Steps towards Decentralized Deterministic Network Coding

Oana Graur and Werner Henkel

{o.graur, w.henkel}@jacobs-university.de

Transmission Systems Group (TrSys)

http://trsys.faculty.jacobs-university.de

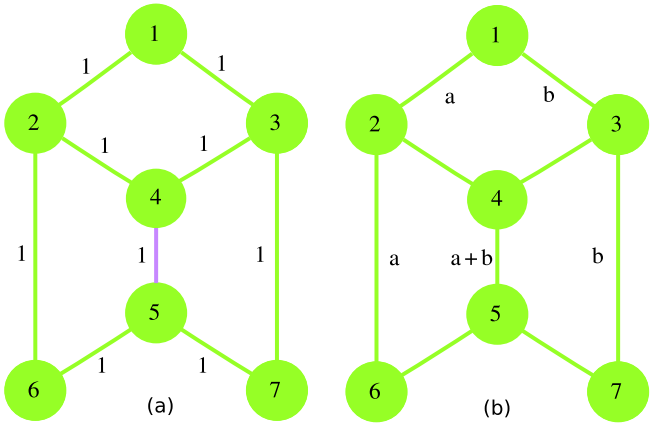
JACOBS
UNIVERSITY

Overview

- Network Coding
  - Linear Network Coding
  - Multicast Rate
- Scale-Free Networks
- Barabasi-Albert (BA) Model
- Packet Loss in Real Networks
  - Inferring Link Loss from Path Loss – Netscope
  - Measurements
  - Results
- Ant Colony Optimization (ACO)
  - Finding the Optimum Coding Nodes
- Hierarchical Network Coding
  - Routing in the Internet
  - Proposed Hierarchical Network Coding Scheme
- Conclusions

Introduction to Network Coding

- The max flow given by the *Max-Flow/Min-Cut Theorem* is hardly achieved in practice (store-and-forward).
- The maximum flow through a network is achievable through the use of network coding [Ahlswede, Cai, and Li].
- System of equations not linearly dependent.



A butterfly network - simple network coding

Multicast Rate

Single-Source Multicast

- Directed acyclic network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with unit edge capacities
- Transmitted vector

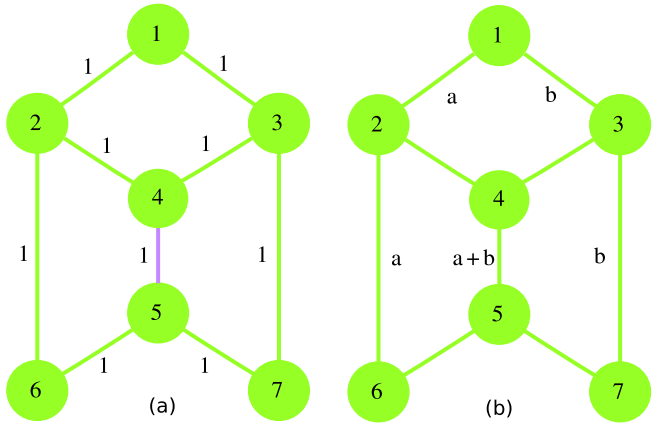
$$\mathbf{x} = [x_1 \; x_2 \; \cdots \; x_\omega], \mathbf{x} \in F^w$$

- Condition for sink t_i to receive $\mathbf{x}$

$$\text{maxflow}(t_i) \geq \omega$$

- The value of *maxflow* for sink t_i is given by the Ford-Fulkerson algorithm.
- The *multicast rate* for a network with multiple sinks is the minimum maxflow rate among all sinks t_i.

$$h = \text{maxflow}(t_i)$$



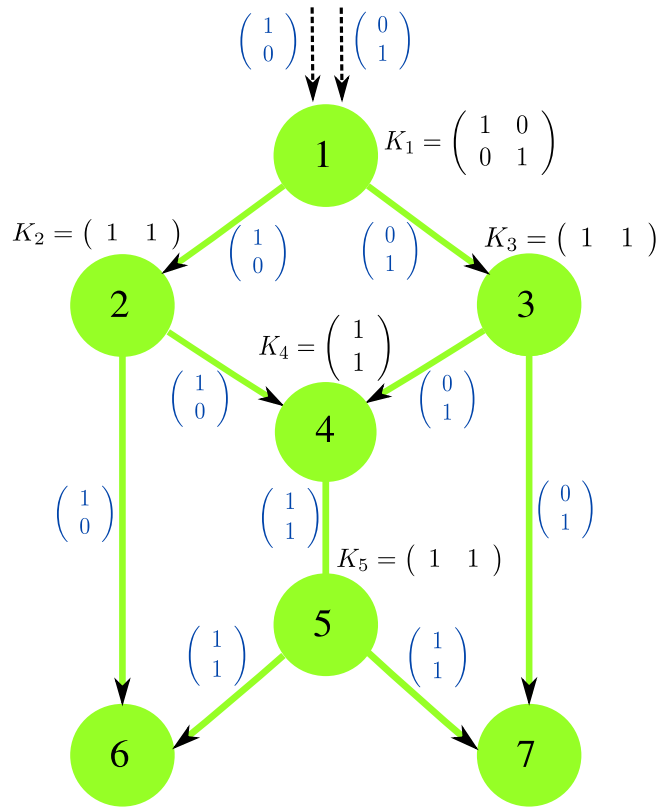
A butterfly network - simple network coding

Linear Network Coding

Local Encoding Kernel [Yeung, Li, Cai]

Let F be a finite field and ω a positive integer. An ω-dimensional F-valued linear network code on an acyclic communication network consists of a scalar $k_{d,e}$, called the local encoding kernel, for every adjacent pair (d, e). Meanwhile, the local encoding kernel at node u can be written as the $|\mathcal{I}n(u)| \times |\mathcal{O}ut(u)|$ matrix $\mathbf{K}_u = [k_{d,e}]$, $d \in \mathcal{I}n(u)$, $e \in \mathcal{O}ut(u)$.

- maps data from incoming edge d to outgoing edge e, at a node



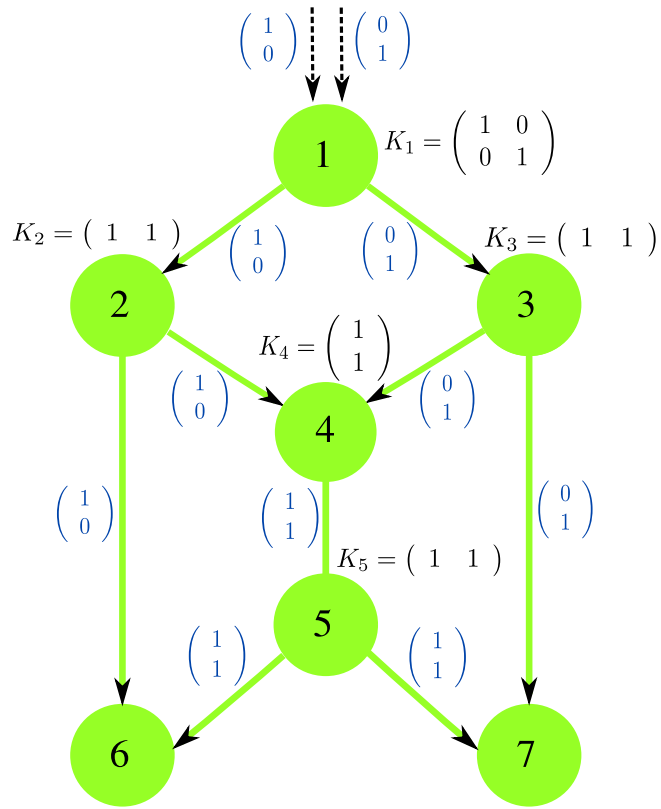
Local encoding kernels (in black) and global encoding kernels (in blue)

Linear Network Coding

Global Encoding Kernel [Yeung, Li, Cai]

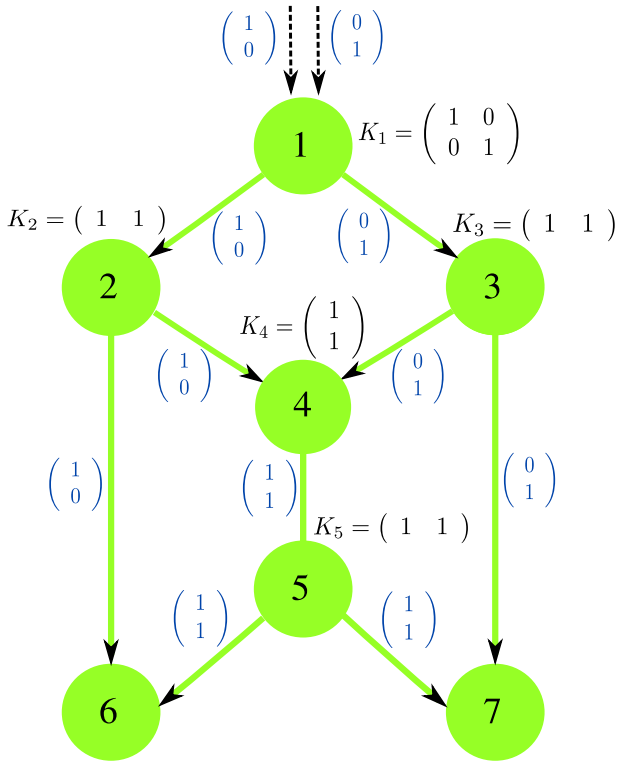
Let F be a finite field and ω a positive integer. An ω-dimensional F-valued linear network code on an acyclic communication network consists of a scalar $k_{d,e}$ in the network as well as ω-dimensional column vector $\mathbf{f}_e$, known as the global encoding kernel, for every channel e such that:

- $\mathbf{f}_e = \sum_{d \in \mathcal{I}n(u)} k_{d,e}\mathbf{f}_d$, where $e \in \mathcal{O}ut(u)$.
- The vectors $\mathbf{f}_e$ for the ω imaginary channels $e \in \mathcal{I}n(s)$ form the natural basis of the vector space F^ω.
- maps the source message $\mathbf{x}$ to the output on edge e



Local encoding kernels (in black) and global encoding kernels (in blue)

Linear Network Coding



$$\mathbf{F}_{u_{out}} = \mathbf{F}_{u_{in}} \mathbf{K}_u$$

$$\mathbf{F}_{u_{in}} = [\mathbf{f}_{d_1} \mathbf{f}_{d_2} \cdots \mathbf{f}_{d_{|\mathcal{I}n(u)|}}]$$

$$\mathbf{F}_{u_{out}} = [\mathbf{f}_{e_1} \mathbf{f}_{e_2} \cdots \mathbf{f}_{e_{|\mathcal{O}ut(u)|}}]$$

$$d \in \mathcal{I}n(u)$$

$$e \in \mathcal{O}ut(u)$$

$$[\mathbf{f}_{e_1} \mathbf{f}_{e_2} \cdots \mathbf{f}_{e_{|\mathcal{O}ut(u)|}}] = [\mathbf{f}_{d_1} \mathbf{f}_{d_2} \cdots \mathbf{f}_{d_{|\mathcal{I}n(u)|}}] \cdot \begin{bmatrix} k_{d_1 e_1} & k_{d_1 e_2} & \cdots & k_{d_1 e_{|\mathcal{O}ut(u)|}} \\ k_{d_2 e_1} & k_{d_2 e_2} & \cdots & k_{d_2 e_{|\mathcal{O}ut(u)|}} \\ \vdots & \vdots & \ddots & \vdots \\ k_{d_{|\mathcal{I}n(u)|} e_1} & k_{d_{|\mathcal{I}n(u)|} e_2} & \cdots & k_{d_{|\mathcal{I}n(u)|} e_{|\mathcal{O}ut(u)|}} \end{bmatrix}$$

Linear Network Coding

- For a certain node u, the symbol leaving the outgoing edge e is given by

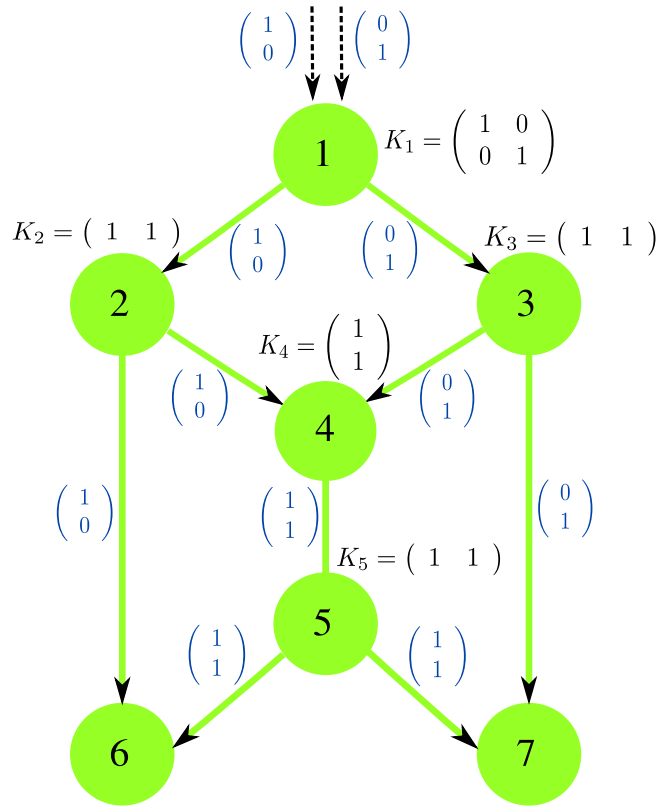
$$\mathbf{x} \cdot \mathbf{f}_e = \mathbf{x} \cdot \sum_{d \in \mathcal{I}n(u)} k_{d,e} \mathbf{f}_d = \sum_{d \in \mathcal{I}n(u)} k_{d,e} (\mathbf{x} \cdot \mathbf{f}_d)$$

- The vector of received symbols $\mathbf{y} = [y_1 \; y_2 \; \cdots \; y_\omega]$ can be written as

$$\mathbf{y} = \mathbf{x}\mathbf{G} \; ,$$

where column j of $\mathbf{G}$ holds the global encoding kernel of the jth incoming link of sink t_i.

- Decoding at sink t_i is possible if $\mathbf{G}$ is nonsingular.



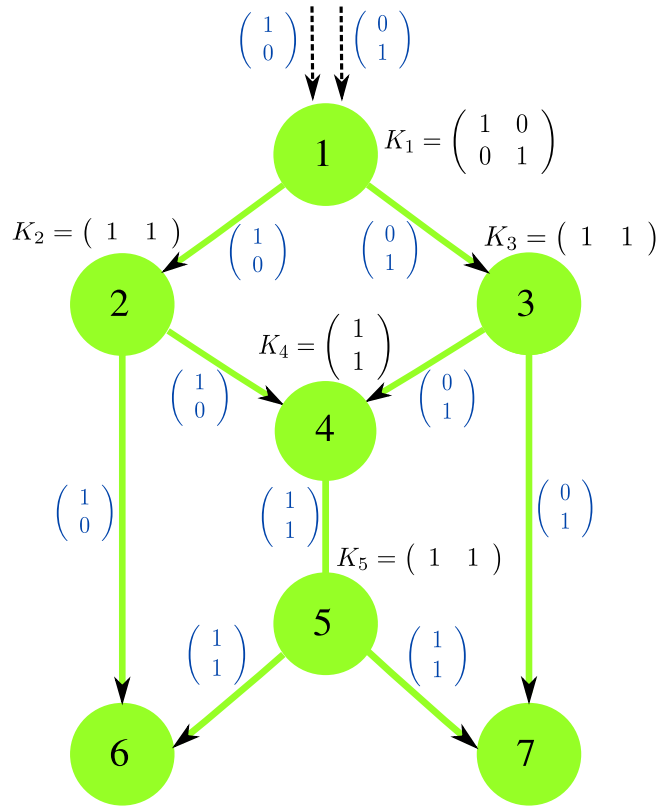
Local encoding kernels (in black) and global encoding kernels (in blue)

Erasures in Network Coding

- congested links (buffer overflows, etc.)
- packet loss $\rightarrow$ Binary Erasure Channel (BEC)
- $\mathcal{P}$ – set of all edges of a certain path L
- p_i – erasure probability of link e_i, $1 \leq i \leq |\mathcal{P}|$
- overall erasure probability of path L is given by

$$P_L = 1 - \prod_{i=1}^{|\mathcal{P}|} (1 - p_i)$$

- If GEK poorly chosen, performance is significantly decreased



Local encoding kernels (in black) and global encoding kernels (in blue)

Network Modeling and Barabasi-Albert (BA) Model

- *Scale free* model for simulation of node degree distribution
  - growth
  - preferential attachment
- Starting with n_0 vertices, at each time increment a new node is connected to the an existing node i with probability

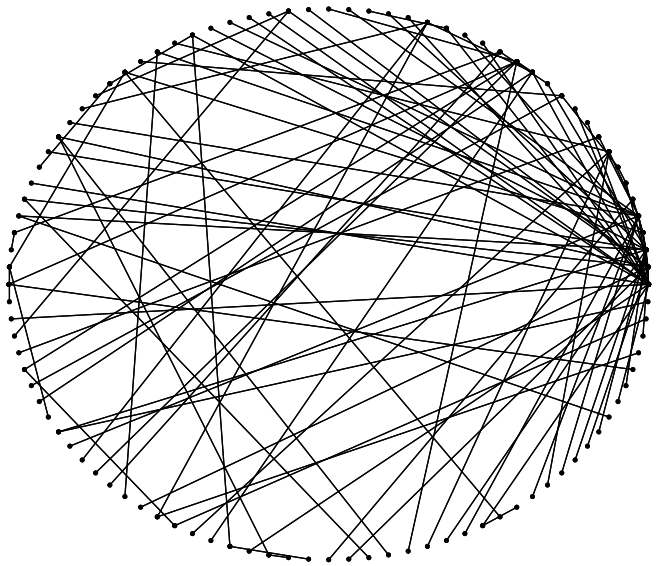
$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}$$

- Degree distribution

$$P(k) = 2m^{1/\beta} k^{-\gamma} \text{ for } \gamma{=}3,\ \beta{=}0.5$$

- *Clustering* coefficient

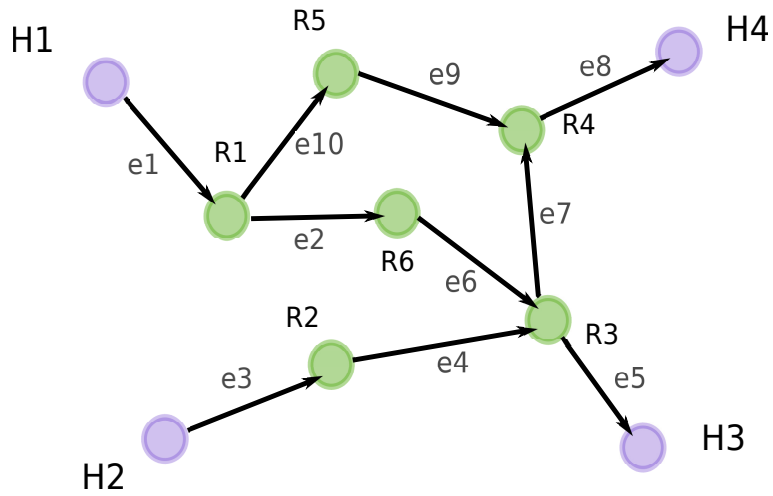
$$C_i = \frac{2E_i}{k_i(k_i-1)}$$



Scale-free network generated by the B-A model

Packet Loss in Real Networks

- limited accesibility to intermediate nodes
- end-to-end measurements possible (path measurements)
- Netscope [Nguyen, Ghita]
  - network tomography
  - infer link loss from path loss measurements
- construct a probability distribution function of the link losses



Random network example

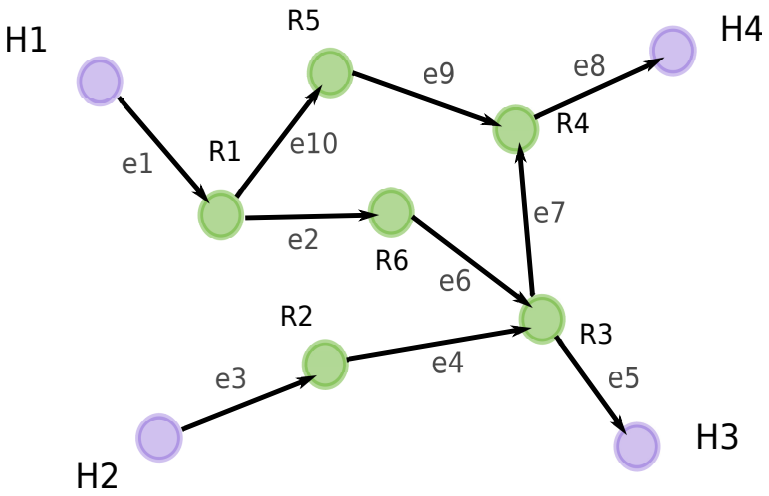
Inferring Link Loss from Path Loss - Netscope

Network: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Edge capacities: $c(u, v) : \mathcal{E} \rightarrow \mathbb{R}$
Number vertices: $n_v = |\mathcal{V}|$
Number edges: $n_e = |\mathcal{E}|$
Set of paths: $\mathcal{P}$
Number of paths: $n_p = |\mathcal{P}|$
End-to-end measurements: $\mathbf{y} = [y_1 y_2 \ldots y_{n_p}]^T$
Link transmission rate estimates: $\mathbf{x} = [x_1 x_2 \ldots x_{n_e}]^T$
Routing matrix: $\mathbf{R}$, size $n_p \times n_e$

$$\mathbf{y} = \mathbf{R}\mathbf{x}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix}$$

Routing matrix is always rank deficient [Ghita, Nguyen].



Random network example

Inferring Link Loss from Path Loss - Netscope

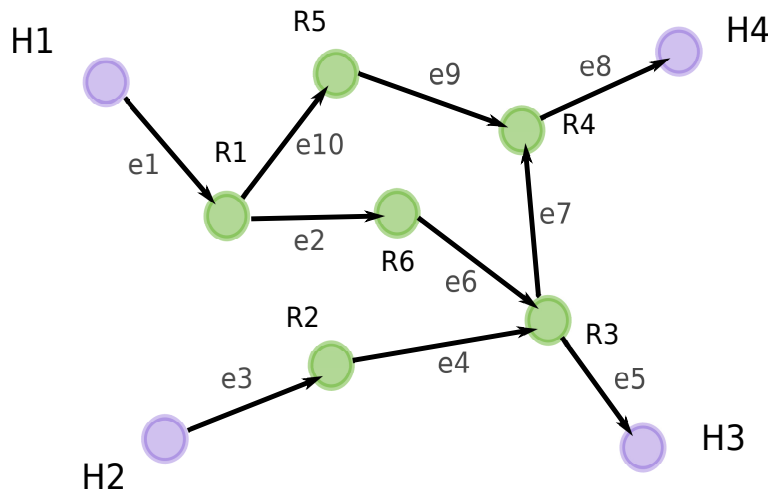
Network: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Edge capacities: $c(u, v) : \mathcal{E} \rightarrow \mathbb{R}$
Number vertices: $n_v = |\mathcal{V}|$
Number edges: $n_e = |\mathcal{E}|$
Set of paths: $\mathcal{P}$
Number of paths: $n_p = |\mathcal{P}|$
End-to-end measurements: $\mathbf{y} = [y_1 y_2 \ldots y_{n_p}]^T$
Link loss estimates: $\mathbf{x} = [x_1 x_2 \ldots x_{n_e}]^T$
Routing matrix: $\mathbf{R}$, size $n_p \times n_e$

$$\mathbf{y} = \mathbf{R}\mathbf{x}$$

- Not all links are lossy, e.g., backbone links.
- Packet loss of a subset of the links (k) can be approximated by zero.

$$k_{opt} = |\mathcal{E}| - rank(\mathbf{R})$$

Nguyen et al. argue that the links with zero transmission loss can be identified by first computing their variance. The more congested a link is, the higher the variance of its loss rate over time.



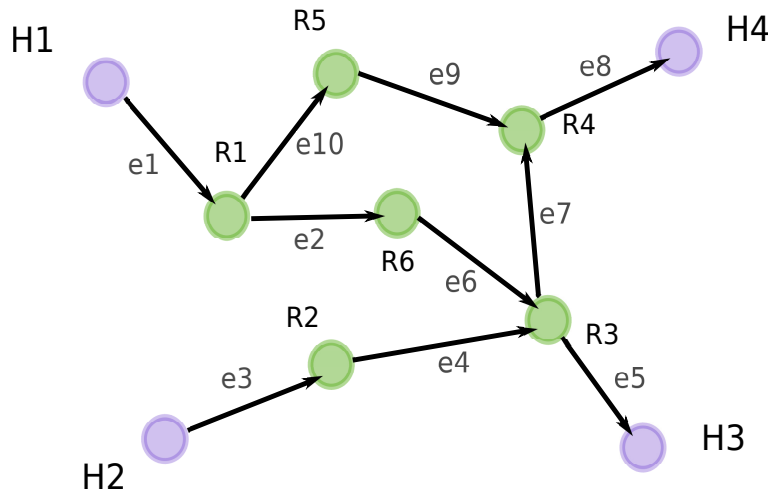
Random network example

Inferring Link Loss from Path Loss - Netscope

Practical aspects and assumptions

- not all links are distinguishable $\rightarrow$ *virtual links*
- paths meet, split and meet again $\rightarrow$ *route fluttering*
- *route fluttering* highly site-dependent (2%–40%)
- *route fluttering* attributed to *load balancing*
- topology varies over time (only 2/3 of routes persist for more than one day)

Reduced routing matrix $\mathbf{R}$, size $n_p \times n_c$, $n_c < n_e$.



Random network example

Inferring Link Loss from Path Loss - Netscope

Estimate of the successful transmission rate on path P_i: $\hat{\phi}_i$
Transmission rate of link e_k for the packets traveling along path P_i: $\hat{\phi}_{i,e_k}$
Loss rate of P_i: 1-ϕ_i, $\phi_i = \mathbb{E}[\hat{\phi}_i]$
Transmission rate of link e_k: $\phi_{e_k} = \mathbb{E}[\hat{\phi}_{e_k}]$
$y_i = \log \hat{\phi}_i$
$x_k = \log \hat{\phi}_{e_k}$

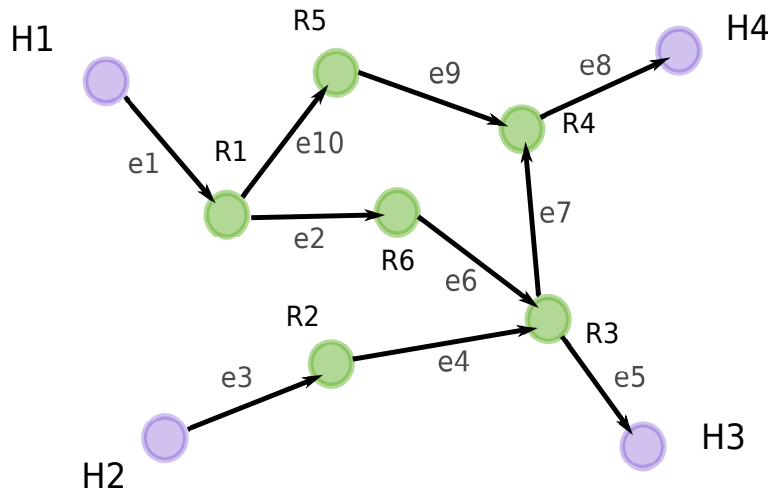
Assumptions:

- $\hat{\phi}_{i,e_k} = \hat{\phi}_{e_k}$
- random variables x_k are independent

Covariance matrix of $\mathbf{x}$:

$$\mathbf{\Gamma_x} = \text{diag}([v_1, v_2, \ldots, v_{n_c}])$$

$$\mathbf{\Sigma} = \mathbf{R\Gamma_x R^T}$$



Random network example

Inferring Link Loss from Path Loss - Netscope

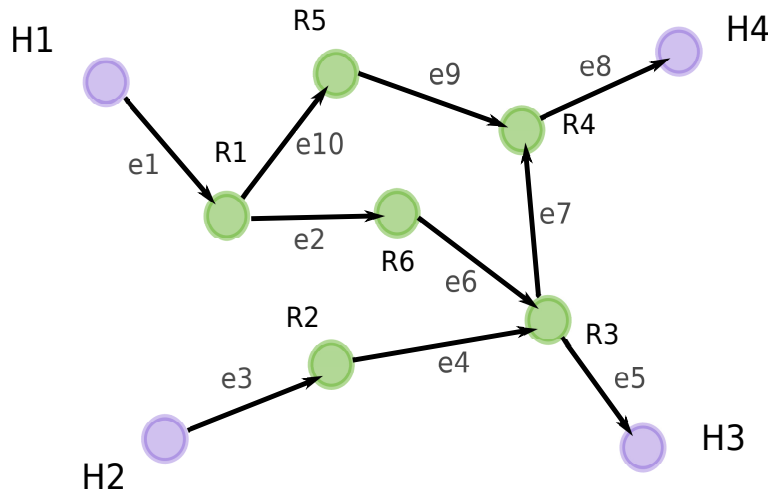
$$\mathbf{y} = \mathbf{R}\mathbf{x}$$

Covariance matrix of $\mathbf{x}$:

$$\mathbf{\Gamma_x} = \mathrm{diag}([v_1, v_2, \ldots, v_{n_c}])$$

$$\mathbf{\Sigma} = \mathbf{R}\mathbf{\Gamma_x}\mathbf{R}^{\mathbf{T}}$$

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma^2{}_{y_1} & \mathrm{cov}(y_1, y_2) & \ldots & \mathrm{cov}(y_1, y_{n_p}) \\ \mathrm{cov}(y_2, y_1) & \sigma^2{}_{y_2} & \ldots & \mathrm{cov}(y_1, y_{n_p}) \\ . & . & \ldots & . \\ . & . & \ldots & . \\ \mathrm{cov}(y_{n_p}, y_1) & \mathrm{cov}(y_{n_p}, y_2) & \ldots & \sigma^2{}_{y_{n_p}} \end{pmatrix}$$



Random network example

Inferring Link Loss from Path Loss - Netscope

Let $\mathbf{A}$ be an *augmented matrix* of dimension $n_p(n_p+1)/2 \times n_c$ whose rows consist of the component-wise product of each pair of different rows from $\mathbf{R}$. The rows of $\mathbf{A}$ are arranged as follows: $\mathbf{A}_{((i-1)\times n_p+(j-i)+1)*} = \mathbf{R_{i*}} \odot \mathbf{R_{j*}}$ for all $1 \leq i \leq j \leq n_p$. E.g.,

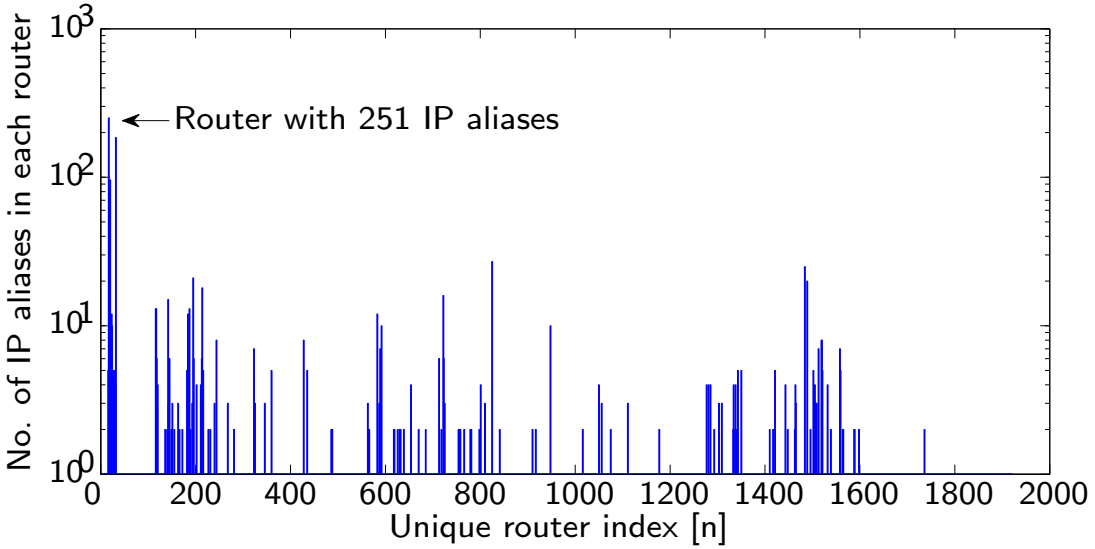
$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R} = \begin{pmatrix} \mathbf{R_{1*}} \\ \mathbf{R_{2*}} \\ \vdots \\ \mathbf{R_{n_p*}} \end{pmatrix} \qquad \mathbf{R} = \begin{pmatrix} \mathbf{R_{*1}} & \mathbf{R_{*2}} & \cdots & \mathbf{R_{*n_p}} \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{R_{1*}} \odot \mathbf{R_{1*}} \\ \mathbf{R_{1*}} \odot \mathbf{R_{2*}} \\ \vdots \\ \mathbf{R_{1*}} \odot \mathbf{R_{n_p*}} \\ \mathbf{R_{2*}} \odot \mathbf{R_{2*}} \\ \mathbf{R_{2*}} \odot \mathbf{R_{3*}} \\ \vdots \\ \mathbf{R_{2*}} \odot \mathbf{R_{n_p*}} \\ \vdots \\ \mathbf{R_{n_p*}} \odot \mathbf{R_{n_p*}} \end{pmatrix}$$

PlanetLab Measurements

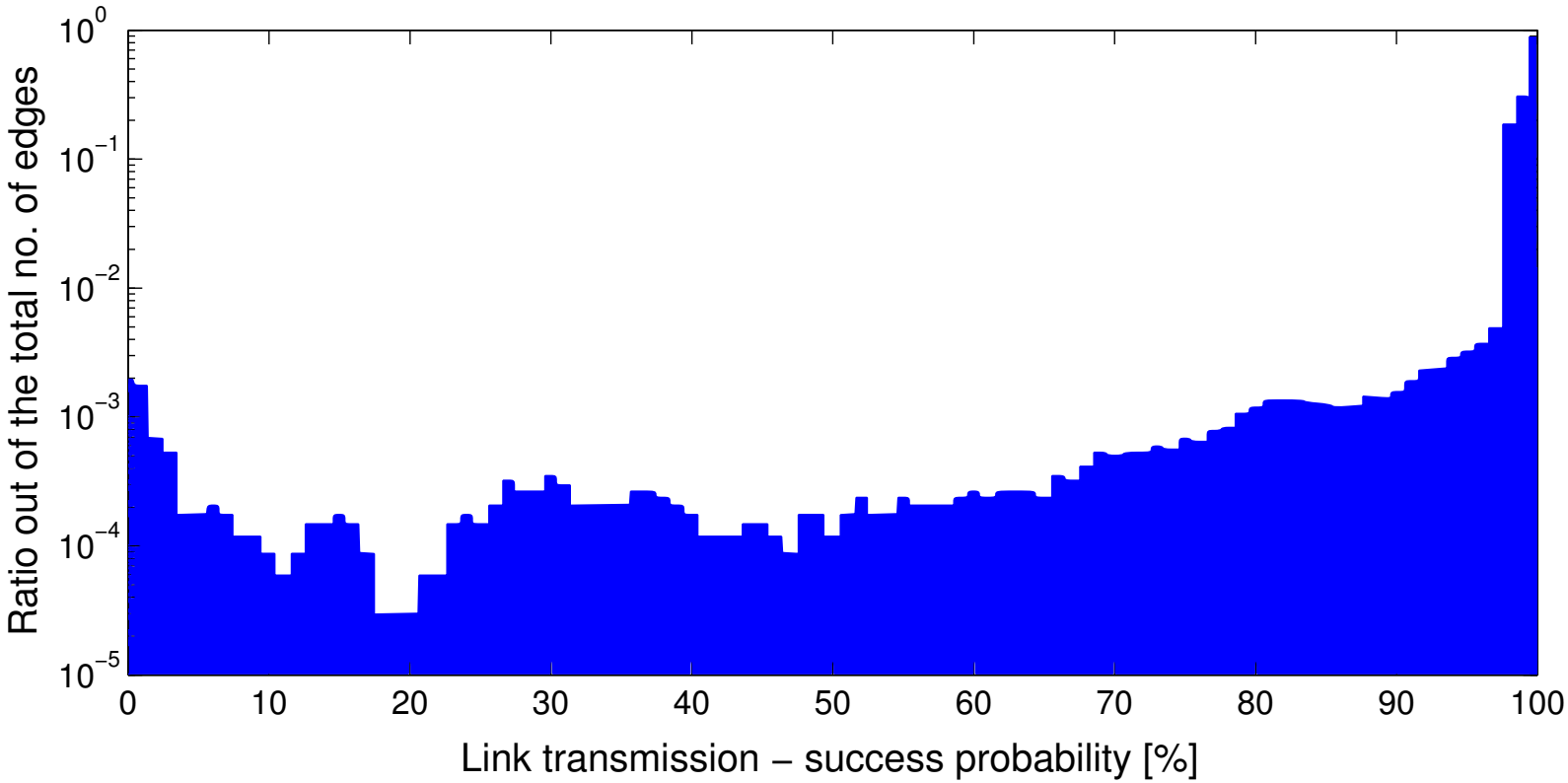
- Internet testbed for research
- 226 end hosts, both *beacons* and *probing destinations*
- *ping*, *traceroute*
- Number of active paths $|\mathcal{P}| = 50850$
- IP aliasing resolution (active probing – *Ally*)
- IP ID [RFC 791] counter and TTL counter comparison
- Number of intermediate routers reduced from 3049 to 1920



IP aliases

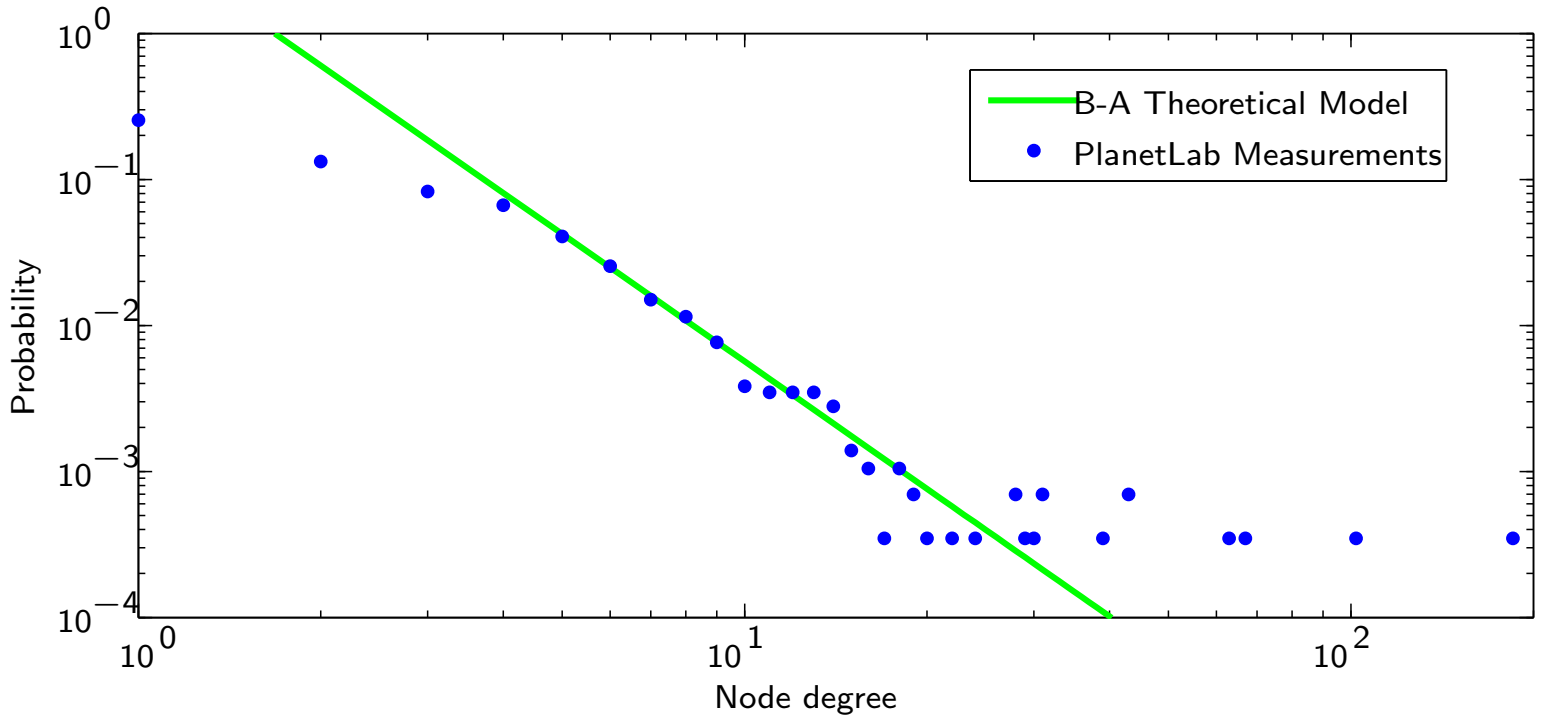
The router with the highest numbers of aliases (251) in the PlanetLab Europe network was identified as being part of *NORDUnet* (Nordic Infrastructure for Research & Education network).

PlanetLab Measurements



Derived link loss *(erasure)* distribution based on path loss measurements, employing network tomography techniques (Netscope). 92% of the links were found to be error-free, while a small percentage of the links had 100% packet loss.

PlanetLab Measurements

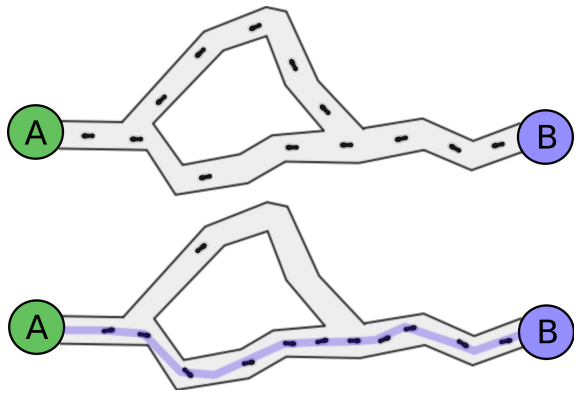


Comparison between the theoretical model and our measurements in the PlanetLab network

Ant Colony Optimization for Finding Shortest Path

Ant Colony Optimization (ACO) [Dorigo]

- Ants walk through the network, looking for food sources. Once a food source is found, ants return to the colony, on the same path, laying down a *pheromone* trail.
- Each ant has a memory of the current path.
- At every node, an ant chooses one among the adjacent links based on previous pheromone levels of the links.
- Pheromone concentration will be more significant on shorter paths.
- Trail intensity diffuses over time.



Ants path

ACO is preferable since it does not require a centralized view of the topology.

Ant Colony Optimization for Finding Shortest Path

Ant Colony Optimization (ACO) [Dorigo]

Network: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Number of vertices: $n = |\mathcal{V}|$
Edge mappings: $w : \mathcal{E} \to \mathbb{R}$
Pheromone level of edge $e_{i,j}$: $\tau_{i,j}(t)$
Number of ants: m

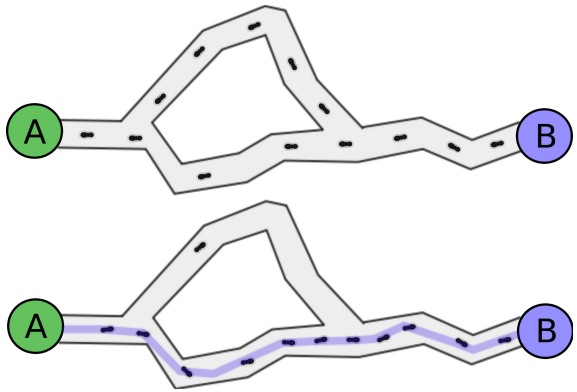
- Transition probability

$$p_{ij}(t) = \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}}{\sum_{sallowed} \tau_{is}^{\alpha}(t)\eta_{is}^{\beta}}$$

- Pheromone update

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij},$$

$$\Delta\tau_{i,j} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k} ,$$



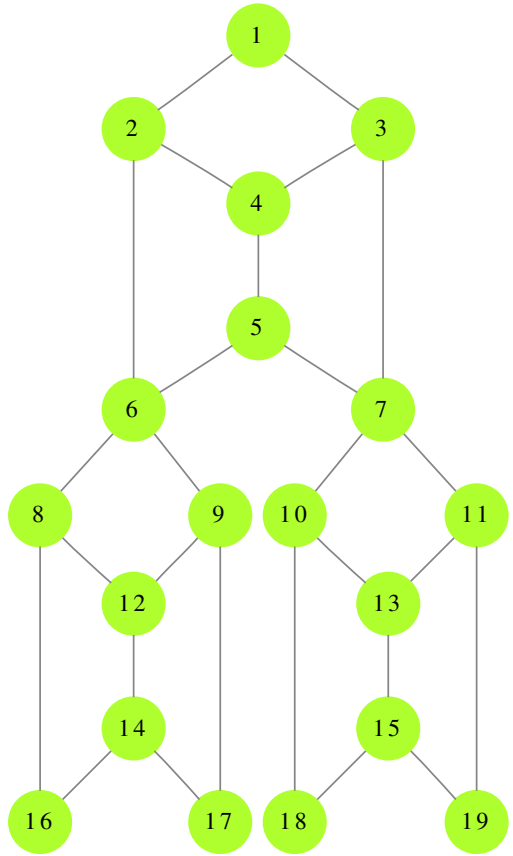
Ants path

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{Q}{L_k} , & \text{if ant } k \text{ deposited pheromones on } e_{i,j} \text{ in time } t \\ 0, & \text{otherwise.} \end{cases}$$

$$L_k = 1 - \prod_{i=1}^{|\mathcal{P}_k|} \phi_{e_i}$$

Finding Disjoint Paths

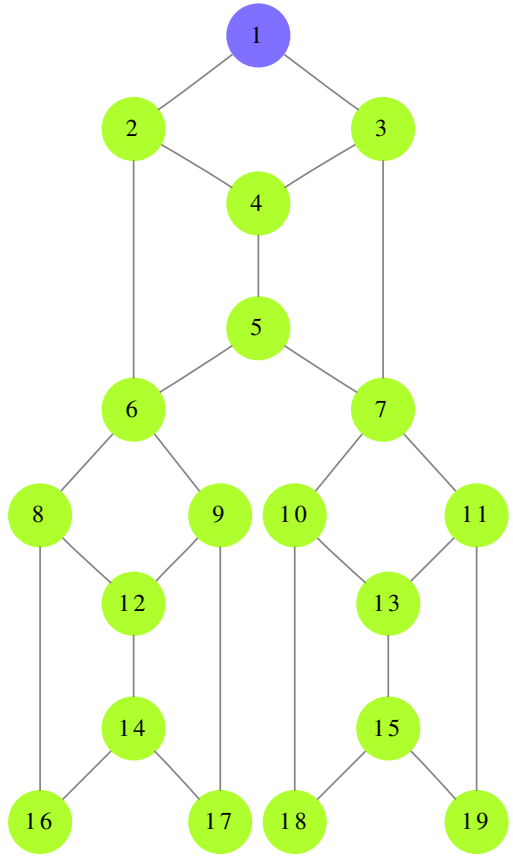
- Select a source node and sink nodes.



Sinks	First disjoint path	Second disjoint path
Sink 16		
Sink 17		
Sink 18		
Sink 19		

Finding Disjoint Paths

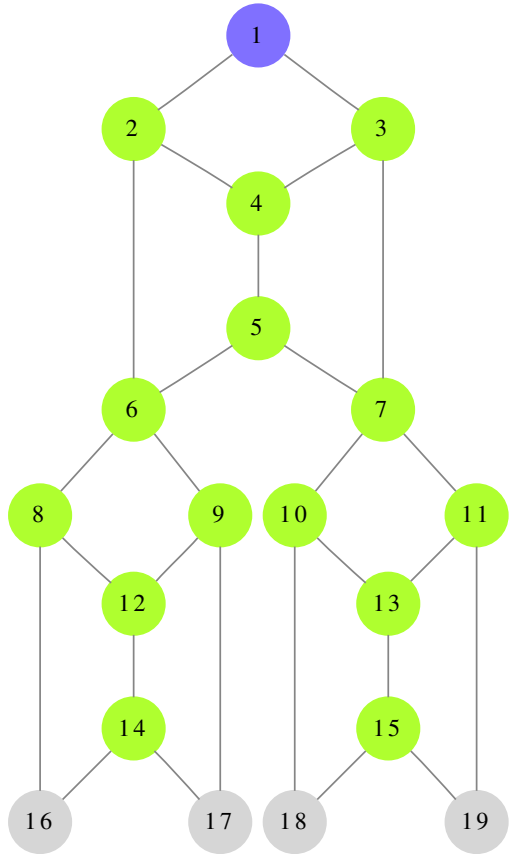
- Select a source node and sink nodes.



Sinks	First disjoint path	Second disjoint path
Sink 16 Sink 17 Sink 18 Sink 19		

Finding Disjoint Paths

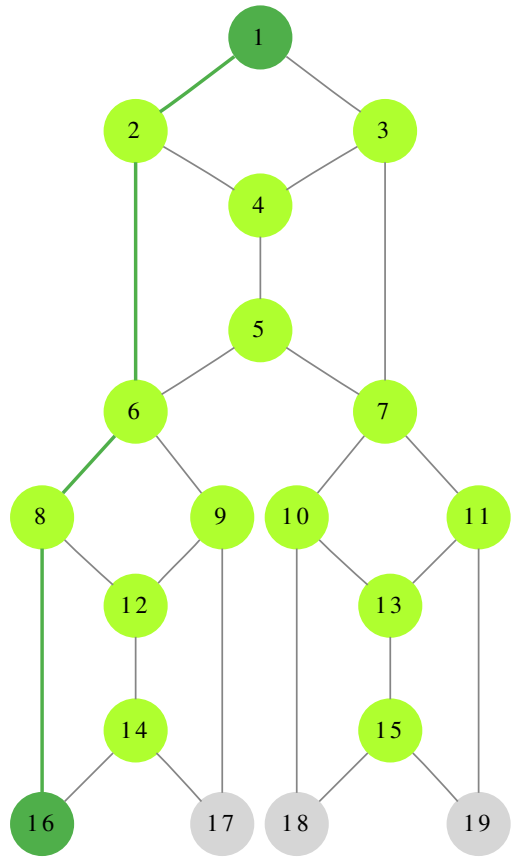
- Select a source node and sink nodes.



Sinks	First disjoint path	Second disjoint path
Sink 16 Sink 17 Sink 18 Sink 19		

Finding Disjoint Paths

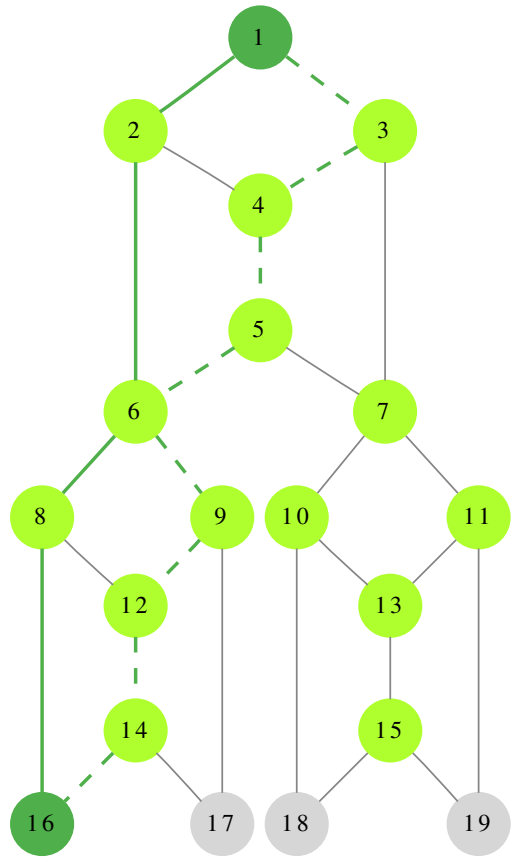
- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



Sinks	First disjoint path	Second disjoint path
Sink 16 Sink 17 Sink 18 Sink 19	$\mathcal{P}_1$: 1→2→6→8→16	

Finding Disjoint Paths

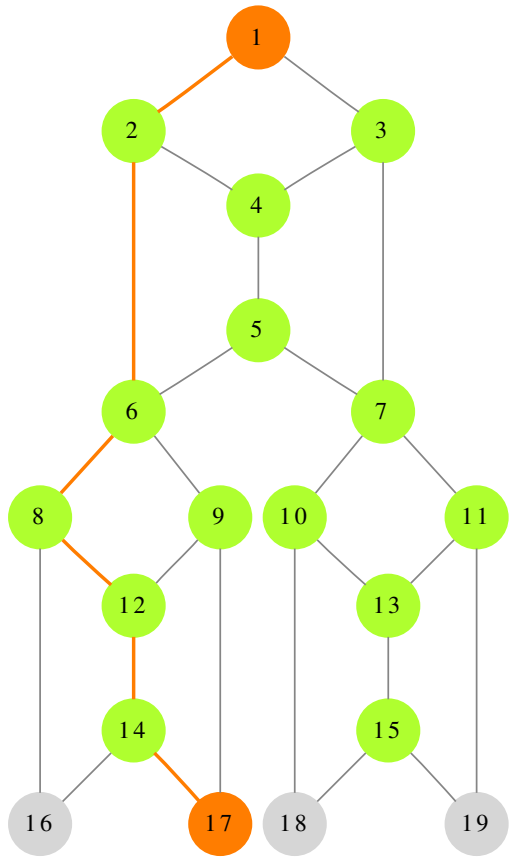
- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



Sinks	First disjoint path	Second disjoint path
Sink 16 Sink 17 Sink 18 Sink 19	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→4 →5→6→9→2 →14→16

Finding Disjoint Paths

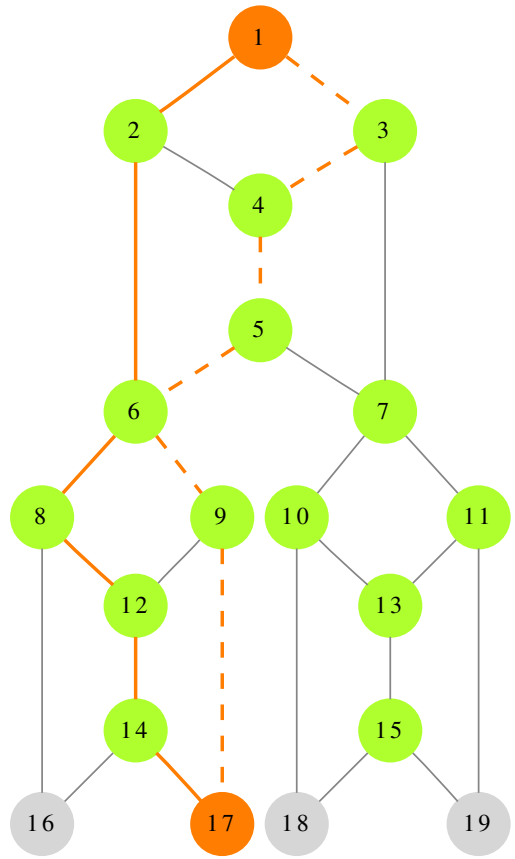
- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



Sinks	First disjoint path	Second disjoint path
Sink 16 Sink 17 Sink 18 Sink 19	$\mathcal{P}_1$: 1→2→6→8→16 $\mathcal{P}_3$: 1→2→6→8→12→14→17	$\mathcal{P}_2$: 1→3→4 →5→6→9→2 →14→16

Finding Disjoint Paths

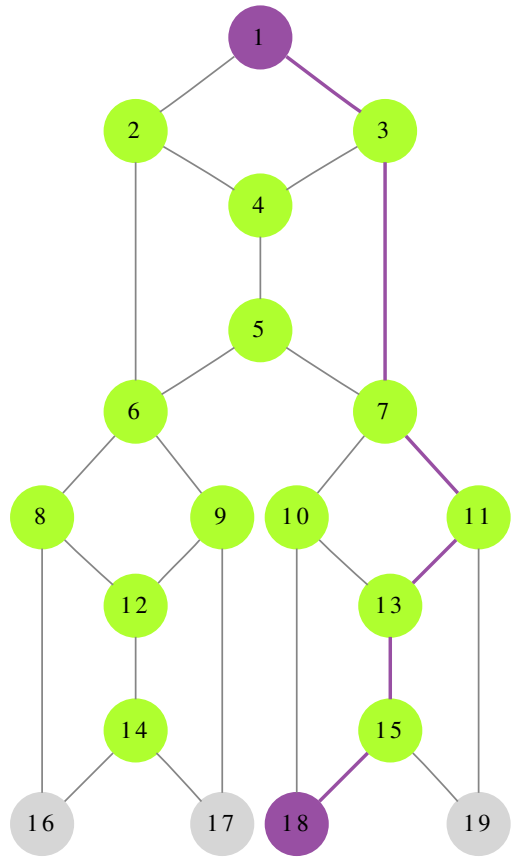
- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



Sinks	First disjoint path	Second disjoint path
Sink 16	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→4 →5→6→9→2 →14→16
Sink 17	$\mathcal{P}_3$: 1→2→6→8→12→14→17	$\mathcal{P}_4$: 1→3→4 →5→6→9→17
Sink 18		
Sink 19		

Finding Disjoint Paths

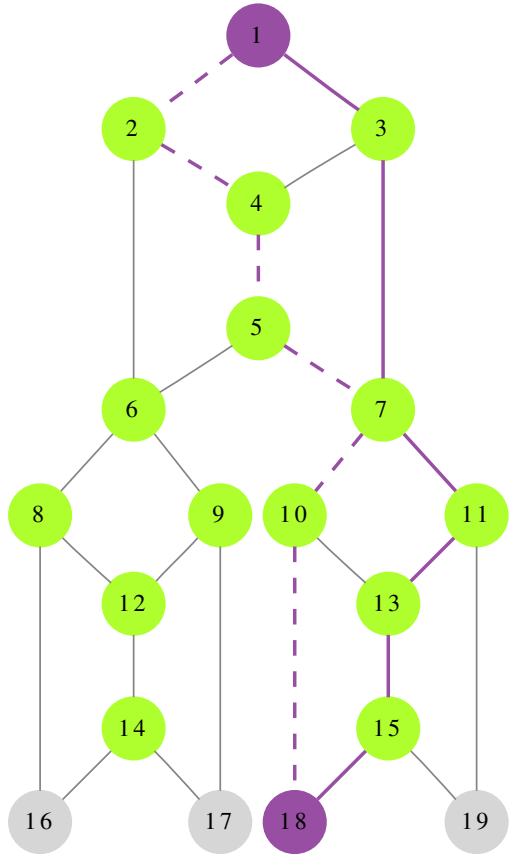
- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



Sinks	First disjoint path	Second disjoint path
Sink 16	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→4 →5→6→9→2 →14→16
Sink 17	$\mathcal{P}_3$: 1→2→6→8→12→14→17	$\mathcal{P}_4$: 1→3→4 →5→6→9→17
Sink 18	$\mathcal{P}_5$: 1→2→4 →5→7→10→18	
Sink 19		

Finding Disjoint Paths

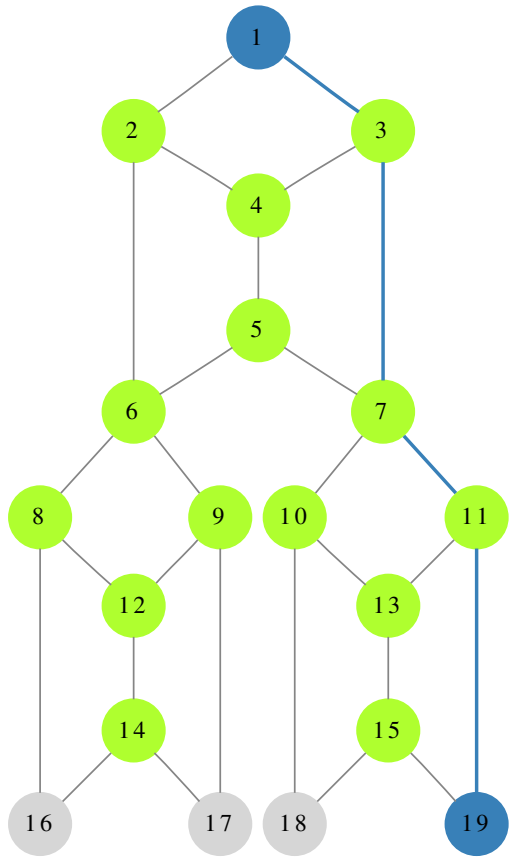
- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



Sinks	First disjoint path	Second disjoint path
Sink 16	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→4 →5→6→9→2 →14→16
Sink 17	$\mathcal{P}_3$: 1→2→6→8→12→14→17	$\mathcal{P}_4$: 1→3→4 →5→6→9→17
Sink 18	$\mathcal{P}_5$: 1→2→4 →5→7→10→18	$\mathcal{P}_6$: 1→3→7→11→13→15→18
Sink 19		

Finding Disjoint Paths

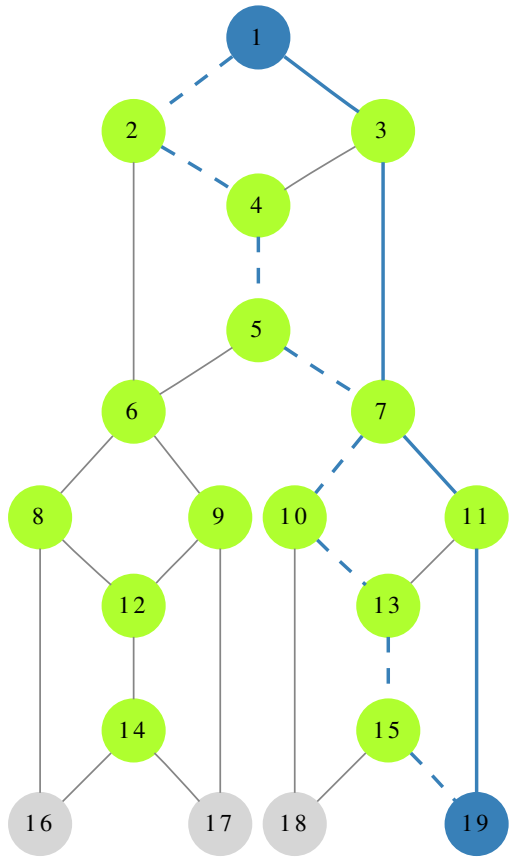
- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



Sinks	First disjoint path	Second disjoint path
Sink 16	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→4 →5→6→9→2 →14→16
Sink 17	$\mathcal{P}_3$: 1→2→6→8→12→14→17	$\mathcal{P}_4$: 1→3→4 →5→6→9→17
Sink 18	$\mathcal{P}_5$: 1→2→4 →5→7→10→18	$\mathcal{P}_6$: 1→3→7→11→13→15→18
Sink 19	$\mathcal{P}_7$: 1→3→7→11→19	

Finding Disjoint Paths

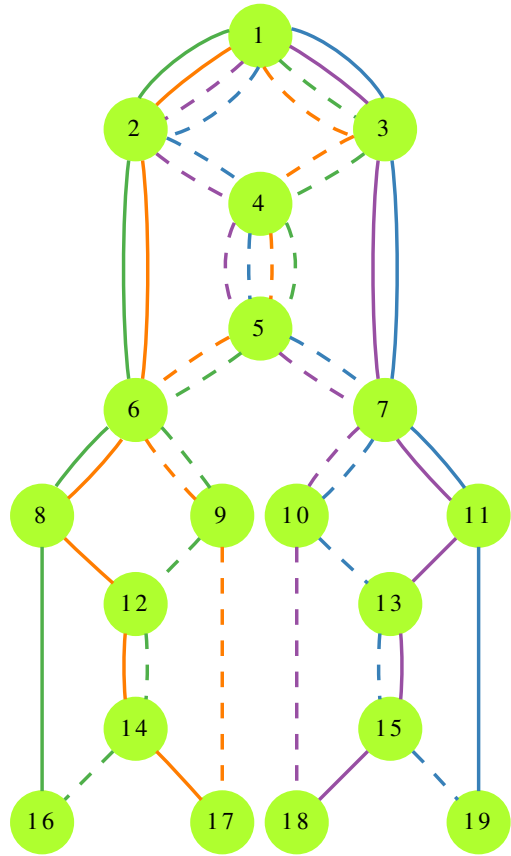
- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



Sinks	First disjoint path	Second disjoint path
Sink 16	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→4 →5→6→9→2 →14→16
Sink 17	$\mathcal{P}_3$: 1→2→6→8→12→14→17	$\mathcal{P}_4$: 1→3→4 →5→6→9→17
Sink 18	$\mathcal{P}_5$: 1→2→4 →5→7→10→18	$\mathcal{P}_6$: 1→3→7→11→13→15→18
Sink 19	$\mathcal{P}_7$: 1→3→7→11→19	$\mathcal{P}_8$: 1→2→4 →5→7→10 →13→ 15→19

Finding Coding Nodes

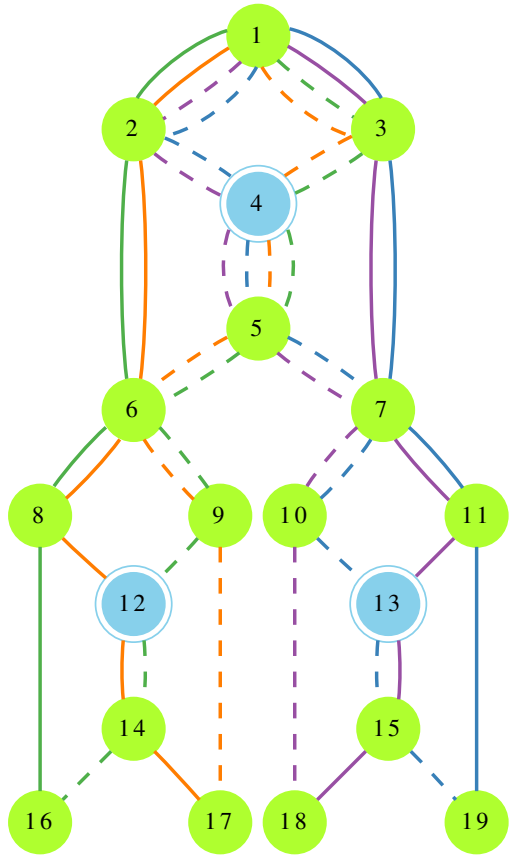
- Obtain coding nodes from the intersection of paths corresponding to different sinks.



Sinks	First disjoint path	Second disjoint path
Sink 16	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→ **4** ⇒**5**→6→9→ **12** ⇒**14**→16
Sink 17	$\mathcal{P}_3$: 1→2→6→8→ **12** ⇒**14**→17	$\mathcal{P}_4$: 1→3→ **4** ⇒**5**→6→9→17
Sink 18	$\mathcal{P}_5$: 1→2→ **4** ⇒**5**→7→10→18	$\mathcal{P}_6$: 1→3→7→11→ **13** ⇒**15**→18
Sink 19	$\mathcal{P}_7$: 1→3→7→11→19	$\mathcal{P}_8$: 1→2→ **4** ⇒**5**→7→10 → **13** ⇒ **15**→19

Finding Coding Nodes

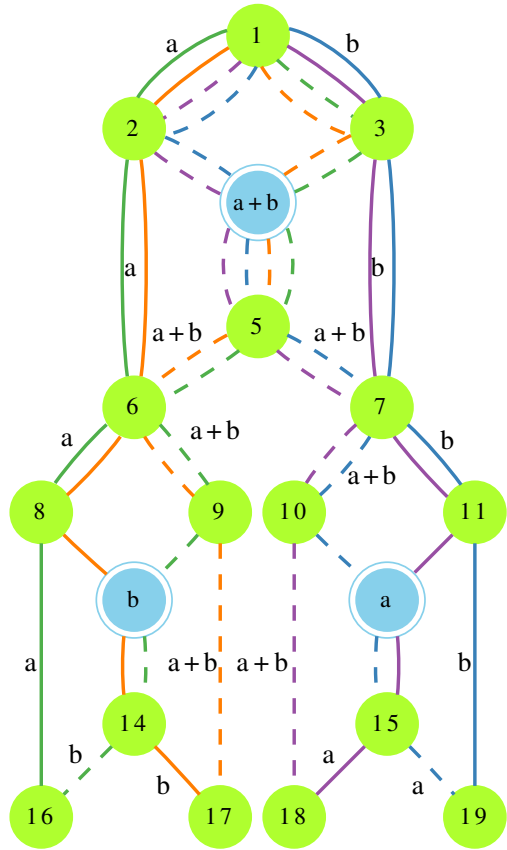
- Obtain coding nodes from the intersection of paths corresponding to different sinks.



Sinks	First disjoint path	Second disjoint path
Sink 16	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→ **4** ⇒**5**→6→9→ **12** ⇒**14**→16
Sink 17	$\mathcal{P}_3$: 1→2→6→8→ **12** ⇒**14**→17	$\mathcal{P}_4$: 1→3→ **4** ⇒**5**→6→9→17
Sink 18	$\mathcal{P}_5$: 1→2→ **4** ⇒**5**→7→10→18	$\mathcal{P}_6$: 1→3→7→11→ **13** ⇒**15**→18
Sink 19	$\mathcal{P}_7$: 1→3→7→11→19	$\mathcal{P}_8$: 1→2→ **4** ⇒**5**→7→10 → **13** ⇒ **15**→19

Finding Coding Nodes

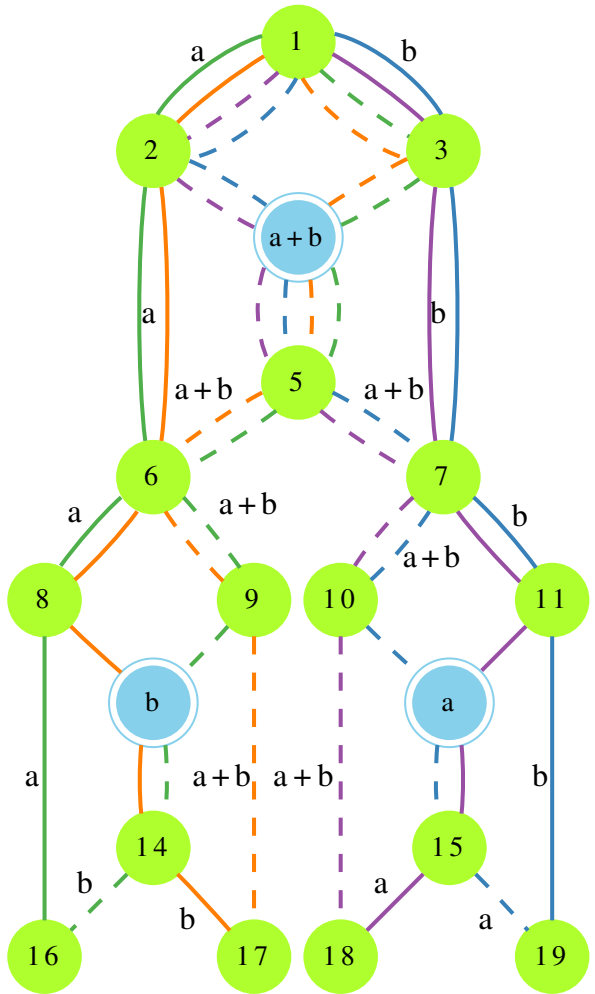
- Obtain coding nodes from the intersection of paths corresponding to different sinks.



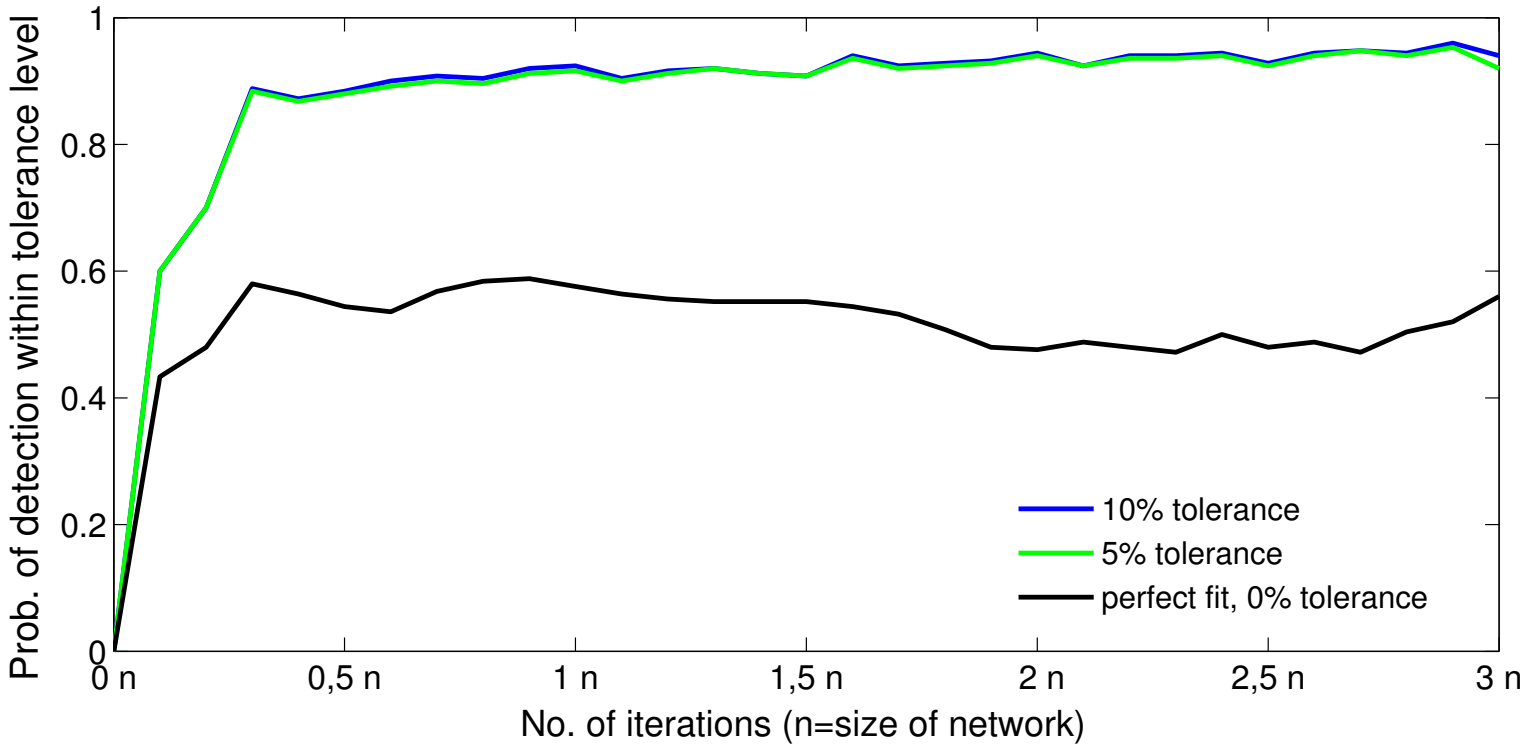
Sinks	First disjoint path	Second disjoint path
Sink 16	$\mathcal{P}_1$: 1→2→6→8→16	$\mathcal{P}_2$: 1→3→ **4** ⇒**5**→6→9→ **12** ⇒**14**→16
Sink 17	$\mathcal{P}_3$: 1→2→6→8→ **12** ⇒**14**→17	$\mathcal{P}_4$: 1→3→ **4** ⇒**5**→6→9→17
Sink 18	$\mathcal{P}_5$: 1→2→ **4** ⇒**5**→7→10→18	$\mathcal{P}_6$: 1→3→7→11→ **13** ⇒**15**→18
Sink 19	$\mathcal{P}_7$: 1→3→7→11→19	$\mathcal{P}_8$: 1→2→ **4** ⇒**5**→7→10 → **13** ⇒ **15**→19

Finding Coding Nodes

- Ensure that at all times each edge passes either coded or not coded messages, not both at the same time, otherwise flip disjoint paths.

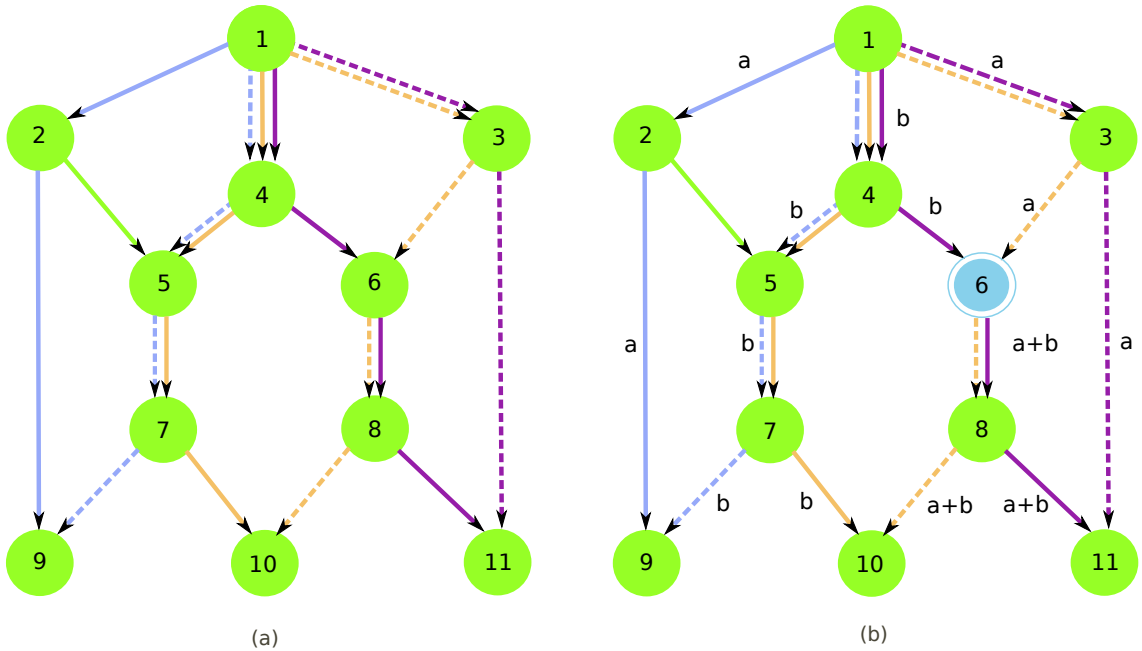


ACO Convergence



$\alpha = 1,\ \beta = 5,\ \rho = 0.99,\ n = m = 156$

Practical Aspects



Counter example

- GEK not uniquely specified $\mathbf{y}_i = \mathbf{x}\mathbf{G}_i$
- $\mathbf{G}_i$ needs to be invertible for every sink t_i

Conclusions

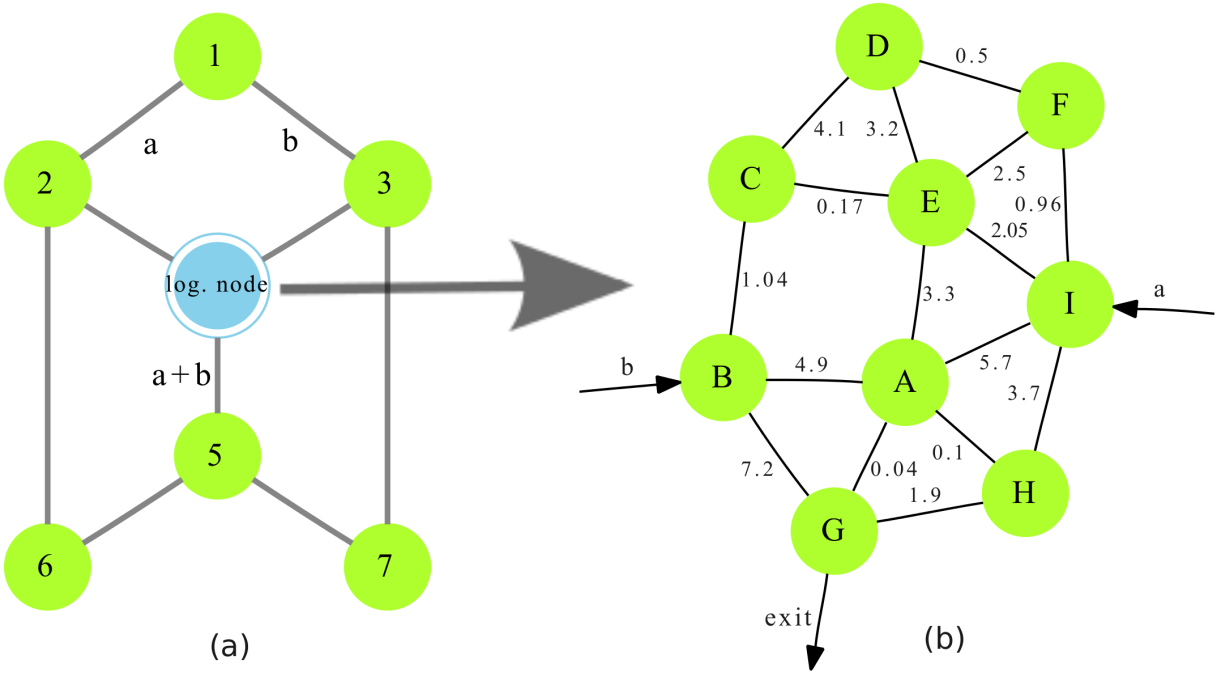
- B-A network model
- inferred link loss from packet loss $\rightarrow$ pdf of link losses
- single-source multicast
- Ant Colony Optimization to determine coding nodes
- solve the problem of GEK not uniquely determined by the coding links choice for certain topologies
- hierarchical network coding scheme

Towards Deterministic Network Coding in Hierarchical Networks

Oana Graur and Werner Henkel
{o.graur, w.henkel}@jacobs-university.de

Transmission Systems Group (TrSys)
http://trsys.faculty.jacobs-university.de

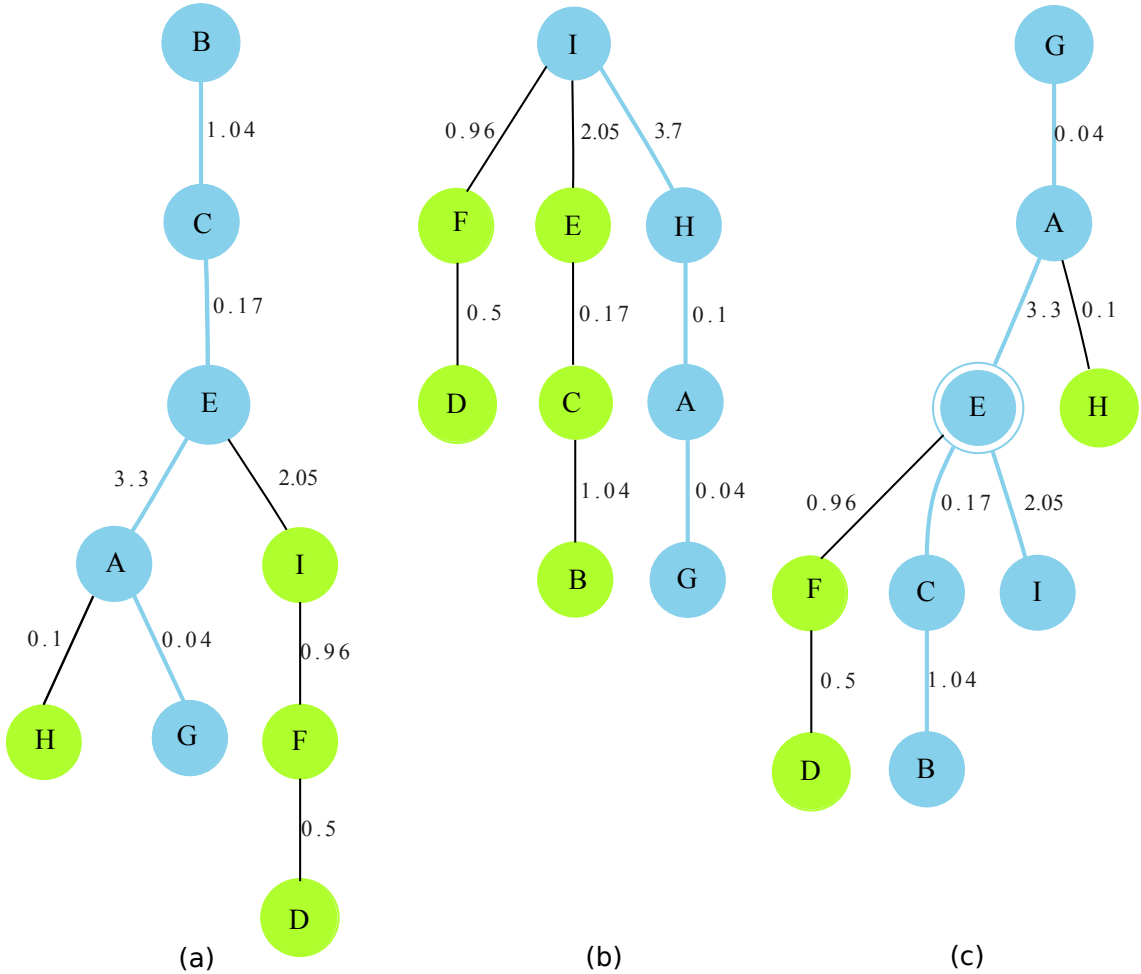
Hierarchical Network Structure



Hierarchical network structure

- Autonomous Systems (AS) → set of routers under a single technical administration with a clearly defined routing policy [RFC 1771].
- Routing protocols → IGP (OSPF), EGP (BGP)
- OSPF builds Dijkstra trees → not network coding aware
- 4400 ASs identified and 9 % of the flow between ASs accounts for 90 % of the bytes transmitted

Hierarchical Network Structure



Dijkstra trees vs. minimum spanning tree

Thank you!