Lectures in Wroclaw

- **Epidemic Algorithms**
    - Monday, April 6th, 2009, 3pm
- **Random Networks**
    - Monday, April 6th, 2009, 6pm
- **Distributed Heterogeneous Hash Tables**
    - Tuesday, April 7th, 2009, 3pm
- **Network Coding**
    - Wednesday, April 8th, 2009, 11am
- **Locality in Peer-to-Peer Networks**
    - Wednesday, April 8th, 2009, 3pm

University of Freiburg
Department of Computer Science
Computer Networks and Telematics
Prof. Dr. Christian Schindelhauer

CoNe
Freiburg

Random Graphs for Peer-to-Peer Overlays

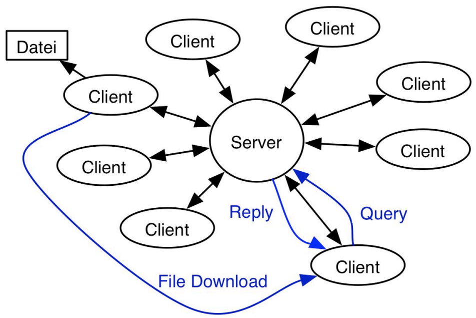
Christian Schindelhauer

joint work with

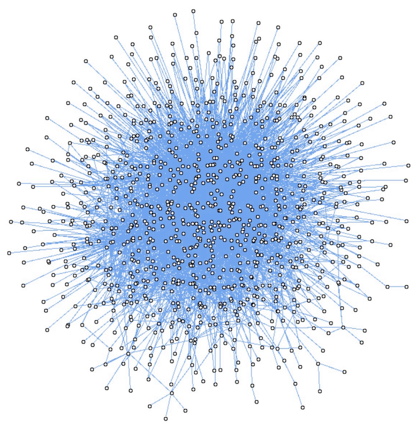
Peter Mahlmann
University of Paderborn

A Short History of Peer-to-Peer-Networks

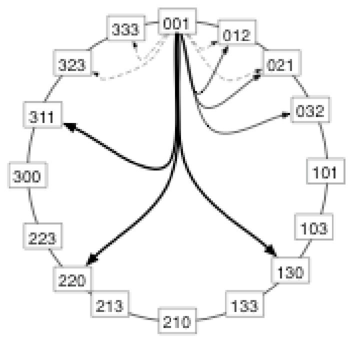
- **1st Generation**
  - Shawn “Napster” Fanning (1999)
  - Centralized client-server database
  - Peer-to-peer: download (mostly mp3-music)
  - Shut down by court order because of copyright infr



- **2nd Generation:**
  - Decentralized, uncontrolled communication network
  - Lookup by broadcasting a query
    - Gnutella (Frankel, Pepper, 2000)
    - eDonkey
    - FastTrack



- **3rd Generation**
  - Efficient data structures (DHT)
    - CAN, Chord, Pastry, Tapestry, ...
  - Anonymity features
    - Freenet, I2P, GNUnet



Peer-to-Peer Networking Facts

- **Hostile environment**
  - Legal situation
  - Egoistic users
  - Networking
    - ISP filter Peer-to-Peer Networking traffic
    - User arrive and leave
    - Several kinds of attacks
    - Local system administrators fight peer-to-peer networks
- **Implication**
  - Use stable robust network structure as a backbone
  - Napster: star
  - CAN: lattice
  - Chord, Pastry, Tapestry: ring + pointers for lookup
  - Gnutella, FastTrack: chaotic "social" network
- **Idea: Use a Random d-regular Network**

Why Random Networks ?

- **Random Graphs ...**
  - Robustness
  - Simplicity
  - Connectivity
  - Diameter
  - Graph expander
  - Security
- **Random Graphs in Peer-to-Peer networks:**
  - Gnutella
  - JXTApose

Dynamic Random Networks ...

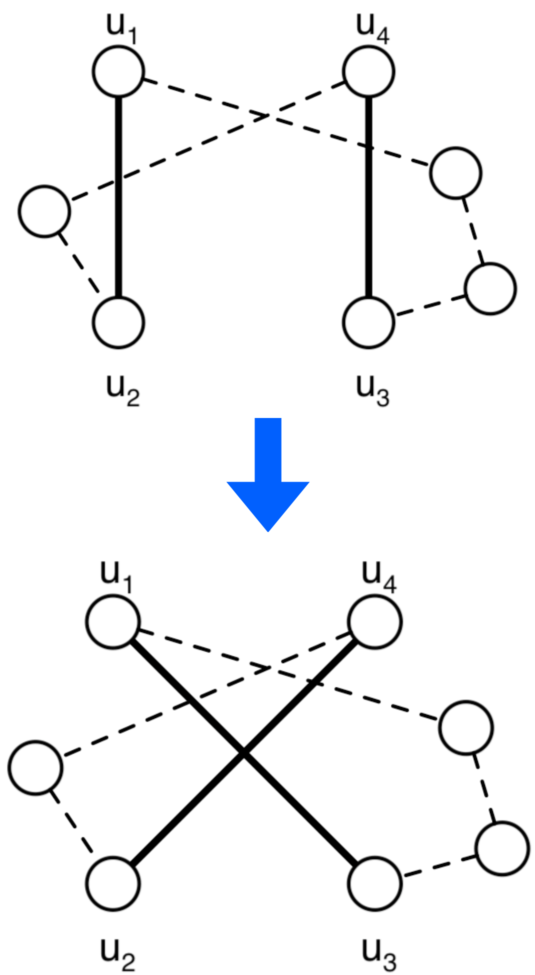
- **Peer-to-Peer networks are highly dynamic ...**

- maintenance operations are needed to preserve properties of random graphs
- which operation can maintain (repair) a random digraph?

Desired properties:	
Soundness	**Operation remains in domain** **(preserves connectivity and out-degree)**
Generality	**every graph of the domain is reachable** **does not converge to specific small graph set**
Feasibility	**can be implemented in a P2P-network**
Convergence Rate	**probability distribution converges quickly**

Simple Switching

- **Simple Switching**
  - choose two random edges
    - $\{u_1,u_2\} \in E$, $\{u_3,u_4\} \in E$
  - such that $\{u_1,u_3\}$, $\{u_2,u_4\} \notin E$
    - add edges $\{u_1,u_3\}$, $\{u_2,u_4\}$ to E
    - remove $\{u_1,u_2\}$ and $\{u_3,u_4\}$ from E
- **McKay, Wormald, 1990**
  - Simple Switching converges to uniform probability distribution of random network
  - Convergence speed:
    - $O(nd^3)$ for $d \in O(n^{1/3})$
- **Simple Switching cannot be used in Peer-to-Peer networks**
  - Simple Switching disconnects the graph with positive probability
  - No network operation can re-connect disconnected graphs



Necessities of Graph Transformation

	Simple-Switching
Graphs	Undirected Graphs
Soundness	?
Generality	↯
Feasibility	✔
Convergence	✔

- **Problem: Simple Switching does not preserve connectivity**
- **Soundness**
  - Graph transformation remains in domain
  - Map connected d-regular graphs to connected d-regular graphs
- **Generality**
  - Works for the complete domain and can lead to any possible graph
- **Feasibility**
  - Can be implemented in P2P network
- **Convergence Rate**
  - The probability distribution converges quickly

Directed Random Graphs

- **Peter Mahlmann, Christian Schindelhauer, Distributed Random Digraph Transformations for Peer-to-Peer Networks, 18th ACM Symposium on Parallelism in Algorithms and Architectures, Cambridge, MA, USA. July 30 - August 2, 2006**

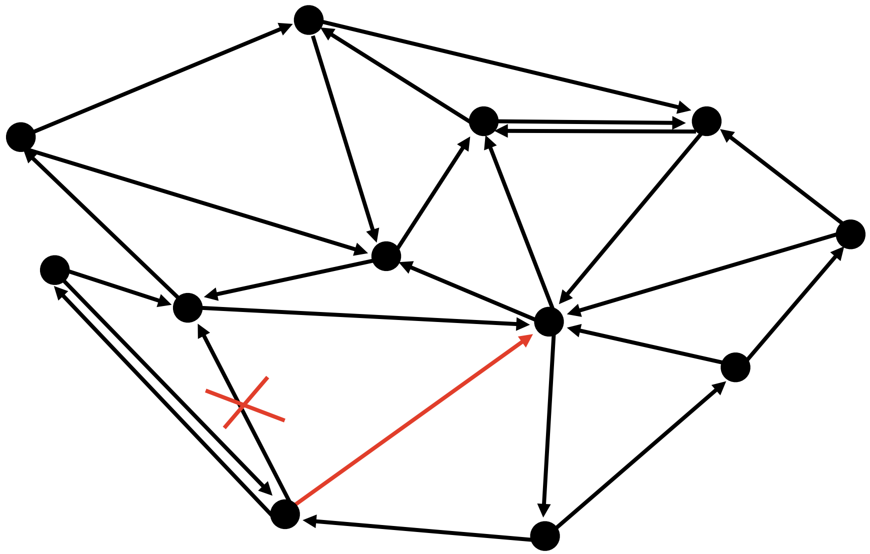
Directed Graphs

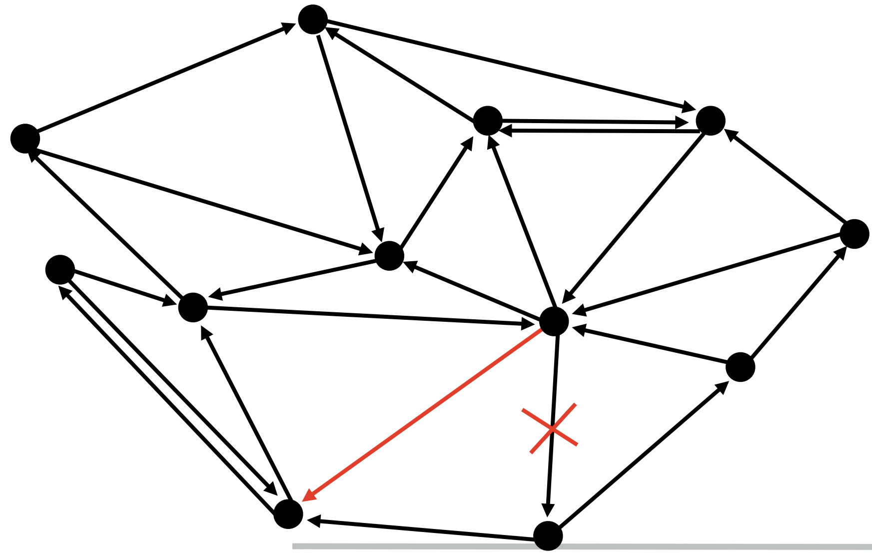
Push Operation:

1. Choose random node *u*
2. Set *v* to *u*
3. While a random event with p= 1/*h* appears
   a) Choose random edge starting at *v* and ending at *v*‘
   1. Set *v* to *v*‘
2. Insert edge (*v*,*v*‘)
a) Remove random edge starting at *v*

Pull Operation:

1. Choose random node *u*
2. Set *v* to *u*
3. While a random event with p= 1/*h* appears
   a) Choose random edge starting at *v* and ending at *v*‘
   1. Set *v* to *v*‘
2. Insert edge (*v*‘,*v*)
a) Remove random edge starting at *v*‘





Simulation of Push-Operations

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

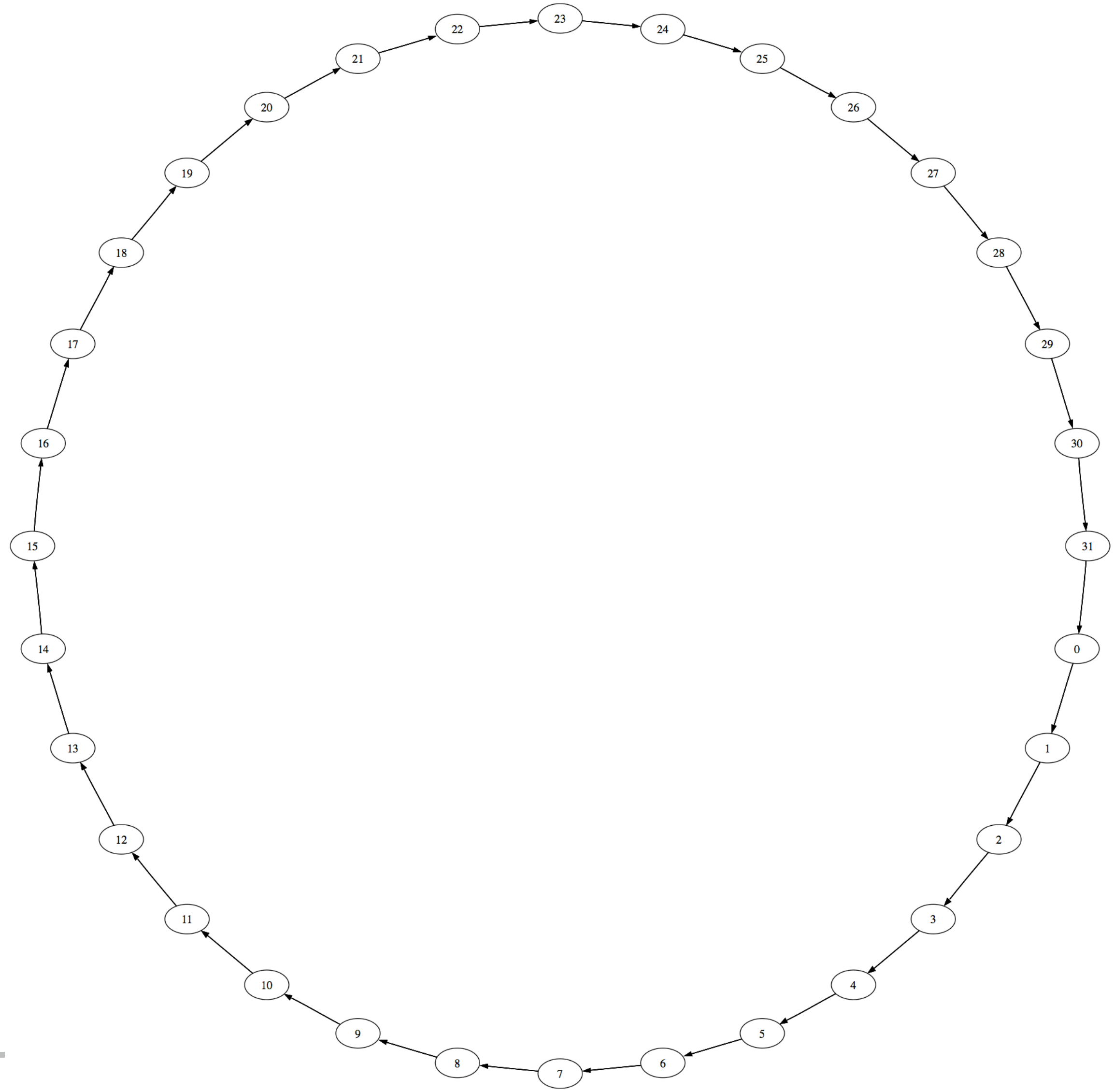
Startsituation

Parameter:

n = 32 Knoten

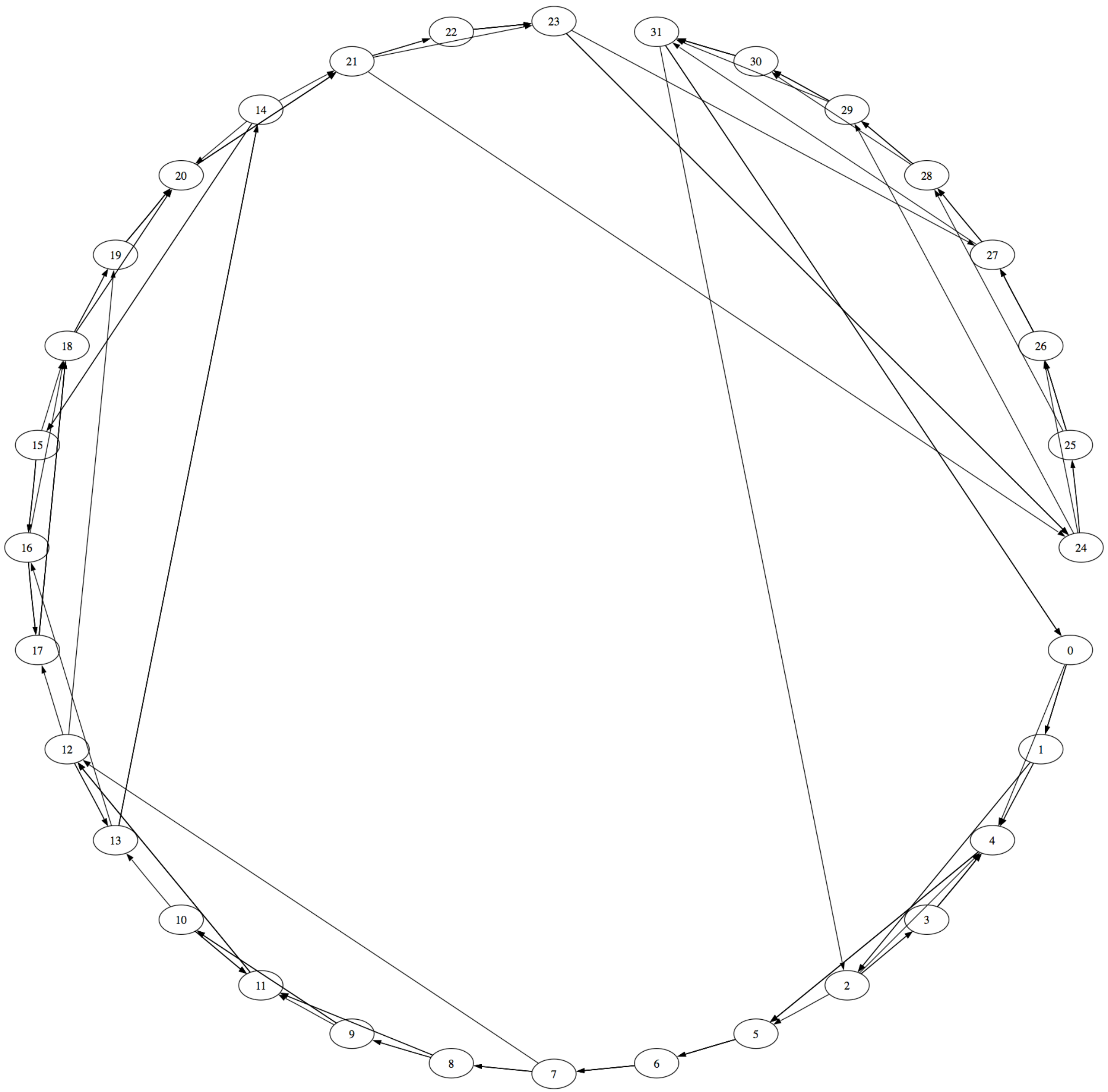
out-degree d = 4

Hop-distance h = 3



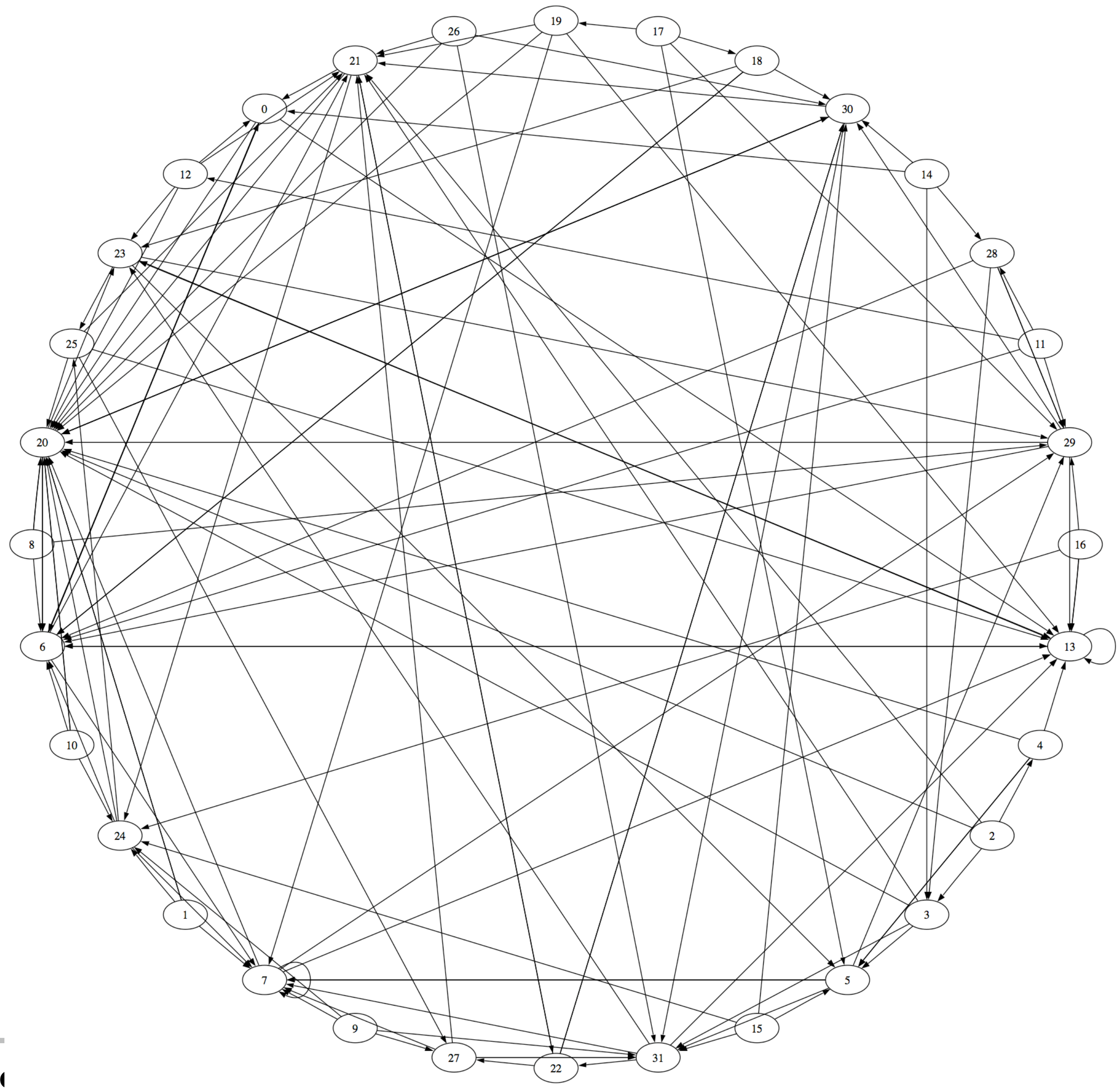
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

1 Iteration Push ...

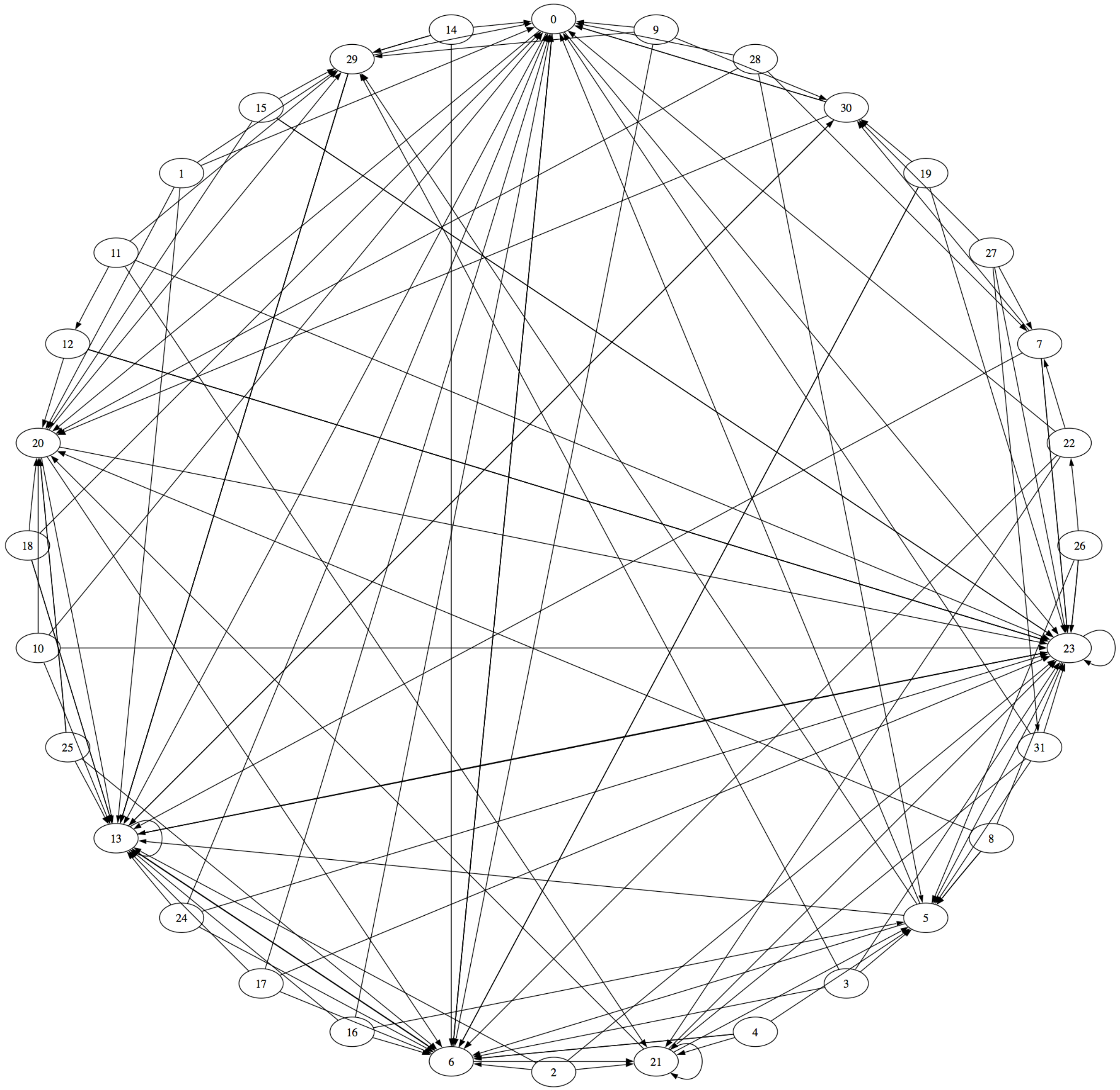


Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

10 Iterations Push ...

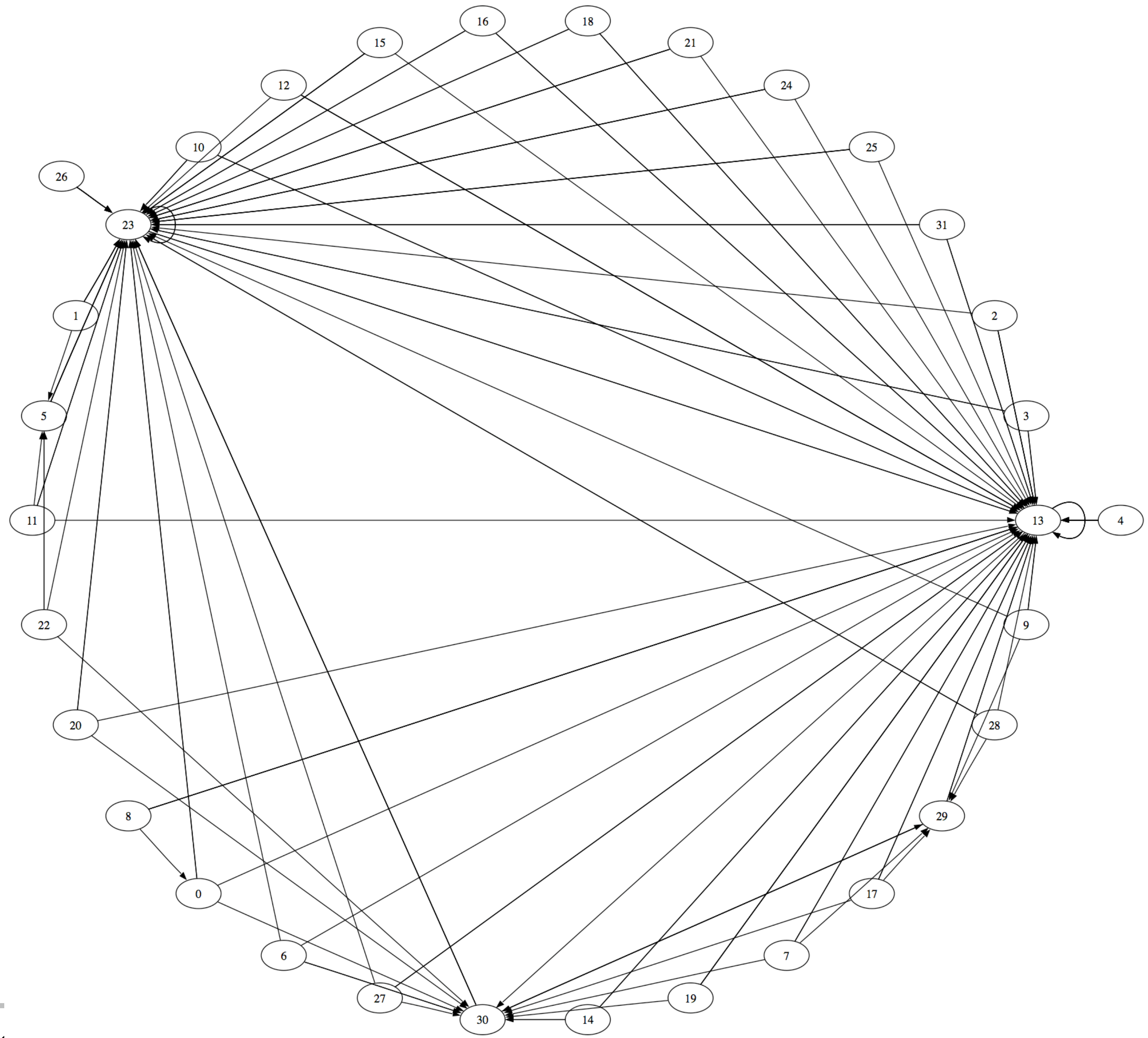


20 Iterations von Push ...



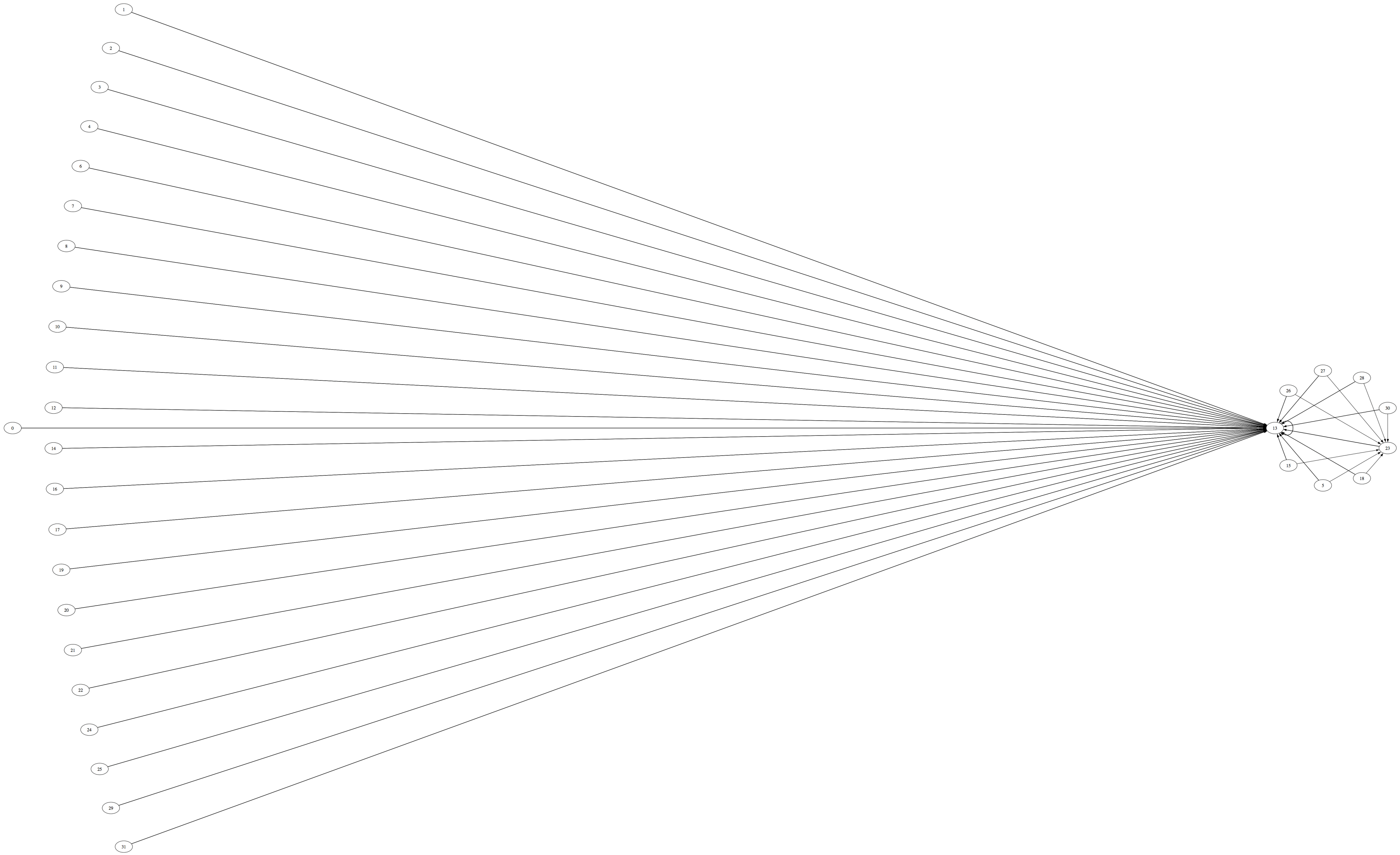
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

30 Iterations Push ...



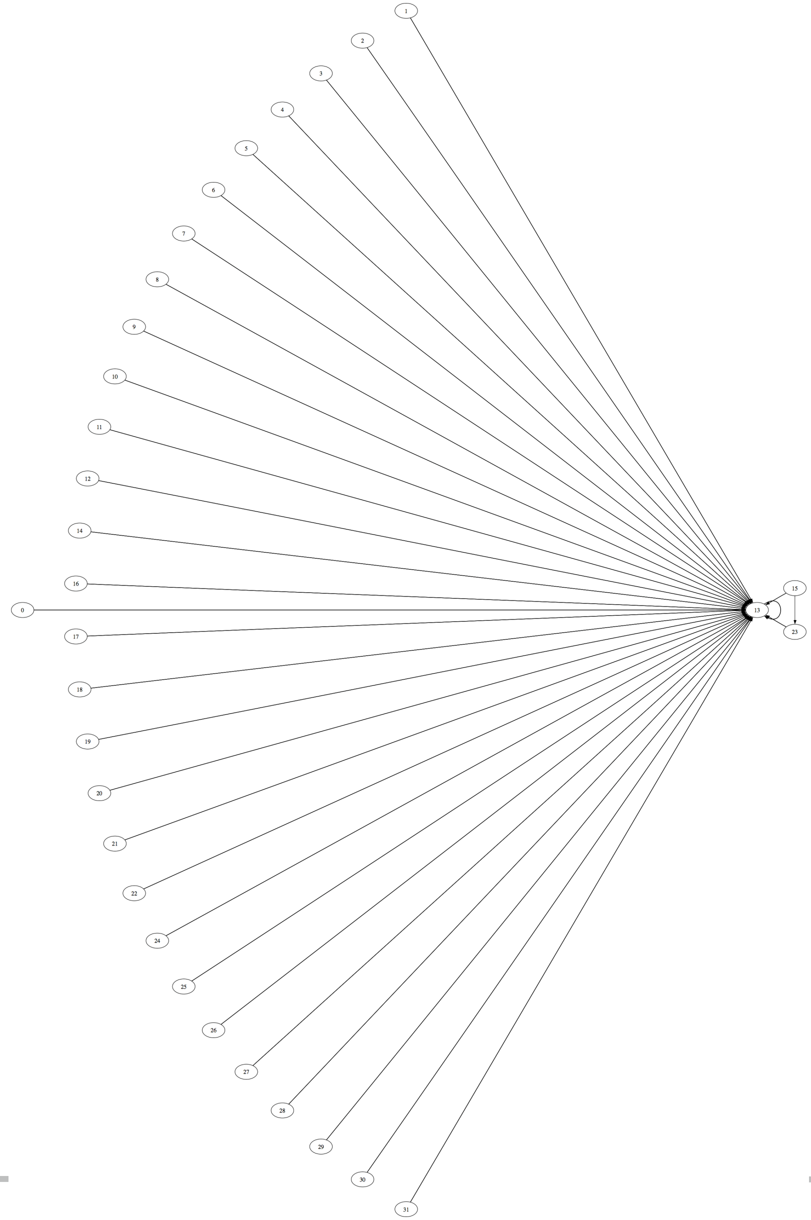
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

40 Iterations Push ...



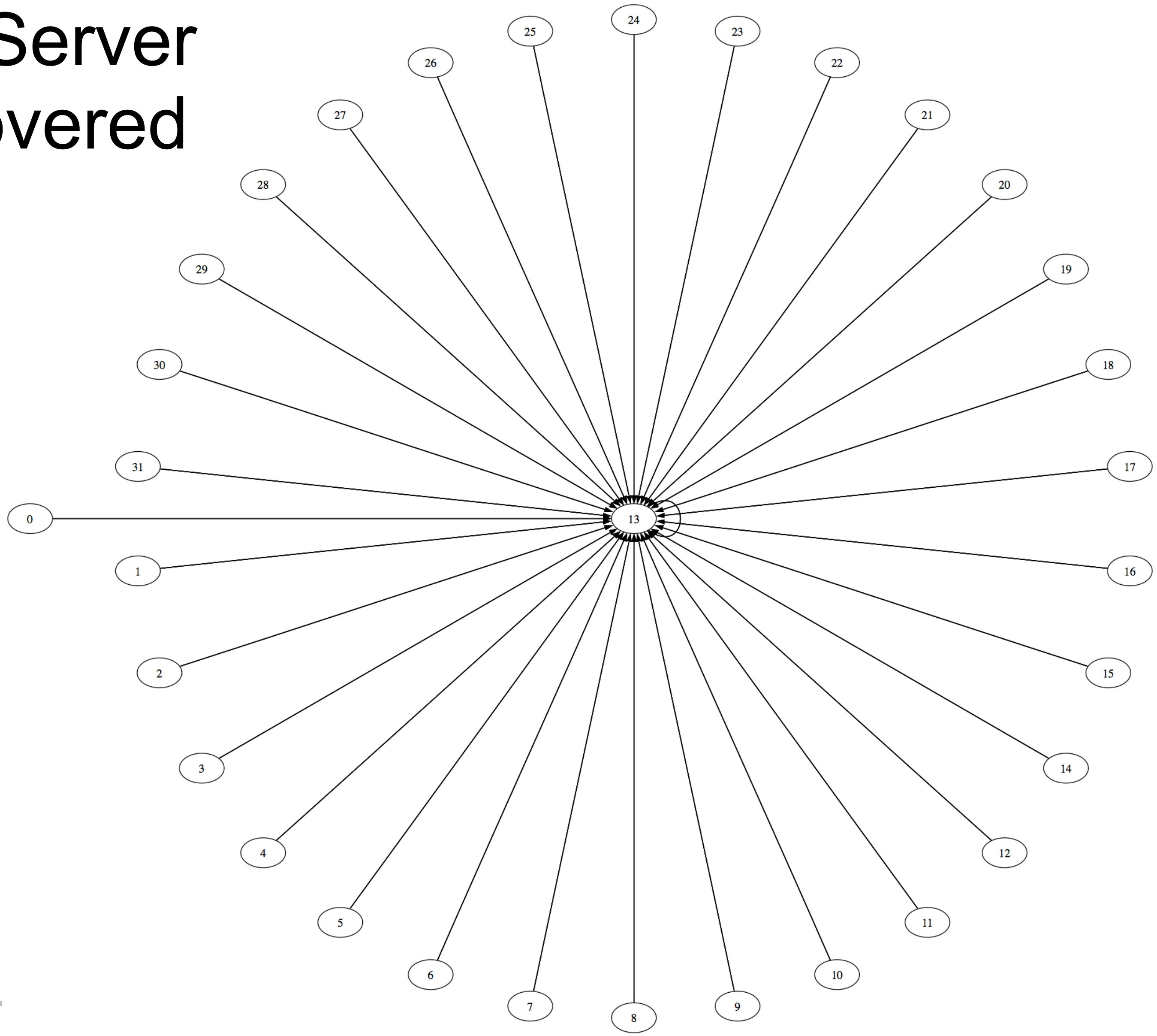
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

50 Iterations Push ...



70 Iterations Push ...

Client-Server rediscovered



Simulation der Pull-Operation ...

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

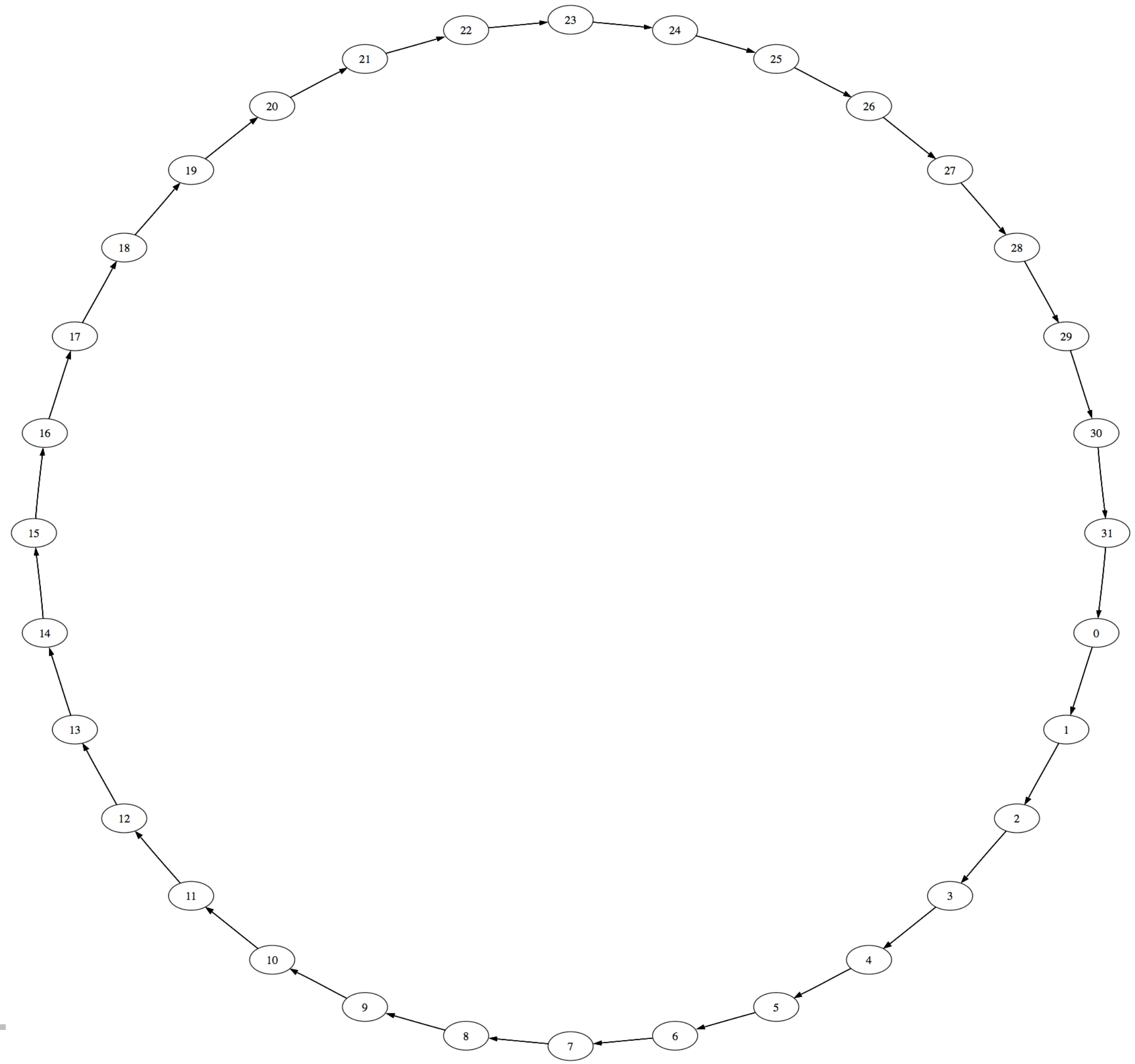
Start situation

Parameter:

n = 32 nodes

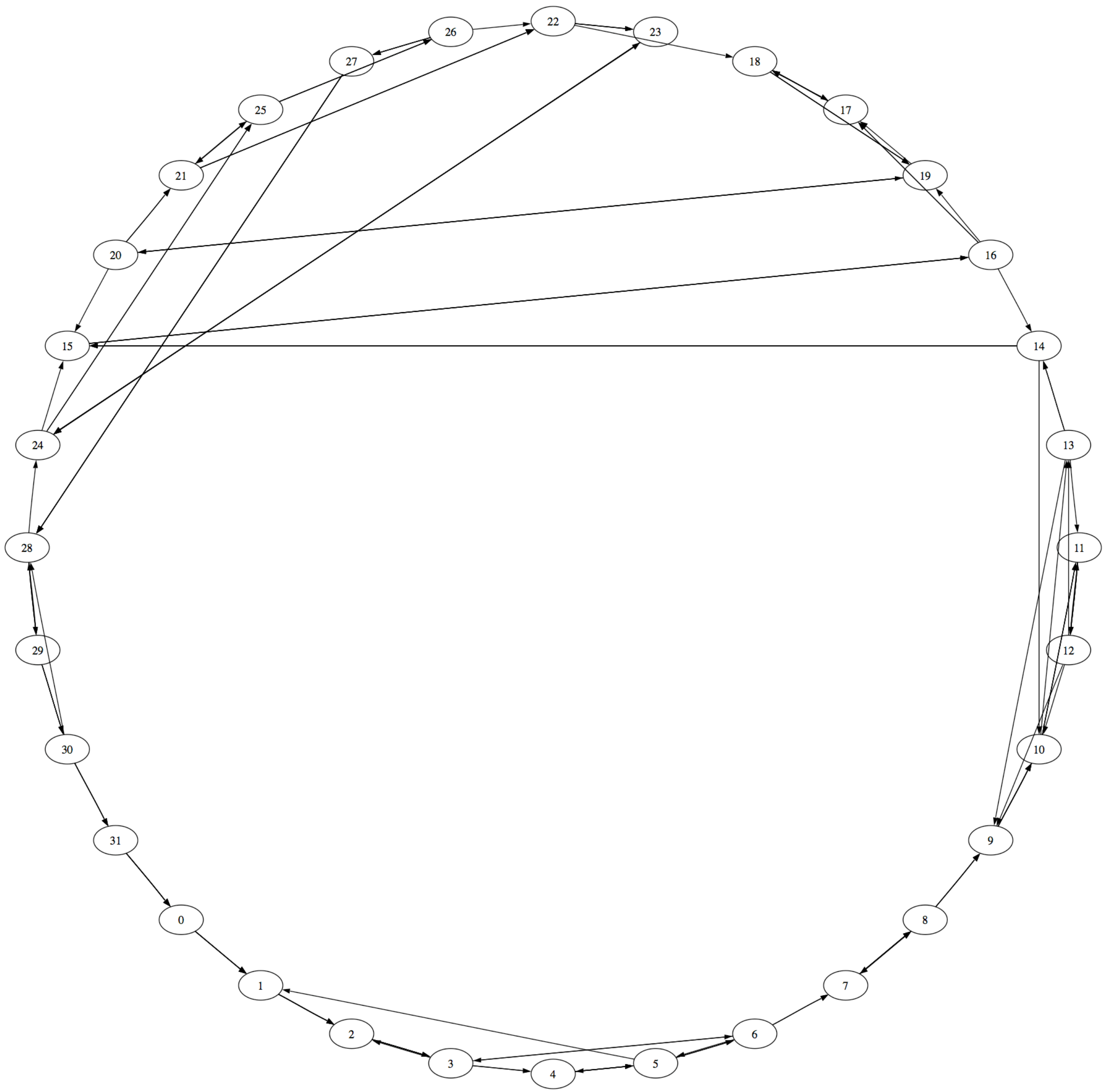
outdegree d = 4

hop distance h = 3

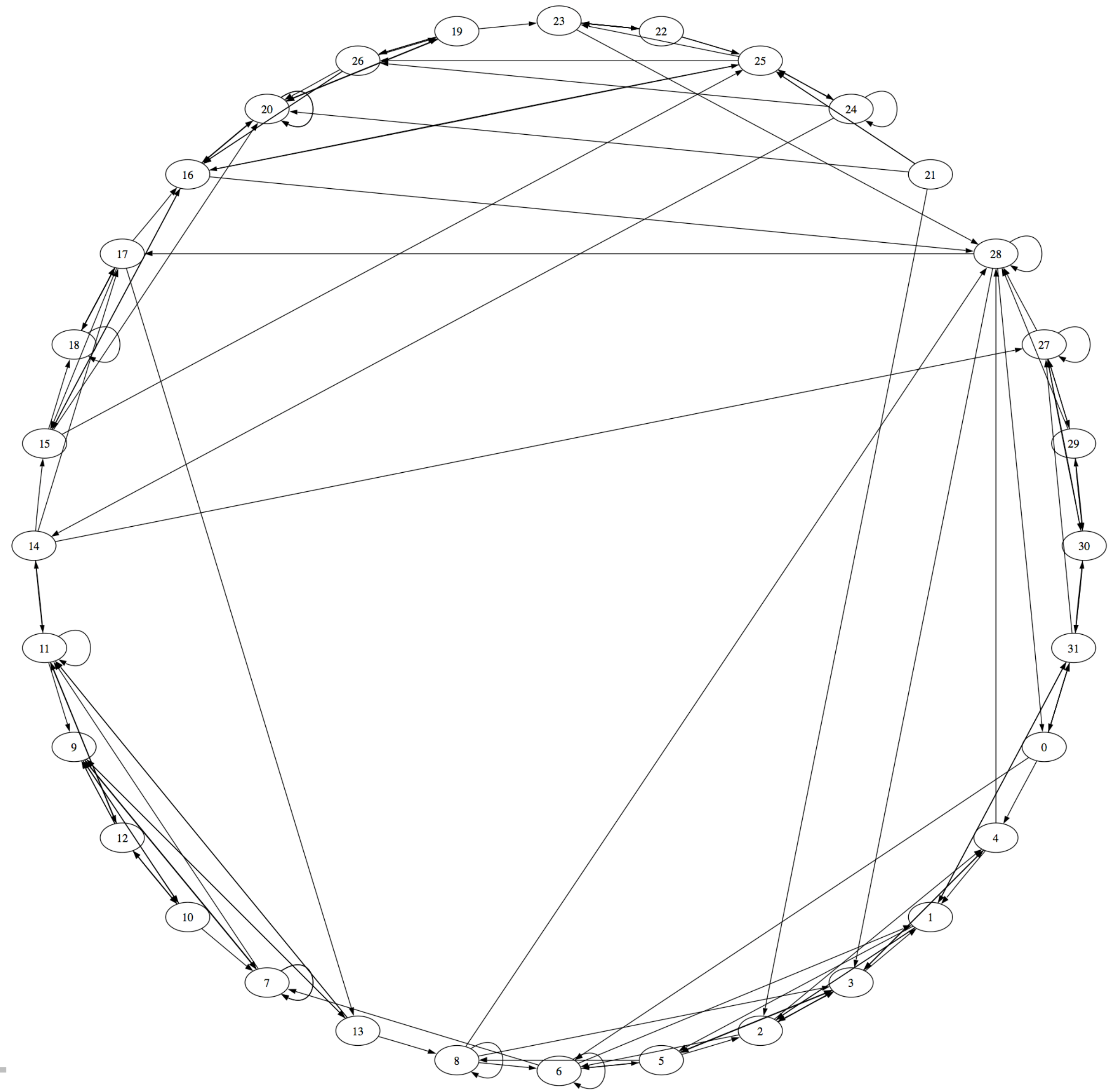


Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

1 Iteration Pull ...

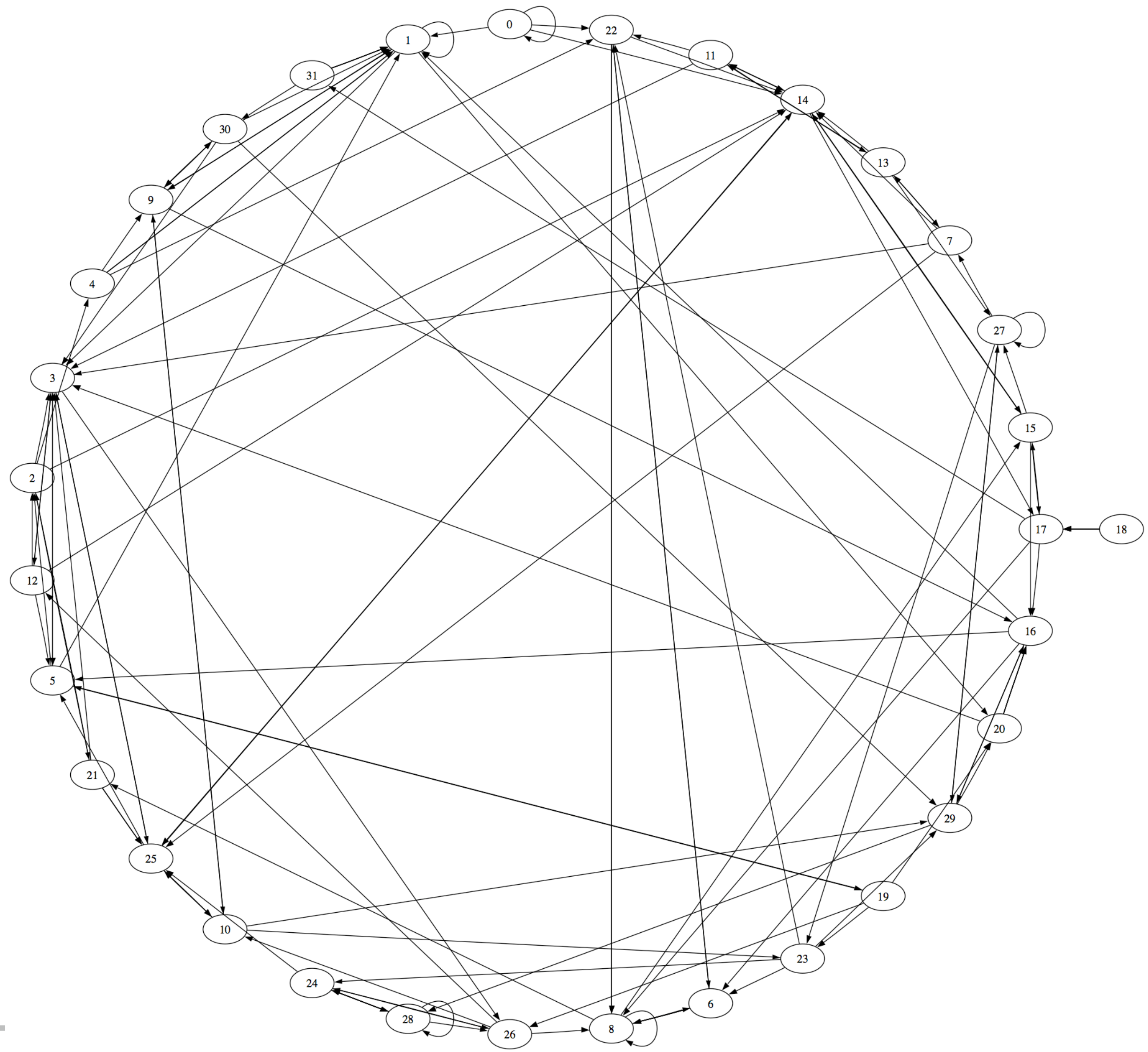


10 Iterations Pull ...



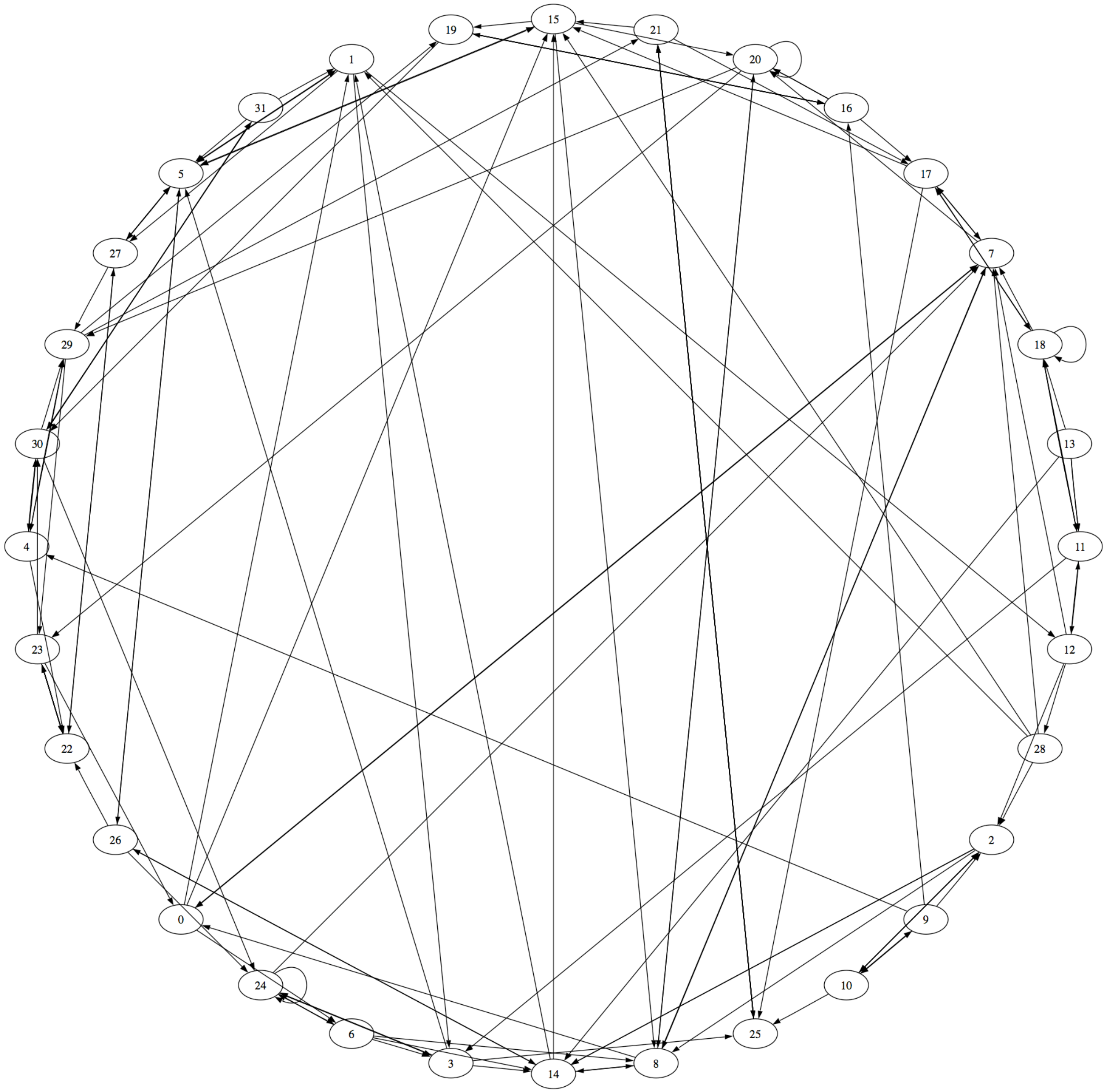
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

20 Iterations Pull ...



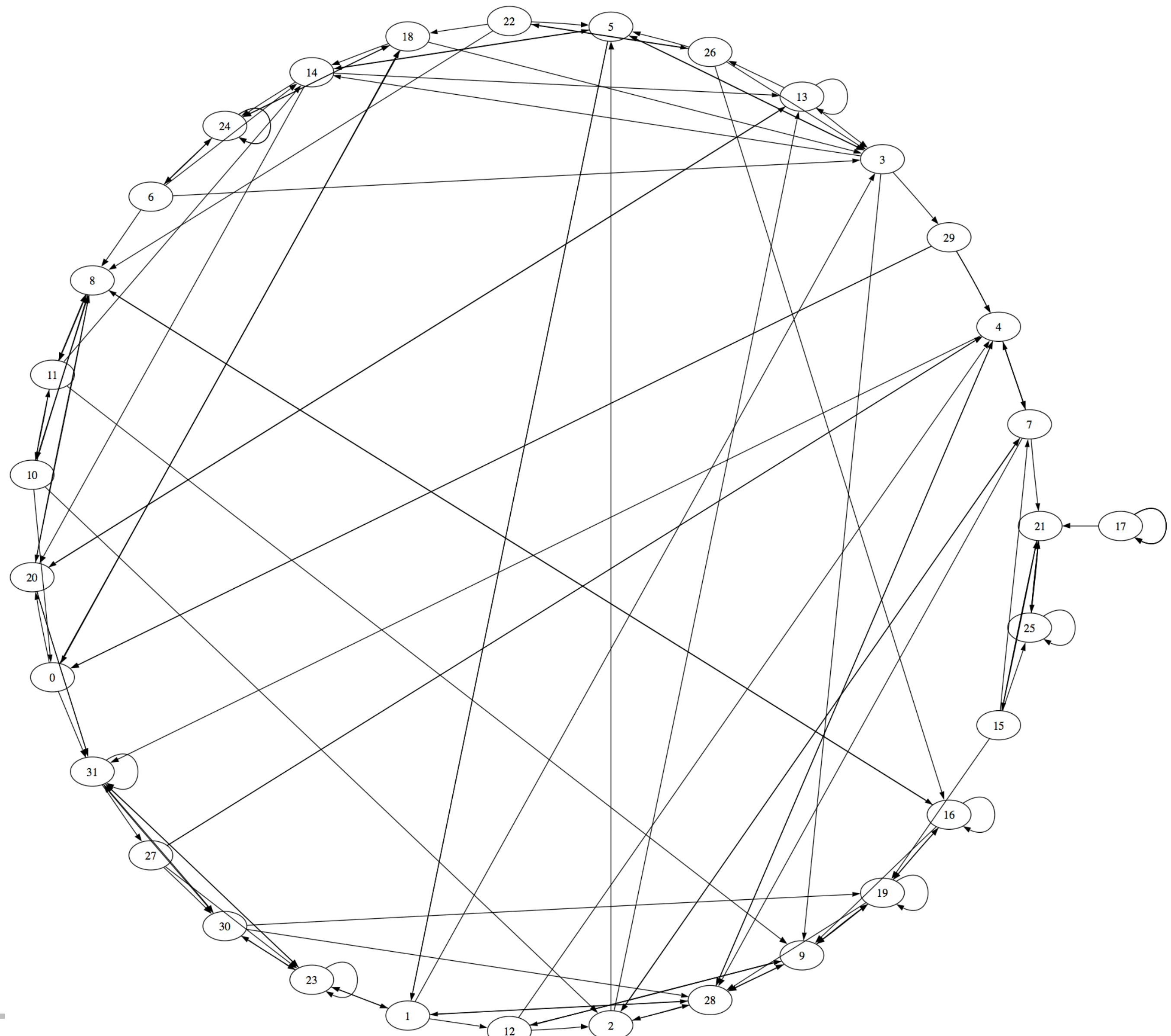
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

30 Iterations Pull ...



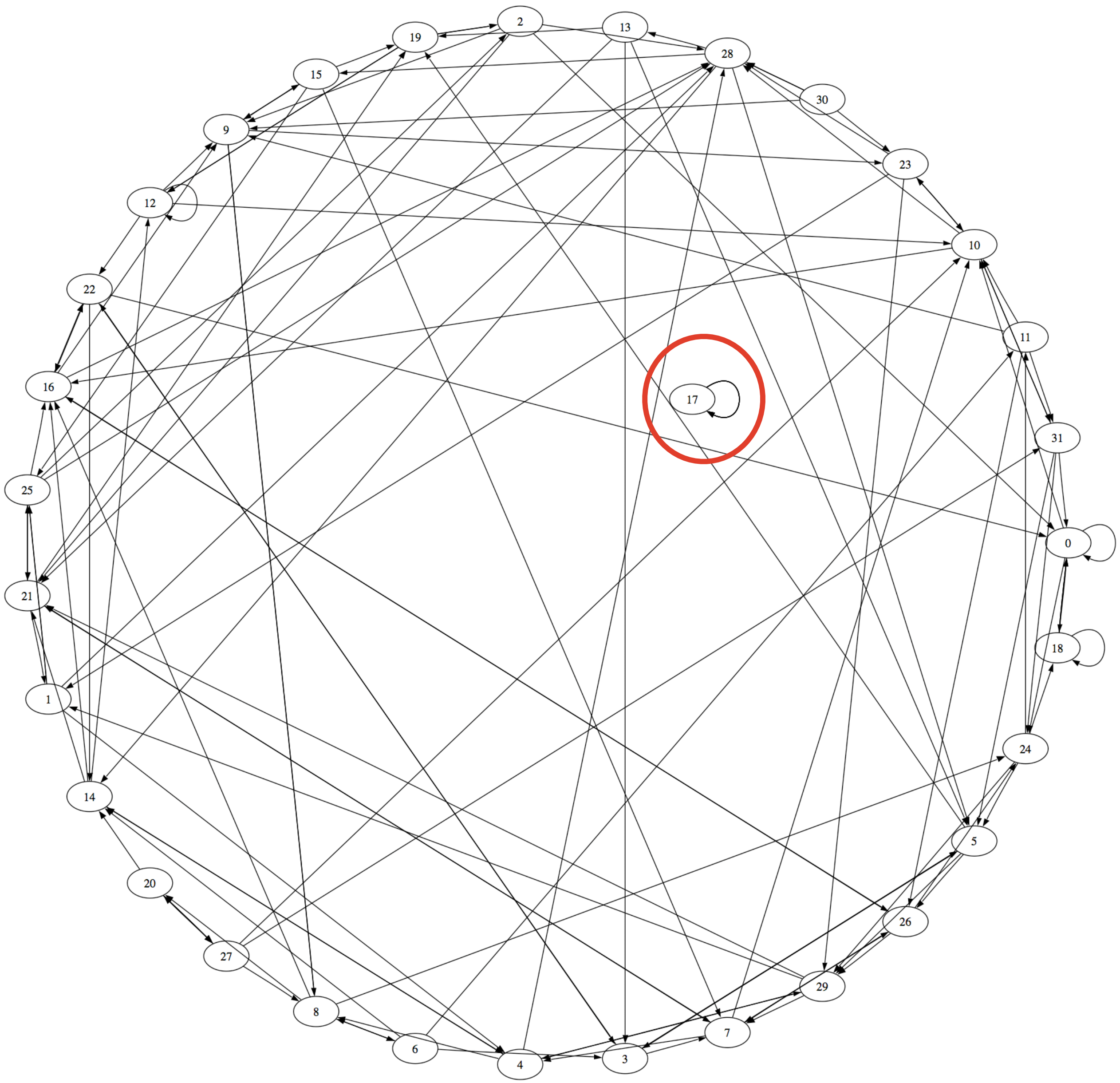
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

40 Iterationen Pull ...



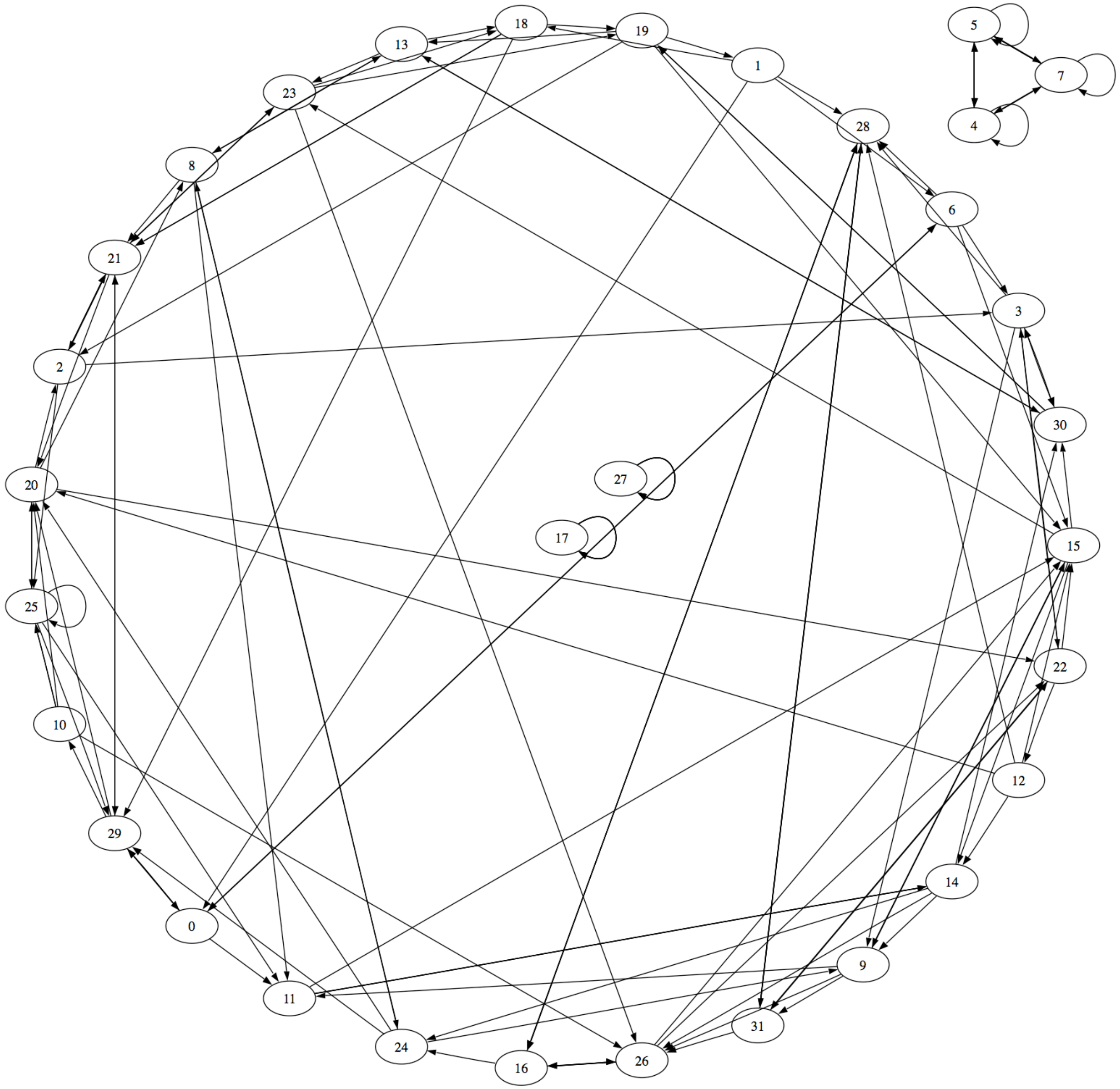
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

50 Iterations Pull ...



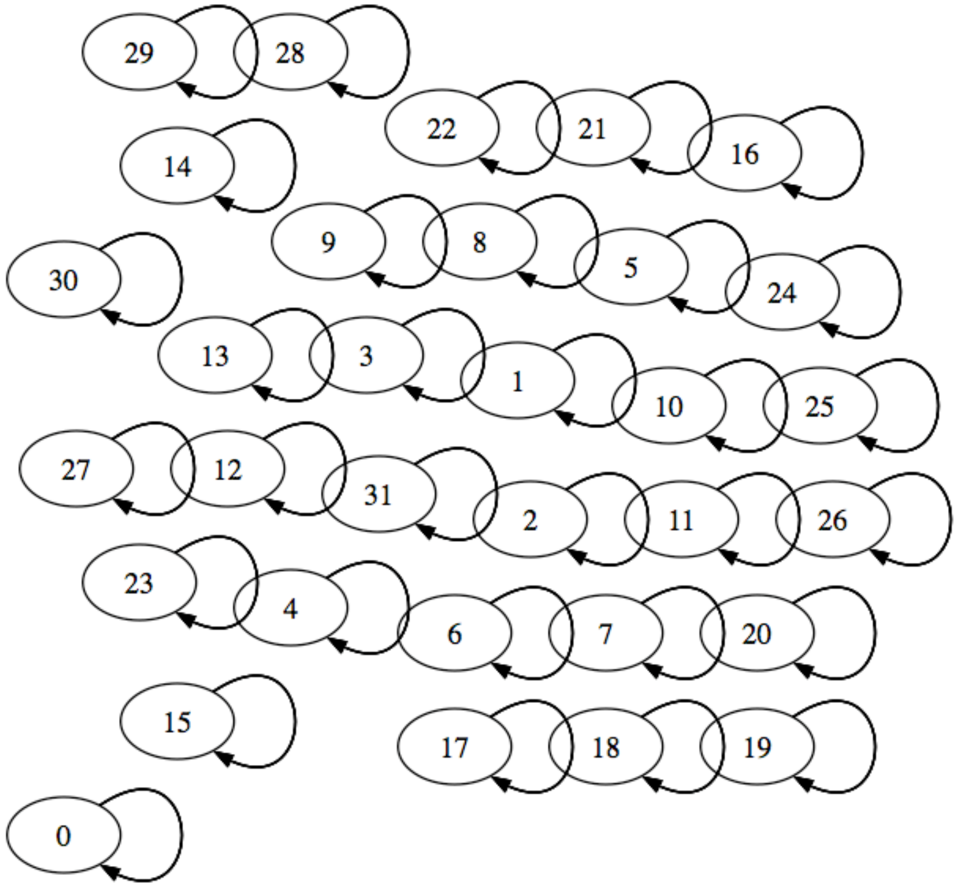
500 Iterations Pull ...

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



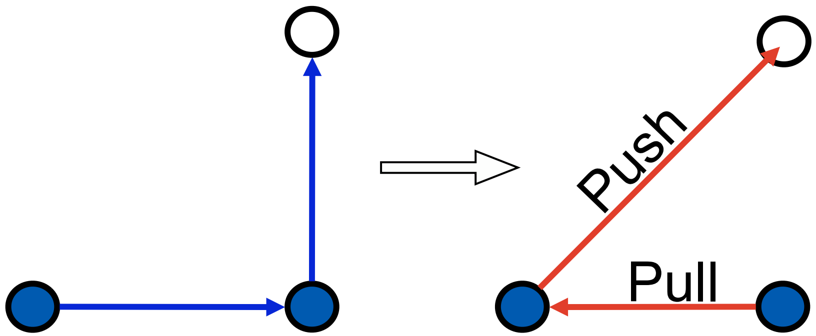
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

5000 Iterations Pull ...



Combination of Push and Pull

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



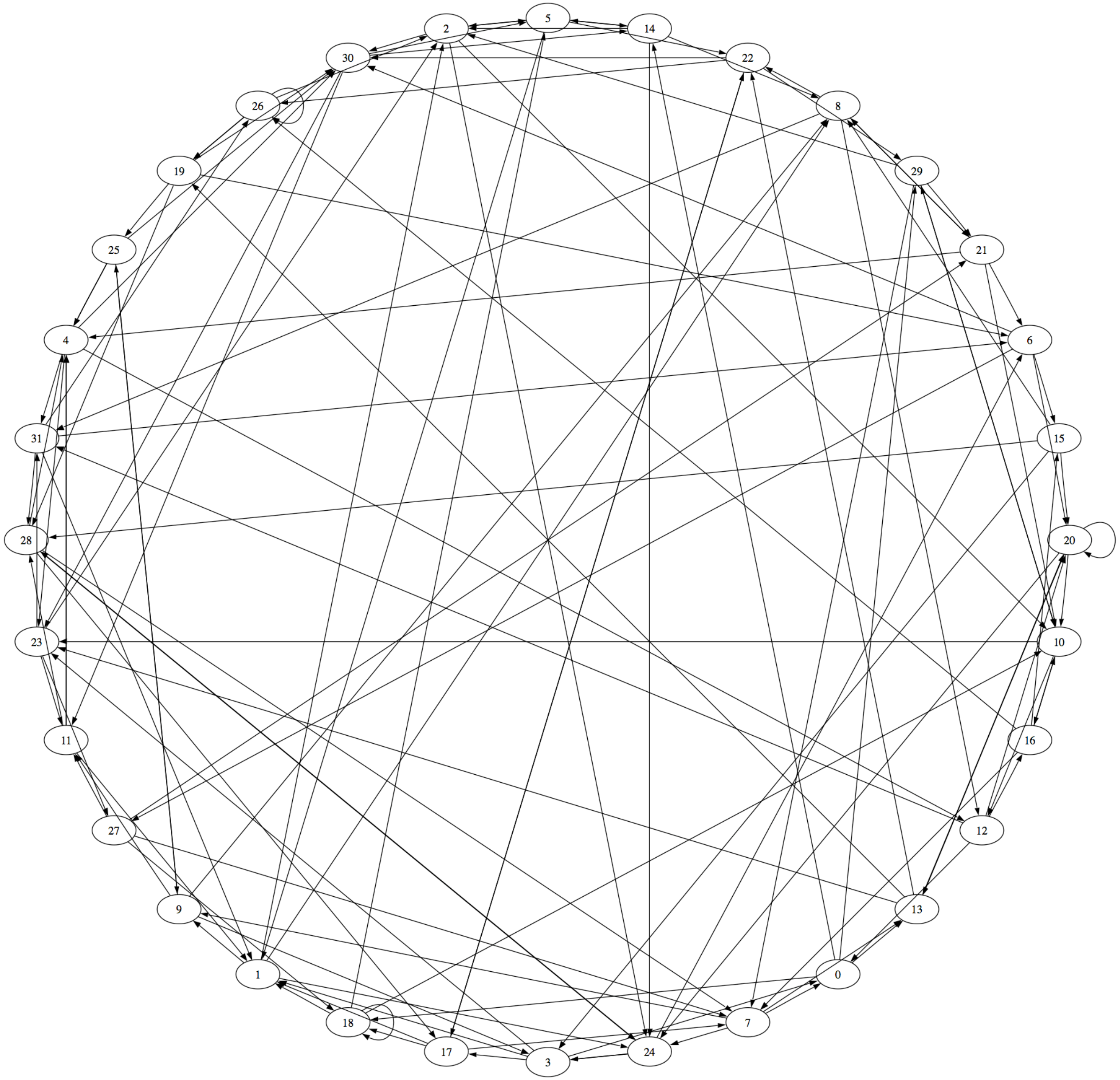
Simulation of Push&Pull-Operations ...

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Same start situation

Parameters
n = 32 nodes
degree d = 4
hop-distance h = 3

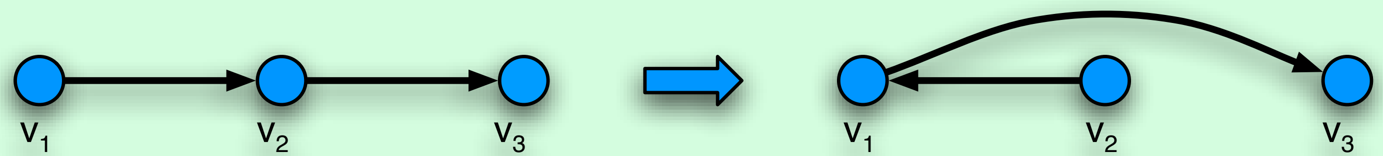
but
1.000.000 iterations



Pointer-Push&Pull for Multi-Digraphs

Pointer-Push&Pull:

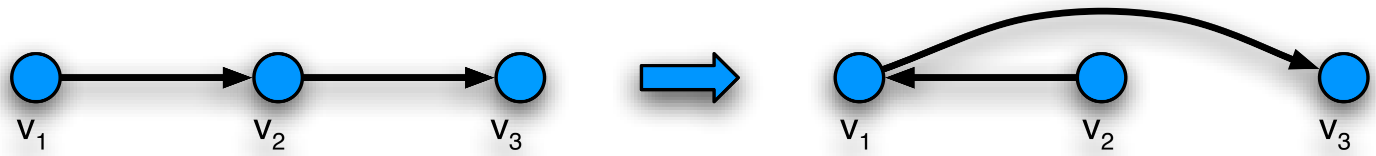
- choose random node $v_1 \in V$
- do random walk v_1, v_2, v_3
- delete edges (v_1,v_2) and (v_2,v_3)
- add edges (v_2,v_1) and (v_1,v_3)



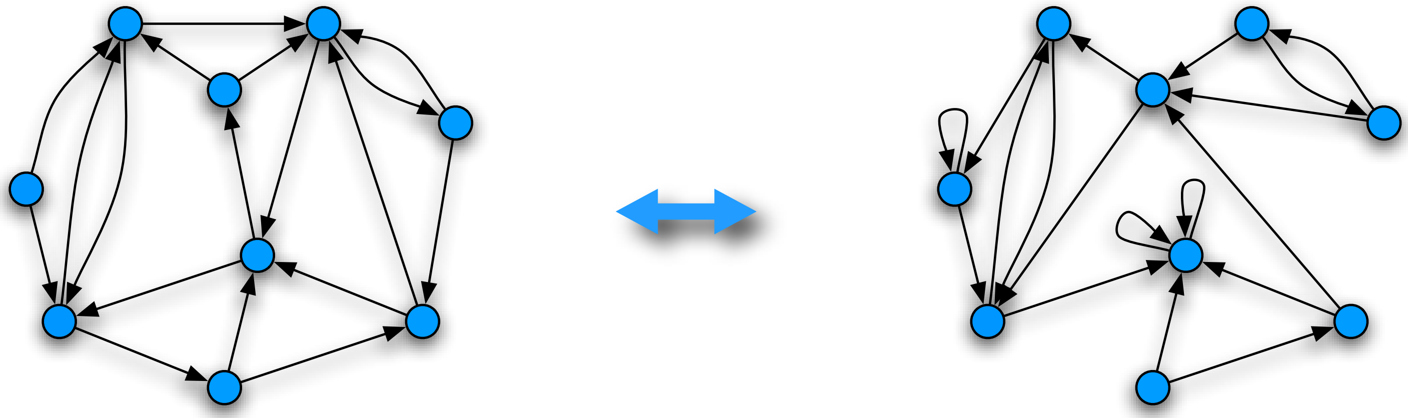
- obviously:
  - preserves connectivity of G
  - does not change out-degrees

➡ Pointer-Push&Pull is **sound** for the domain of out-regular connected multi-digraphs

Pointer-Push&Pull: Reachability

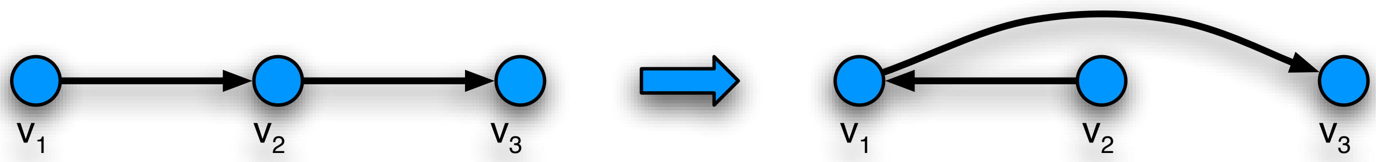


Lemma *A series of random Pointer-Push&Pull operations can transform an arbitrary connected out-regular multi-digraph, to every other graph within this domain*



Computer Networks and Telematics
University of Freiburg
Christian Schindelhauer

Pointer-Push&Pull: Uniformity



What is the stationary prob. distribution generated by Pointer-Push&Pull?

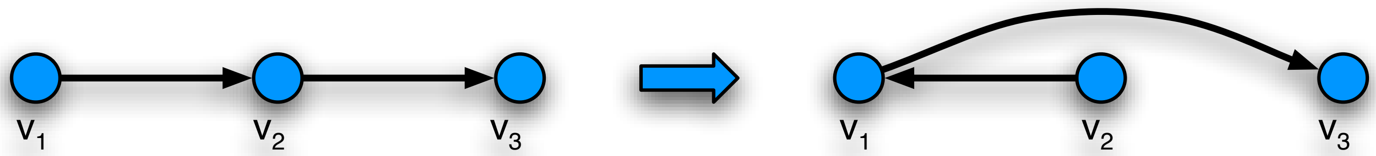
- depends on random walk

example: ***node oriented random walk***

- choose random neighboring node with $p=1/d$ respectively
- due to multi-edges possibly less than d neighbors
- if no node was chosen operation is canceled

$$P[G \xrightarrow{\mathcal{PP}} G'] = P[G' \xrightarrow{\mathcal{PP}} G]$$

Uniform Generality

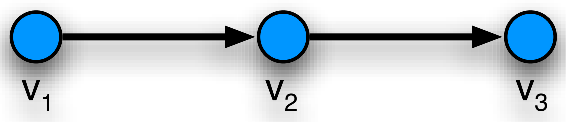


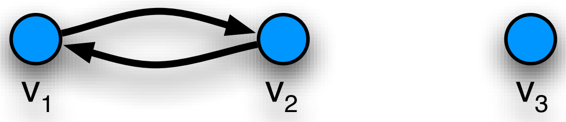
Theorem: *Let G′ be a d-out-regular connected multi-digraph with n nodes. Applying Pointer-Push&Pull operations repeatedly will construct every d-out-regular connected multi-digraph with the same probability in the limit, i.e.*

$$\lim_{t \to \infty} P[G' \xrightarrow{t} G] = \frac{1}{|\mathcal{MDG}_{n,d}|}$$

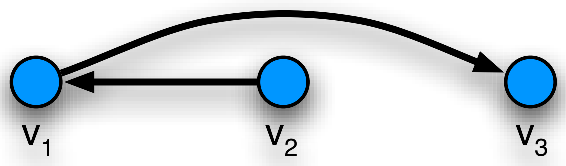
Feasibility ...

A Pointer-Push&Pull operation in the network ...





(2) v_2 replaces (v_2,v_3) by (v_2,v_1) and sends ID of v_3 to v_1



- only 2 messages between two nodes, carrying the information of one edge only
- verification of neighborhood is mandatory in dynamic networks

⇒ combine neighbor-check with Pointer-Push&Pull

Properties of Pointer-Push&Pull

	Pointer-Push&Pull
Graphs	Directed Multigraphs
Soundness	✔
Generality	✔
Feasibility	✔
Convergence	?

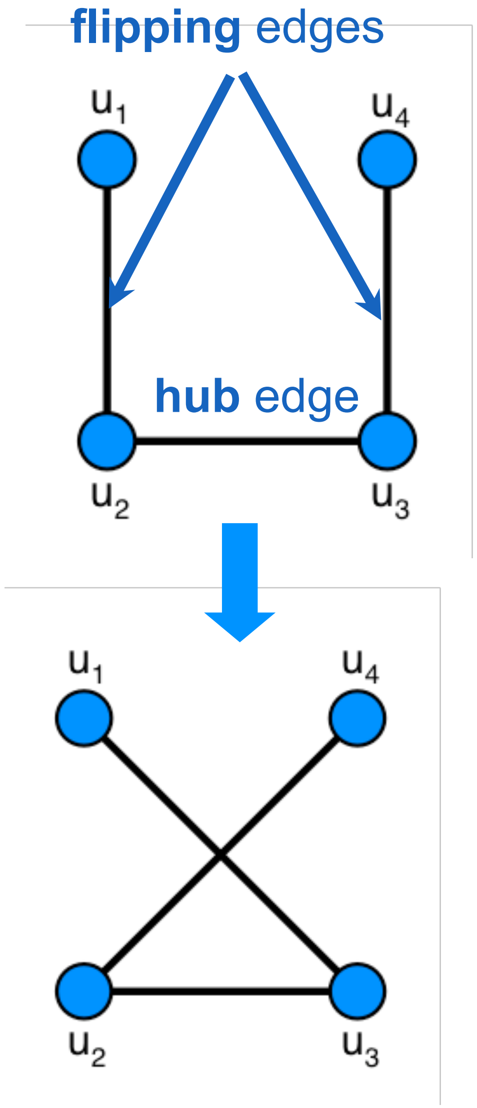
- strength of Pointer-Push&Pull is its **simplicity**
- generates truly random digraphs
- the price you have to pay: multi-edges

Open Problems:

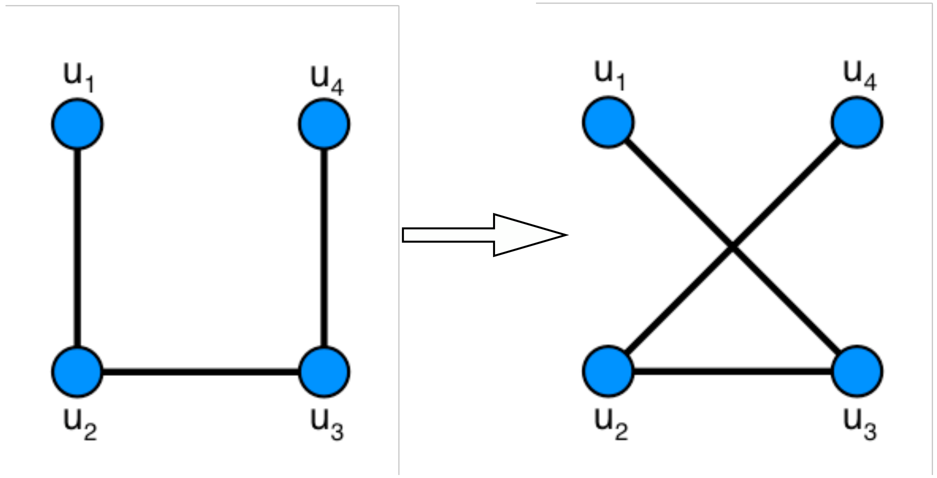
- convergence rate is unknown, conjecture $O(dn \log n)$
- is there a similar operation for simple digraphs?

The 1-Flipper (F^1)

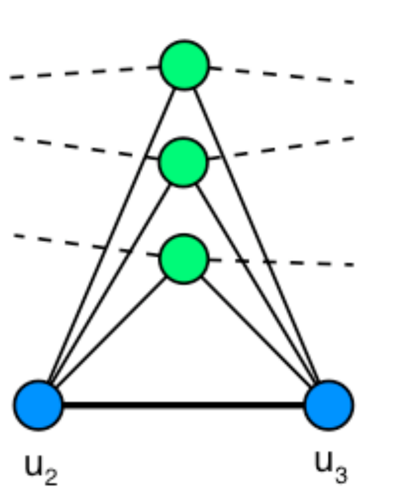
- **The operation**
  - choose random edge $\{u_2,u_3\} \in E$,
    - hub edge
  - choose random node $u_1 \in N(u_2)$
    - 1st flipping edge
  - choose random node $u_4 \in N(u_3)$
    - 2nd flipping edge
  - if $\{u_1,u_3\}, \{u_2,u_4\} \notin E$
    - flip edges, i.e.
    - add edges $\{u_1,u_3\}, \{u_2,u_4\}$ to E
    - remove $\{u_1,u_2\}$ and $\{u_3,u_4\}$ from E



1-Flipper is sound



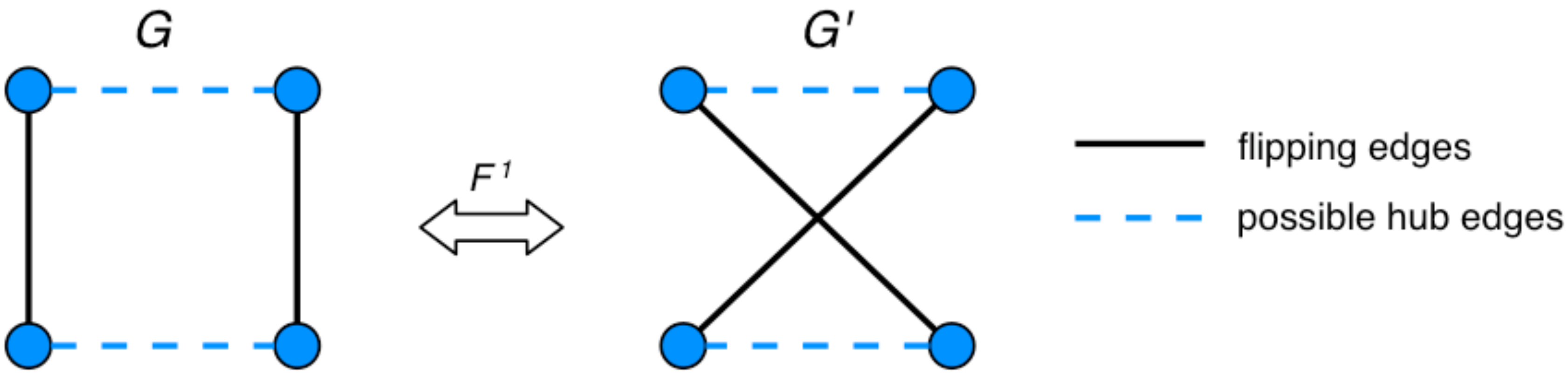
- **Soundness:**
  - 1-Flipper preserves d-regularity
    - follows from the definition
  - 1-Flipper preserves connectivity
    - because of the hub edge
- **Observation:**
  - For all d > 2 there is a connected d-regular graph G such that $P[G \xrightarrow{F^1} G] \neq 0$
  - For all d ≥ 2 and for all d-regular connected graphs at least one 1-Flipper-operation changes the graph with positive probability
    - This does not imply generality



1-Flipper is symmetric

- **Lemma (symmetry):**
  - For all undirected regular graphs G,G′:

$$P[G \xrightarrow{F^1} G'] = P[G' \xrightarrow{F^1} G]$$



Computer Networks and Telematics
University of Freiburg
Christian Schindelhauer

1-Flipper provides generality

- **Lemma (reachability):**
  - For all pairs G,G′ of connected d-regular graphs there exists a sequence of 1-Flipper operations transforming G into G′.

Computer Networks and Telematics
University of Freiburg
Christian Schindelhauer

1-Flipper properties: uniformity

- **Theorem (uniformity):**
  - Let G_0 be a d-regular connected graph with n nodes and d > 2. Then in the limit the 1-Flipper operation constructs all connected d-regular graphs with the same probability:

$$\lim_{t \to \infty} P[G_0 \xrightarrow{t} G] = \frac{1}{|\mathcal{C}_{n,d}|}$$

Computer Networks and Telematics
University of Freiburg
Christian Schindelhauer

1-Flipper properties: Expansion

- **Definition (edge boundary):**
  - The edge boundary δS of a set $S \subset V$ is the set of edges with exactly one endpoint in S.
- **Definition (expansion):**
  A graph G=(V,E) has expansion $\beta > 0$
  - if for all node sets S with $|S| \leq |V|/2$:
  - $|\delta S| \geq \beta\ |S|$
- **Since for d $\in \omega(1)$ a random connected d-regular graph is a $\theta(d)$ expander asymptotically almost surely (a.a.s: in the limit with probability 1), we have**
- **Theorem:**
  - For d > 2 consider any d-regular connected Graph G_0. Then in the limit the 1-Flipper operation establishes an expander graph after a sufficiently large number of applications a.a.s.

Computer Networks and Telematics
University of Freiburg
Christian Schindelhauer

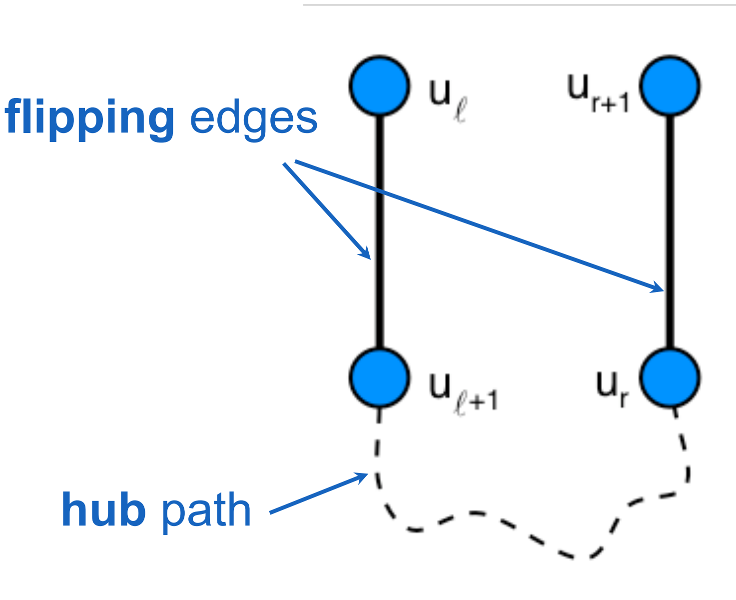
Flipper

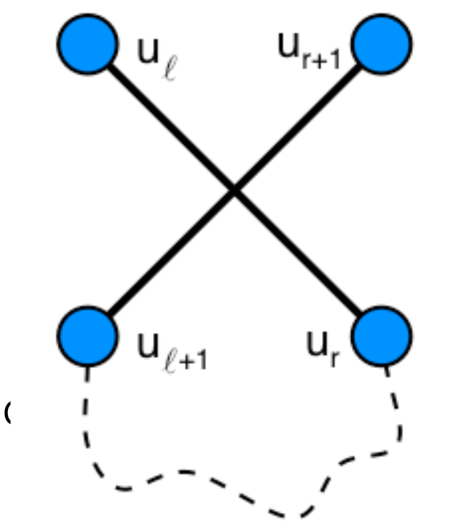
	Flipper
Graphs	Undirected Graphs
Soundness	✔
Generality	✔
Feasibility	✔
Convergence	?

- **Flipper involves 4 nodes**
- **Generates truly random graphs**
- **Open Problems:**
  - convergence rate is unknown, conjecture $O(dn \log n)$

The k-Flipper (F^k)

- **The operation**
  - choose random node
  - random walk P′ in G
  - choose hub path with nodes
    - $\{u_l, u_r\}, \{u_{l+1}, u_{r+1}\}$ occur only once in P′
  - if $\{u_l, u_r\}, \{u_{l+1}, u_{r+1}\} \notin E$
    - add edges $\{u_l, u_r\}, \{u_{l+1}, u_{r+1}\}$ to E
    - remove $\{u_l, u_{l+1}\}$ and $\{u_r, u_{r+1}\}$ from E





k-Flipper: Properties ...

- **k-Flipper preserves connectivity and d-regularity**
  - proof analogously to the 1-Flipper
- **k-Flipper provides reachable,**
  - since the 1-Flipper provides reachability
  - k-Flipper can emulate 1-Flipper
- **But: k-Flipper is not symmetric:**
  - a new proof for expansion property is needed

Computer Networks and Telematics
University of Freiburg
Christian Schindelhauer

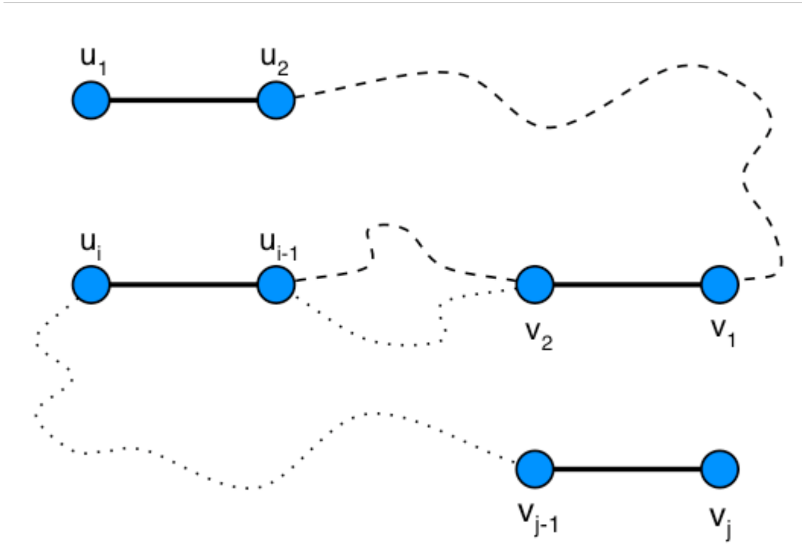
k-Flipper: Expanders ...

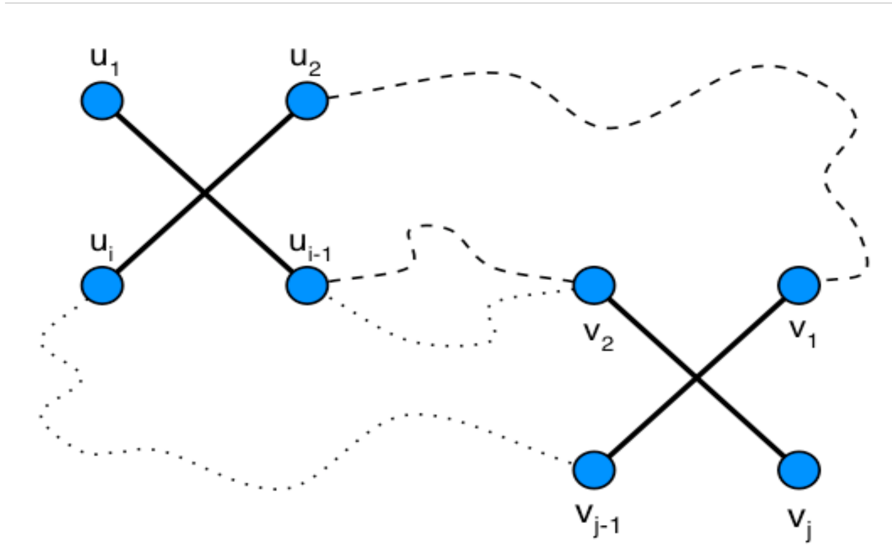
- **Theorem:**
  - For $d \in \Omega(\log n)$ applying $O(dn)$ Random $\Theta(d^2 n^2 \log 1/\epsilon)$ -Flipper operations transforms any given connected d-regular graph into a connected d-regular graph with expansion $\Theta(d)$ w.h.p..

Computer Networks and Telematics
University of Freiburg
Christian Schindelhauer

Concurrency ...

- **In a P2P-network there are concurrent Flipper operations**
  - No central coordination
  - Concurrent Flipper operations can speed up the convergence process
  - However concurrent Flipper operations can disconnect the network





k-Flipper

	k-Flipper large k	k-Flipper small k
Graphs	Undirected Graphs	Undirected Graphs
Soundness	✔	✔
Generality	✔	✔
Feasibility	↯	✔
Convergence	✔	?

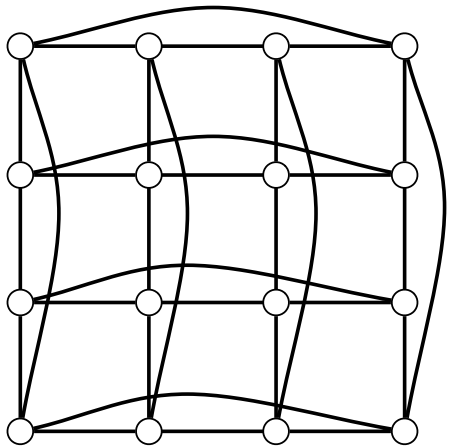
- **Convergence only proven for too long paths**
  - Operation is not feasible then.
  - Does k-Flipper quickly converge for small k?
- **Open problem:**
  - Which k is optimal?

All Graph Transformation

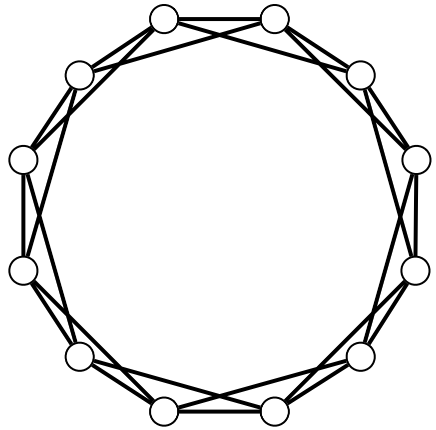
	Simple-Switching	Flipper	Pointer-Push&Pull	k-Flipper small k	k-Flipper large k
Graphs	Undirected Graphs	Undirected Graphs	Directed Multigraphs	Undirected Graphs	Undirected Graphs
Soundness	?	✔	✔	✔	✔
Generality	↘	✔	✔	✔	✔
Feasibility	✔	✔	✔	✔	↘
Conver-gence	✔	?	?	?	✔

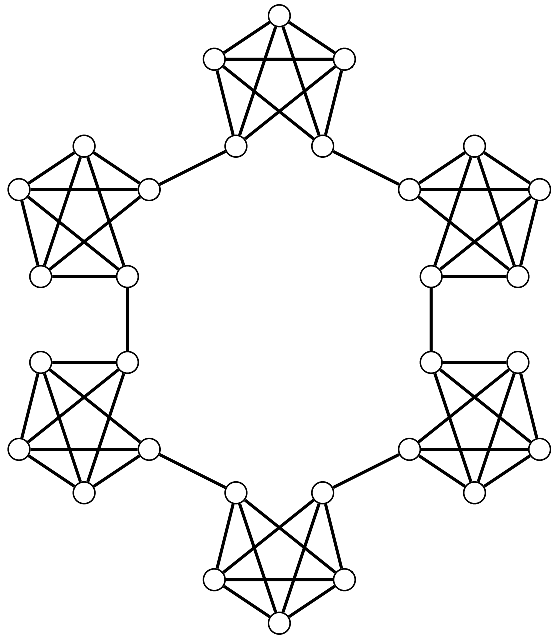
- **Open Problems**
  - Conjecture: Flipper converges in after O(dn log n) operations to a truly random graph
    - best known bound: $O(d^{56}n^{53})$ (Feder et al. 2006)
  - Conjecture: k-Flipper converges faster, but involves more nodes and flags
  - Conjecture: k-Flipper does not pay out
- **Empirical Simulations**
  - Estimate expansion by eingenvalue gap
  - Estimate eigenvalue gap by iterated multiplication of a start vector

Start Graphs

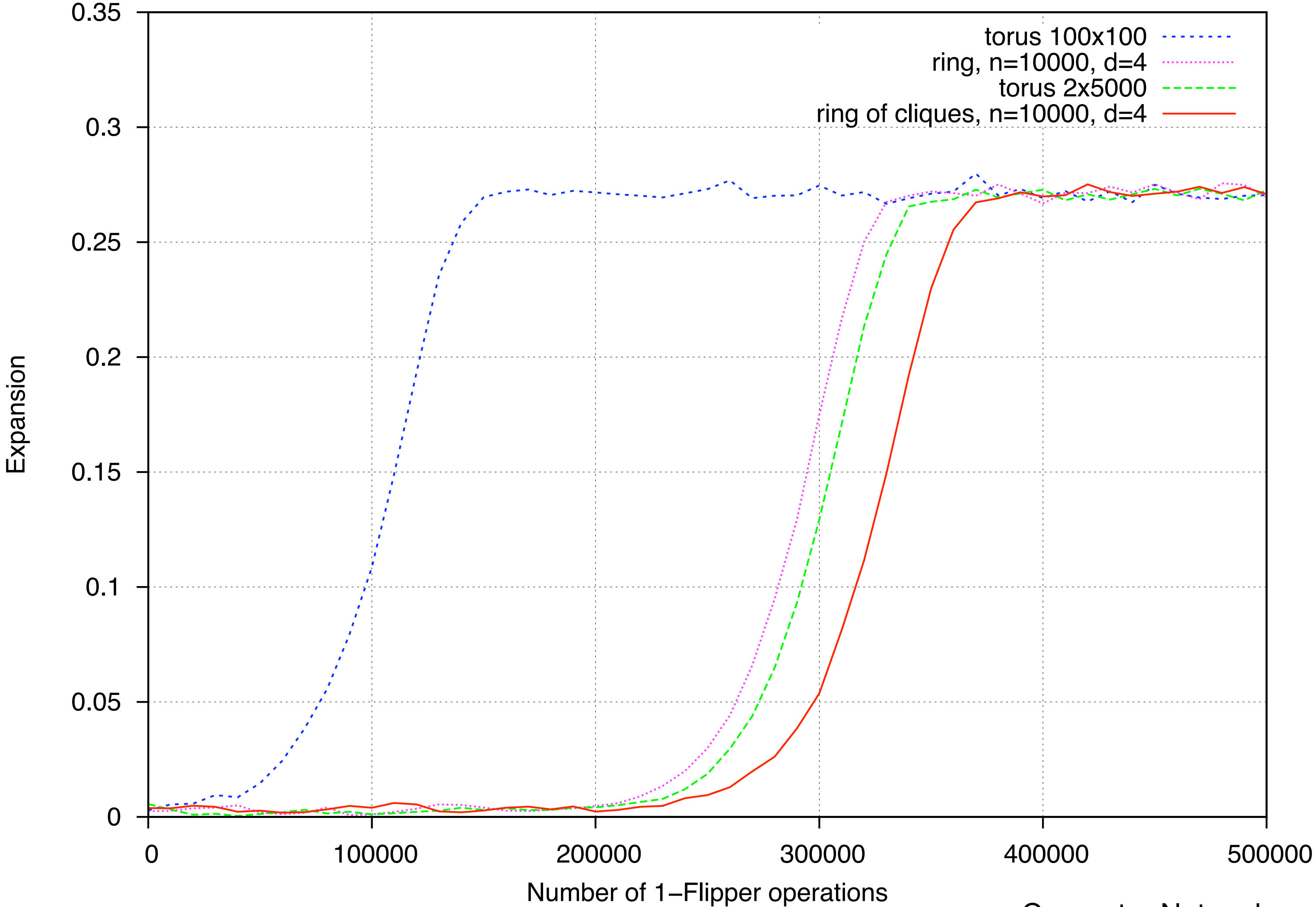


- Ring with neighbor edges
- Torus
- Ring of cliques

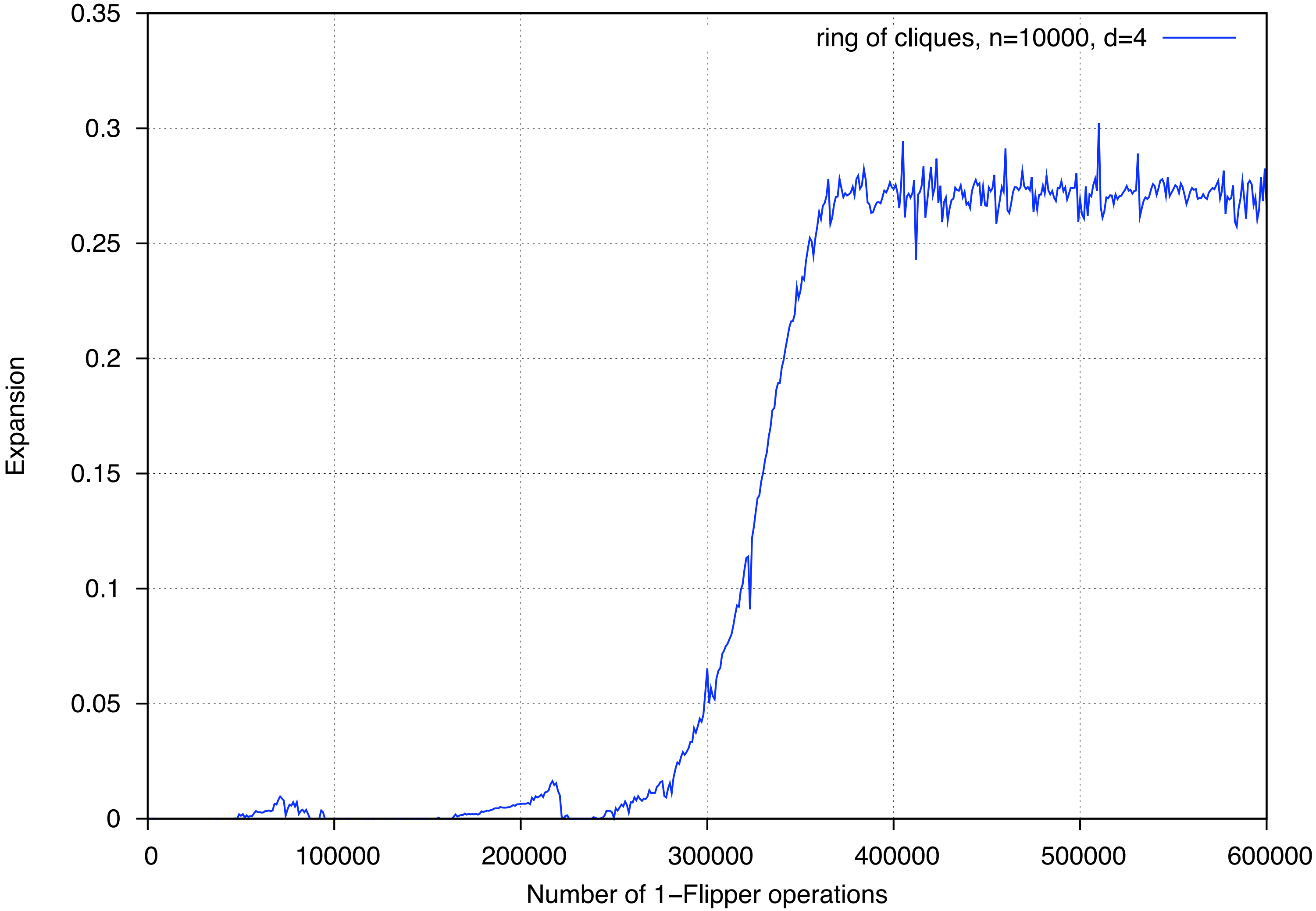




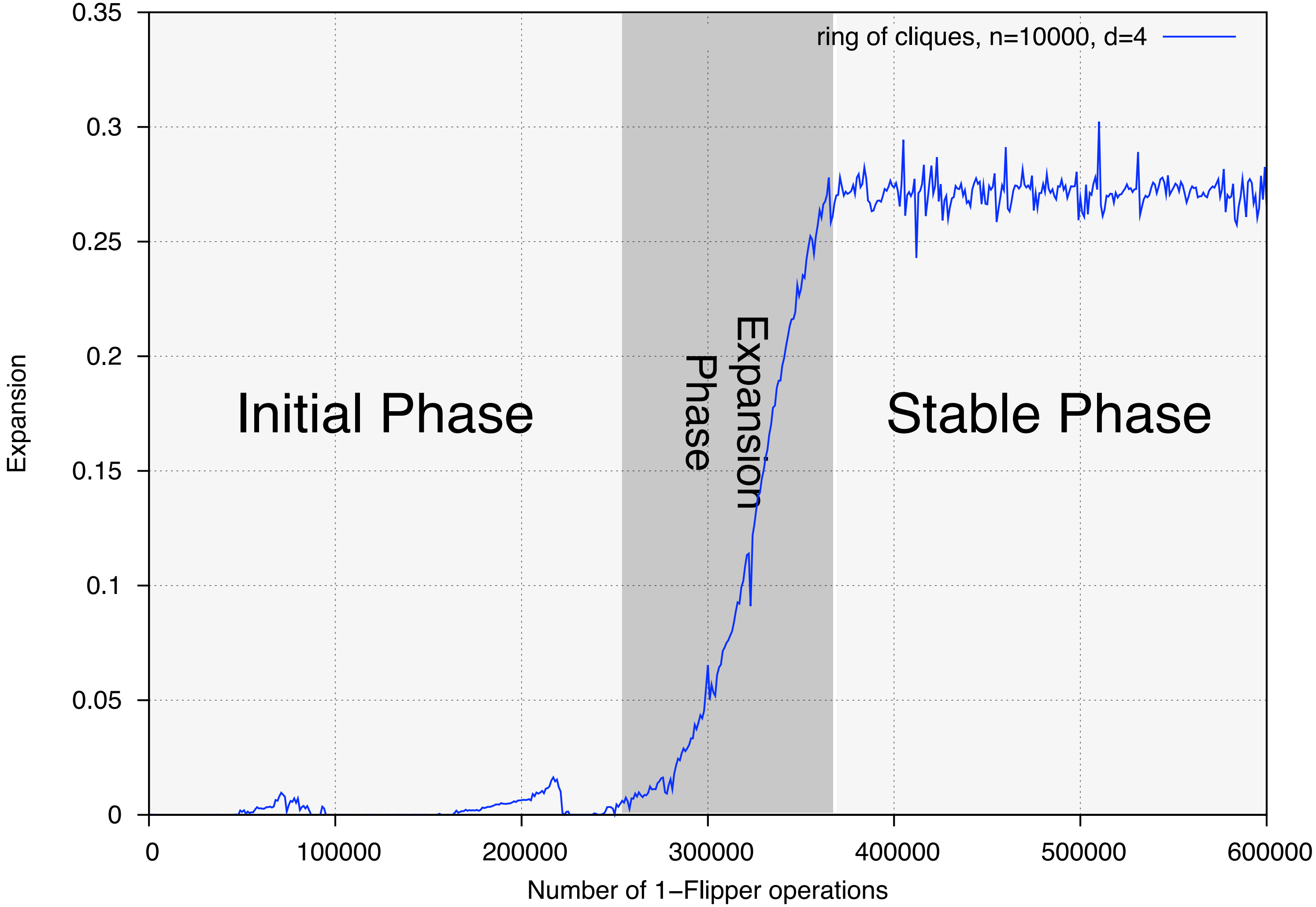
Flipper
Influence of the Start Graph



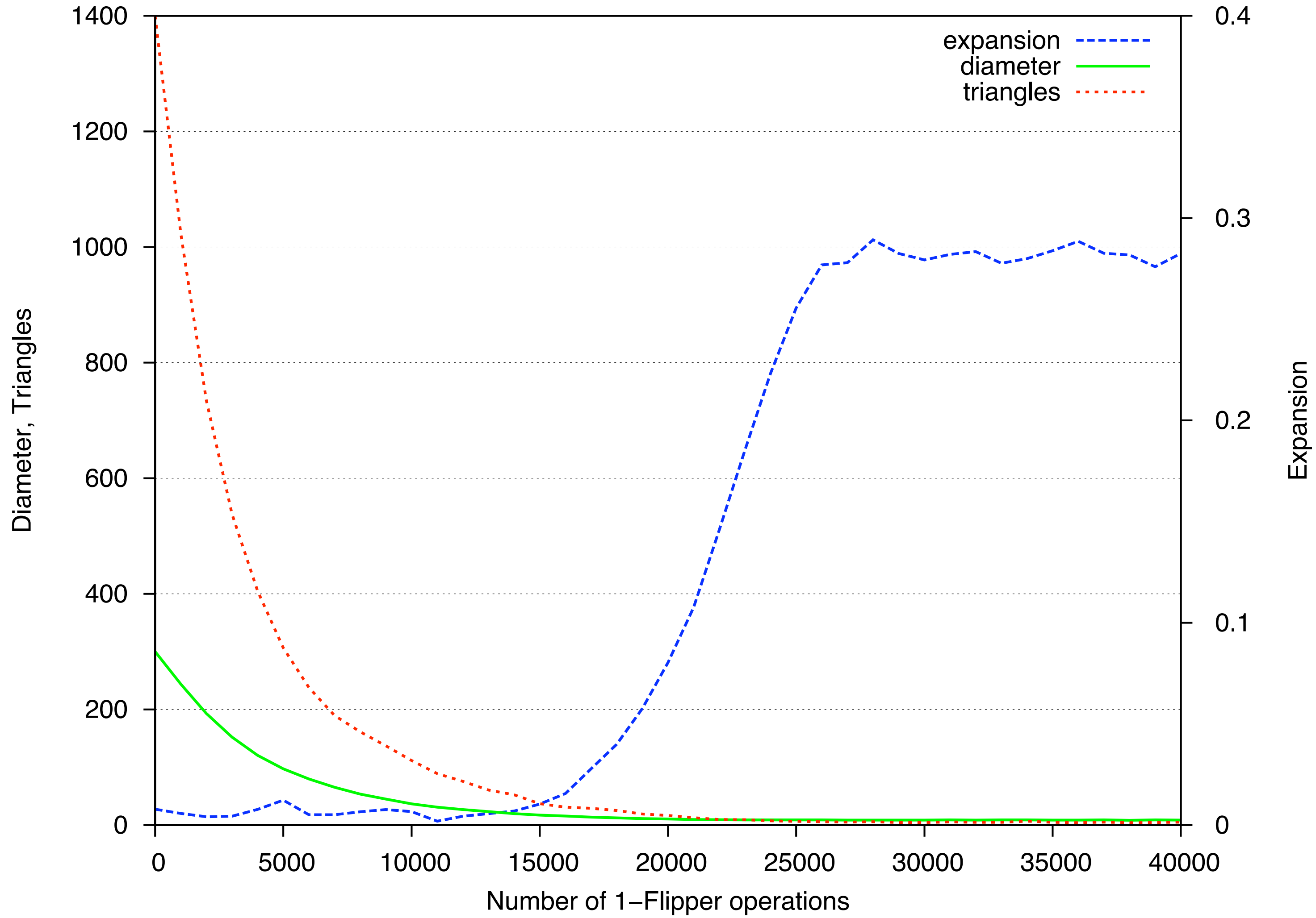
Development of Expansion



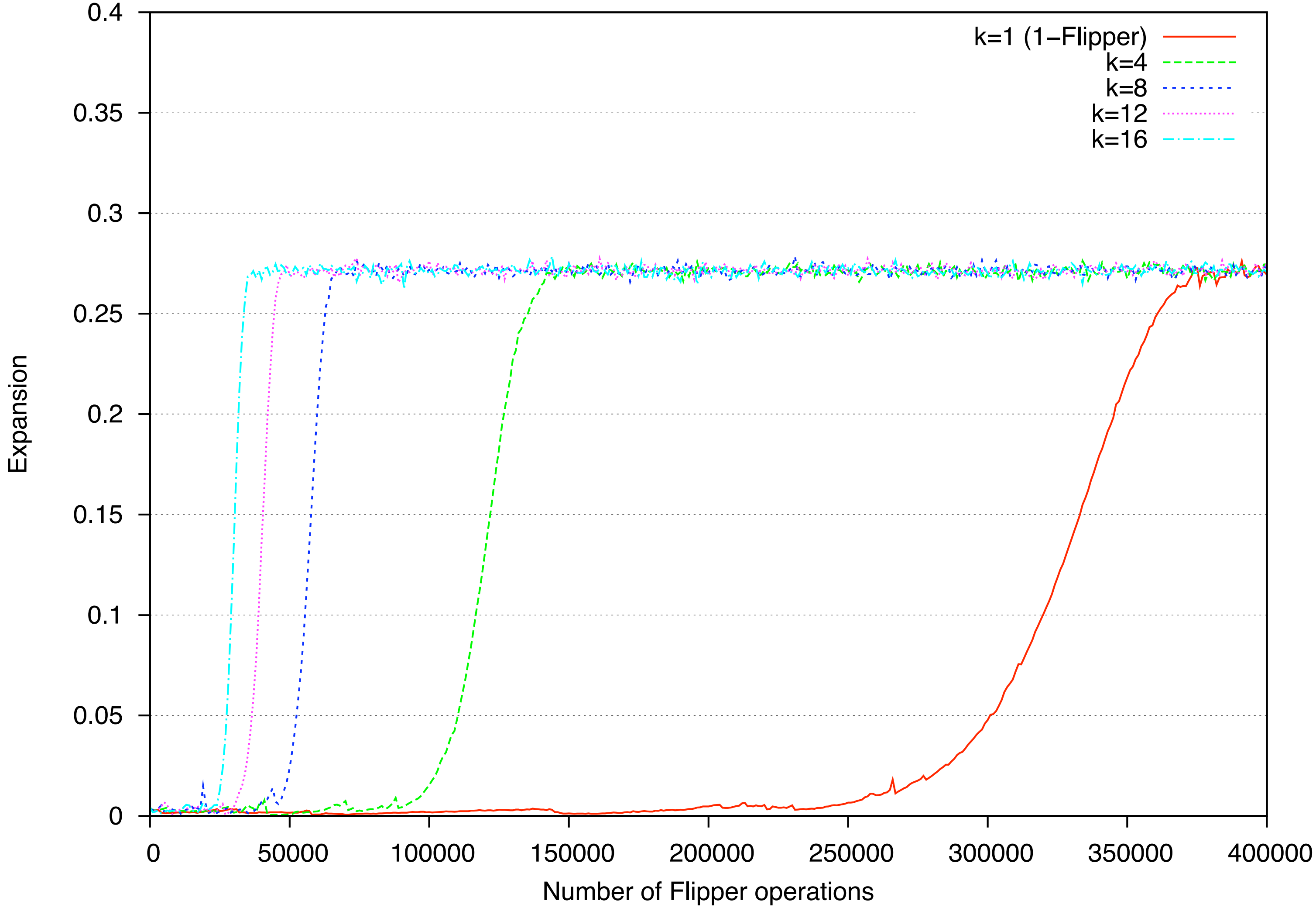
Development of Expansion



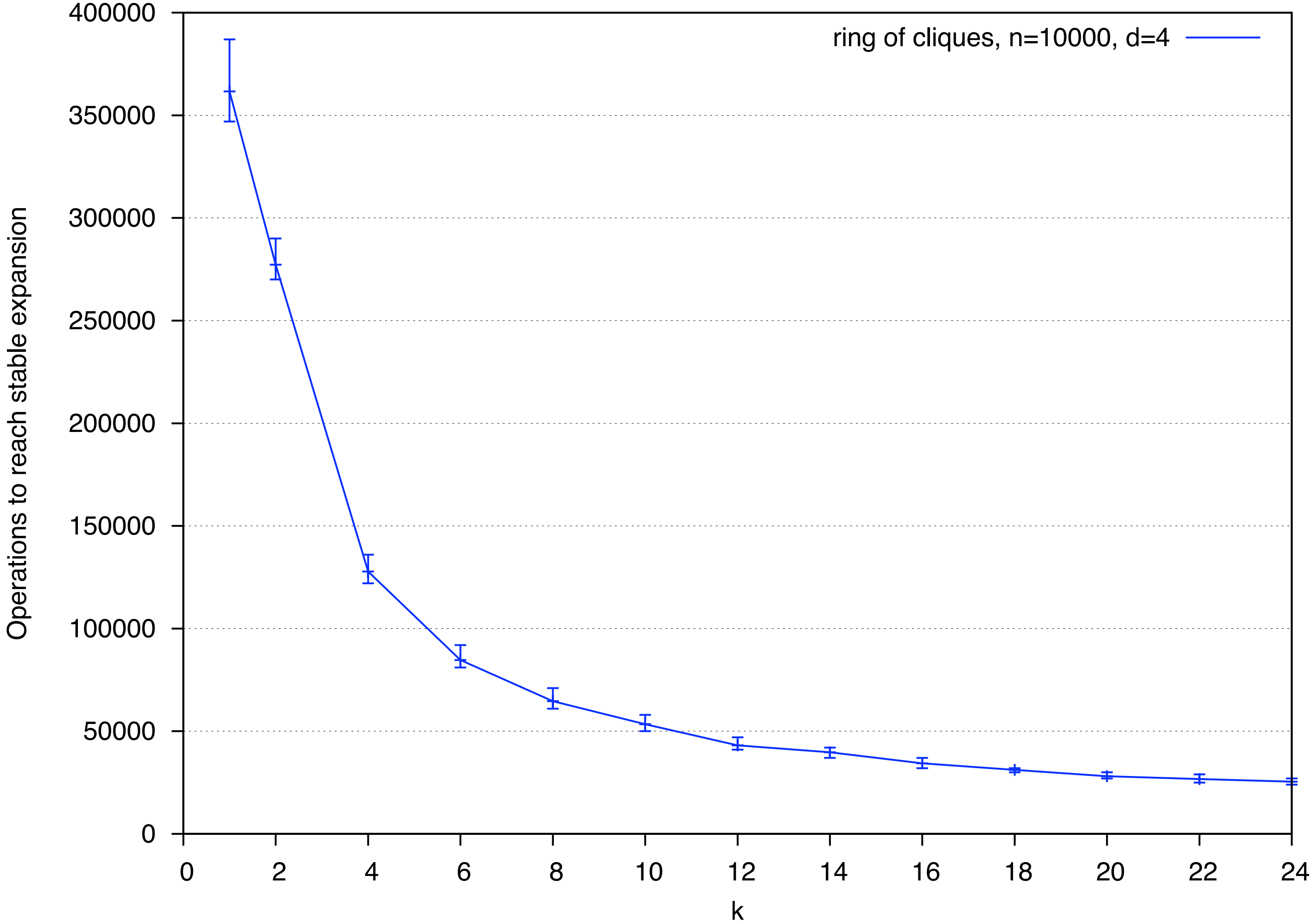
Expansion, Diameter & Triangles



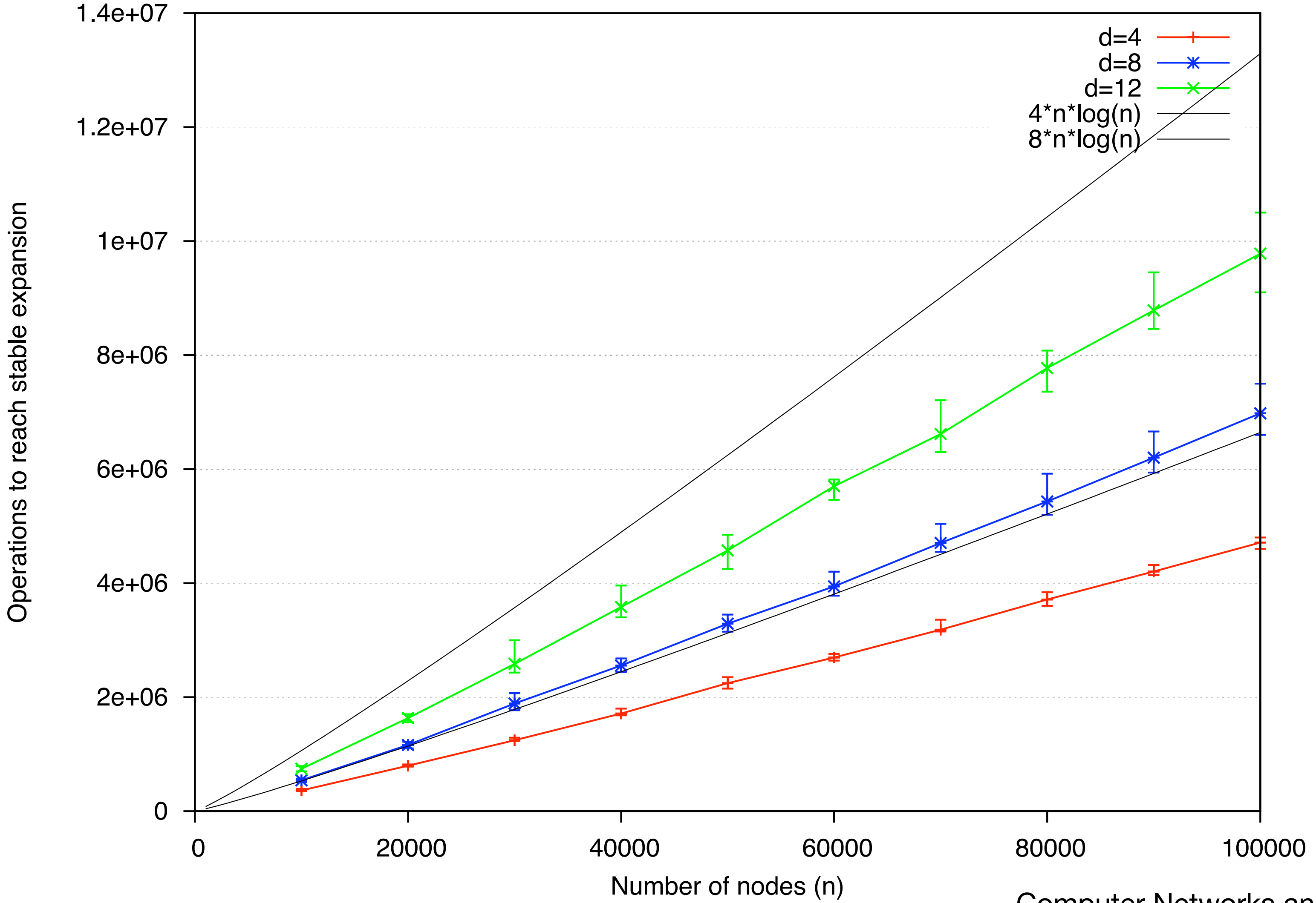
k-Flipper
Start Graph: Ring of Cliques



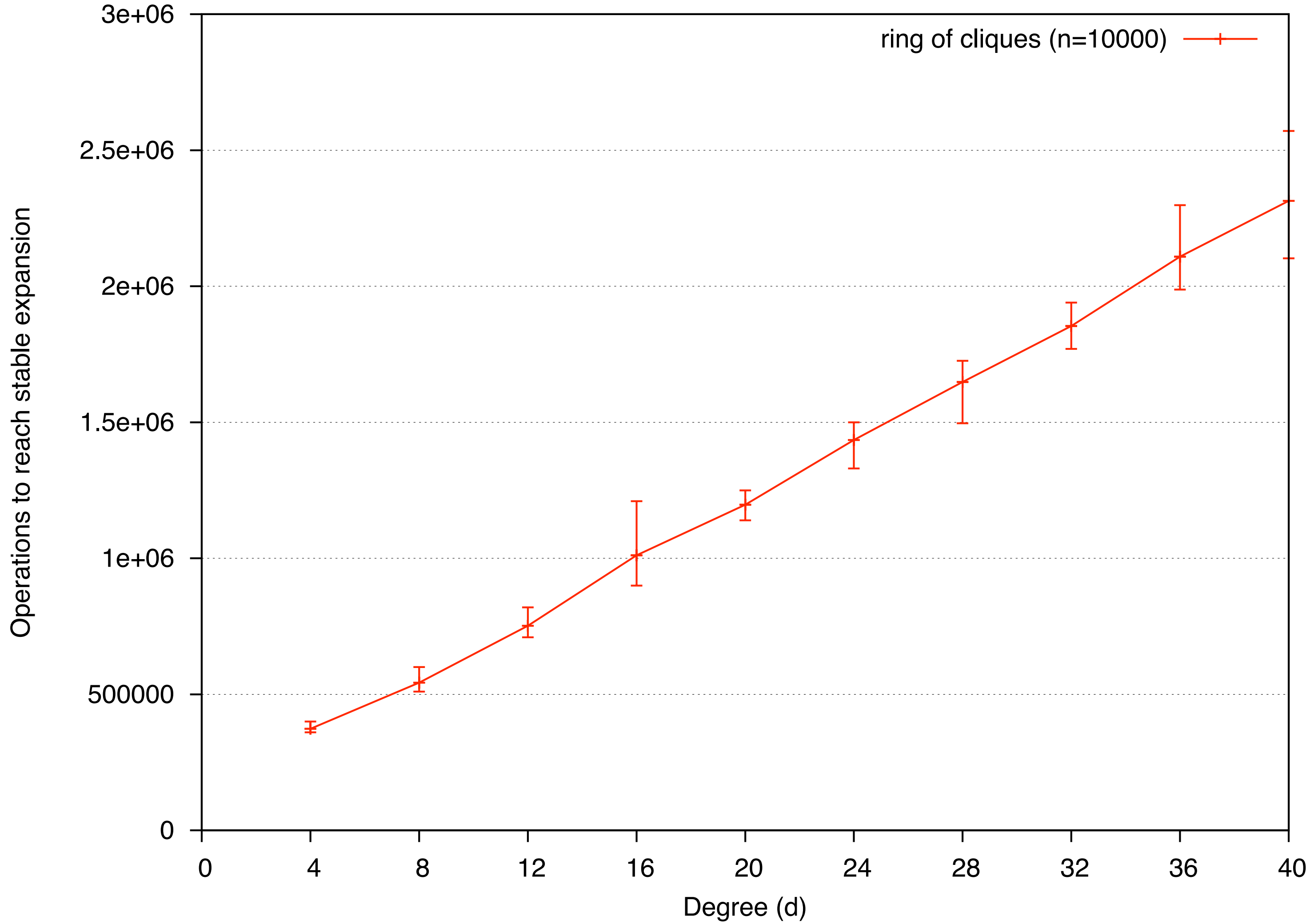
k-Flipper
Start Graph: Ring of Cliques



Convergence of Flipper



Convergence of Flipper Varying Degree

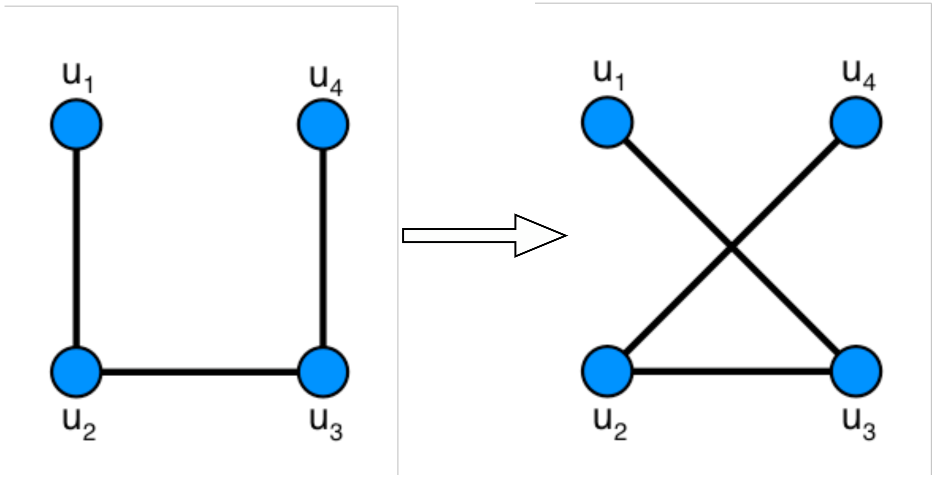


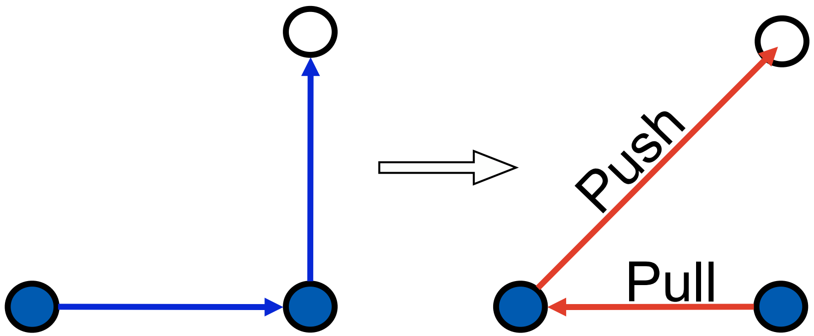
All Graph Transformation

	Simple-Switching	Flipper	Pointer-Push&Pull	k-Flipper small k	k-Flipper large k
Graphs	Undirected Graphs	Undirected Graphs	Directed Multigraphs	Undirected Graphs	Undirected Graphs
Soundness	?	✔	✔	✔	✔
Generality	↯	✔	✔	✔	✔
Feasibility	✔	✔	✔	✔	↯
Convergence	✔	✔	?	✔	✔

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Good Peer-to-Peer-Operations





Topology-Management

- **T-Man: Fast Gossip-based Construction of Large-Scale Overlay Topologies**
  Mark Jelasity Ozalp Babaoglu, 1994

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Verteilte Topologie-Konstruktion

```
do at a random time once in each
consecutive interval of T time units
   p ← selectPeer()
   myDescriptor ← (myAddress,myProfile)
   buffer ← merge(view,{myDescriptor})
   buffer ← merge(buffer,rnd.view)
   send buffer to p
   receive buffer_p from p
   buffer ← merge(buffer_p,view)
   view ← selectView(buffer)
```

(a) active thread

```
do forever
   receive buffer_q from q
   myDescriptor ← (myAddress,myprofile)
   buffer ← merge(view,{myDescriptor})
   buffer ← merge(buffer,rnd.view)
   send buffer to q
   buffer ← merge(buffer_q,view)
   view ← selectView(buffer)
```

(b) passive thread

Fig. 1. The T-MAN protocol.

Finding a Torus

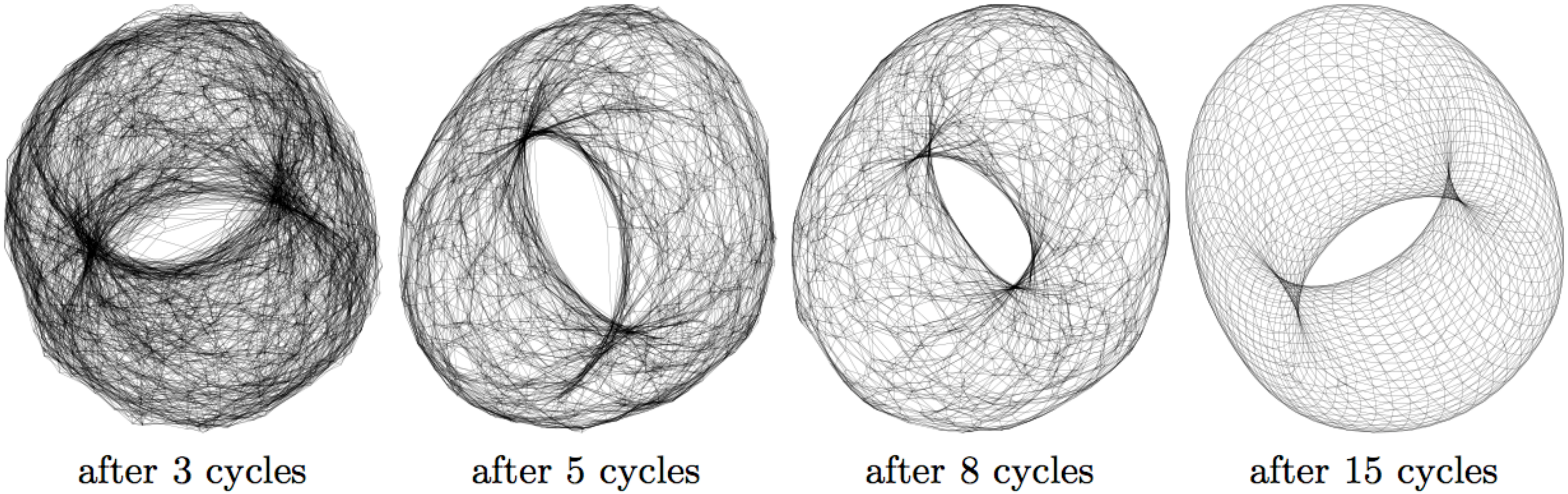
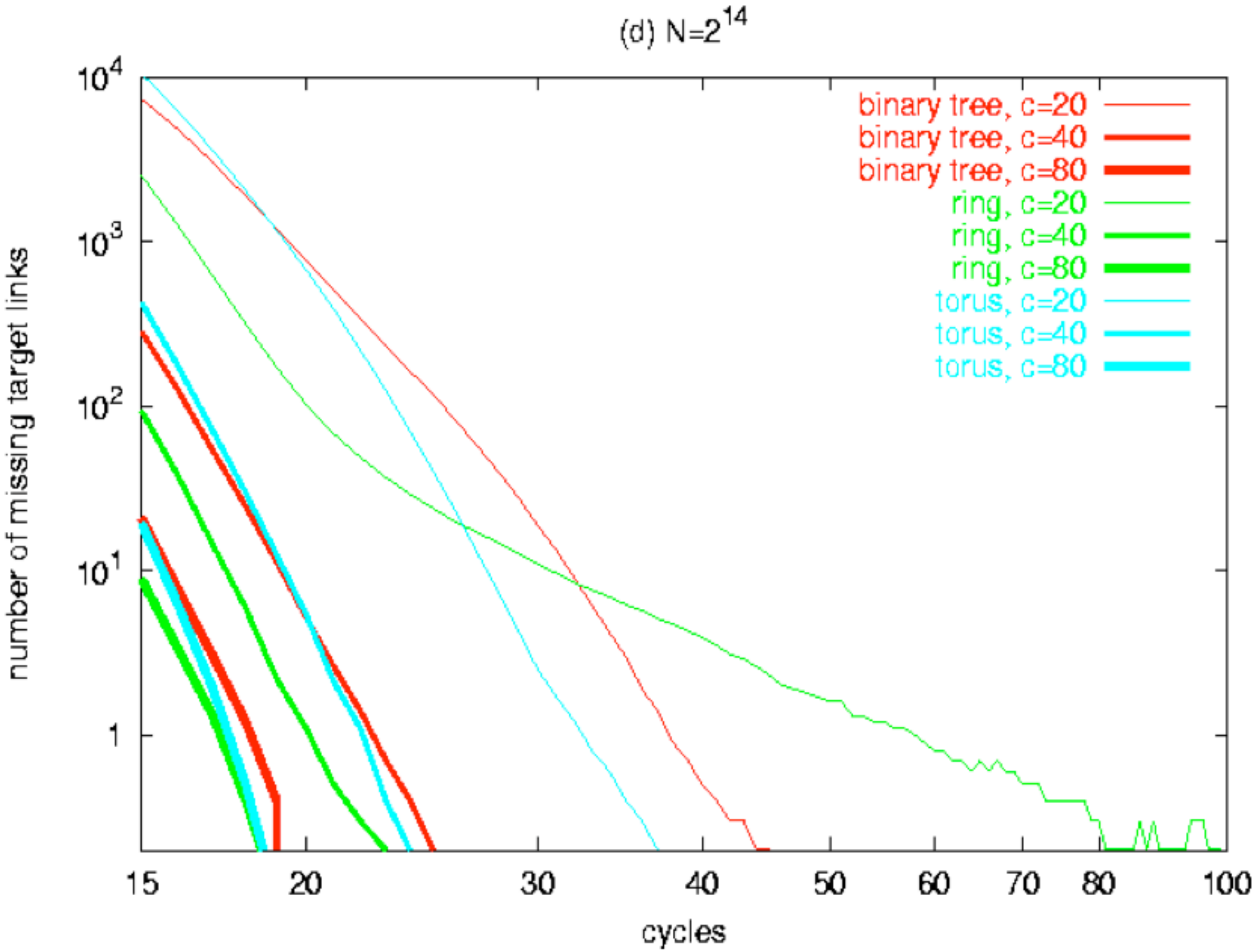


Fig. 2. Illustrative example of constructing a torus over $50 \times 50 = 2500$ nodes, starting from a uniform random topology with $c = 20$. For clarity, only the nearest 4 neighbors (out of 20) of each node are displayed.

Convergence of T-MAN



T-Chord

- **Chord on demand, A Montresor, M Jelasity, O Babaoglu - Peer-to-Peer Computing, 2005. P2P 2005.**

Main Technique T-Man

The T-Man algorithm

// *view* is a collection of neighbors

Init: $view = rnd.view \cup \{ (myaddress, mydescriptor) \}$

// active thread
// executed by p
do once every δ time units
q = selectNeighbor(view)
msg_p = extract(*view*, *q*)
send msg_p **to** q
receive msg_q **from** q
view = merge(*view*, msg_q)

A "round" of length δ

// passive thread
// executed by p
do forever
receive msg_q from *
msg_p = extract(*view*, *q*)
send msg_p **to** q
view = merge(*view*, msg_q)

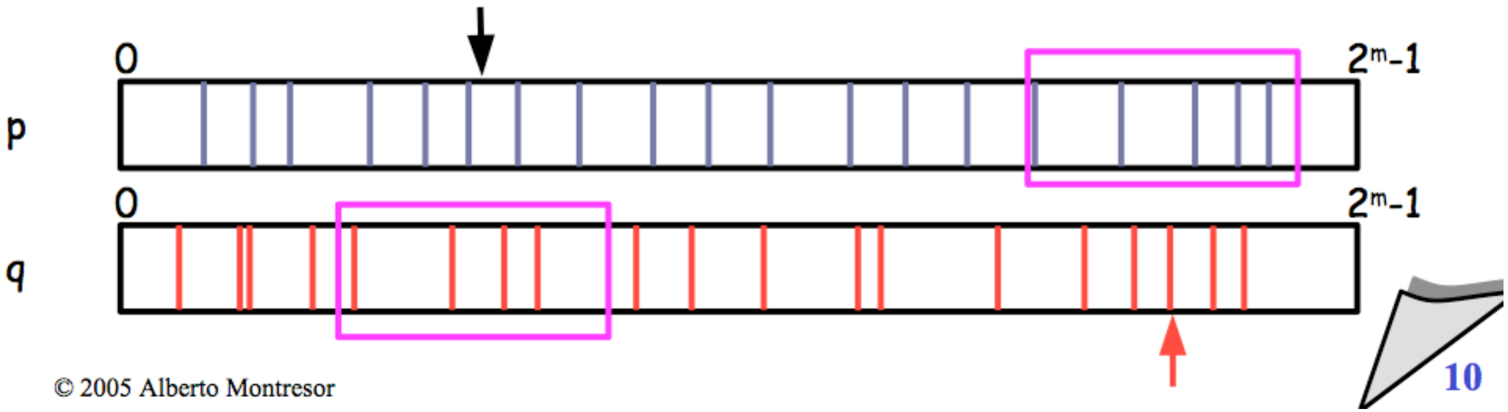
© 2005 Alberto Montresor

9

Adaption for Chord

T-Man for T-Chord

- selectPeer():
  - randomly select a peer ***q*** from the ***r*** nodes in my view that are *nearest to* **p** *in terms of ID distance*
- extract():
  - send to ***q*** the ***r*** nodes in local view that are *nearest to* **q**
  - ***q*** responds with the ***r*** nodes in its view that are *nearest to* **p**
- merge():
  - both ***p*** and ***q*** *merge* the received nodes to their view

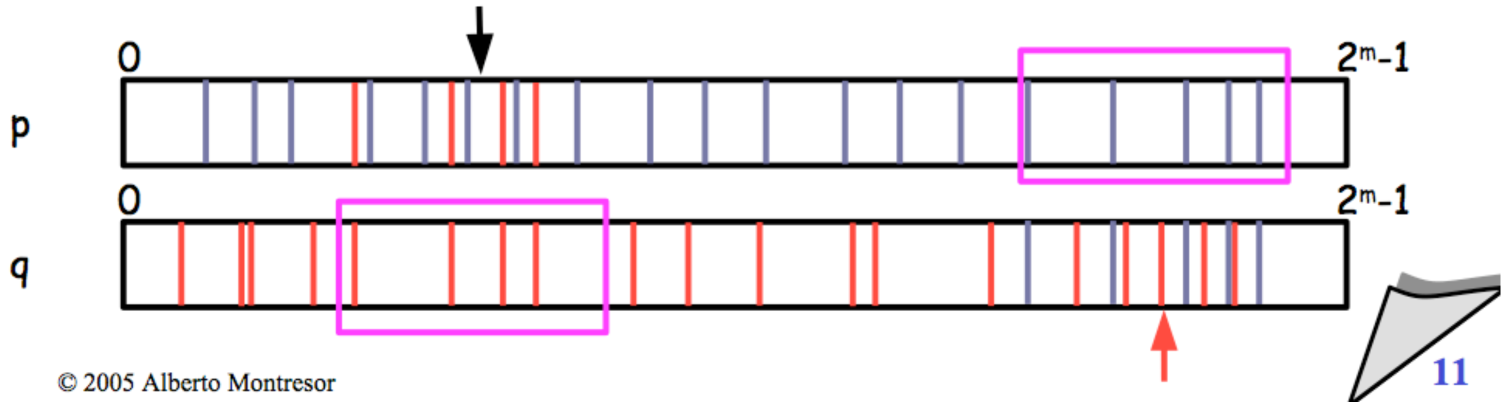


© 2005 Alberto Montresor

After Exchange of Links

T-Man for T-Chord

- selectPeer():
  - randomly select a peer ***q*** from the ***r*** nodes in my view that are *nearest to* **p** *in terms of ID distance*
- extract():
  - send to ***q*** the ***r*** nodes in local view that are *nearest to* **q**
  - ***q*** responds with the ***r*** nodes in its view that are *nearest to* **p**
- merge():
  - both ***p*** and ***q*** *merge* the received nodes to their view



© 2005 Alberto Montresor

University of Freiburg
Department of Computer Science
Computer Networks and Telematics
Prof. Dr. Christian Schindelhauer

CoNe
Freiburg

Random Graphs for Peer-to-Peer Overlays

Christian Schindelhauer

joint work with

Peter Mahlmann
University of Paderborn