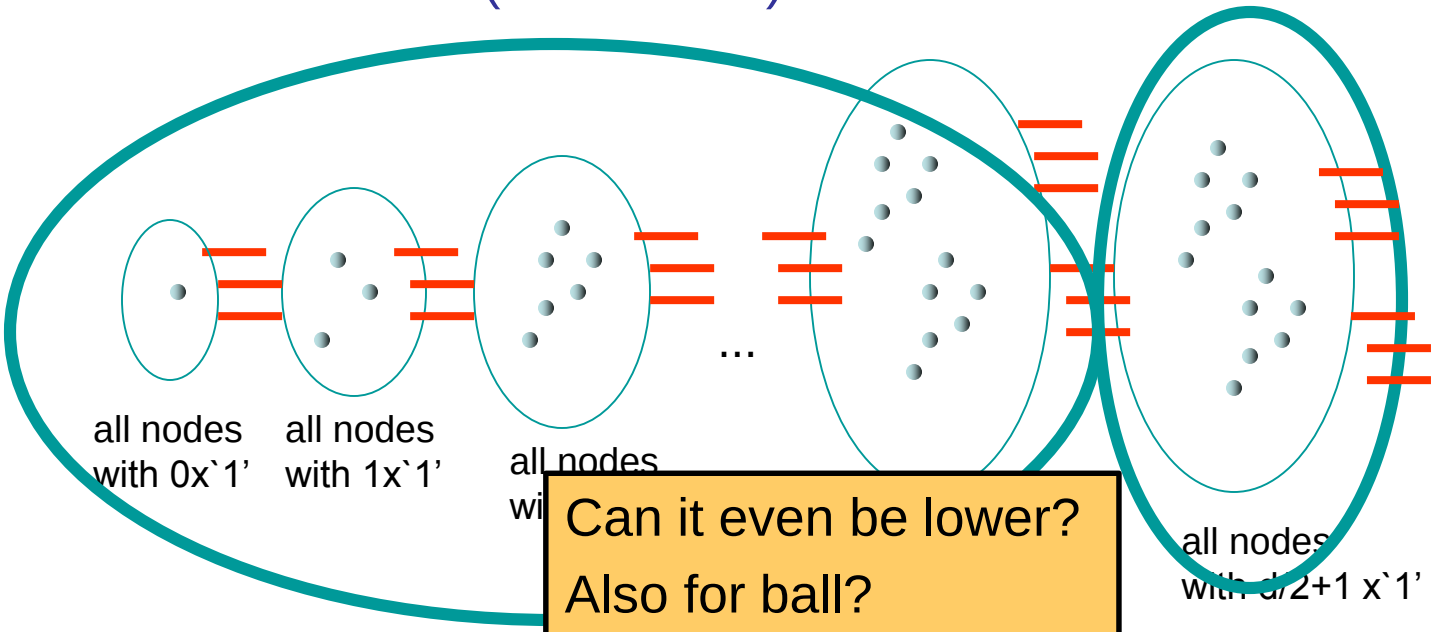
Size of neighborhood = number of possibilities to place d/2+1 `1's at d positions.

Idea:

Total number of nodes with between 0 and d/2 ,1-bits?

U (~n/2 nodes)

$\Gamma(U)$ = binomial(d,d/2+1)



Can it even be lower?

Also for ball?

Expansion $1/\sqrt{(\log n)}$ then follows from computing the ratio...

Many networks are hypercubic!

Many computer networks are variants or generalizations of hypercubes!

E.g., **peer-to-peer systems** (Chord, Pastry, Kademlia, ...)

E.g., **datacenter** topologies (container-based datacenters, BCube, MDCCube, ...)

E.g., **parallel architectures** (butterfly variants, etc.)

Many networks are hypercubic!

BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers*

Chuanxiong Guo[1], Guohan Lu[1], Dan Li[1], Haitao Wu[1], Xuan Zhang[1,2], Yunfeng Shi[1,3], Chen Tian[1,4], Yongguang Zhang[1], Songwu Lu[1,5]
1: Microsoft Research Asia, 2: Tsinghua, 3: PKU, 4: HUST, 5: UCLA
{chguo,lguohan,danil,hwu}@microsoft.com, xuan-zhang05@mails.tsinghua.edu.cn, shiyunfeng@pku.edu.cn, tianchen@mail.hust.edu.cn, ygz@microsoft.com, slu@cs.ucla.edu

ABSTRACT

This paper presents BCube, a new network architecture specifically designed for shipping-container based, modular data centers. At the core of the BCube architecture is its server-centric network structure, where servers with multi-

1. INTRODUCTION

Shipping-container based, modular data center (MDC) offers a new way in which data centers are built and deployed [13, 16, 23, 24, 25]. In an MDC, up to a few thousands of servers are interconnected via switches to form the network

emlia, ...)

acenters,

BCube, MDCCube, ...)

E.g., **parallel arcl**

MDCube: A High Performance Network Structure for Modular Data Center Interconnection

Haitao Wu, Guohan Lu, Dan Li, Chuanxiong Guo, Yongguang Zhang
Microsoft Research Asia (MSRA), China
{hwu, lguohan, danil, chguo, ygz}@microsoft.com

ABSTRACT

Shipping-container-based data centers have been introduced as building blocks for constructing mega-data centers. However, it is a challenge on how to interconnect those containers together with reasonable cost and cabling complexity, due to the fact that a mega-data center can have hundreds or even thousands of containers and the aggregate bandwidth among containers can easily reach tera-bit per second. As a new

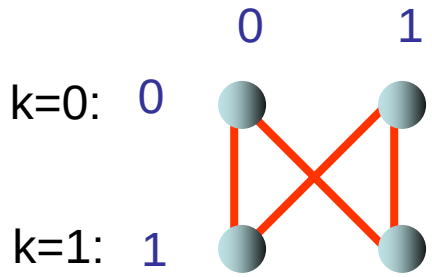
1. INTRODUCTION

Large data centers are built around the world to provide various online services. The deployment trend shows that the number of servers in data centers continues to grow. Companies like Amazon, Google, and Microsoft are building mega-data centers for cloud computing [13]. The recently proposed "shipping container" data center [13, 9, 11, 15, 16, 17] takes a modular approach, which is called Modu-

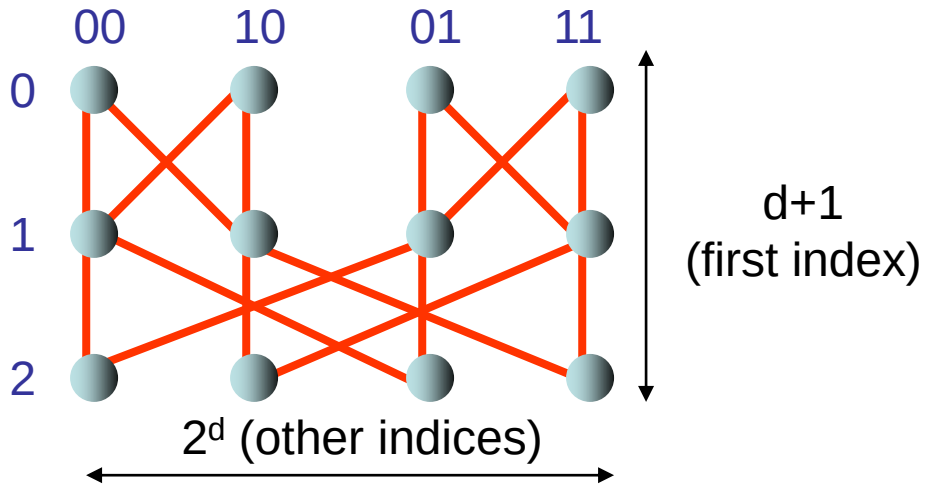
Butterfly: A rolled-out hypercube

Butterfly has bitstrings like hypercube, nodes also connected if one bit difference, but **not at every position**…

d=1: k={0,1}

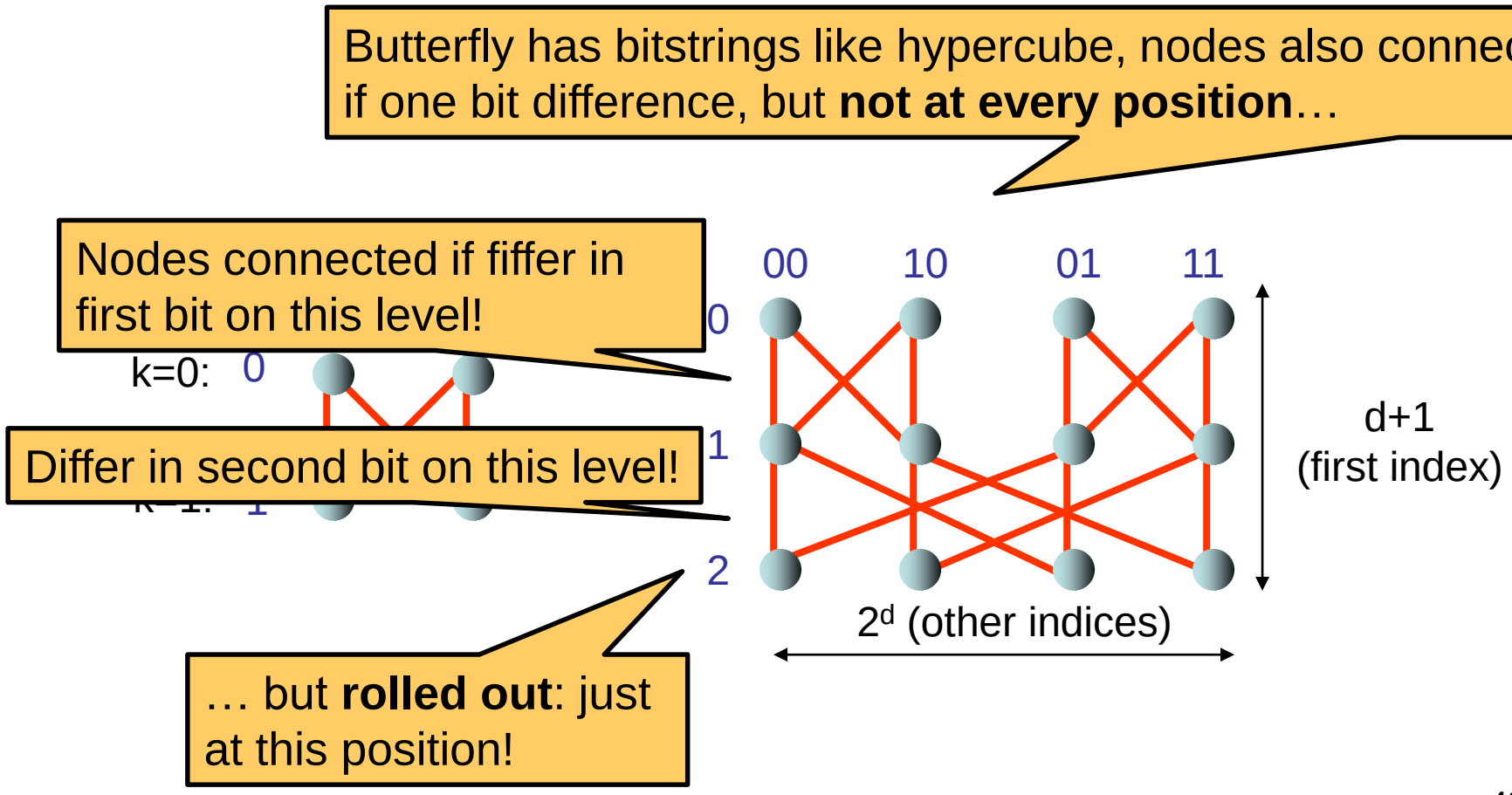


d=2:



… but **rolled out**: just at this position!

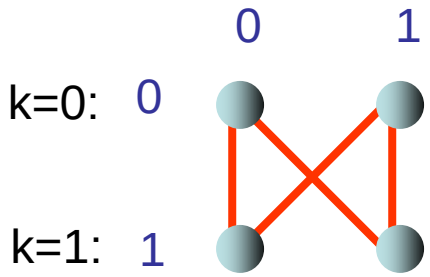
Butterfly: A rolled-out hypercube



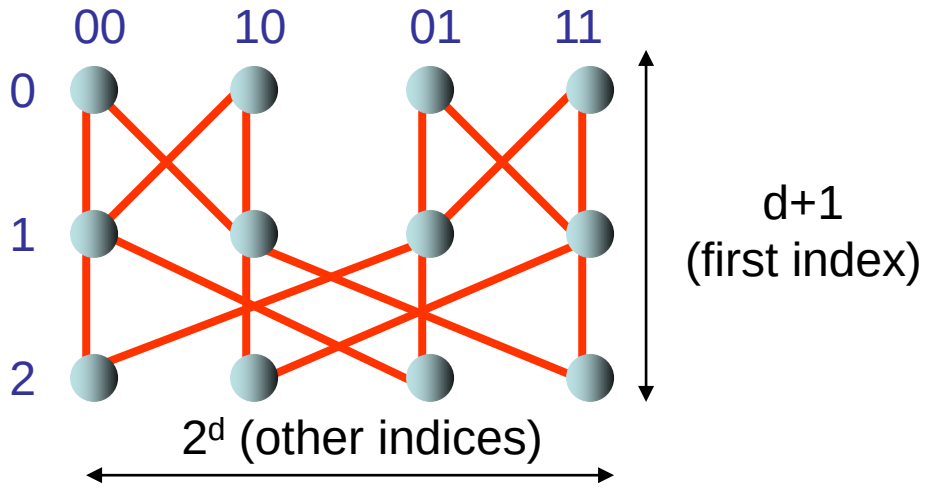
Butterfly: A rolled-out hypercube

Accordingly: 2-dimensional identifier: with the „rollout dimension“.

d=1: k={0,1}



d=2:



Butterfly: A rolled-out hypercube

Formally…

Butterfly graph: (e.g., for parallel architectures)

Nodes $V = \{(k, b_1...b_d), k \in \{0,...,d\}, b \in \{0,1\}^d\}$ (2-dim: „number+bitstring“)

Undirected edges E = for all i: $(k-1, b_1...b_k...b_d)$

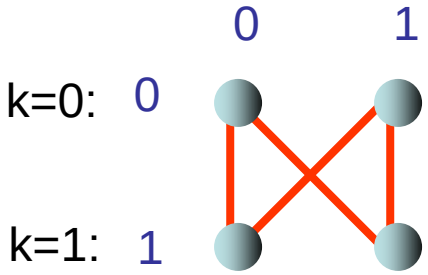
connected to $(k, b_1...b_k...b_d)$ and $(k, b_1...1\text{-}b_k...b_d)$

(i.e., to nodes on next level with same and opposite bit at only this position)

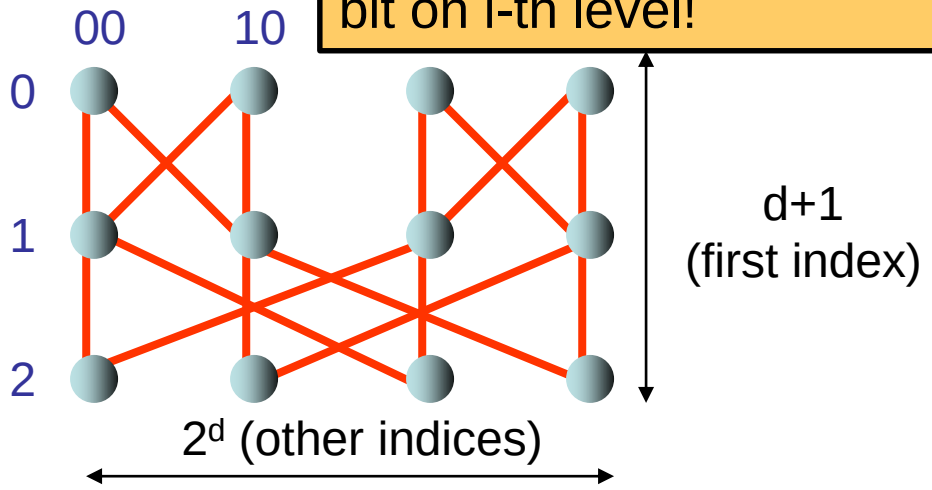
Essentially a rolled-out hypercube!

Roll-out hypercube into multiple levels: connect i-th bit on i-th level!

d=1: k={0,1}



d=2:



Butterfly: A rolled-out hypercube

Butterfly graph: (e.g., for parallel architectures)

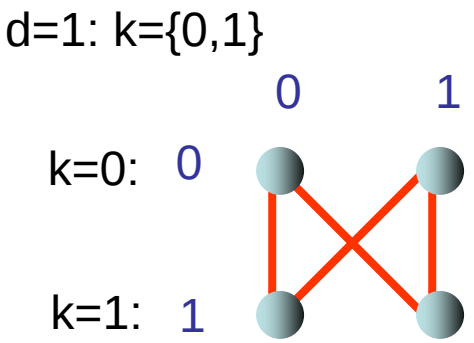
Nodes $V = \{(k, b_1...b_d), k \in \{0,...,d\}, b \in \{0,1\}^d\}$ (2-dim: „number+bitstring“)

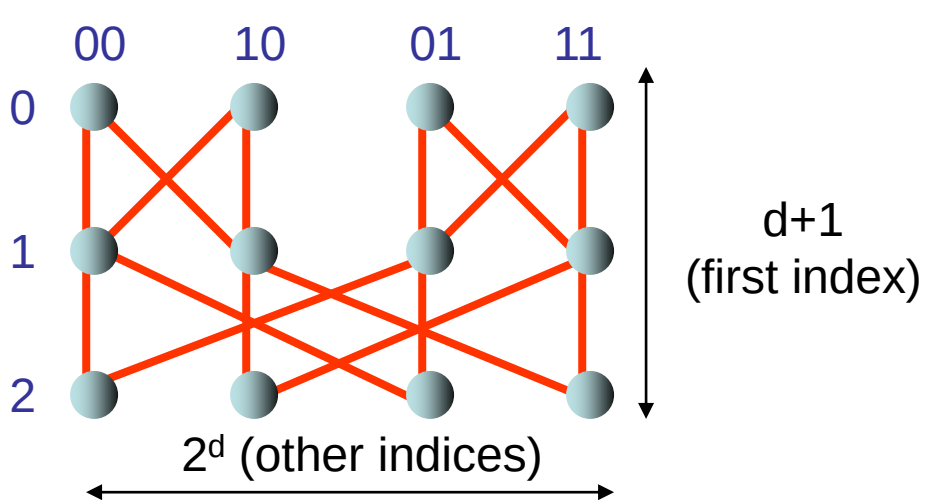
Undirected edges E = for all i: $(k-1, b_1...b_k...b_d)$

connected to $(k, b_1...b_k...b_d)$ and $(k, b_1...1\text{-}b_k...b_d)$

(i.e., to nodes on next level with same and opposite bit at only this position)

Essentially a rolled-out hypercube!





Diameter, Degree, Expansion?
How many nodes in total?

Butterfly: A rolled-out hypercube

Butterfly graph: (e.g., for parallel architectures)

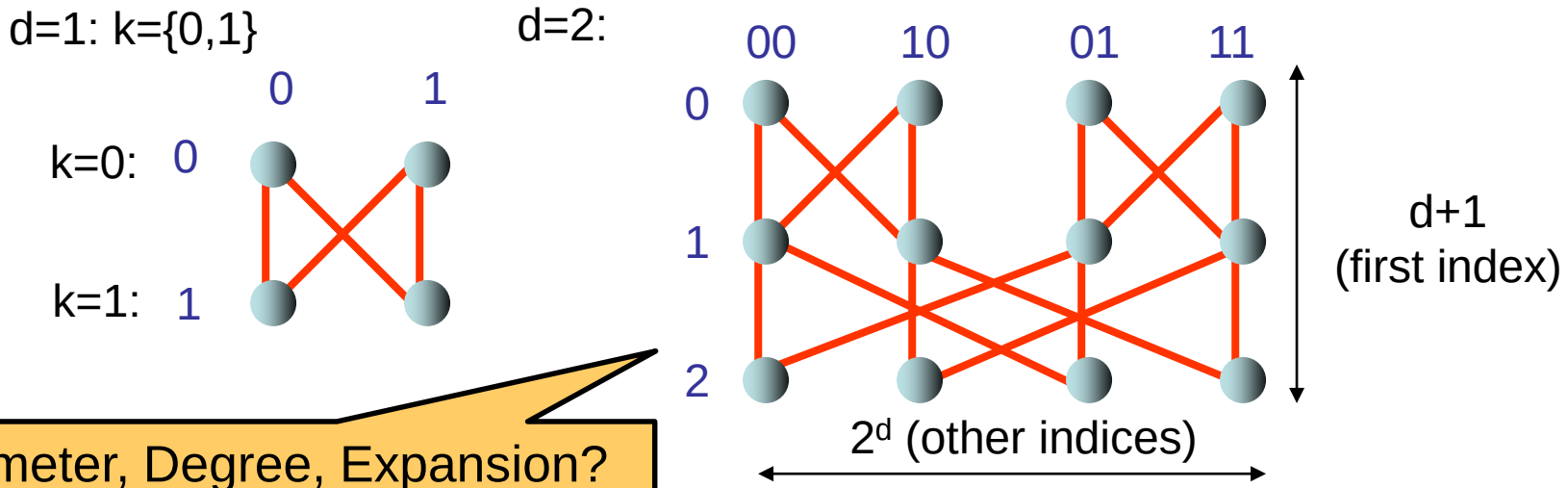
Nodes $V = \{(k, b_1...b_d), k \in \{0,...,d\}, b \in \{0,1\}^d\}$ (2-dim: „number+bitstring“)

Undirected edges E = for all i: $(k-1, b_1...b_k...b_d)$

connected to $(k, b_1...b_k...b_d)$ and $(k, b_1...1\text{-}b_k...b_d)$

(i.e., to nodes on next level with same and opposite bit at only this position)

Essentially a rolled-out hypercube!



Diameter, Degree, Expansion?
How many nodes in total?

Degree 4, Diameter 2d (e.g., go to corresponding „bottom“, then up)

Butterfly: A rolled-out hypercube

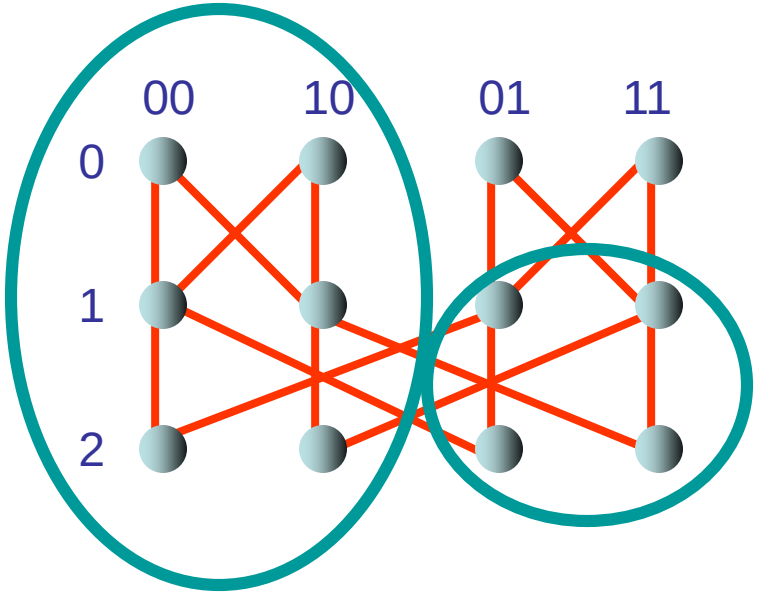
Butterfly graph:

Nodes V = {(k, b_1...b_d), k ϵ {0,...,d}, b ϵ {0,1}^{d}}

Edges E = for all i: (k-1, b_1...b_k...b_d)

connected to (k, b_1...b_k...b_d) and (k, b_1...1-b_k...b_d)

Expansion:

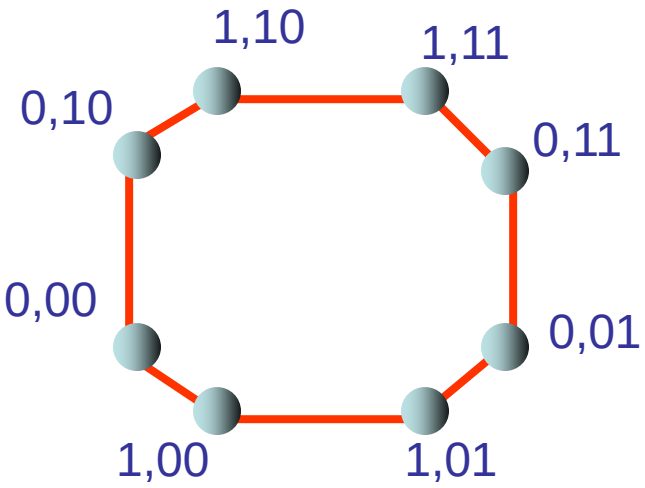


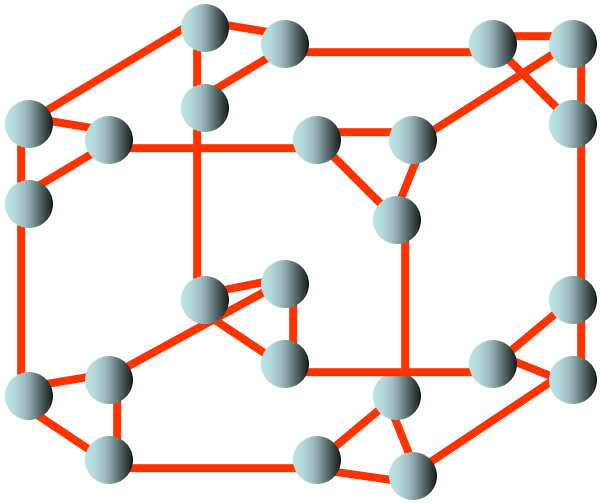
Left half only connected to right half at k=d. So neighborhood n/d nodes.

U has n/2 nodes.

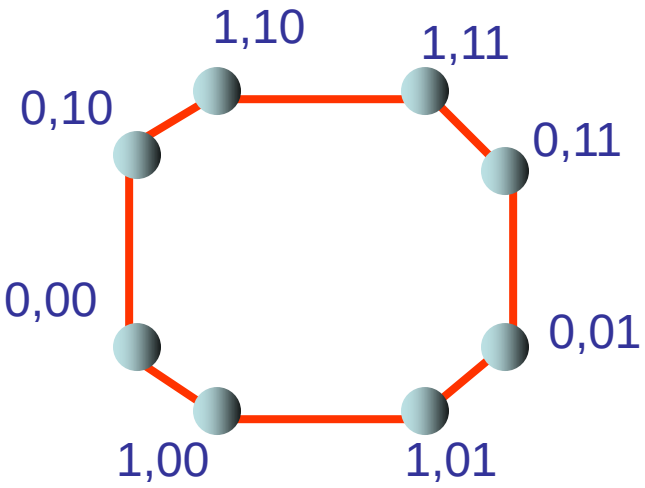
Expansion ~ 1/d.

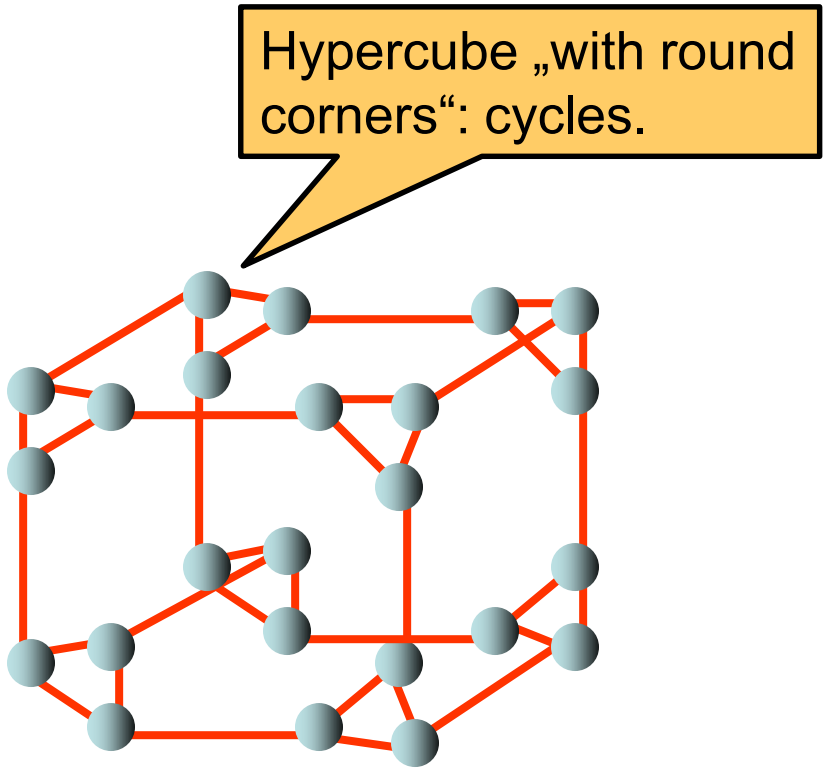
CCC: Hypercube with cyclic corners





CCC: Hypercube with cyclic corners





CCC: Hypercube with cyclic corners

Formally

Cube-Connected Cycles: Hypercube with „replaced corners“ (split into d nodes!)

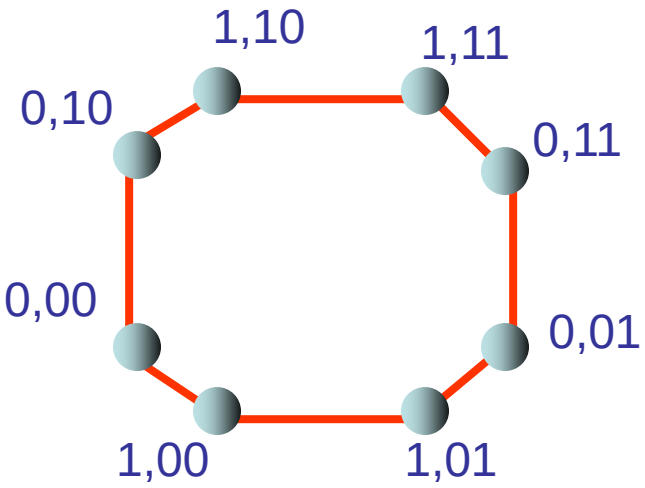
Nodes $V = \{(k, b_1...b_d)\ k \in \{0,...,d-1\}, b \in \{0,1\}^d\}$

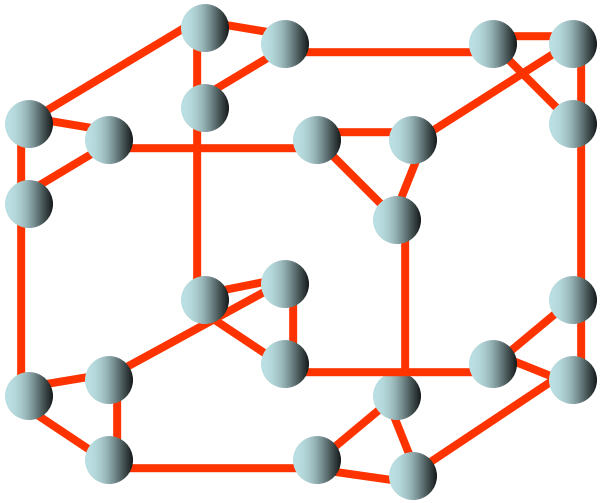
Edges E = for all i: $(k, b_1...b_k...b_d)$

connected to $(k-1, b_1...b_k...b_d)$, $(k+1, b_1...b_k...b_d)$ and $(k, b_1...1-b_k...b_d)$

(for each dimension one node!)

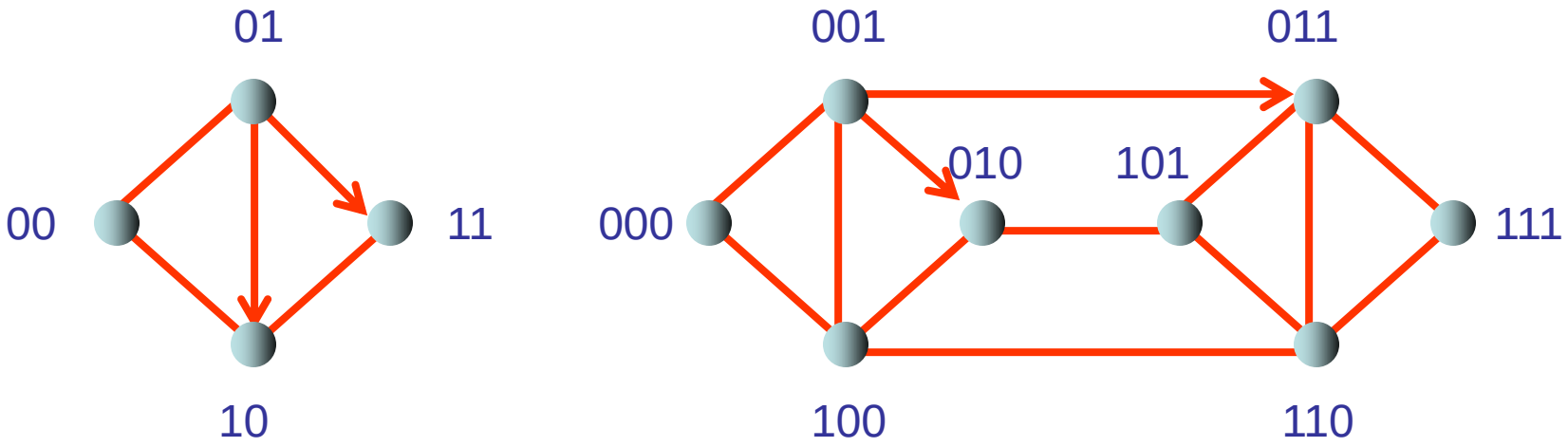
Example:



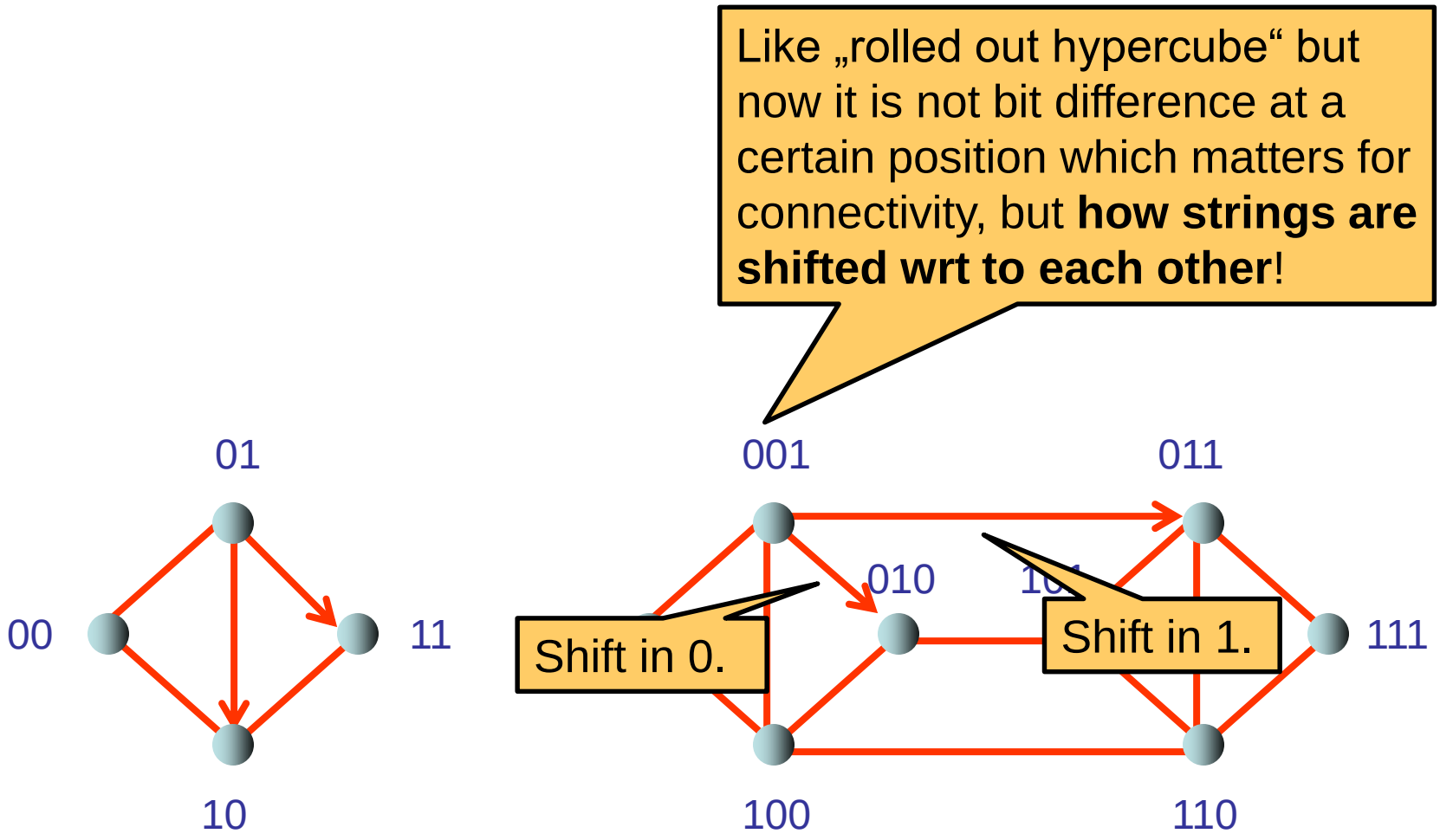


De Bruijn Graph: Pull-in bits from the back!

Like „rolled out hypercube“ but now it is not bit difference at a certain position which matters for connectivity, but **how strings are shifted wrt to each other**!



De Bruijn Graph: Pull-in bits from the back!



De Bruijn Graph: Pull-in bits from the back!

De Bruijn Graph:

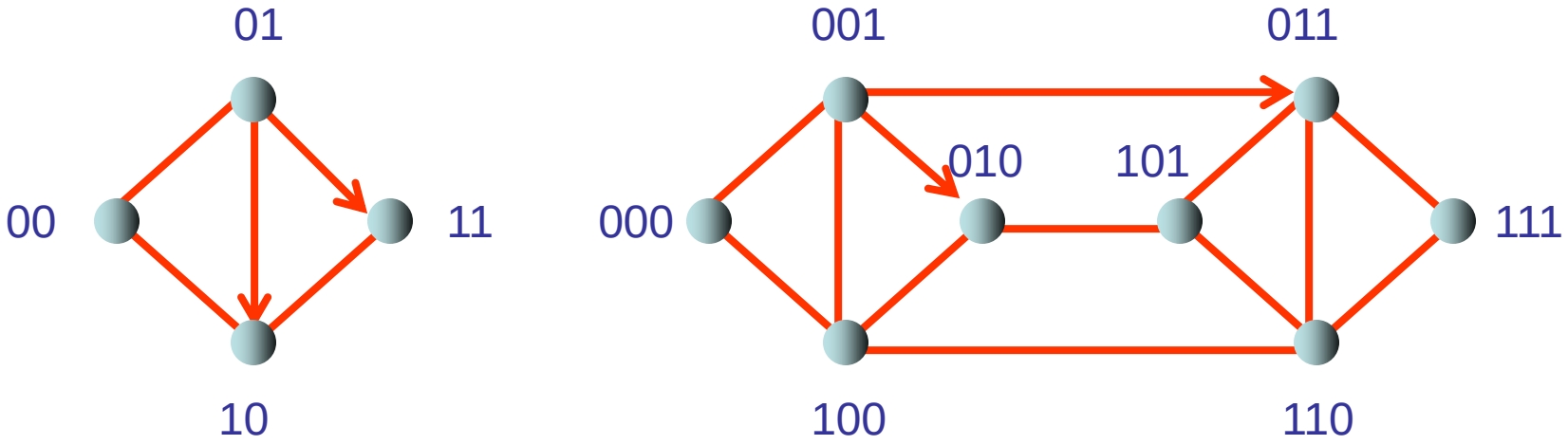
Formally…

Nodes V = $\{(b_1...b_d) \in \{0,1\}^d\}$ (bitstrings...)

(Undirected) edges E = for all i: $(b_1...b_k...b_d)$

connected to $(b_2...b_d0)$ and $(b_2...\ b_d1)$ („shift left and add 0 and 1“)

Example (undirected version):



De Bruijn Graph: Pull-in bits from the back!

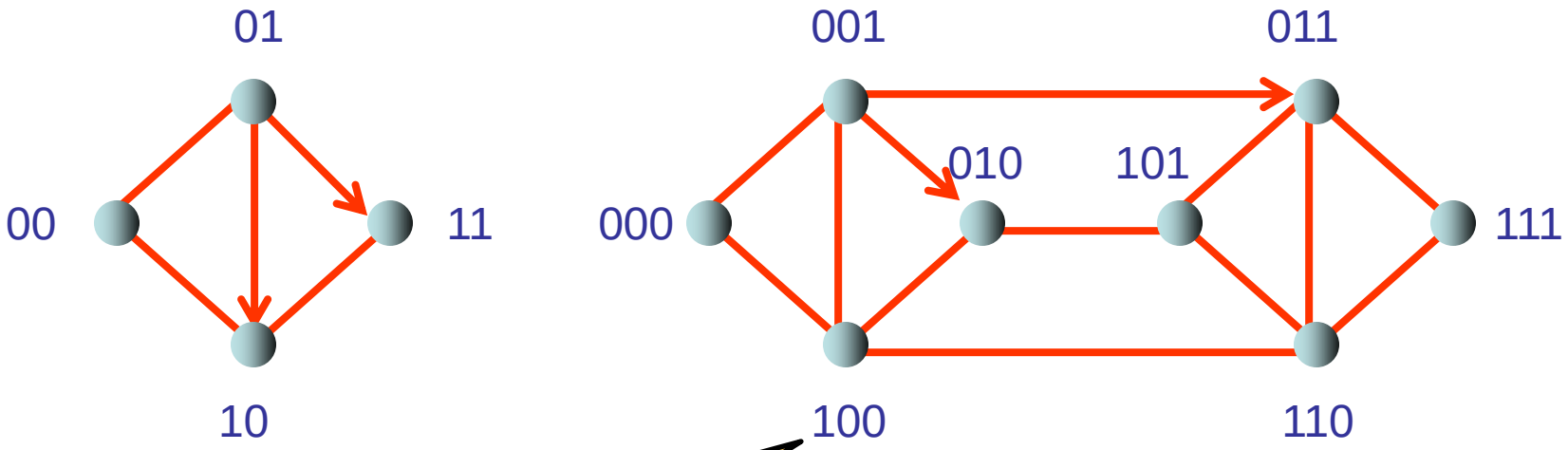
De Bruijn Graph:

Nodes V = $\{(b_1...b_d) \in \{0,1\}^d\}$ (bitstrings...)

(Undirected) edges E = for all i: $(b_1...b_k...b_d)$

connected to $(b_2...b_d0)$ and $(b_2...\ b_d1)$ („shift left and add 0 and 1“)

Example (undirected version):



How to do routing on this graph?

De Bruijn Graph: Pull-in bits from the back!

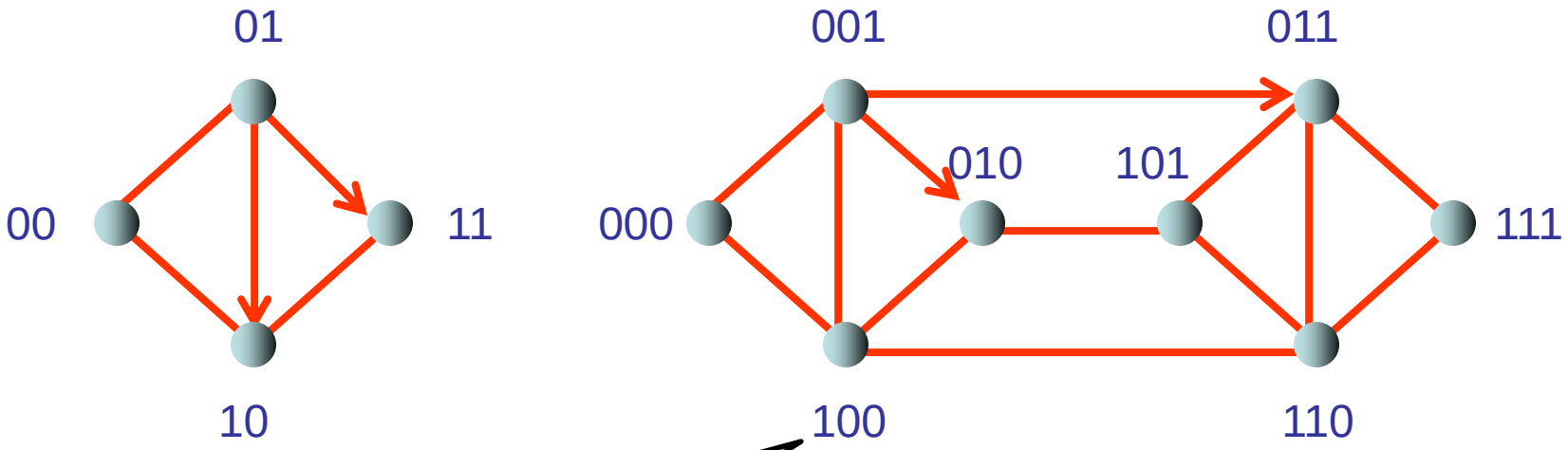
De Bruijn Graph:

Nodes V = $\{(b_1...b_d) \in \{0,1\}^d\}$ (bitstrings...)

(Undirected) edges E = for all i: $(b_1...b_k...b_d)$

connected to $(b_2...b_d0)$ and $(b_2...\ b_d1)$ („shift left and add 0 and 1“)

Example (undirected version):



How to do routing on this graph?

Fill in destination bits from the left!

We have seen graphs with diameter-degree pairs:

log(n)-log(n)

log(n)-O(1)

…

Is there a graph with:

O(1)-log(n)

max(diameter,degree)< O(log n)?

…

One can show the following theorem

Theorem

Each network with n nodes and max degree d>2 must have a diameter of at least log(n)/log(d-1)-1.

One can show the following theorem

Theorem

Each network with n nodes and max degree d>2 must have a diameter of at least log(n)/log(d-1)-1.

Implications: Constant diameter networks need a linear degree! But also: log(n)-log(n) is not the best tradeoff for minimizing the maximum of degree and diameter!

One can show the following theorem

Theorem

Each network with n nodes and max degree d>2 must have a diameter of at least log(n)/log(d-1)-1.

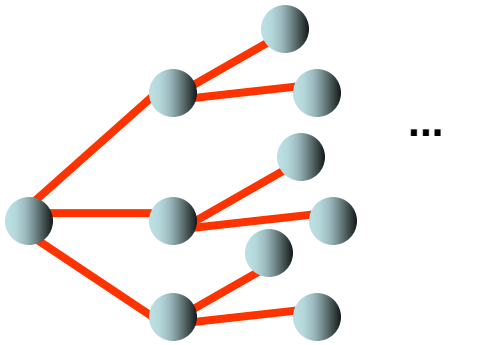
How to prove this?

Implications: Constant diameter networks need a linear degree! But also: log(n)-log(n) is not the best tradeoff for minimizing the maximum of degree and diameter!

One can show the following theorem

Theorem

Each network with n nodes and max degree d>2 must have a diameter of at least log(n)/log(d-1)-1.

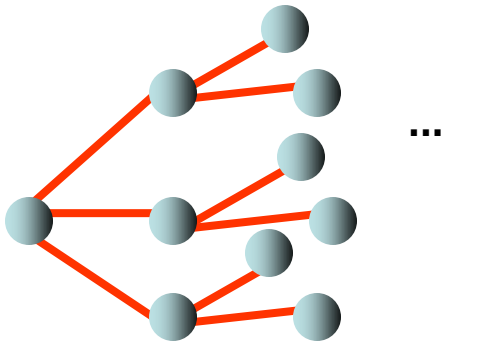


Proof by simply counting how many nodes can be reached given a certain degree!

What is the degree-diameter tradeoff? Idea? Proof?

Theorem

Each network with n nodes and max degree d>2 must have a diameter of at least log(n)/log(d-1)-1.



Proof by simply counting how many nodes can be reached given a certain degree!

In two steps, at most

d (d-1)

additional nodes can be reached!

So in *k* steps at most:

$$1 + \sum_{i=0}^{k-1} d \cdot (d-1)^i = 1 + d \cdot \frac{(d-1)^k - 1}{(d-1) - 1} \leq \frac{d \cdot (d-1)^k}{d-2}$$

Formally: need to reach n nodes (if connected, i.e., finite diameter).

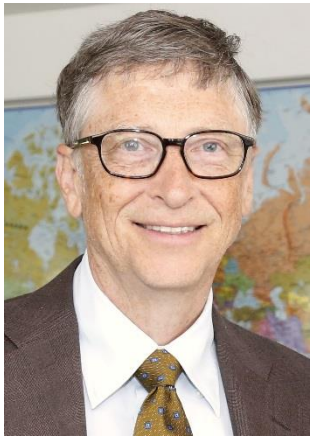
To ensure it is connected this must be at least n, so:

$$(d-1)^k \geq \frac{(d-2) \cdot n}{d} \quad \Leftrightarrow \quad k \geq \log_{d-1}\left(\frac{(d-2) \cdot n}{d}\right) \quad \Leftrightarrow \quad k \geq \log_{d-1} n + \log_{d-1}\left(\frac{d-2}{d}\right)$$

Reformulating this yields the claim... ☺

HOMEWORK: Understand the Pancake Graph

O(log n / loglog n) diameter and degree!



Homework: Pancake Graphs P_n

Define P_n as follows: the vertex set is

$$V(P_n) = \{v_1 v_2 \dots v_n \mid v_i \in [n] \text{ and } v_i \neq v_j \ \forall i \neq j\} \tag{1}$$

where we use $[n] = \{1, 2, \dots, n\}$. In other words, $V(P_n) = S_n$, the group of all permutations on n elements. There exists an edge of dimension i for $2 \leq i \leq n$ when

$$e_i = (u_1 u_2 \dots u_i \dots u_n, v_1 v_2 \dots v_i \dots v_n) \in E(P_n) \iff$$
$$v_j = u_{i-j+1} \text{ for } 1 \leq j \leq i \text{ and } v_j = u_j \text{ for } i < j \leq n \tag{2}$$

or, we can say that an edge e_i represents a *prefix reversal*

$$v_1 v_2 \dots v_i v_{i+1} \dots v_n \longleftrightarrow v_i \dots v_2 v_1 v_{i+1} \dots v_n. \tag{3}$$

For the following questions, where appropriate, give your answers in terms of $N := |V(P_n)|$ (approximately), the number of vertices, as well as n.

a) Draw (nicely!) P_n for $n = 2, 3, 4$. Try to describe a pattern for drawing P_n for any n.

b) What is the degree of each vertex in P_n?

c) Can you give bounds on the diameter $D(P_n)$ of the pancake network?