Recent graph-theoretic progress of network coding

From characterization theorems to new graph algorithms.

Chih-Chun Wang
Center for Wireless Systems and Applications
School of ECE
Purdue University

Ness B. Shroff
Departments of ECE and CSE
The Ohio State University

Content

- The challenge of characterizing inter-session network coding.
  - Existing results: information-theoretic and graph-theoretic.

Content

- The challenge of characterizing inter-session network coding.
  - Existing results: information-theoretic and graph-theoretic.
  - Pairwise intersession network coding — One step toward solving the intrinsically hard problem. [ISIT07, Allerton 07, submitted to IT].

Content

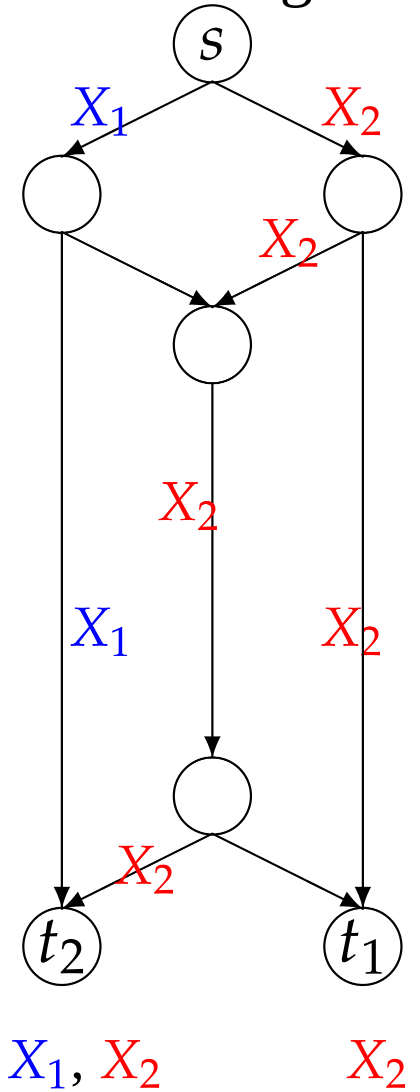
- The challenge of characterizing inter-session network coding.
  - Existing results: information-theoretic and graph-theoretic.
  - Pairwise intersession network coding — One step toward solving the intrinsically hard problem. [ISIT07, Allerton 07, submitted to IT].
- Implementation of practical intra-session network coding.
  - Bandwidth efficiency governed by the min-cut max-flow theorem. [Ahlswede *et al*. 00], [Li *et al*. 03]

Content

- The challenge of characterizing inter-session network coding.
  - Existing results: information-theoretic and graph-theoretic.
  - Pairwise intersession network coding — One step toward solving the intrinsically hard problem. [ISIT07, Allerton 07, submitted to IT].
- Implementation of practical intra-session network coding.
  - Bandwidth efficiency governed by the min-cut max-flow theorem. [Ahlswede *et al.* 00], [Li *et al.* 03]
  - Trimming network coding traffic by network coding — A new class of max-flow algorithms [ISIT 08, submitted to IT].

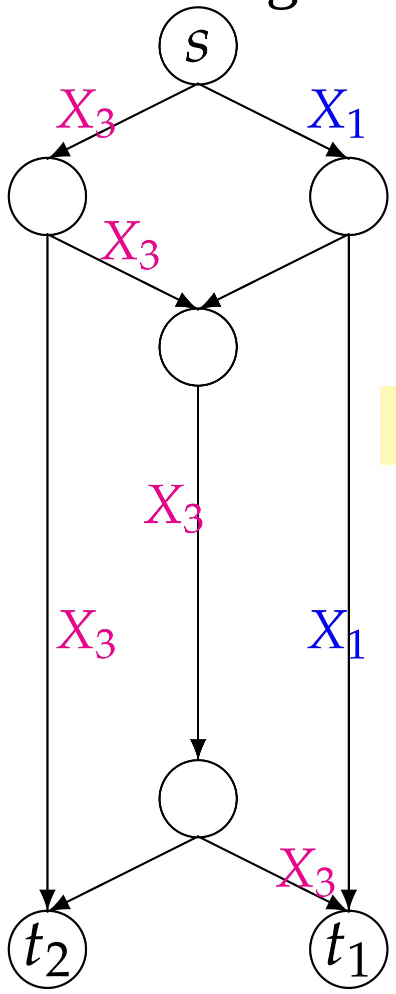
Single Session — Intra-session Network Coding

The Routing Solution



Single Session — Intra-session Network Coding

The Routing Solution

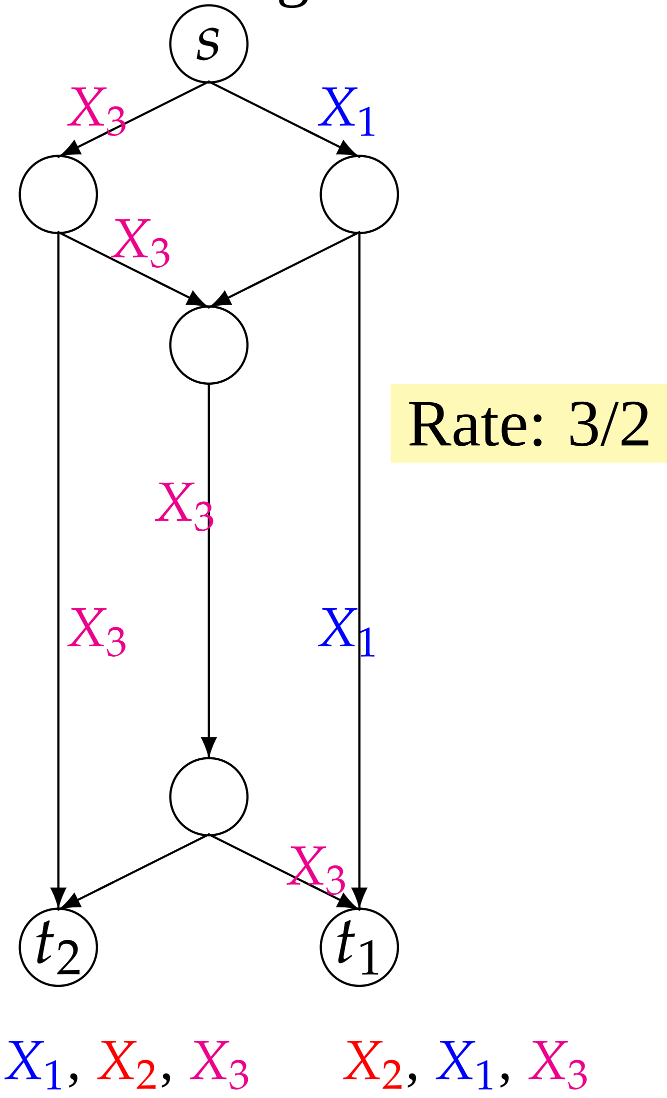


Rate: 3/2

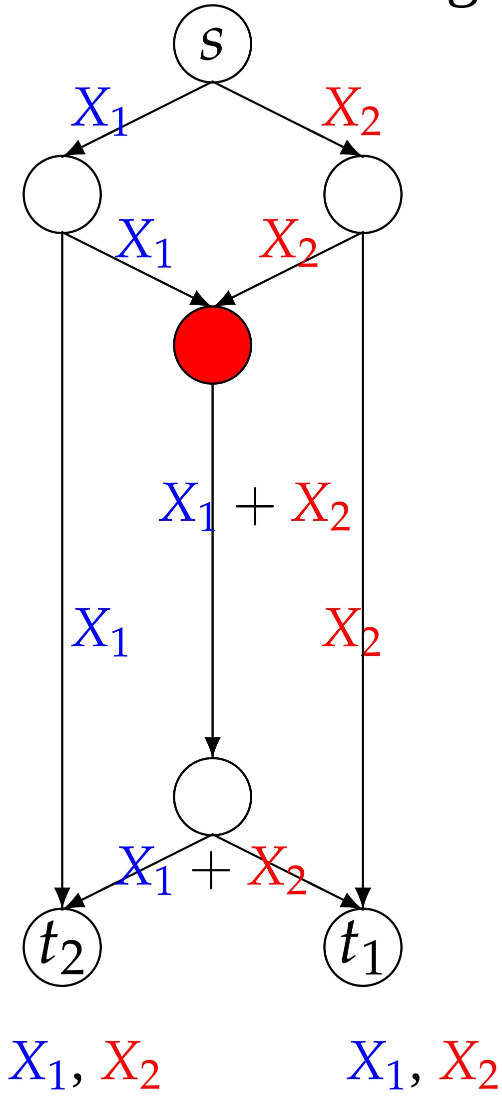
X_1, X_2, X_3 X_2, X_1, X_3

Single Session — Intra-session Network Coding

The Routing Solution

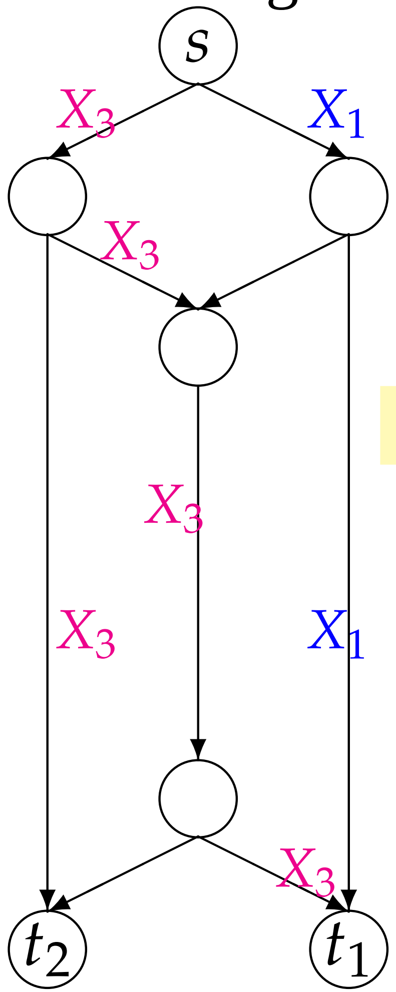


The Network Coding Solution



Single Session — Intra-session Network Coding

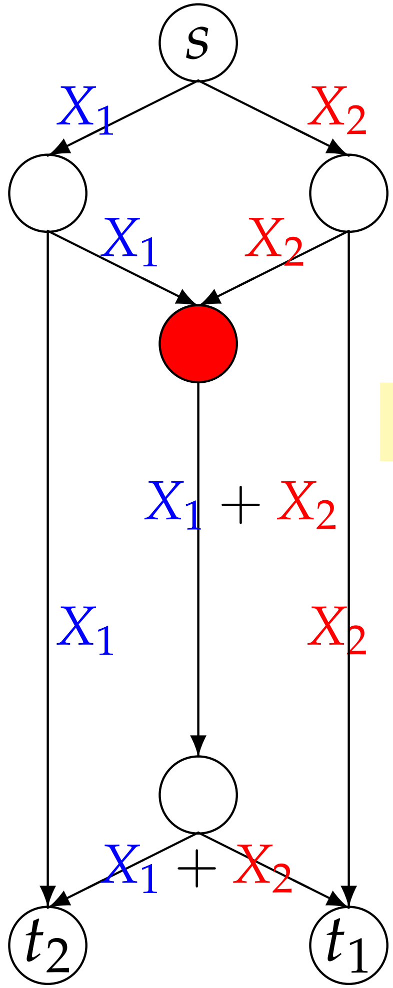
The Routing Solution



Rate: 3/2

X_1, X_2, X_3 X_2, X_1, X_3

The Network Coding Solution

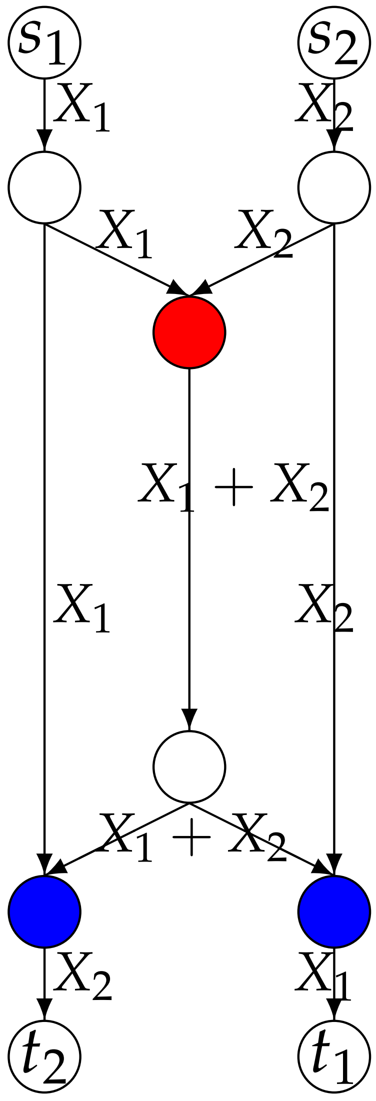


Rate: 2

X_1, X_2 X_1, X_2

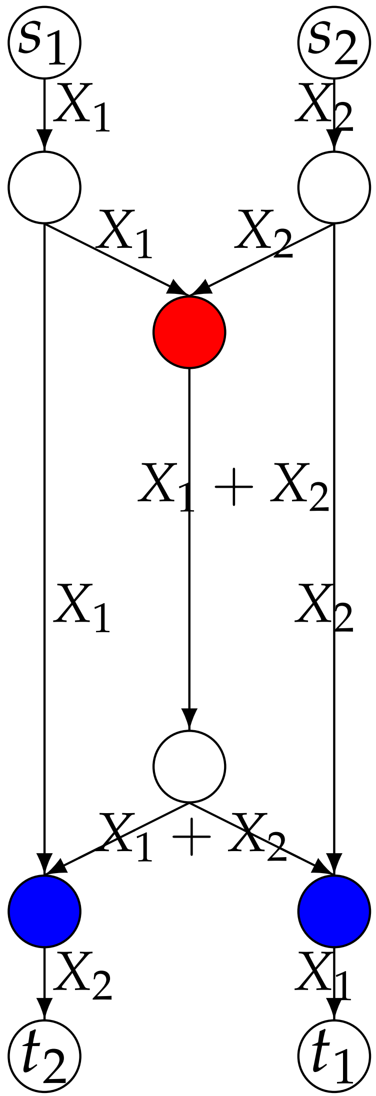
Multiple Sessions

- Inter-session network coding: The benefit is also apparent.



Multiple Sessions

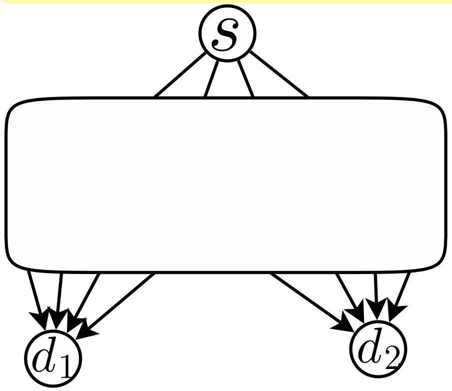
- Inter-session network coding: The benefit is also apparent.



For intra- and inter-session network coding, the corresponding hardness of realizing the coding benefits are fundamentally different.

Intra- versus Inter-session

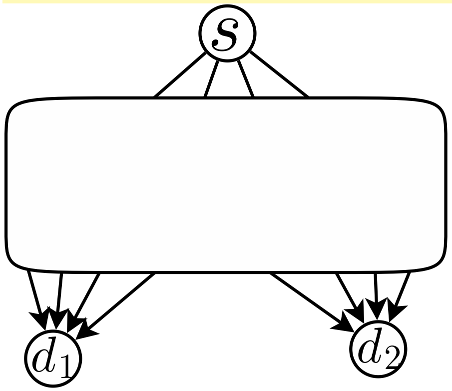
Intrasession network coding



$$M_1 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}, \quad M_2 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

Intra- versus Inter-session

Intrasession network coding

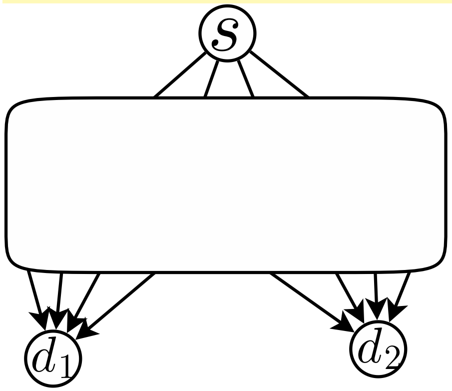


$$M_1 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}, \quad M_2 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

Only require $\det(\cdot) \neq 0$ condition.

Intra- versus Inter-session

Intrasession network coding



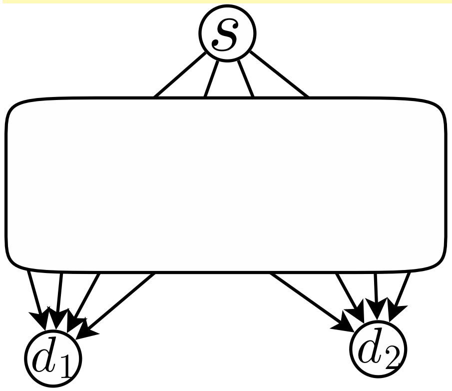
$$M_1 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}, \quad M_2 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

Only require $\det(\cdot) \neq 0$ condition.

So easy for a large $\mathsf{GF}(q)$, even a random network coding can do it.

Intra- versus Inter-session

Intrasession network coding

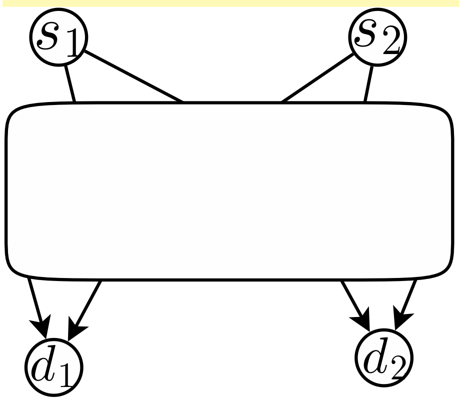


$$M_1 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}, \quad M_2 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

Only require $\det(\cdot) \neq 0$ condition.

So easy for a large $\mathsf{GF}(q)$, even a random network coding can do it.

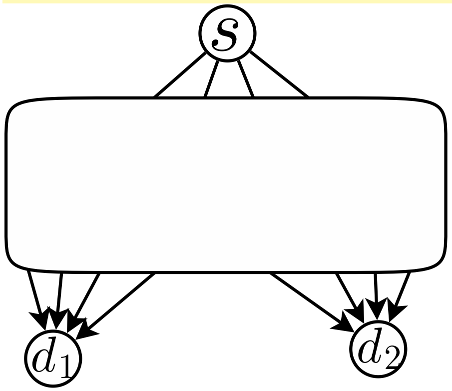
Intersession network coding



$$M_1 = \left[\begin{array}{cc|cc} \square & \square & \square & \square \\ \square & \square & \square & \square \end{array}\right], \quad M_2 = \left[\begin{array}{cc|cc} \square & \square & \square & \square \\ \square & \square & \square & \square \end{array}\right]$$

Intra- versus Inter-session

Intrasession network coding

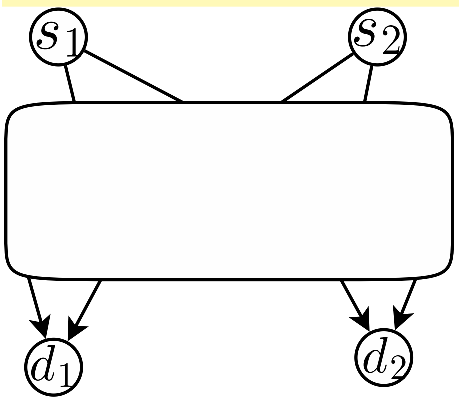


$$M_1 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}, \quad M_2 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

Only require $\det(\cdot) \neq 0$ condition.

So easy for a large $\mathsf{GF}(q)$, even a random network coding can do it.

Intersession network coding

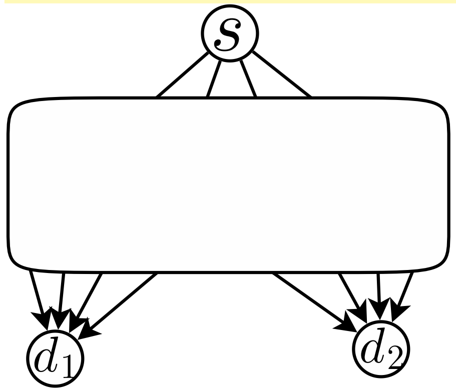


$$M_1 = \left[\begin{array}{cc|cc} \square & \square & \square & \square \\ \square & \square & \square & \square \end{array}\right], \quad M_2 = \left[\begin{array}{cc|cc} \square & \square & \square & \square \\ \square & \square & \square & \square \end{array}\right]$$

Require $\begin{bmatrix} \square & \square \\ \square & \square \end{bmatrix} = 0$.

Intra- versus Inter-session

Intrasession network coding

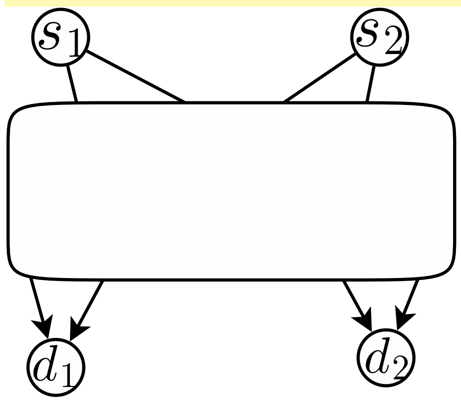


$$M_1 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}, \quad M_2 = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

Only require $\det(\cdot) \neq 0$ condition.

So easy for a large $\mathsf{GF}(q)$, even a random network coding can do it.

Intersession network coding



$$M_1 = \left[\begin{array}{cc|cc} \square & \square & \square & \square \\ \square & \square & \square & \square \end{array}\right], \quad M_2 = \left[\begin{array}{cc|cc} \square & \square & \square & \square \\ \square & \square & \square & \square \end{array}\right]$$

Require $\begin{bmatrix} \square & \square \\ \square & \square \end{bmatrix} = 0$. **Much harder!!**

Bounds for Intersession NC

- General graphs, $K \geq 2$ (Unicast) Sessions.

Bounds for Intersession NC

- General graphs, $K \geq 2$ (Unicast) Sessions.
- Pure inform.-theoretic approaches: Fundamental regions: [Song *et al.* 03], [Yan *et al.* 07], entropy calculus [Jain *et al.* 06]
  I.e. construct random variables satisfying entropy inequalities.

Bounds for Intersession NC

- General graphs, $K \geq 2$ (Unicast) Sessions.
- Pure inform.-theoretic approaches: Fundamental regions: [Song *et al.* 03], [Yan *et al.* 07], entropy calculus [Jain *et al.* 06]
  I.e. construct random variables satisfying entropy inequalities.
- Capacity outer bounds (nec. condition):
  - The cut conditions + Inform.-theoretic (side-information)

Bounds for Intersession NC

- General graphs, $K \geq 2$ (Unicast) Sessions.
- Pure inform.-theoretic approaches: Fundamental regions: [Song *et al.* 03], [Yan *et al.* 07], entropy calculus [Jain *et al.* 06]
  I.e. construct random variables satisfying entropy inequalities.
- Capacity outer bounds (nec. condition):
  - The cut conditions + Inform.-theoretic (side-information)
  - The network-sharing bound [2], the information dominance condition [1], and the edge-cut bounds [Kramer *et al.* 06].

Bounds for Intersession NC

- General graphs, $K \geq 2$ (Unicast) Sessions.
- Pure inform.-theoretic approaches: Fundamental regions: [Song *et al.* 03], [Yan *et al.* 07], entropy calculus [Jain *et al.* 06]
  I.e. construct random variables satisfying entropy inequalities.
- Capacity outer bounds (nec. condition):
  - The cut conditions + Inform.-theoretic (side-information)
  - The network-sharing bound [2], the information dominance condition [1], and the edge-cut bounds [Kramer *et al.* 06].
- Capacity inner bound (suff. condition, achievability):
  - The modified flow conditions + Linear programming.

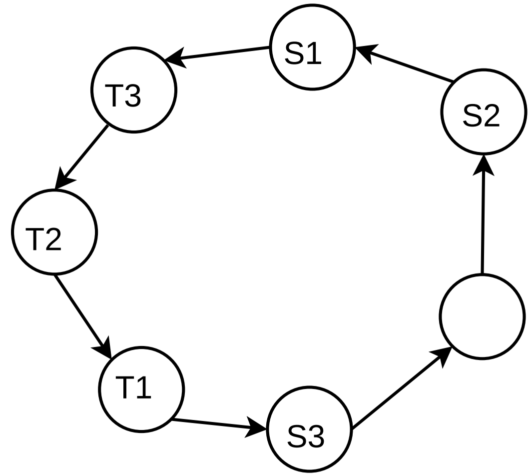
Bounds for Intersession NC

- General graphs, $K \geq 2$ (Unicast) Sessions.
- Pure inform.-theoretic approaches: Fundamental regions: [Song *et al.* 03], [Yan *et al.* 07], entropy calculus [Jain *et al.* 06]
  I.e. construct random variables satisfying entropy inequalities.
- Capacity outer bounds (nec. condition):
  - The cut conditions + Inform.-theoretic (side-information)
  - The network-sharing bound [2], the information dominance condition [1], and the edge-cut bounds [Kramer *et al.* 06].
- Capacity inner bound (suff. condition, achievability):
  - The modified flow conditions + Linear programming.
  - Butterfly-based construction [Traskov *et al.* 06], pollution-treatment [Wu 06].

Special Graphs w. Known Cap.

- Directed Cycles [1]

$$\sum_{i \text{ separated by } e} r_i \leq c(e)$$

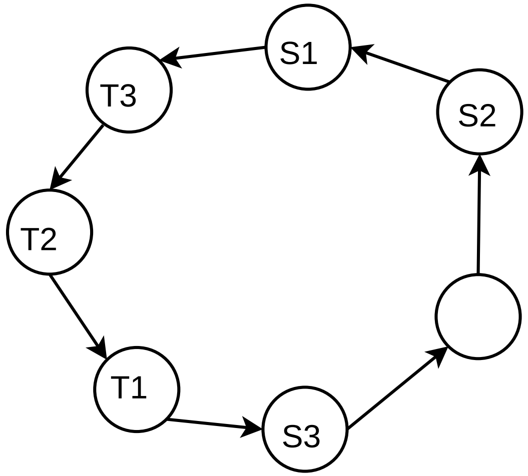


[1] Harvey *et al.* 06, IEEE Trans. IT; [2] Yan *et al.* 06, IEEE Trans. IT

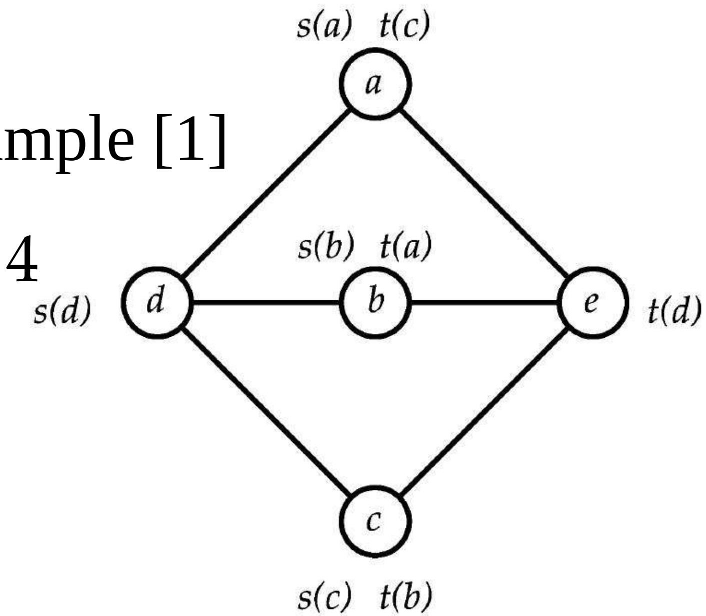
Special Graphs w. Known Cap.

- Directed Cycles [1]

$$\sum_{i \text{ separated by } e} r_i \leq c(e)$$



- The undirected Okamura-Seymour example [1]
  - Network coding = routing. $r = 3/4$

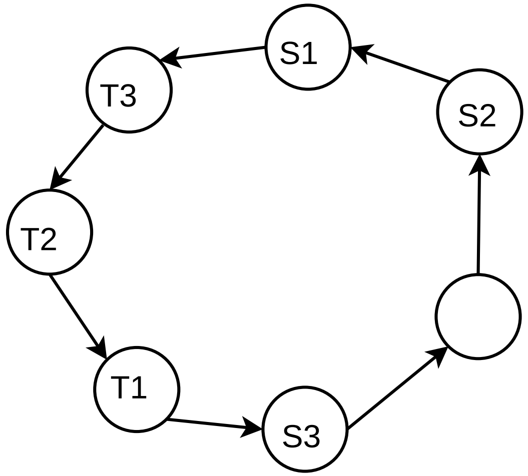


[1] Harvey *et al.* 06, IEEE Trans. IT; [2] Yan *et al.* 06, IEEE Trans. IT

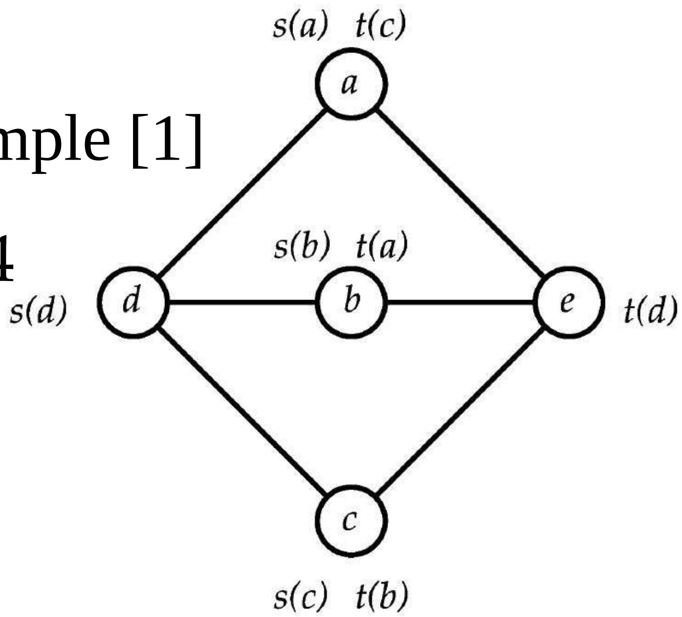
Special Graphs w. Known Cap.

- Directed Cycles [1]

$$\sum_{i \text{ separated by } e} r_i \leq c(e)$$



- The undirected Okamura-Seymour example [1]
  - Network coding = routing. $r = 3/4$



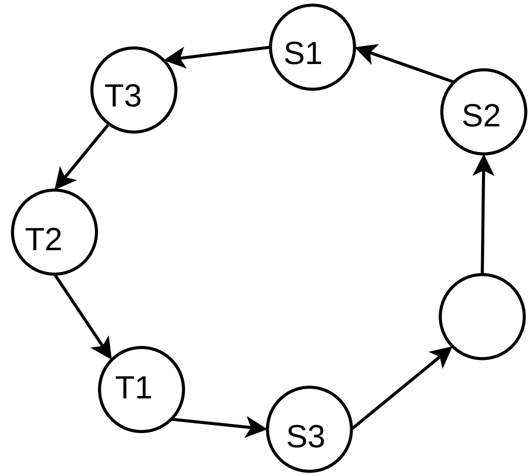
- Directed, acyclic, degree 2, three-layer networks [2]

[1] Harvey *et al.* 06, IEEE Trans. IT; [2] Yan *et al.* 06, IEEE Trans. IT

Special Graphs w. Known Cap.

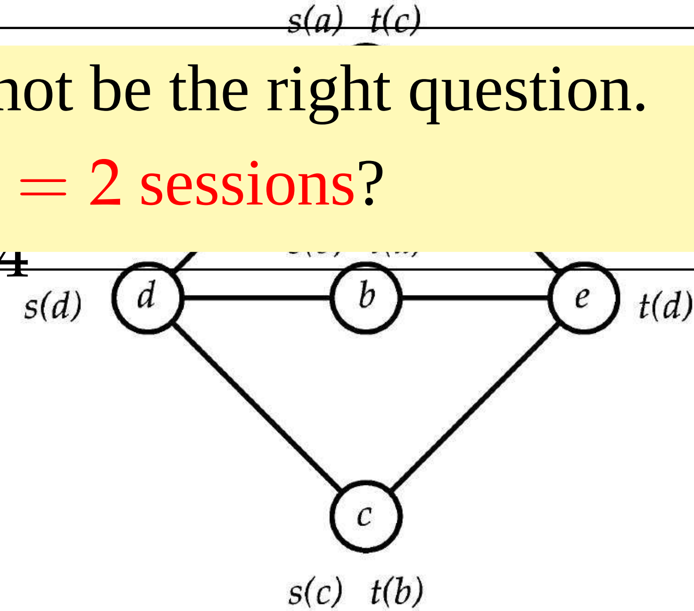
- Directed Cycles [1]

$$\sum_{i \text{ separated by } e} r_i \leq c(e)$$



"Special graph & $k > 1$ sessions" may not be the right question.
How about general graph & $k = 2$ sessions?

- Network coding = routing: $r = 3/4$

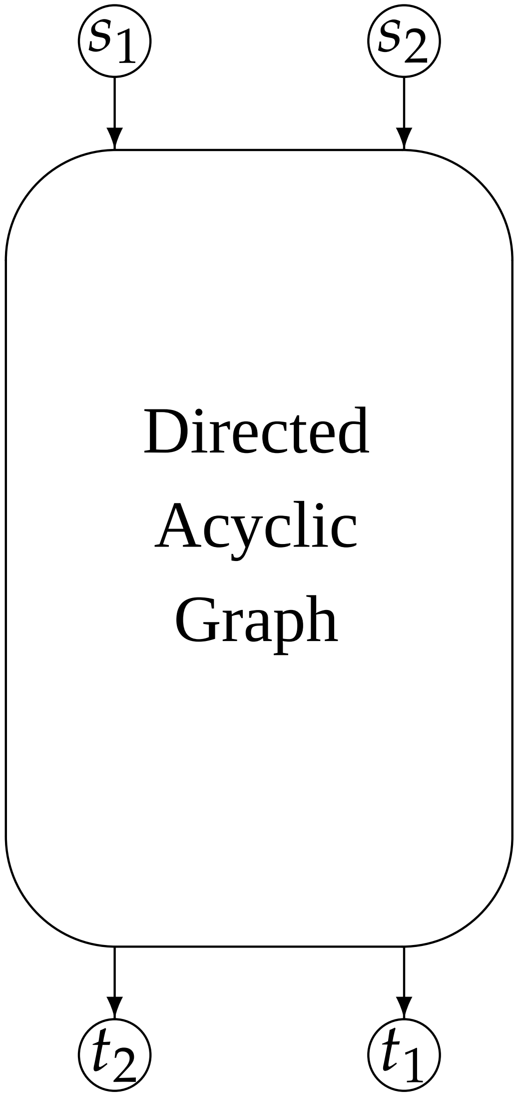


- Directed, acyclic, degree 2, three-layer networks [2]

[1] Harvey *et al.* 06, IEEE Trans. IT; [2] Yan *et al.* 06, IEEE Trans. IT

Two Simple Unicast Sessions

When can we send X_1 and X_2 simultaneously?

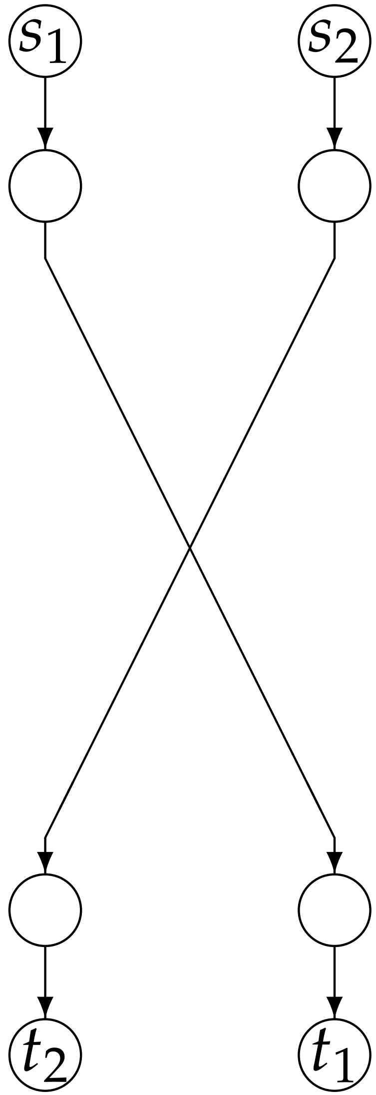


Two Simple Unicast Sessions

When can we send X_1 and X_2 simultaneously?

Routing solutions

$\Longleftrightarrow$ Edge disjoint paths



Two Simple Unicast Sessions

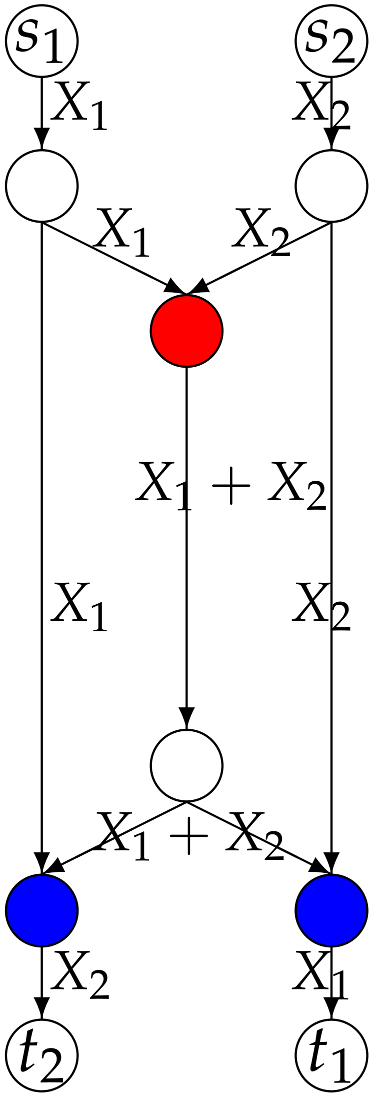
When can we send X_1 and X_2 simultaneously?

Routing solutions

$\Longleftrightarrow$ Edge disjoint paths

The existence of a butterfly

$\Longrightarrow$ Network coding solutions



Two Simple Unicast Sessions

When can we send X_1 and X_2 simultaneously?

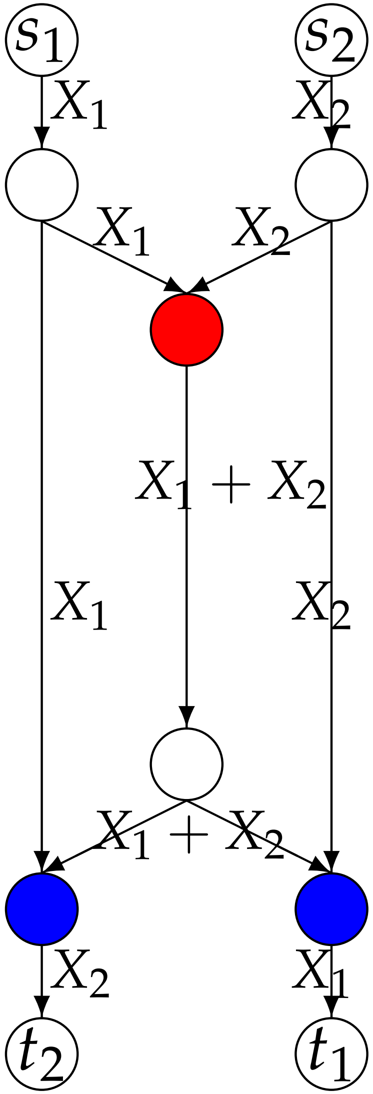
Routing solutions

$\Longleftrightarrow$ Edge disjoint paths

The existence of a butterfly

$\Longrightarrow$ Network coding solutions

Vice versa?



Two Simple Unicast Sessions

When can we send X_1 and X_2 simultaneously?

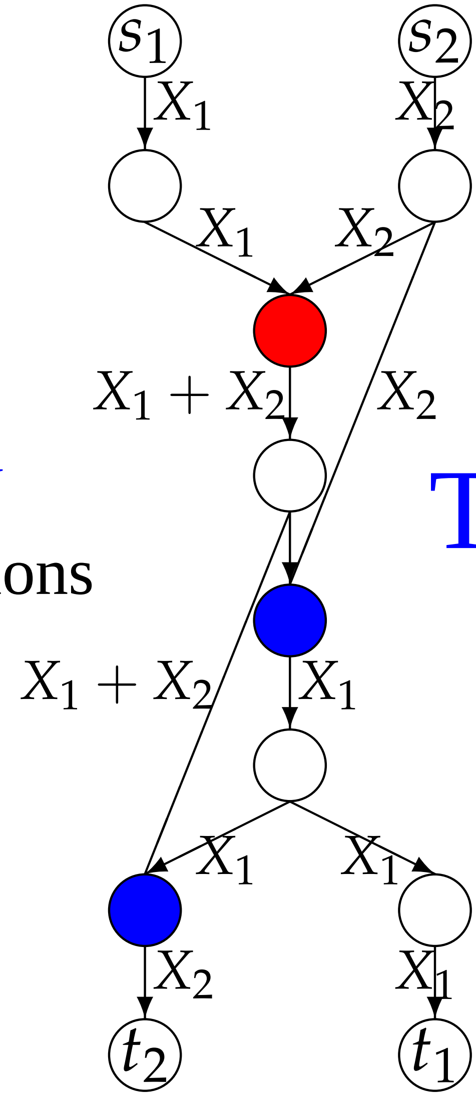
Routing solutions

$\Longleftrightarrow$ Edge disjoint paths

The existence of a butterfly

$\Longrightarrow$ Network coding solutions

Vice versa?



The grail structure

Two Simple Unicast Sessions

When can we send X_1 and X_2 simultaneously?

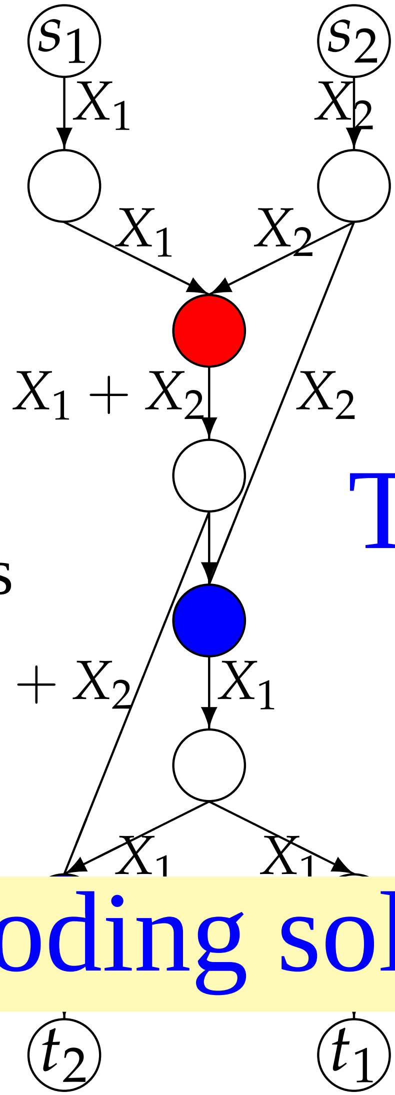
Routing solutions

$\Longleftrightarrow$ Edge disjoint paths

The existence of a butterfly

$\Longrightarrow$ Network coding solutions

Vice versa?



The grail structure

Q: Network coding solutions $\Leftrightarrow$???

Two Simple Multicast Sessions

When can we send X_1 and X_2 simultaneously?

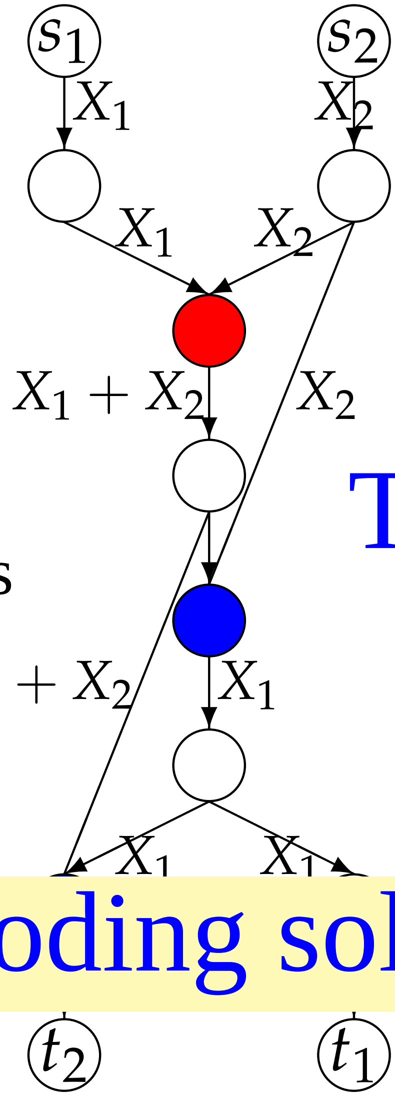
Routing solutions

$\Longleftrightarrow$ Edge disjoint paths

The existence of a butterfly

$\Longrightarrow$ Network coding solutions

Vice versa?



The grail structure

Q: Network coding solutions $\Leftrightarrow$???

Two Simple Multicast Sessions

When can we send X_1 and X_2 simultaneously?

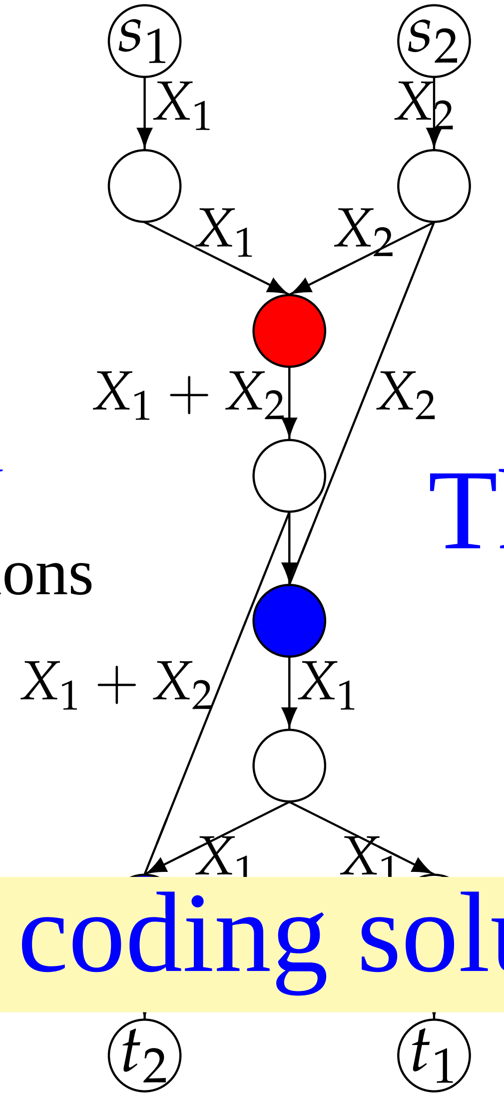
Routing solutions

$\iff$ Edge disjoint paths

The existence of a butterfly

$\implies$ Network coding solutions

Vice versa?



Cyclic graphs?

The grail structure

Q: Network coding solutions $\Leftrightarrow$???

The Char. Thm. For 2 Unicasts

- Setting: General finite directed acyclic graphs, unit edge capacity, (s_1, t_1) & (s_2, t_2), two integer symbols X_1 and X_2.
- Number of Coinciding Paths of edge e: $\mathcal{P} = \{P_1, \cdots, P_k\}$, and $\mathsf{ncp}_{\mathcal{P}}(e) = |\{P \in \mathcal{P} : e \in P\}|$.

The Char. Thm. For 2 Unicasts

- Setting: General finite directed acyclic graphs, unit edge capacity, (s_1, t_1) & (s_2, t_2), two integer symbols X_1 and X_2.
- Number of Coinciding Paths of edge e: $\mathcal{P} = \{P_1, \cdots, P_k\}$, and $\mathsf{ncp}_{\mathcal{P}}(e) = |\{P \in \mathcal{P} : e \in P\}|$.

Theorem 1 *Network coding* $\Longleftrightarrow$ *one of the following two holds.*

1. $\exists \mathcal{P} = \{P_{s_1,t_1}, P_{s_2,t_2}\}$, *such that*

$$\max_{e \in E} \mathsf{ncp}_{\mathcal{P}}(e) \leq 1.$$

2. $\exists \mathcal{P} = \{P_{s_1,t_1}, P_{s_2,t_2}, P_{s_2,t_1}\}$ *and* $\mathcal{Q} = \{Q_{s_1,t_1}, Q_{s_2,t_2}, Q_{s_1,t_2}\}$ *s.t.*

$$\max_{e \in E} \mathsf{ncp}_{\mathcal{P}}(e) \leq 2 \text{ and } \max_{e \in E} \mathsf{ncp}_{\mathcal{Q}}(e) \leq 2.$$

The Char. Thm. For 2 Unicasts

- Setting: General finite directed acyclic graphs, unit edge capacity, (s_1, t_1) & (s_2, t_2), two integer symbols X_1 and X_2.
- Number of Coinciding Paths of edge e: $\mathcal{P} = \{P_1, \cdots, P_k\}$, and $\mathsf{ncp}_{\mathcal{P}}(e) = |\{P \in \mathcal{P} : e \in P\}|$.

Theorem 1 *Network coding $\iff$ one of the following two holds.*

1. $\exists \mathcal{P} = \{P_{s_1,t_1}, P_{s_2,t_2}\}$, *such that*

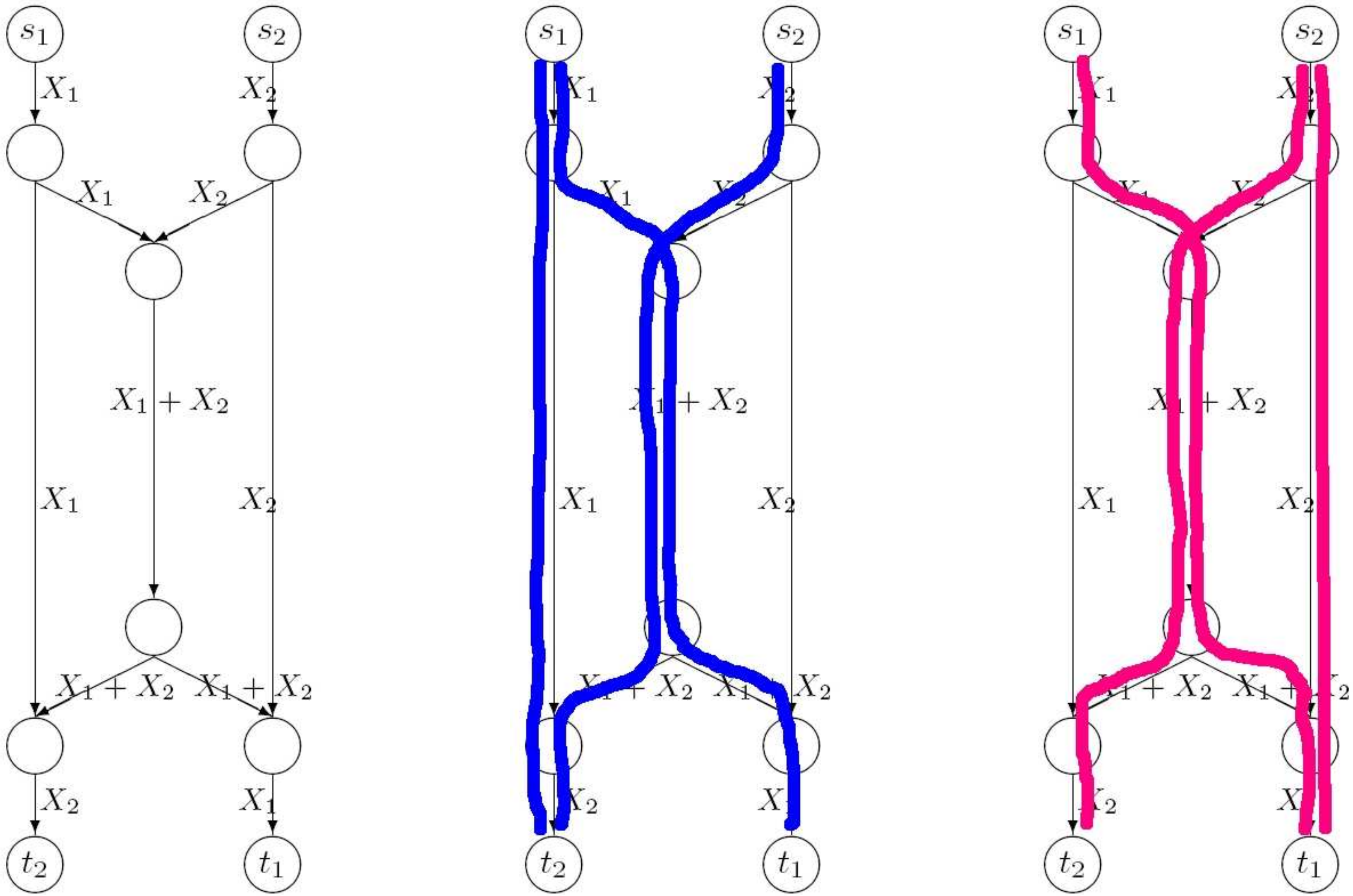
$$\max_{e \in E} \mathsf{ncp}_{\mathcal{P}}(e) \leq 1.$$

2. $\exists \mathcal{P} = \{P_{s_1,t_1}, P_{s_2,t_2}, P_{s_2,t_1}\}$ *and* $\mathcal{Q} = \{Q_{s_1,t_1}, Q_{s_2,t_2}, Q_{s_1,t_2}\}$ *s.t.*

$$\max_{e \in E} \mathsf{ncp}_{\mathcal{P}}(e) \leq 2 \text{ and } \max_{e \in E} \mathsf{ncp}_{\mathcal{Q}}(e) \leq 2.$$

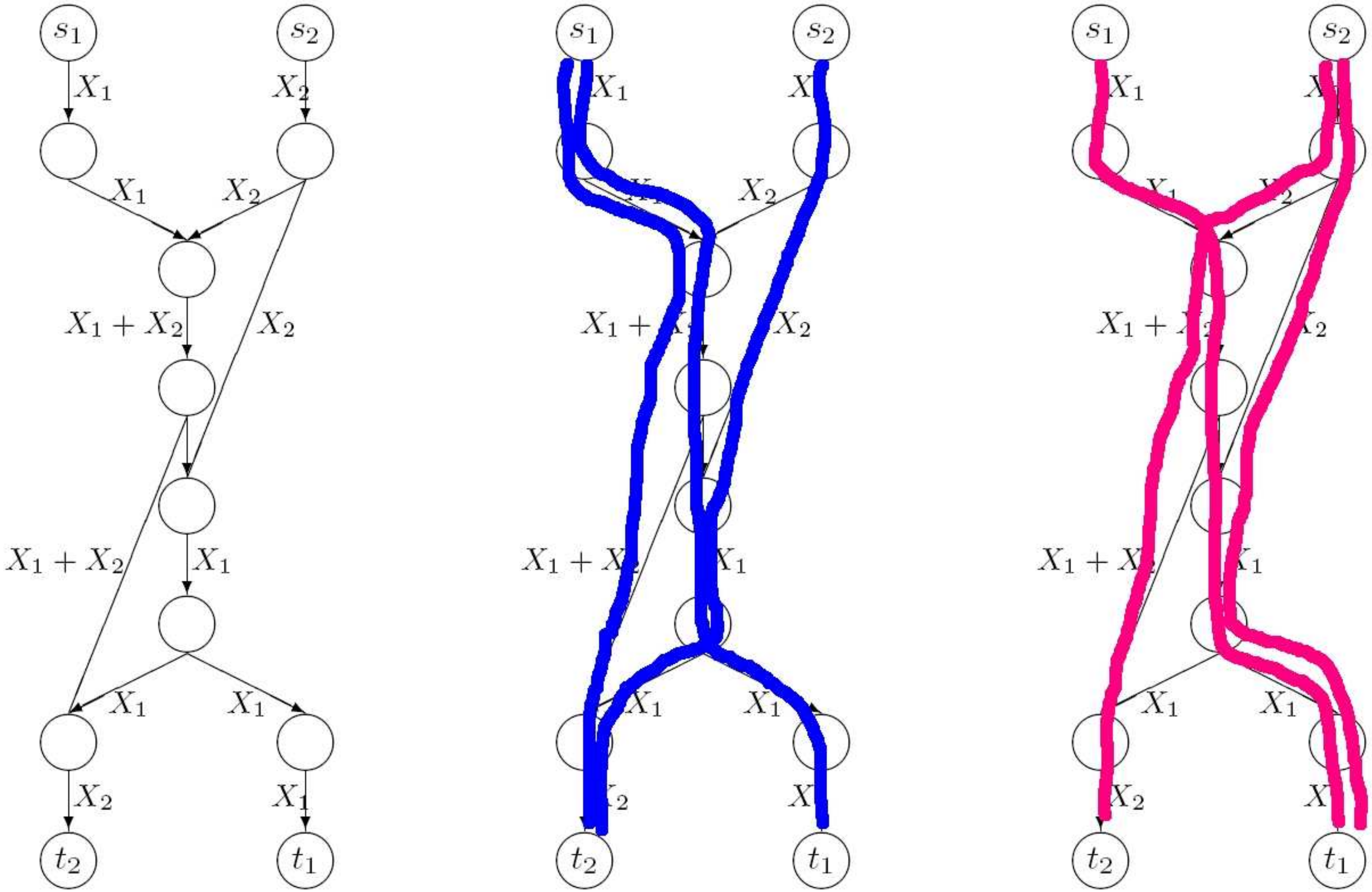
Routing: edge disjointness vs. Network coding: controlled overlaps.

Feasible Example: The Butterfly



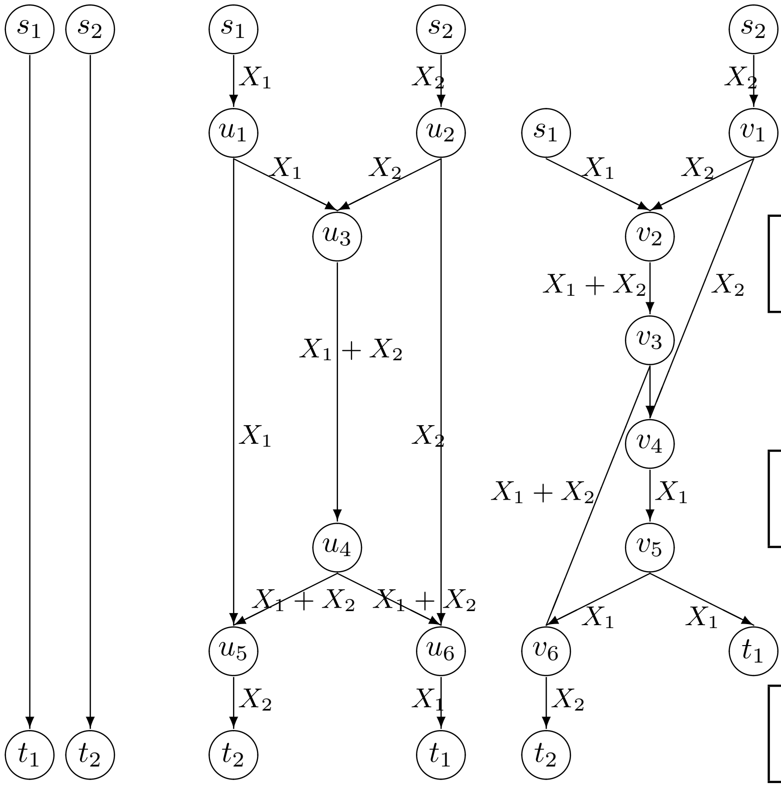
$\mathcal{Q} = \{Q_{s_1,t_1}, Q_{s_2,t_2}, Q_{s_1,t_2}\}$ $\mathcal{P} = \{P_{s_1,t_1}, P_{s_2,t_2}, P_{s_2,t_1}\}$

Feasible Example 2: The Grail



$\mathcal{Q} = \{Q_{s_1,t_1}, Q_{s_2,t_2}, Q_{s_1,t_2}\}$ $\mathcal{P} = \{P_{s_1,t_1}, P_{s_2,t_2}, P_{s_2,t_1}\}$

Strengthened Results for 2 Uni-casts



The graph is 2-EDP, a full butterfly, or a full grail.

by figures $\Downarrow$ $\Uparrow$ is also true

There exists a network coding solution.

$\Updownarrow$

There exist $\mathcal{P}$ and $\mathcal{Q}$ as described in Theorem 1.

Is this controlled edge overlap condition of the right form?

Is this controlled edge overlap condition of the right form?

Possible (at least for linear network coding).

Is this controlled edge overlap condition of the right form?

Possible (at least for linear network coding).

- Linear network coding focuses on integer-valued “rank" conditions while non-linear network coding focuses on real-valued “entropy."

Is this controlled edge overlap condition of the right form?

Possible (at least for linear network coding).

- Linear network coding focuses on integer-valued “rank" conditions while non-linear network coding focuses on real-valued “entropy."
- Controlled edge overlap serves as the char. thm. for other settings:
  - 2 simple unicast traffic in acyclic networks.

Is this controlled edge overlap condition of the right form?

Possible (at least for linear network coding).

- Linear network coding focuses on integer-valued “rank" conditions while non-linear network coding focuses on real-valued “entropy."
- Controlled edge overlap serves as the char. thm. for other settings:
  - 2 simple unicast traffic in acyclic networks.
  - 2 simple multicast traffic in acyclic networks.

Is this controlled edge overlap condition of the right form?

Possible (at least for linear network coding).

- Linear network coding focuses on integer-valued “rank" conditions while non-linear network coding focuses on real-valued “entropy."
- Controlled edge overlap serves as the char. thm. for other settings:
  - 2 simple unicast traffic in acyclic networks.
  - 2 simple multicast traffic in acyclic networks.
  - 2 simple unicast traffic in cyclic networks.

The Char. Thm. For 2 Multicasts

- Setting: General finite directed acyclic graphs, unit edge capacity, $(s_1, \{t_{1,i}\}_i)$ & $(s_2, \{t_{2,j}\}_j)$, two integer symbols X_1 and X_2.

The Char. Thm. For 2 Multicasts

- Setting: General finite directed acyclic graphs, unit edge capacity, $(s_1, \{t_{1,i}\}_i)$ & $(s_2, \{t_{2,j}\}_j)$, two integer symbols X_1 and X_2.

Theorem 2 *The existence of intersession network coding* $\Leftrightarrow$

$$\exists \mathcal{P} = \{P_{s_1,t_{1,i}}, P_{s_2,t_{1,i}} : \forall i\} \cup \{P_{s_2,t_{2,j}} : \forall j\},$$
$$\exists \mathcal{Q} = \{Q_{s_2,t_{2,j}}, Q_{s_1,t_{2,j}} : \forall j\} \cup \{Q_{s_1,t_{1,i}} : \forall i\},$$

such that

$$\max_{e \in E} \mathsf{ncp}_{\{P_{s_1,t_{1,i}}, P_{s_2,t_{1,i}}, P_{s_2,t_{2,j}}\}}(e) \leq 2, \quad \forall i, j,$$

and

$$\max_{e \in E} \mathsf{ncp}_{\{Q_{s_2,t_{2,j}}, Q_{s_1,t_{2,j}}, Q_{s_1,t_{1,i}}\}}(e) \leq 2, \quad \forall i, j.$$

The Char. Thm. For 2 Multicasts

- Setting: General finite directed acyclic graphs, unit edge capacity, $(s_1, \{t_{1,i}\}_i)$ & $(s_2, \{t_{2,j}\}_j)$, two integer symbols X_1 and X_2.

Theorem 2 *The existence of intersession network coding* $\Leftrightarrow$

$$\exists \mathcal{P} = \{\overset{1\longrightarrow 1}{P_{s_1,t_{1,i}}}, \overset{2\longrightarrow 1}{P_{s_2,t_{1,i}}} : \forall i\} \cup \{\overset{2\longrightarrow 2}{P_{s_2,t_{2,j}}} : \forall j\},$$

$$\exists \mathcal{Q} = \{\underset{2\longrightarrow 2}{Q_{s_2,t_{2,j}}}, \underset{1\longrightarrow 2}{Q_{s_1,t_{2,j}}} : \forall j\} \cup \{\underset{1\longrightarrow 1}{Q_{s_1,t_{1,i}}} : \forall i\},$$

such that

$$\max_{e\in E} \mathsf{ncp}_{\{P_{s_1,t_{1,i}}, P_{s_2,t_{1,i}}, P_{s_2,t_{2,j}}\}}(e) \leq 2, \quad \forall i, j,$$

and

$$\max_{e\in E} \mathsf{ncp}_{\{Q_{s_2,t_{2,j}}, Q_{s_1,t_{2,j}}, Q_{s_1,t_{1,i}}\}}(e) \leq 2, \quad \forall i, j.$$

The Char. Thm. For 2 Multicasts

- Setting: General finite directed acyclic graphs, unit edge capacity, $(s_1, \{t_{1,i}\}_i)$ & $(s_2, \{t_{2,j}\}_j)$, two integer symbols X_1 and X_2.

Theorem 2 *The existence of intersession network coding* $\Leftrightarrow$

$$\exists \mathcal{P} = \{\overset{1\longrightarrow 1}{P_{s_1,t_{1,i}}}, \overset{2\longrightarrow 1}{P_{s_2,t_{1,i}}} : \forall i\} \cup \{\overset{2\longrightarrow 2}{P_{s_2,t_{2,j}}} : \forall j\},$$

$$\exists \mathcal{Q} = \{\underset{2\longrightarrow 2}{Q_{s_2,t_{2,j}}}, \underset{1\longrightarrow 2}{Q_{s_1,t_{2,j}}} : \forall j\} \cup \{\underset{1\longrightarrow 1}{Q_{s_1,t_{1,i}}} : \forall i\},$$

such that

Choose paths for i and j separately.

$$\max_{e\in E} \mathsf{ncp}_{\{P_{s_1,t_{1,i}}, P_{s_2,t_{1,i}}, P_{s_2,t_{2,j}}\}}(e) \leq 2, \quad \forall i, j,$$

and

$$\max_{e\in E} \mathsf{ncp}_{\{Q_{s_2,t_{2,j}}, Q_{s_1,t_{2,j}}, Q_{s_1,t_{1,i}}\}}(e) \leq 2, \quad \forall i, j.$$

Then the conditions have to be satisfied for all (i, j) combinations.

2 Unicasts in Cyclic Networks

- Each edge: 1 GF(q) symbol/sec and propagation delay 1 sec.

2 Unicasts in Cyclic Networks

- Each edge: 1 $\mathsf{GF}(q)$ symbol/sec and propagation delay 1 sec.
- Two sessions (s_1, d_1) and (s_2, d_2).

2 Unicasts in Cyclic Networks

- Each edge: 1 GF(q) symbol/sec and propagation delay 1 sec.
- Two sessions (s_1, d_1) and (s_2, d_2).
- Send two strings of symbols $X_1, X_2, \cdots, X_t$ and $Y_1, Y_2, \cdots, Y_t$.

2 Unicasts in Cyclic Networks

- Each edge: 1 GF(q) symbol/sec and propagation delay 1 sec.
- Two sessions (s_1, d_1) and (s_2, d_2).
- Send two strings of symbols $X_1, X_2, \cdots, X_t$ and $Y_1, Y_2, \cdots, Y_t$.
- A network coding solution exists iff

$$\frac{1}{T} I([X]_1^T; [M_{d_1}]_1^T) > (1-\epsilon)\log(q)$$
$$\text{and } \frac{1}{T} I([Y]_1^T; [M_{d_2}]_1^T) > (1-\epsilon)\log(q),$$

2 Unicasts in Cyclic Networks

- Each edge: 1 GF(q) symbol/sec and propagation delay 1 sec.
- Two sessions (s_1, d_1) and (s_2, d_2).
- Send two strings of symbols $X_1, X_2, \cdots, X_t$ and $Y_1, Y_2, \cdots, Y_t$.
- A network coding solution exists iff

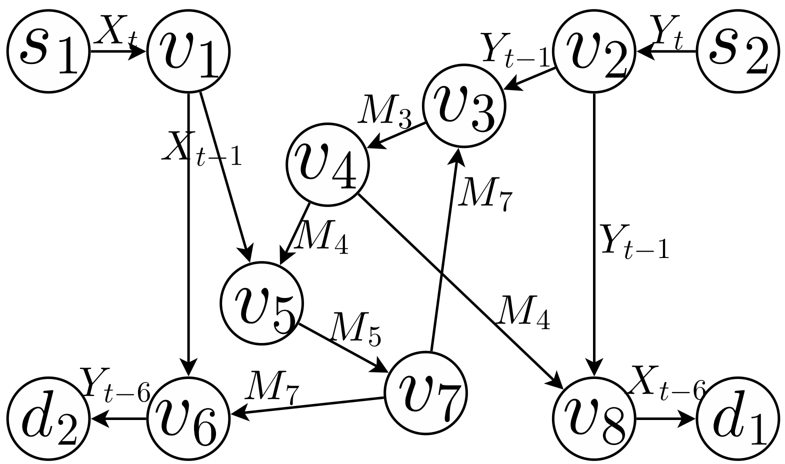
$$\frac{1}{T} I([X]_1^T; [M_{d_1}]_1^T) > (1-\epsilon)\log(q)$$

$$\text{and } \frac{1}{T} I([Y]_1^T; [M_{d_2}]_1^T) > (1-\epsilon)\log(q),$$

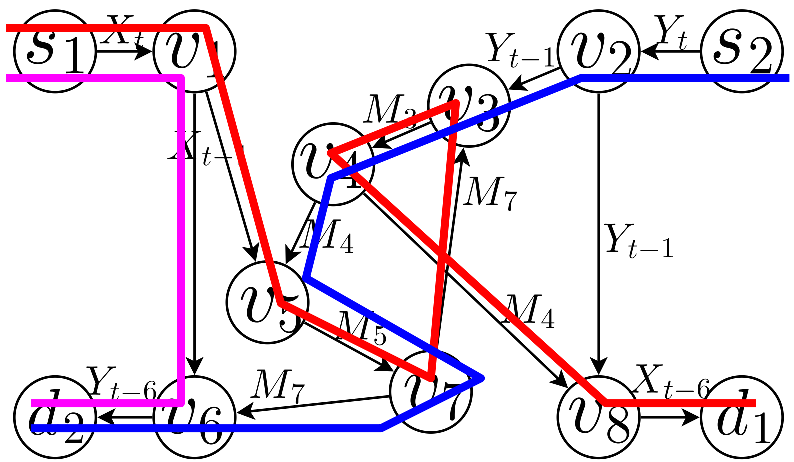
Theorem 3 *Network coding $\Longleftrightarrow$ one of the following two holds.*

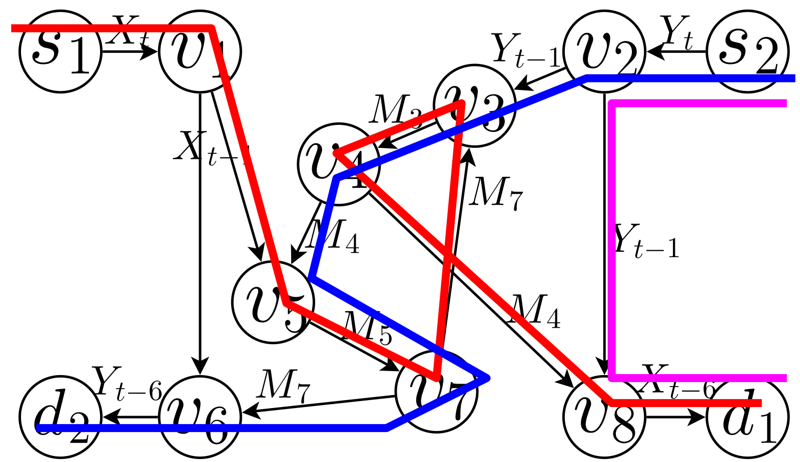
1. $\exists \mathcal{P} = \{P_{s_1,t_1}, P_{s_2,t_2}\}$ *that are edge-disjoint.*
2. $\exists \mathcal{P} = \{P_{s_1,t_1}, P_{s_2,t_2}, P_{s_2,t_1}\}$ *and* $\mathcal{Q} = \{Q_{s_1,t_1}, Q_{s_2,t_2}, Q_{s_1,t_2}\}$ *that have controlled edge overlap.*

One Cyclic Example

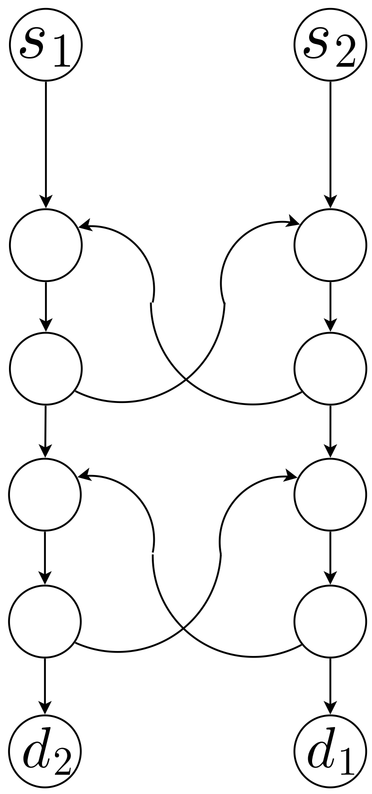


$$
\begin{aligned}
M_3 &= X_{t-4} + Y_{t-2} \\
M_4 &= X_{t-5} + Y_{t-3} \\
M_5 &= X_{t-2} + Y_{t-4} \\
M_7 &= X_{t-3} + Y_{t-5}
\end{aligned}
$$

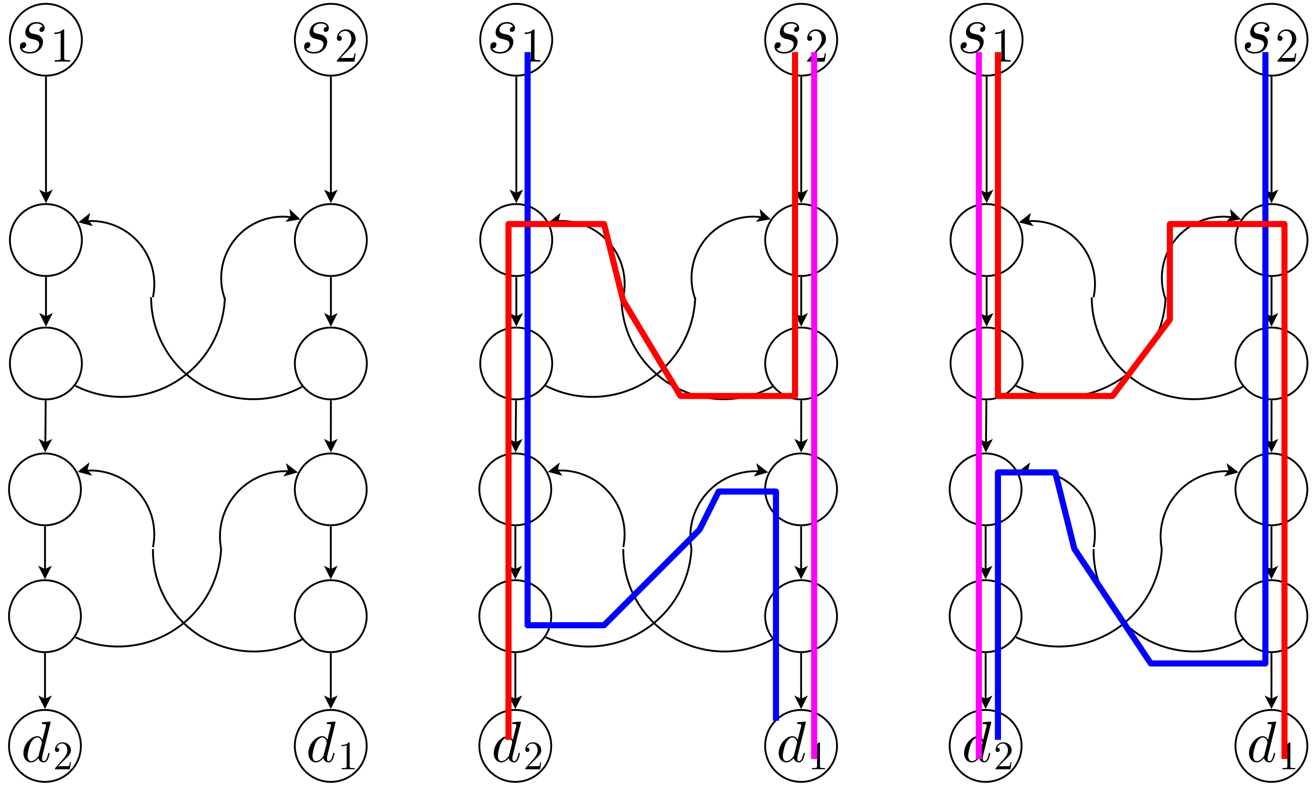




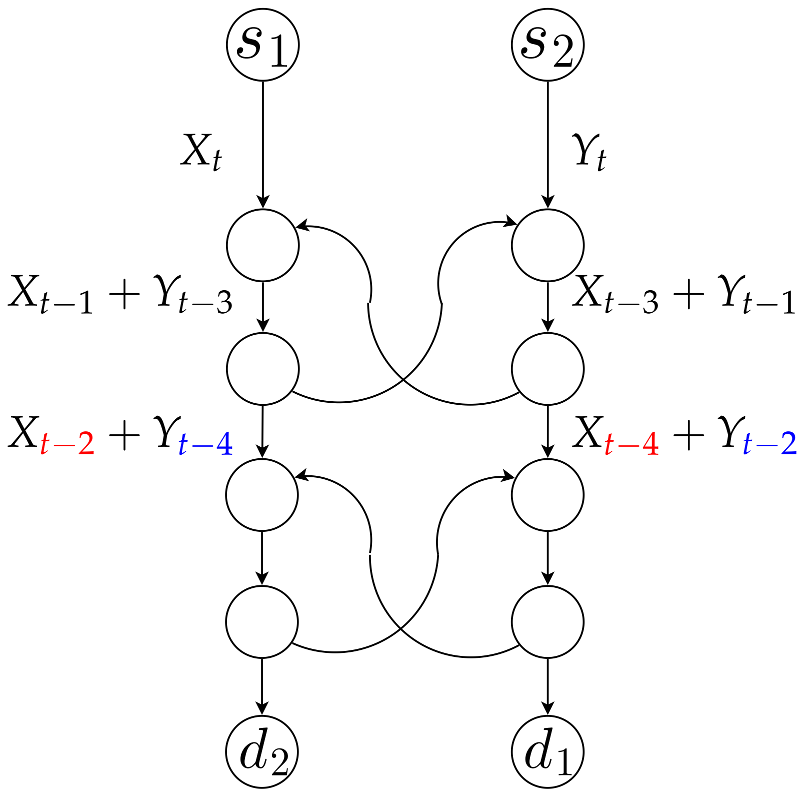
A Special Example

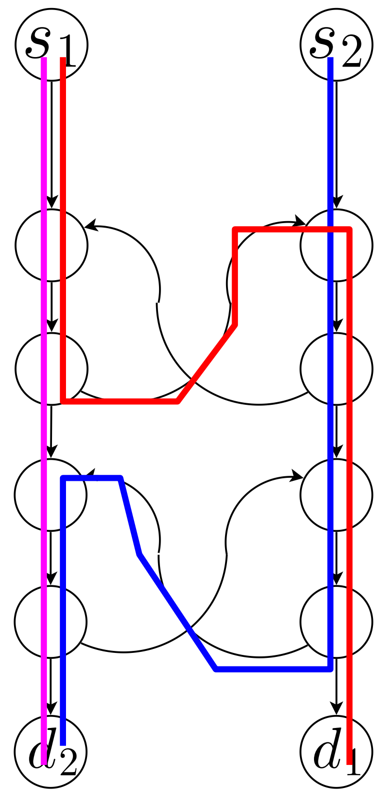


A Special Example



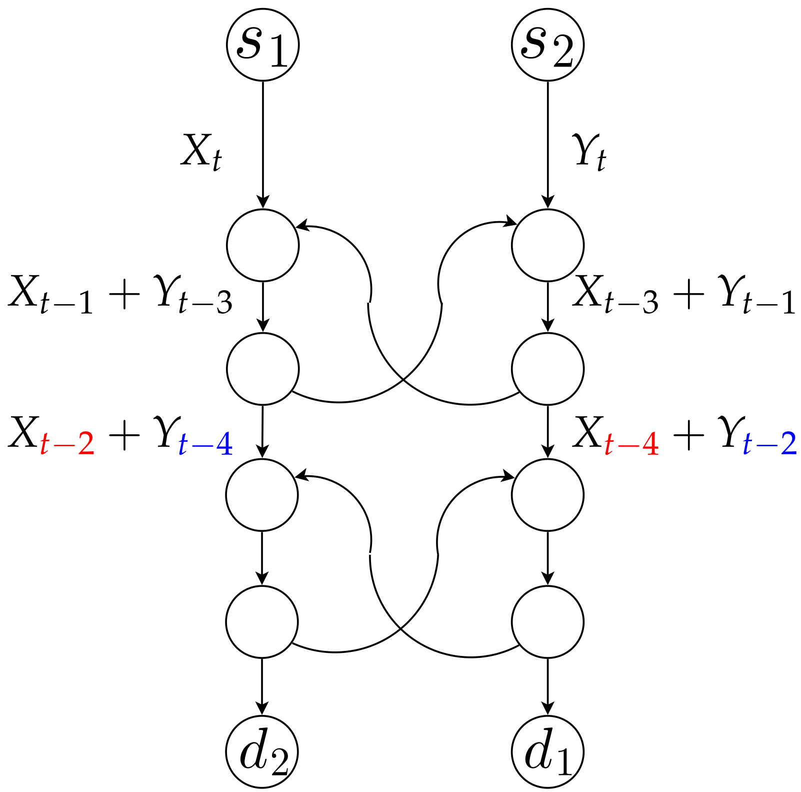
A Special Example

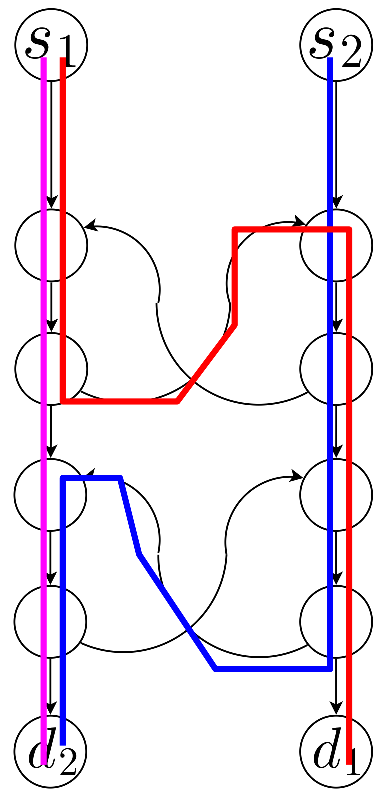




- A non-trivial example due to the causality of delays.

A Special Example





- A non-trivial example due to the causality of delays.
- The achievability is proven by FILO queues.

Implications

- The new basic unit of communications — from **edge-disjoint paths** to **controlled-edge-overlap paths**. Rate control and wireless scheduling [Khreishah *et al.* 07 & 08].

Implications

- The new basic unit of communications — from **edge-disjoint paths** to **controlled-edge-overlap paths**. Rate control and wireless scheduling [Khreishah *et al*. 07 & 08].
- **Sufficiency** of linear network codes for 2 unicasts.

Implications

- The new basic unit of communications — from **edge-disjoint paths** to **controlled-edge-overlap paths**. Rate control and wireless scheduling [Khreishah *et al.* 07 & 08].
- **Sufficiency** of linear network codes for 2 unicasts.
- **Complexity** of deciding the feasibility of 2 unicasts:

	Non-coded (edge-disjoint)	Ntwk Coding (controlled overlap)
acyclic	Poly$(\|G\|)$	
cyclic	NP-complete	

Implications

- The new basic unit of communications — from **edge-disjoint paths** to **controlled-edge-overlap paths**. Rate control and wireless scheduling [Khreishah *et al.* 07 & 08].
- **Sufficiency** of linear network codes for 2 unicasts.
- **Complexity** of deciding the feasibility of 2 unicasts:

	Non-coded (edge-disjoint)	Ntwk Coding (controlled overlap)
acyclic	Poly$(\|G\|)$	Poly$(\|G\|)$
cyclic	NP-complete	Poly$(\|G\|)$

Implications

- The new basic unit of communications — from **edge-disjoint paths** to **controlled-edge-overlap paths**. Rate control and wireless scheduling [Khreishah *et al*. 07 & 08].
- **Sufficiency** of linear network codes for 2 unicasts.
- **Complexity** of deciding the feasibility of 2 unicasts:

	Non-coded (edge-disjoint)	Ntwk Coding (controlled overlap)
acyclic	Poly($|G|$)	Poly($|G|$)
cyclic	NP-complete	Poly($|G|$)

- **Bandwidth optimality.** No need to use other than the paths with controlled edge overlap.

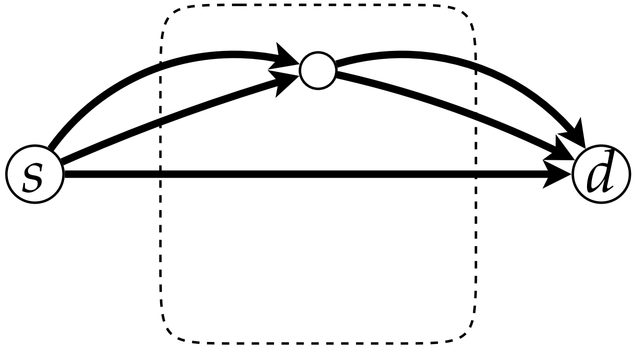
Part 1: Characterization of Intersession Network Coding

Part 1: Characterization of Intersession Network Coding

Part 2: Algorithmic study of Intrasession Network Coding

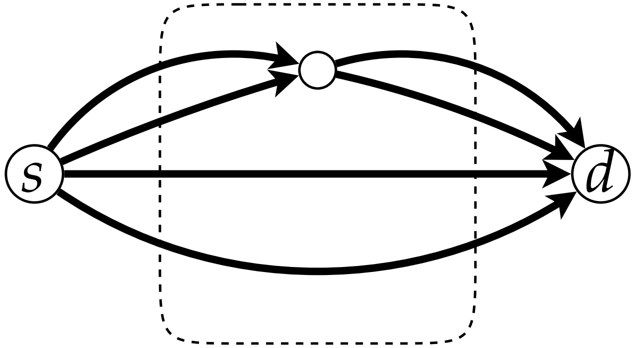
Bandwidth-efficiency

(s, d)-Flow, max (s, d)-flow, and the max-flow value (MFV).



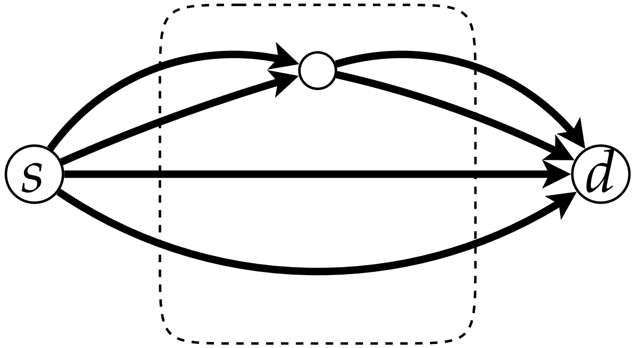
Bandwidth-efficiency

(s, d)-Flow, max (s, d)-flow, and the max-flow value (MFV).



Bandwidth-efficiency

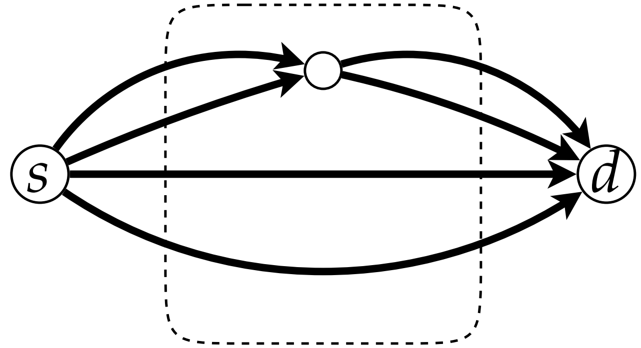
(s, d)-Flow, max (s, d)-flow, and the max-flow value (MFV).



Why study the max flow problem?

Bandwidth-efficiency

(s, d)-Flow, max (s, d)-flow, and the max-flow value (MFV).

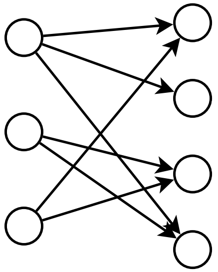


Why study the max flow problem?

- CS: a classic optimization problem:

Bandwidth-efficiency

(s,d)-Flow, max (s,d)-flow, and the max-flow value (MFV).

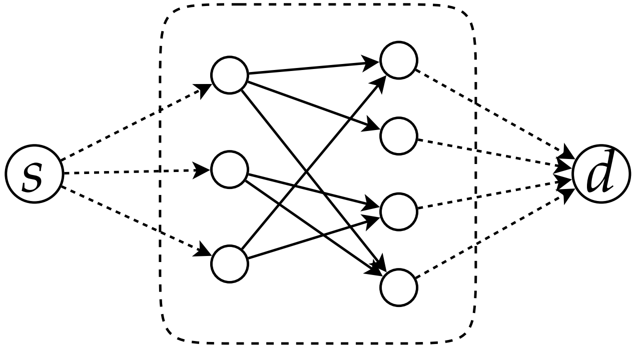


Why study the max flow problem?

- CS: a classic optimization problem:
  ex: finding maximum matching,

Bandwidth-efficiency

(s, d)-Flow, max (s, d)-flow, and the max-flow value (MFV).



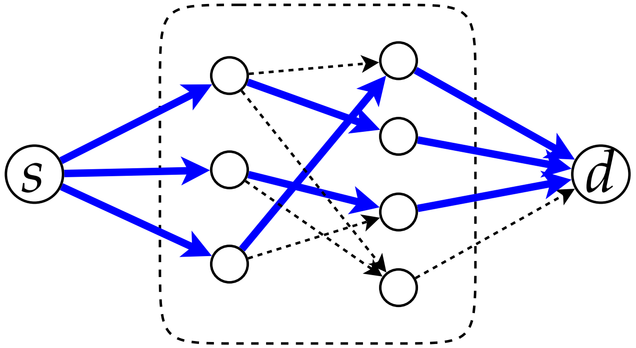
Why study the max flow problem?

- CS: a classic optimization problem:

  ex: finding maximum matching,

Bandwidth-efficiency

(s,d)-Flow, max (s,d)-flow, and the max-flow value (MFV).

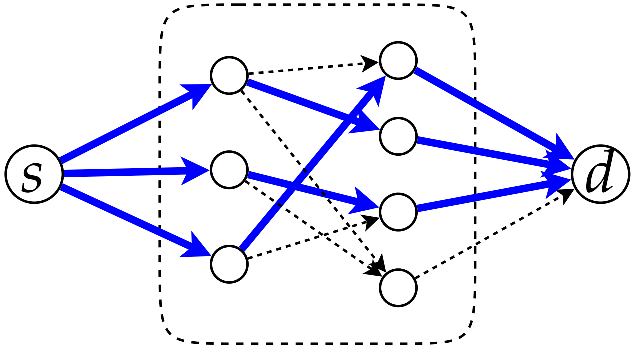


Why study the max flow problem?

- CS: a classic optimization problem:
  ex: finding maximum matching,

Bandwidth-efficiency

(s,d)-Flow, max (s,d)-flow, and the max-flow value (MFV).

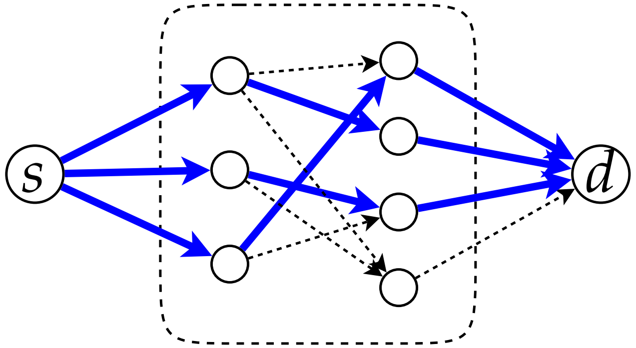


Why study the max flow problem?

- CS: a classic optimization problem:
  ex: finding maximum matching,
  finding the minimum separation (min-cut).

Bandwidth-efficiency

(s, d)-Flow, max (s, d)-flow, and the max-flow value (MFV).

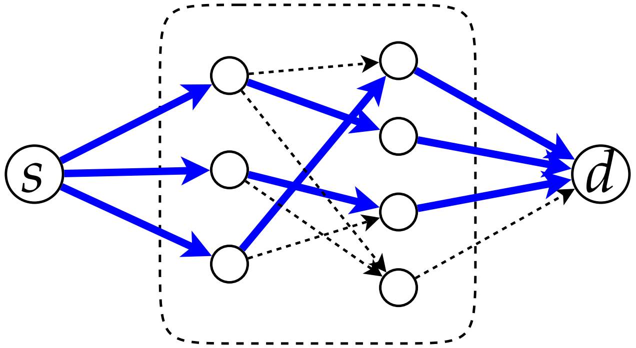


Why study the max flow problem?

- CS: a classic optimization problem:
  ex: finding maximum matching,
  finding the minimum separation (min-cut).
- EE: Bandwidth-efficient network coding solutions.
  - A multicast rate r is supportable iff $r \leq \text{MFV}_i$ for all source-destination pairs (s, d_i) [Ahlswede *et al.* 00], [Li *et al.* 03].

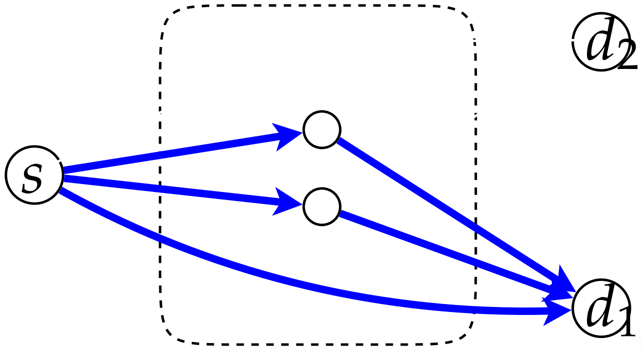
Bandwidth-efficiency

(s,d)-Flow, max (s,d)-flow, and the max-flow value (MFV).



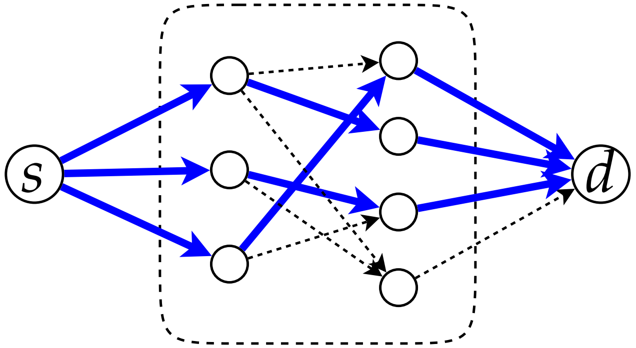
Why study the max flow problem?

- CS: a classic optimization problem:
  ex: finding maximum matching,
  finding the minimum separation (min-cut).
- EE: Bandwidth-efficient network coding solutions.
  - A multicast rate r is supportable iff $r \leq \mathrm{MFV}_i$ for all source-destination pairs (s, d_i) [Ahlswede *et al.* 00], [Li *et al.* 03].



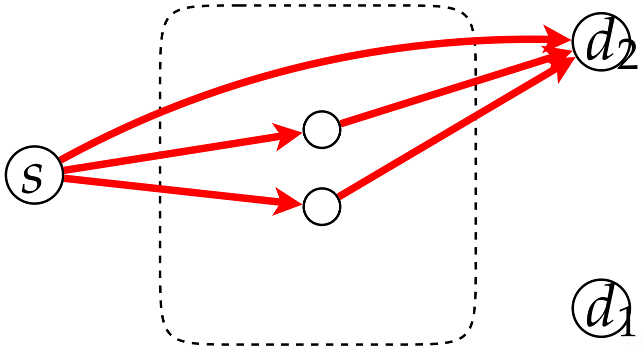
Bandwidth-efficiency

(s, d)-Flow, max (s, d)-flow, and the max-flow value (MFV).



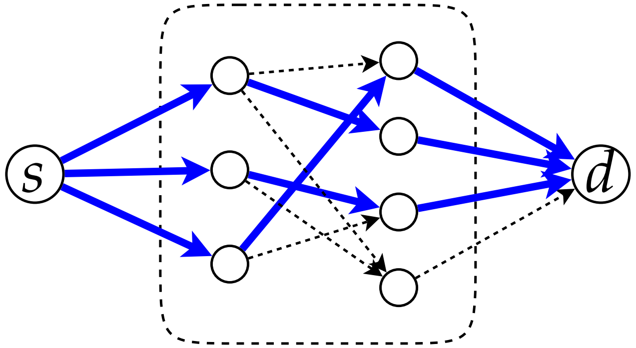
Why study the max flow problem?

- CS: a classic optimization problem:
  ex: finding maximum matching,
  finding the minimum separation (min-cut).
- EE: Bandwidth-efficient network coding solutions.
  - A multicast rate r is supportable iff $r \leq \text{MFV}_i$ for all source-destination pairs (s, d_i) [Ahlswede *et al.* 00], [Li *et al.* 03].



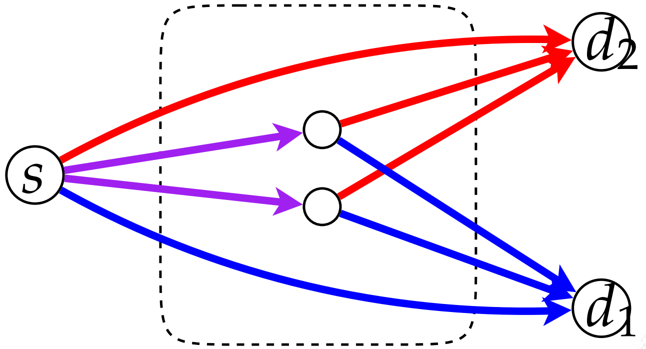
Bandwidth-efficiency

(s,d)-Flow, max (s,d)-flow, and the max-flow value (MFV).



Why study the max flow problem?

- CS: a classic optimization problem:
  ex: finding maximum matching,
  finding the minimum separation (min-cut).
- EE: Bandwidth-efficient network coding solutions.
  - A multicast rate r is supportable iff $r \leq \mathrm{MFV}_i$ for all source-destination pairs (s, d_i) [Ahlswede *et al.* 00], [Li *et al.* 03].



Existing Max-Flow Algorithms

- Linear-programming (LP) based max-flow algorithms
  - $$\max_{f_e \geq 0} \sum_{e \in \mathrm{Out}(s)} f_e$$
    $$\text{subject to} \quad \forall v, \sum_{e \in \mathrm{In}(v)} f_e = \sum_{e' \in \mathrm{Out}(v)} f_{e'}$$

Existing Max-Flow Algorithms

- Linear-programming (LP) based max-flow algorithms
  - $$\max_{f_e \geq 0} \quad \sum_{e \in \mathrm{Out}(s)} f_e$$
    $$\text{subject to} \quad \forall v, \sum_{e \in \mathrm{In}(v)} f_e = \sum_{e' \in \mathrm{Out}(v)} f_{e'}$$
  - Suitable for different objective functions, ex: $\min \sum_e c_e$.

Existing Max-Flow Algorithms

- Linear-programming (LP) based max-flow algorithms
  - $$\max_{f_e \geq 0} \quad \sum_{e \in \text{Out}(s)} f_e$$
    $$\text{subject to} \quad \forall v, \sum_{e \in \text{In}(v)} f_e = \sum_{e' \in \text{Out}(v)} f_{e'}$$
  - Suitable for different objective functions, ex: $\min \sum_e c_e$.
  - Complexity: queue-length exchange,

Existing Max-Flow Algorithms

- Linear-programming (LP) based max-flow algorithms
  - $$\max_{f_e \geq 0} \sum_{e \in \mathrm{Out}(s)} f_e$$
    $$\text{subject to} \quad \forall v, \sum_{e \in \mathrm{In}(v)} f_e = \sum_{e' \in \mathrm{Out}(v)} f_{e'}$$
  - Suitable for different objective functions, ex: $\min \sum_e c_e$.
  - Complexity: queue-length exchange,
  - Convergence speed: small step sizes of the gradient methods,

Existing Max-Flow Algorithms

- Linear-programming (LP) based max-flow algorithms
  - $$\max_{f_e \geq 0} \sum_{e \in \mathrm{Out}(s)} f_e$$
    $$\text{subject to} \quad \forall v, \sum_{e \in \mathrm{In}(v)} f_e = \sum_{e' \in \mathrm{Out}(v)} f_{e'}$$
  - Suitable for different objective functions, ex: $\min \sum_e c_e$.
  - Complexity: queue-length exchange,
  - Convergence speed: small step sizes of the gradient methods,
  - Fractional rate vs. packet-by-packet coding operations.
    - Time-averaging? Practical generation size (# of to-be-mixed packets) is 30–100.

Existing Max-Flow Algorithms

- Graph-theoretic max-flow algorithms
  - Ford-Fulkerson 1956: Residue graph vs. augmenting path
  - Edmonds-Karp 1972: Breadth-first search + FF
  - Dinitz blocking flow algorithm 1970.

Existing Max-Flow Algorithms

- Graph-theoretic max-flow algorithms
  - Ford-Fulkerson 1956: Residue graph vs. augmenting path
  - Edmonds-Karp 1972: Breadth-first search + FF
  - Dinitz blocking flow algorithm 1970.
  - Push & relabel algorithm [Goldberg, Tarjan 1988]:

Existing Max-Flow Algorithms

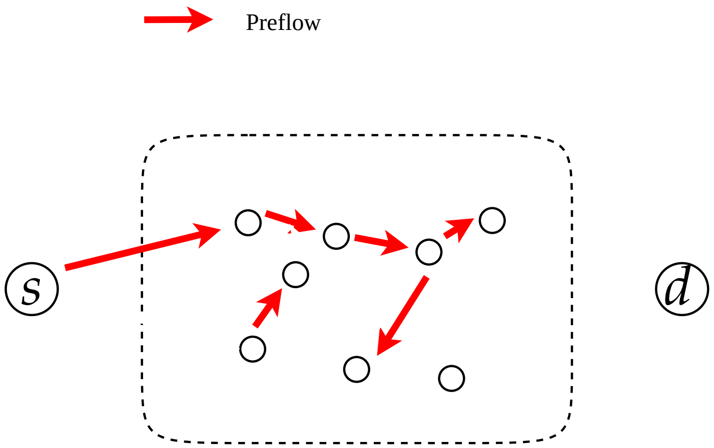
- Graph-theoretic max-flow algorithms
  - Ford-Fulkerson 1956: Residue graph vs. augmenting path
  - Edmonds-Karp 1972: Breadth-first search + FF
  - Dinitz blocking flow algorithm 1970.
  - Push & relabel algorithm [Goldberg, Tarjan 1988]:
    - Fully distributed implementation.

Existing Max-Flow Algorithms

- Graph-theoretic max-flow algorithms
  - Ford-Fulkerson 1956: Residue graph vs. augmenting path
  - Edmonds-Karp 1972: Breadth-first search + FF
  - Dinitz blocking flow algorithm 1970.
  - Push & relabel algorithm [Goldberg, Tarjan 1988]:
    - Fully distributed implementation.
    - Based on the non-coded paradigm.
    - "Preflows" are not allowed to be mixed with each other.

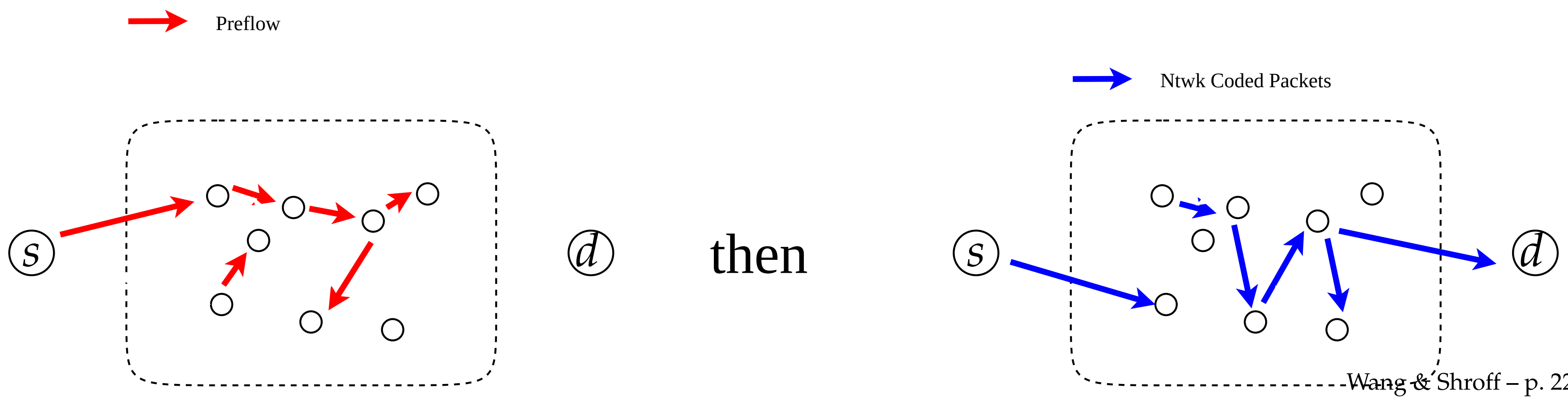
Existing Max-Flow Algorithms

- Graph-theoretic max-flow algorithms
  - Ford-Fulkerson 1956: Residue graph vs. augmenting path
  - Edmonds-Karp 1972: Breadth-first search + FF
  - Dinitz blocking flow algorithm 1970.
  - Push & relabel algorithm [Goldberg, Tarjan 1988]:
    - Fully distributed implementation.
    - Based on the non-coded paradigm.
    - "Preflows" are not allowed to be mixed with each other.



Existing Max-Flow Algorithms

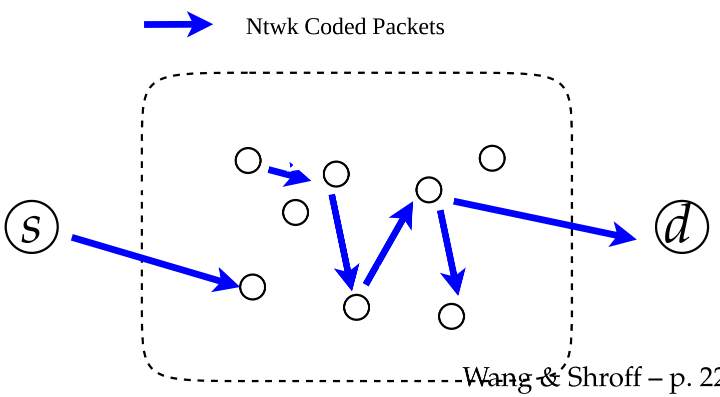
- Graph-theoretic max-flow algorithms
  - Ford-Fulkerson 1956: Residue graph vs. augmenting path
  - Edmonds-Karp 1972: Breadth-first search + FF
  - Dinitz blocking flow algorithm 1970.
  - Push & relabel algorithm [Goldberg, Tarjan 1988]:
    - Fully distributed implementation.
    - Based on the non-coded paradigm.
    - “Preflows" are not allowed to be mixed with each other.



Existing Max-Flow Algorithms

- Graph-theoretic max-flow algorithms
  - Ford-Fulkerson 1956: Residue graph vs. augmenting path
  - Edmonds-Karp 1972: Breadth-first search + FF
  - Dinitz blocking flow algorithm 1970.
  - Push & relabel algorithm [Goldberg, Tarjan 1988]:
    - Fully distributed implementation.
    - Based on the non-coded paradigm.
    - “Preflows" are not allowed to be mixed with each other.

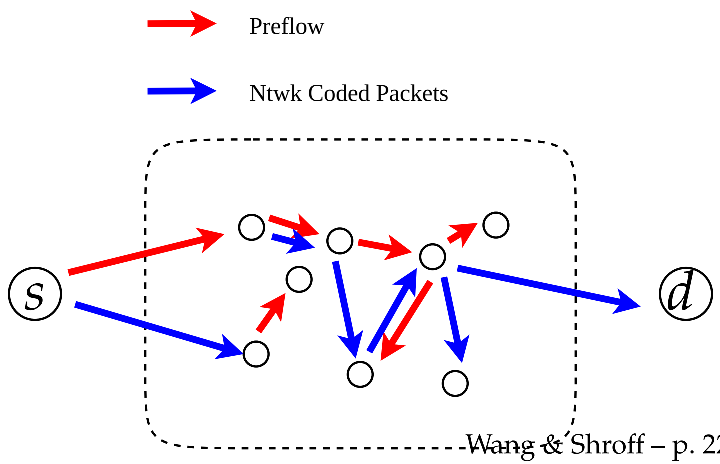
or



Existing Max-Flow Algorithms

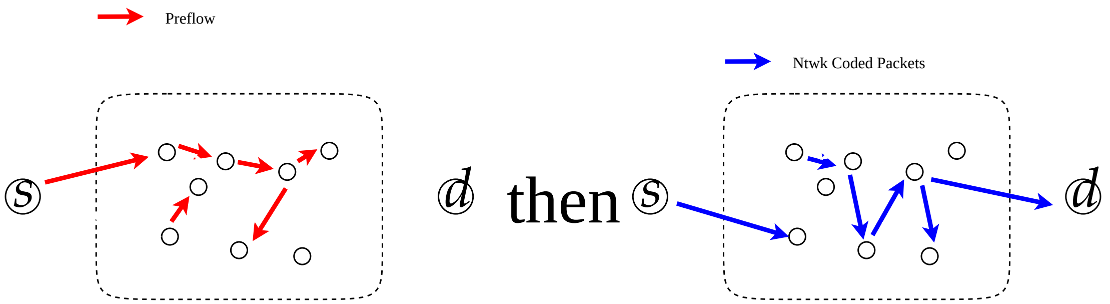
- Graph-theoretic max-flow algorithms
  - Ford-Fulkerson 1956: Residue graph vs. augmenting path
  - Edmonds-Karp 1972: Breadth-first search + FF
  - Dinitz blocking flow algorithm 1970.
  - Push & relabel algorithm [Goldberg, Tarjan 1988]:
    - Fully distributed implementation.
    - Based on the non-coded paradigm.
    - “Preflows" are not allowed to be mixed with each other.

or



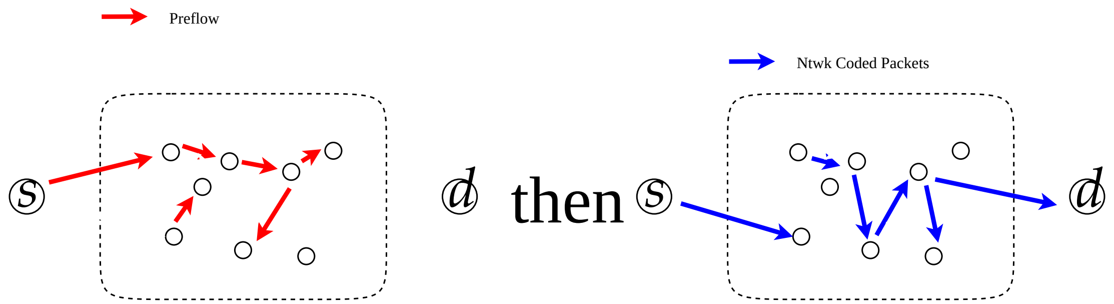
Delay Minimality of NC

- The sequential approach:



Delay Minimality of NC

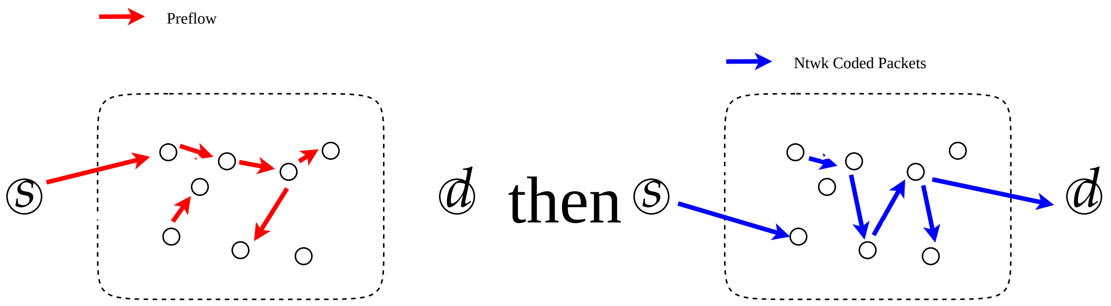
- The sequential approach:



  - Induces delay

Delay Minimality of NC

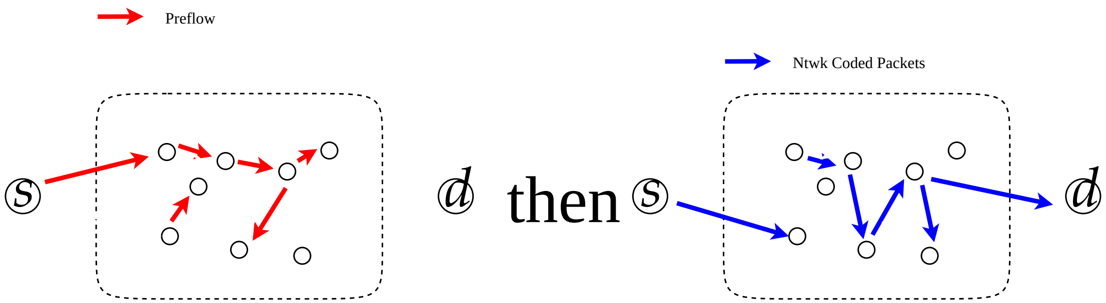
- The sequential approach:



  - Induces delay
  - Network coding offers no throughput advantage than multipath routing for unicast.

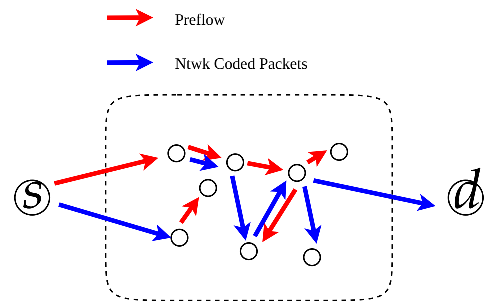
Delay Minimality of NC

- The sequential approach:



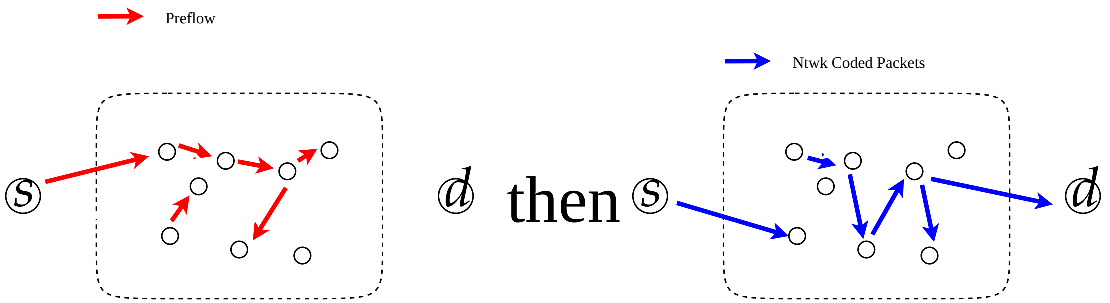
  - Induces delay
  - Network coding offers no throughput advantage than multipath routing for unicast.

- The parallel approach reduces the delay:

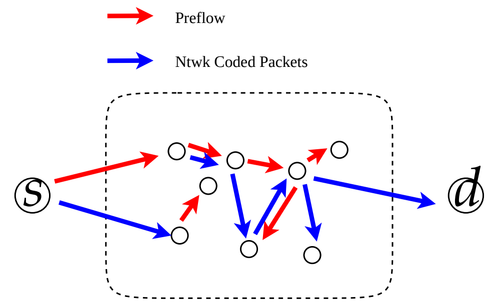


Delay Minimality of NC

- The sequential approach:



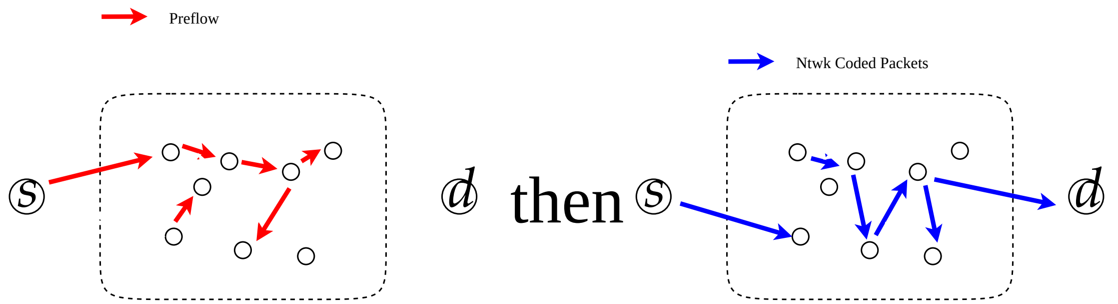
  - Induces delay
  - Network coding offers no throughput advantage than multipath routing for unicast.
- The parallel approach reduces the delay:



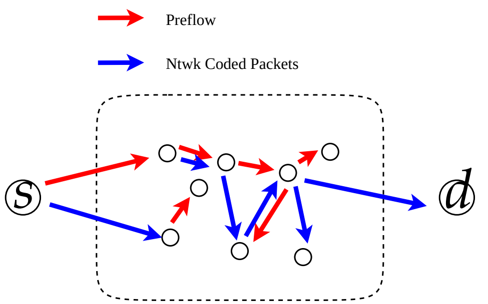
  - NC achieves the min-cut max-flow rate without knowing the max flow.
  - One simply performs random mixing + broadcasting.
  - Network coding is delay-optimal.

Delay Minimality of NC

- The sequential approach:



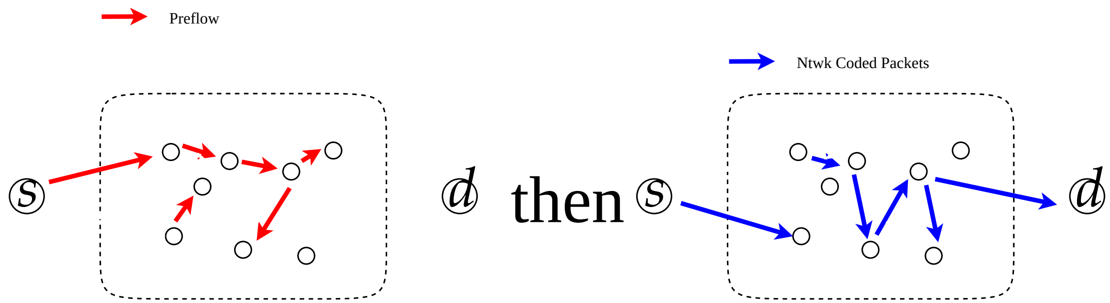
  - Induces delay
  - Network coding offers no throughput advantage than multipath routing for unicast.
- The parallel approach reduces the delay:



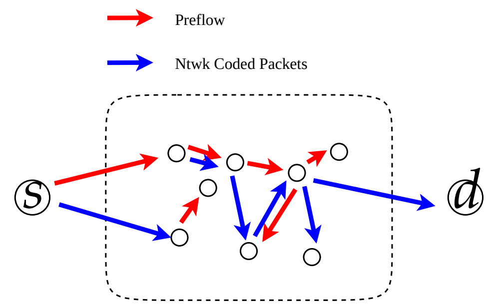
  - NC achieves the min-cut max-flow rate without knowing the max flow.
  - One simply performs random mixing + broadcasting.
  - Network coding is delay-optimal.
  - Coding eliminates the need to decide which edge to send.

Delay Minimality of NC

- The sequential approach:



  - Induces delay
  - Network coding offers no throughput advantage than multipath routing for unicast.
- The parallel approach reduces the delay:



  - NC achieves the min-cut max-flow rate without knowing the max flow.
  - One simply performs random mixing + broadcasting.
  - Network coding is delay-optimal.
  - Coding eliminates the need to decide which edge to send.
  - Significant control and communication overhead.

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence $\longrightarrow$ Run network coding

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence $\longrightarrow$ Run network coding $\longrightarrow$ Bandwidth optimality

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence $\longrightarrow$ Run network coding $\longrightarrow$ Bandwidth optimality
- A new coding-theoretic approach:

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence $\longrightarrow$ Run network coding $\longrightarrow$ Bandwidth optimality
- A new coding-theoretic approach:
  Run network coding

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence $\longrightarrow$ Run network coding $\longrightarrow$ Bandwidth optimality

- A new coding-theoretic approach:
  Run network coding $\longrightarrow$ Repeatedly stop the traffic on redundant edges

  Redundant edges are the edges such that the removal of which will not interrupt the network coded traffic.

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence $\longrightarrow$ Run network coding $\longrightarrow$ Bandwidth optimality

- A new coding-theoretic approach:
  Run network coding $\longrightarrow$ Repeatedly stop the traffic on redundant edges $\longrightarrow$ Bandwidth optimality

  Redundant edges are the edges such that the removal of which will not interrupt the network coded traffic.

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence $\longrightarrow$ Run network coding $\longrightarrow$ Bandwidth optimality
- A new coding-theoretic approach: Delay optimal.
  Run network coding $\longrightarrow$ Repeatedly stop the traffic on redundant edges $\longrightarrow$ Bandwidth optimality

  Redundant edges are the edges such that the removal of which will not interrupt the network coded traffic.

A Coding-Theoretic Approach

- Classic sequential graph-theoretic approach:
  Run the max-flow algorithm until convergence ⟶ Run network coding ⟶ Bandwidth optimality
- A new coding-theoretic approach: Delay optimal.
  Run network coding ⟶ Repeatedly stop the traffic on redundant edges ⟶ Bandwidth optimality

  Redundant edges are the edges such that the removal of which will not interrupt the network coded traffic.

The key question: How to find distributedly the redundant edges?

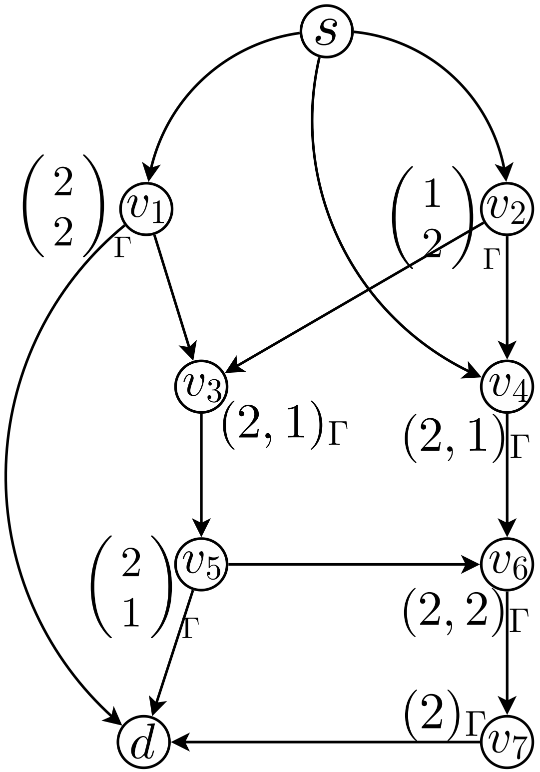
New Approach of Coded Feedback

Network coding on $\mathsf{GF}(3)$

New Approach of Coded Feed-back

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

Network coding on GF(3)

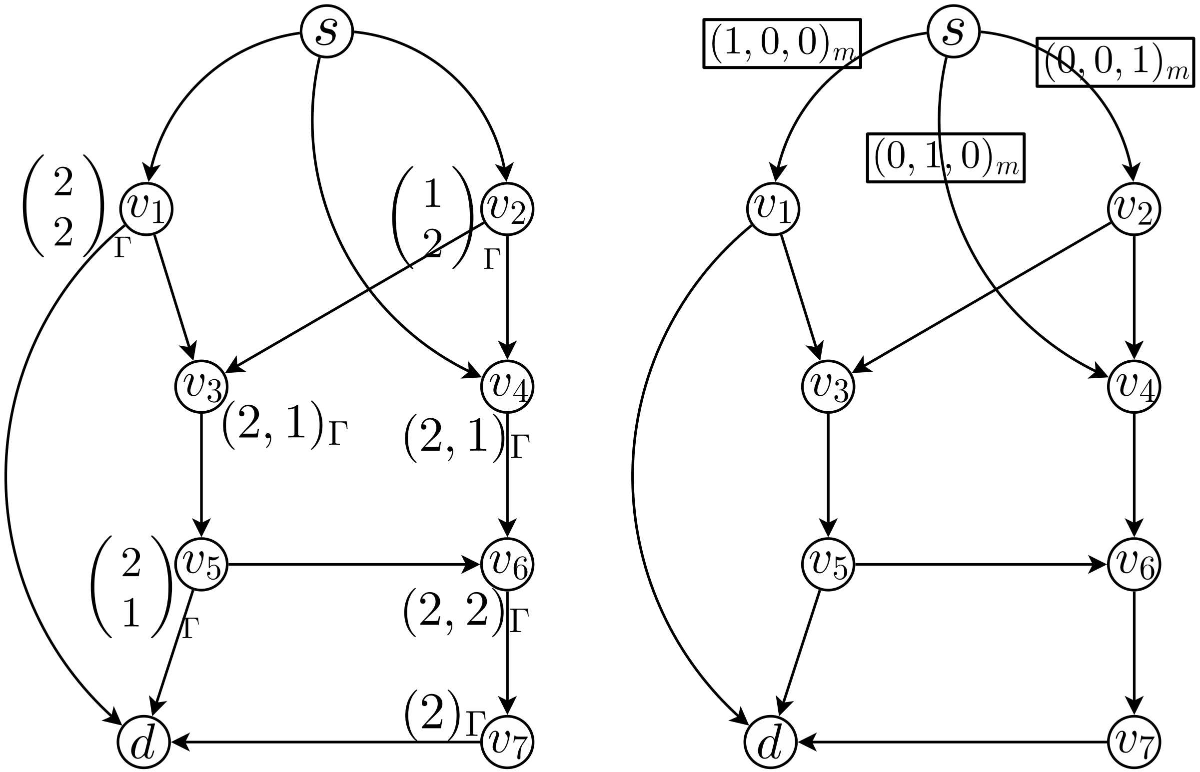


New Approach of Coded Feedback

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on GF(3)

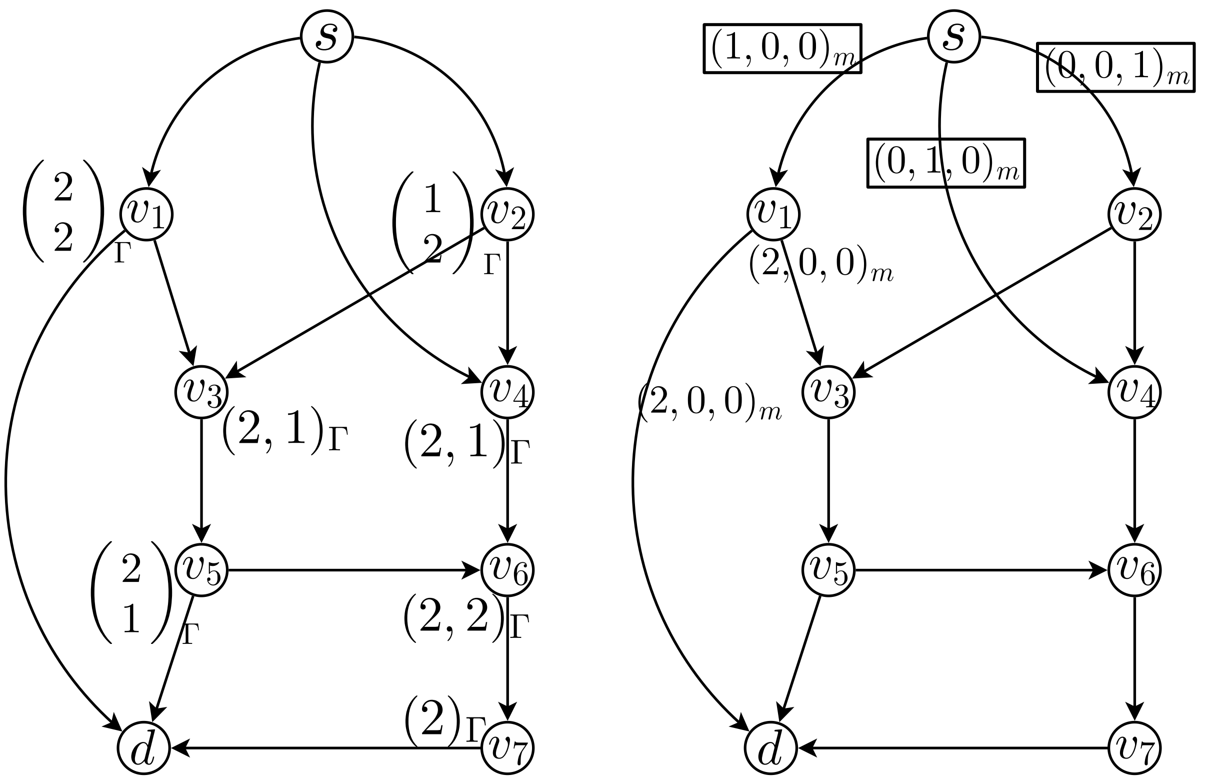


New Approach of Coded Feedback

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on GF(3)

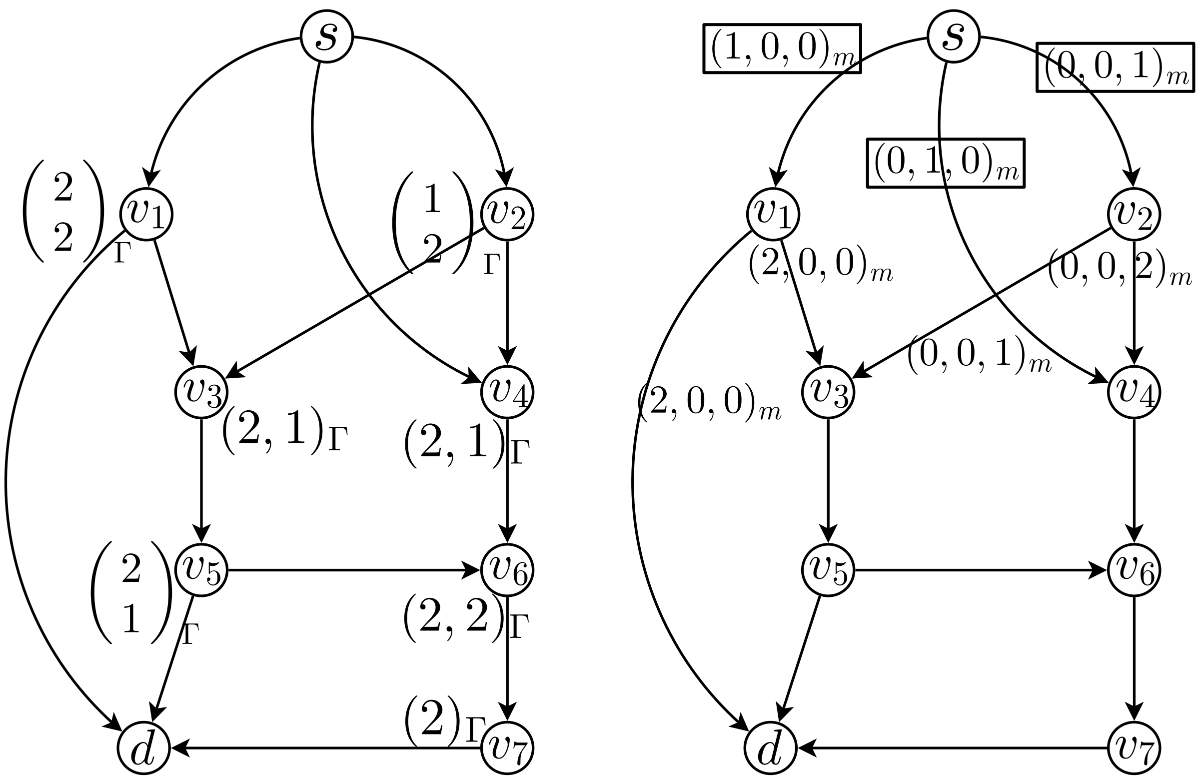


New Approach of Coded Feed-back

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on $\mathsf{GF}(3)$

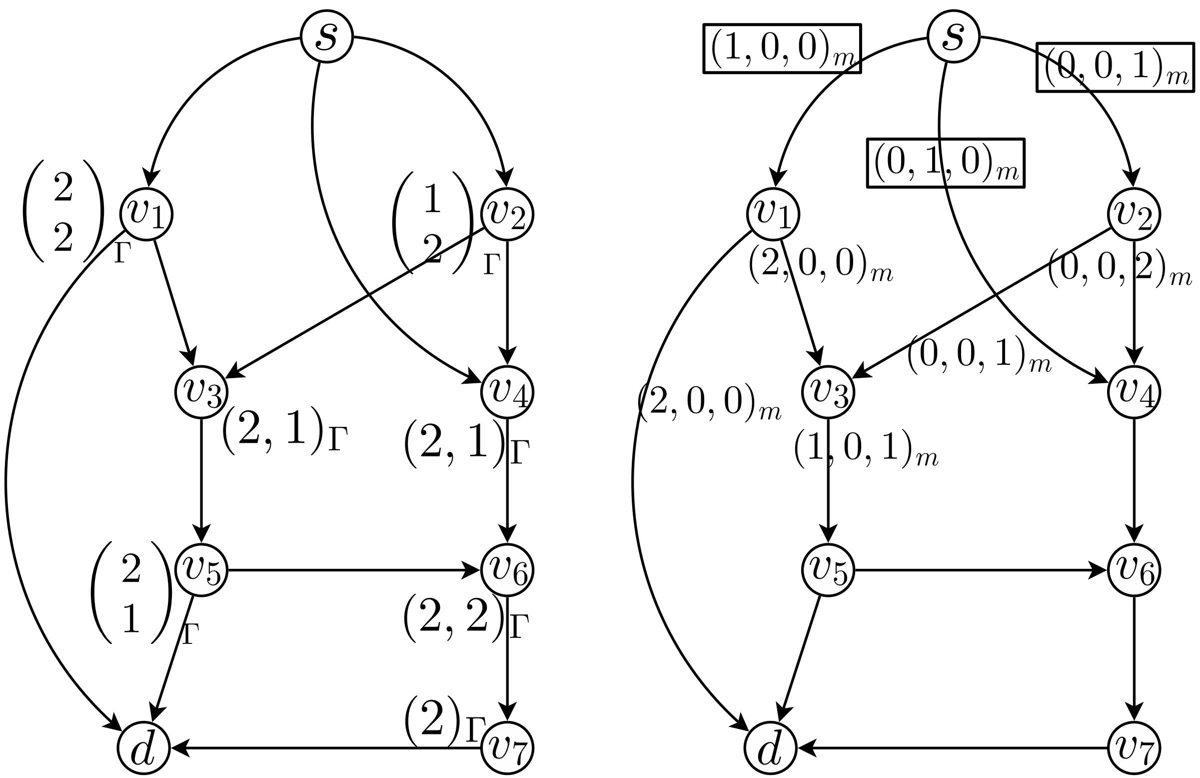


New Approach of Coded Feedback

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on GF(3)

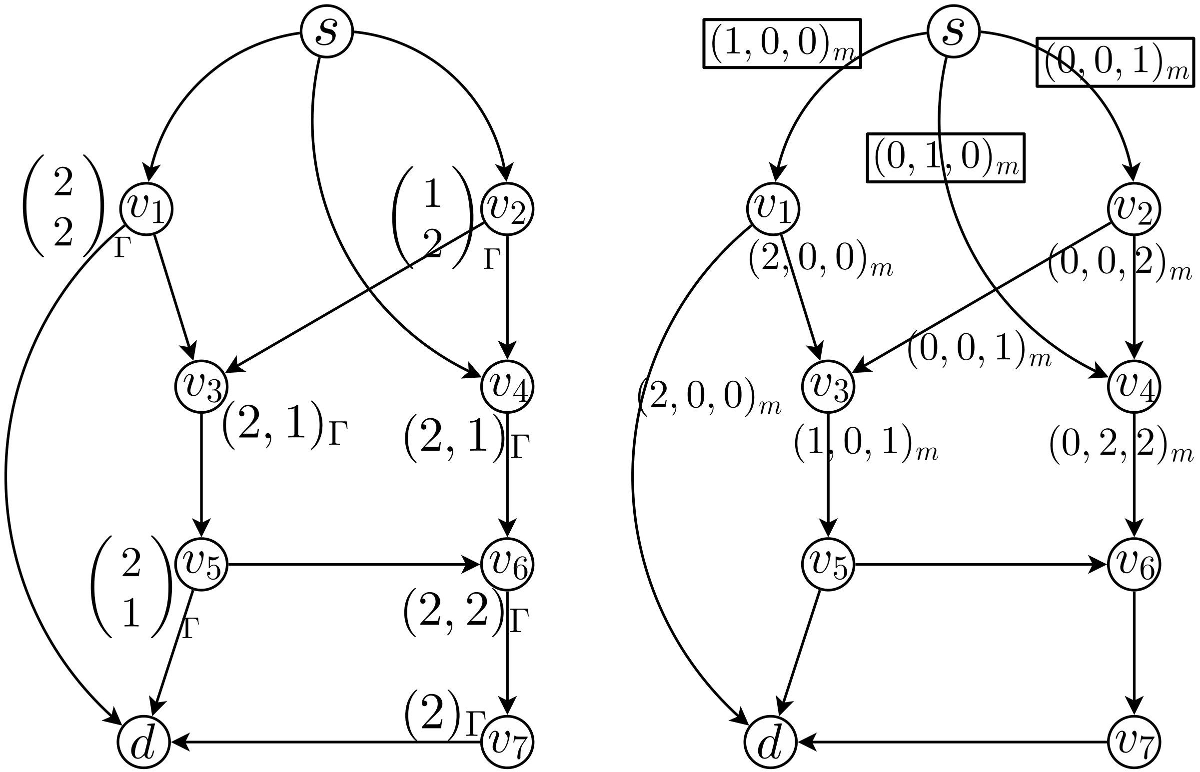


New Approach of Coded Feedback

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on GF(3)

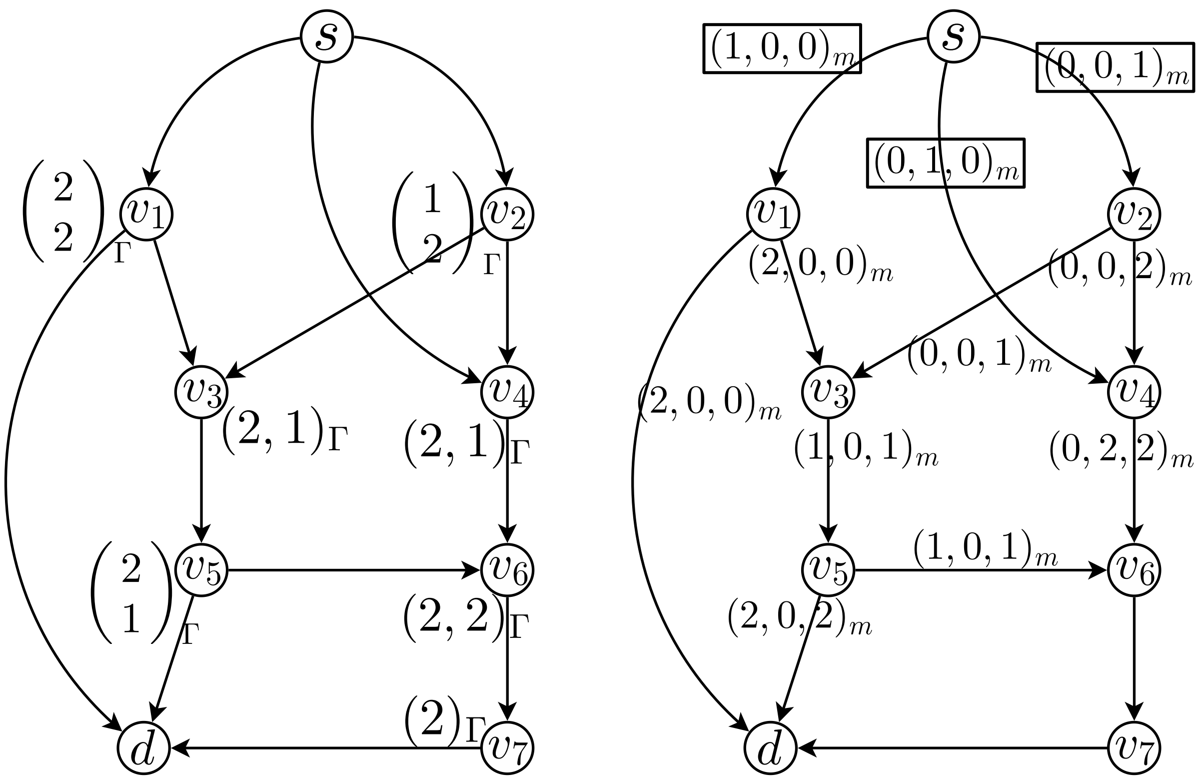


New Approach of Coded Feedback

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on GF(3)

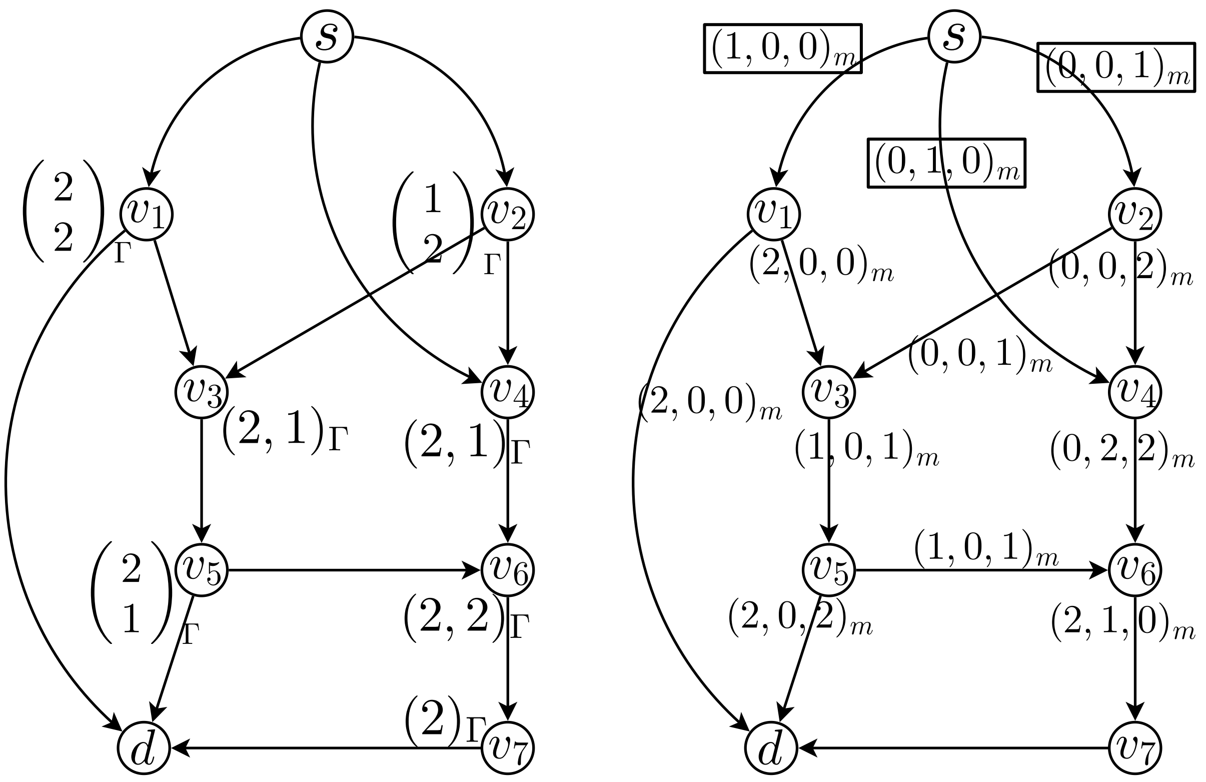


New Approach of Coded Feed-back

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on $\mathsf{GF}(3)$

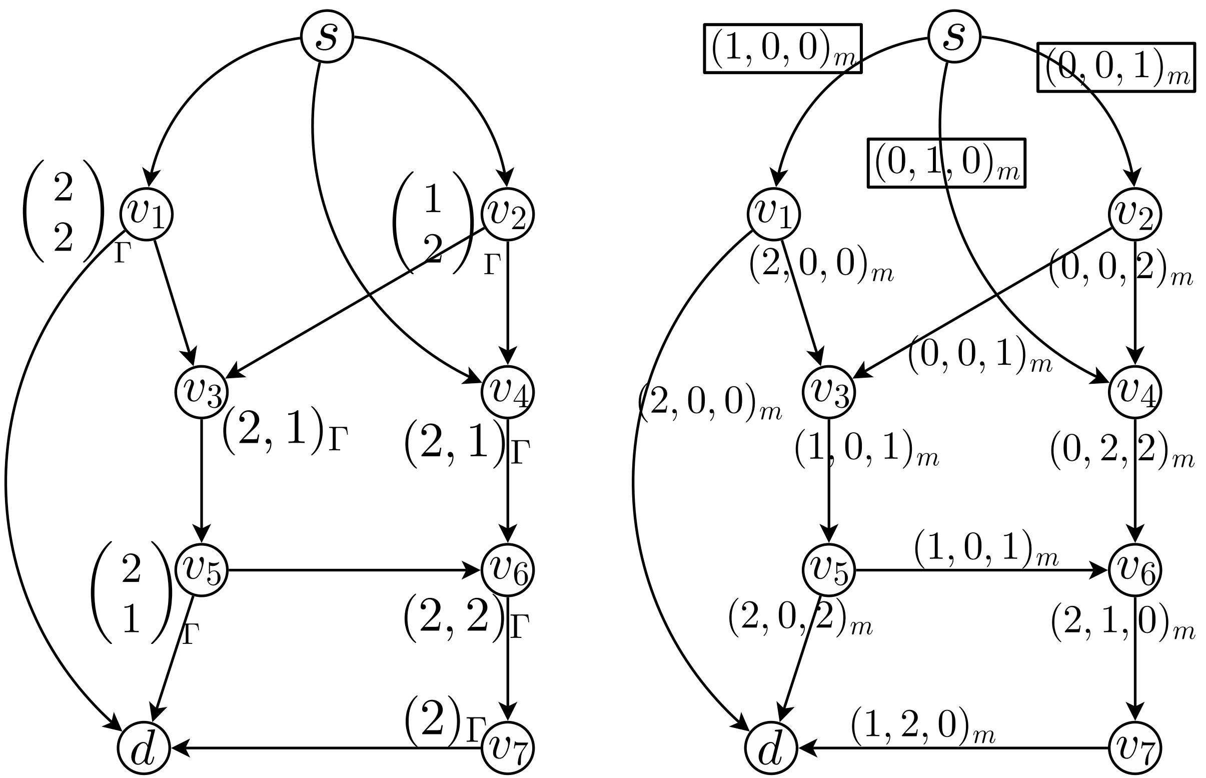


New Approach of Coded Feed-back

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on GF(3)



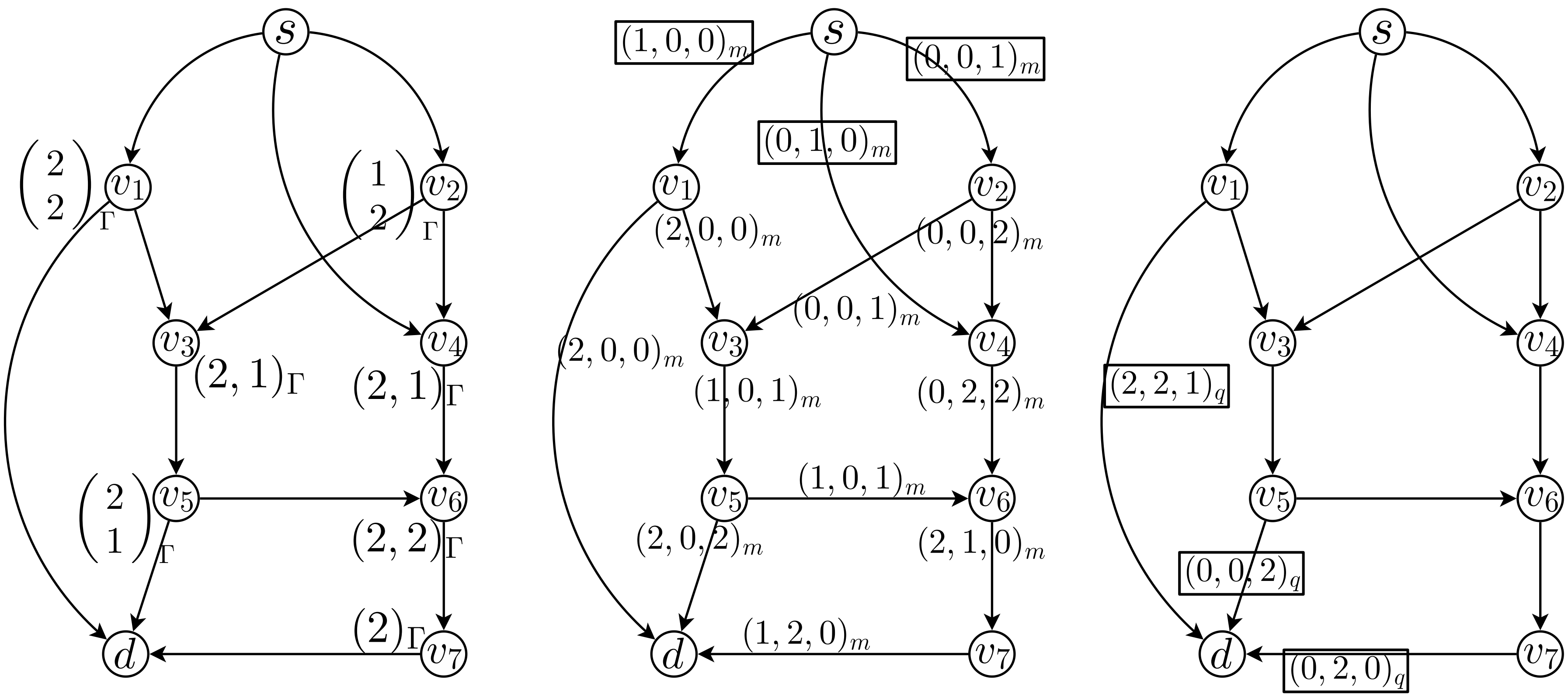
New Approach of Coded Feed-back

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Network coding on GF(3)

Step 3: Compute the coded feedback q_e



New Approach of Coded Feedback

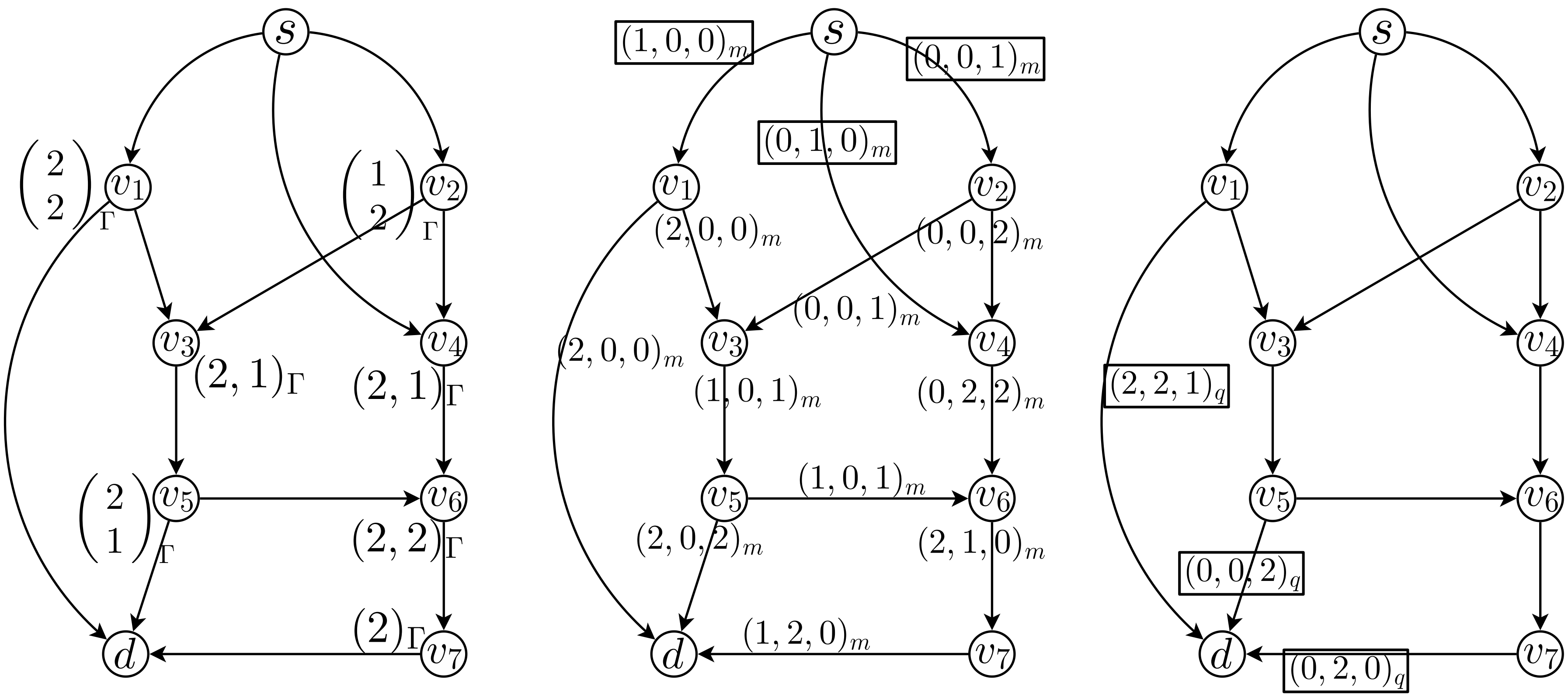
Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on GF(3)

♡ **Orthogonal Coded Feedback**



New Approach of Coded Feedback

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

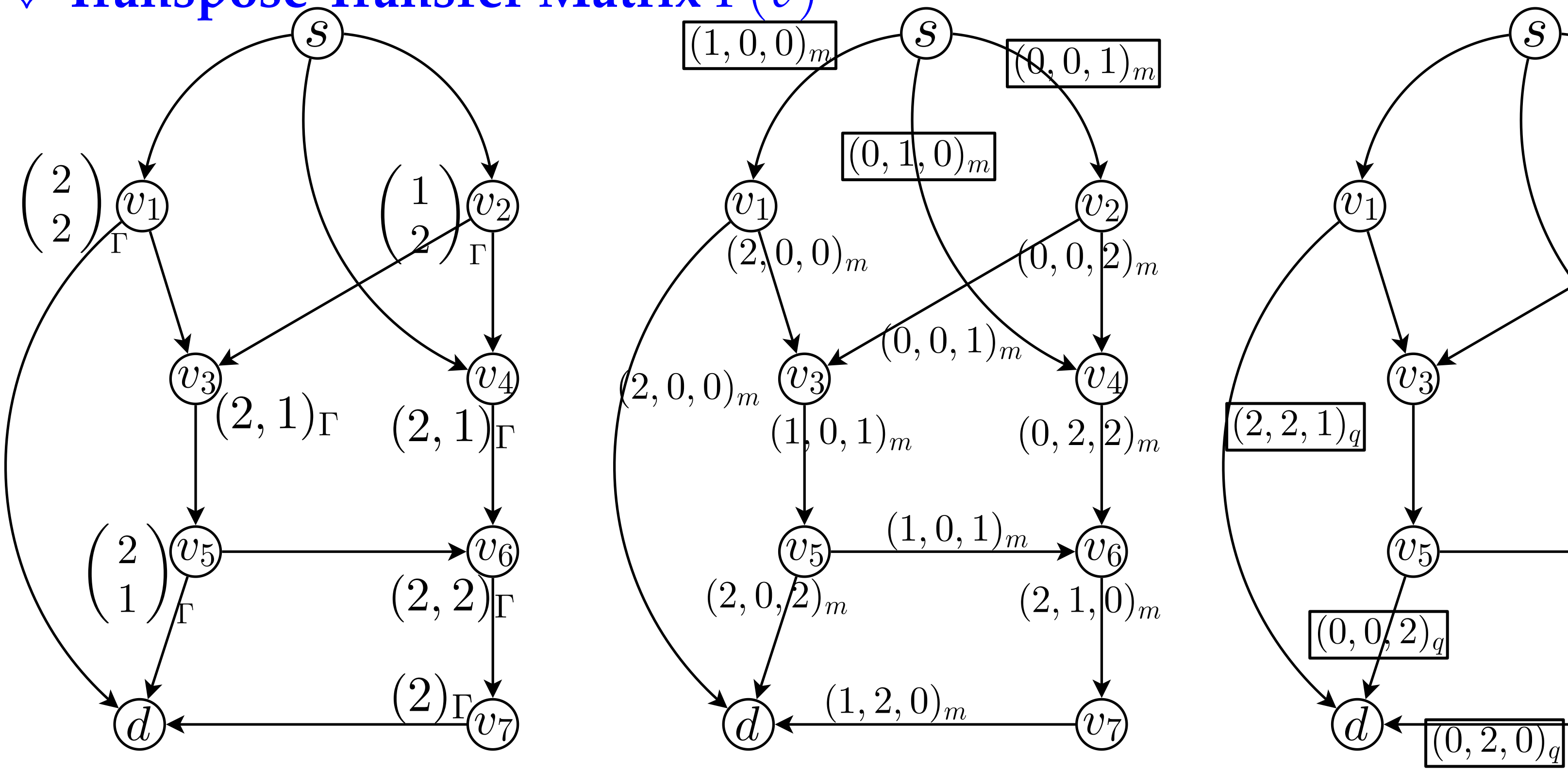
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on GF(3)

♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix $\Gamma(v)^T$**



New Approach of Coded Feedback

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

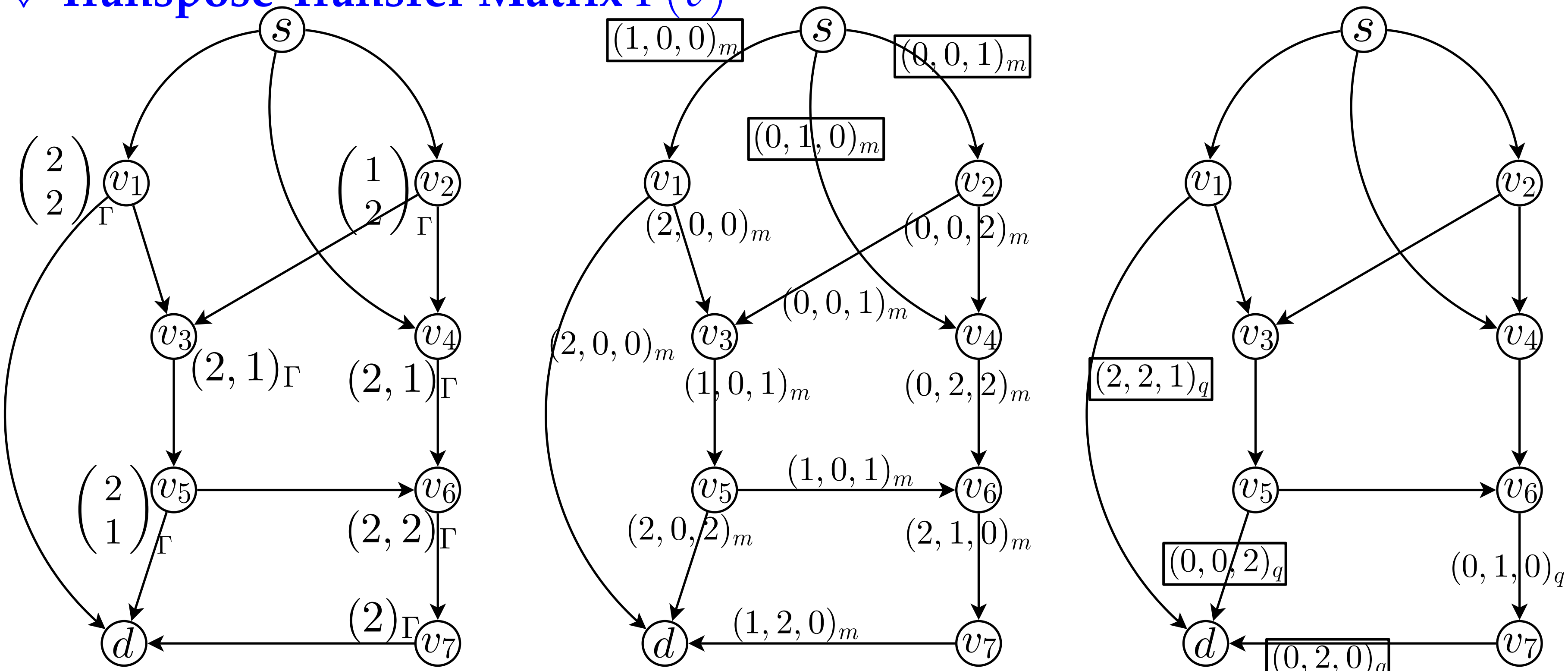
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on GF(3)

♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix $\Gamma(v)^T$**



New Approach of Coded Feedback

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

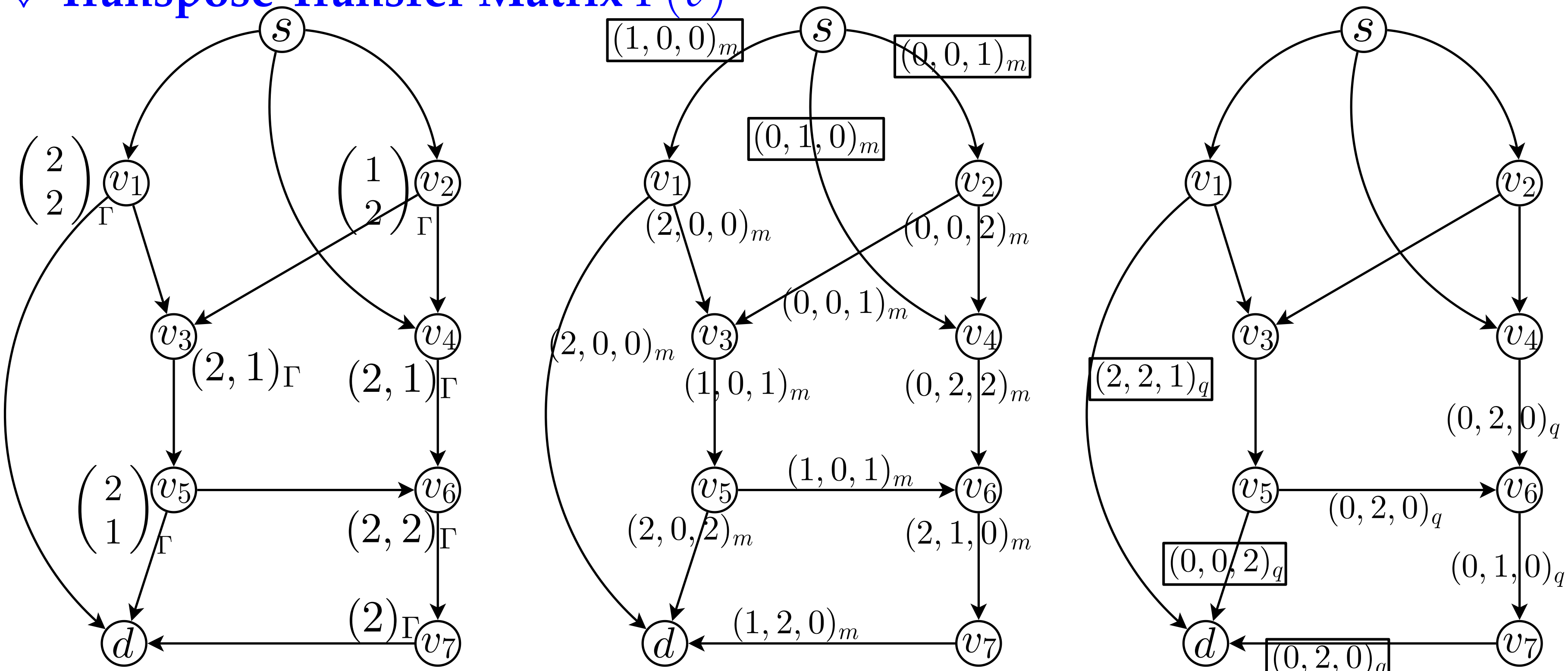
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on $\mathsf{GF}(3)$

♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix** $\Gamma(v)^{\mathrm{T}}$



New Approach of Coded Feed-back

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

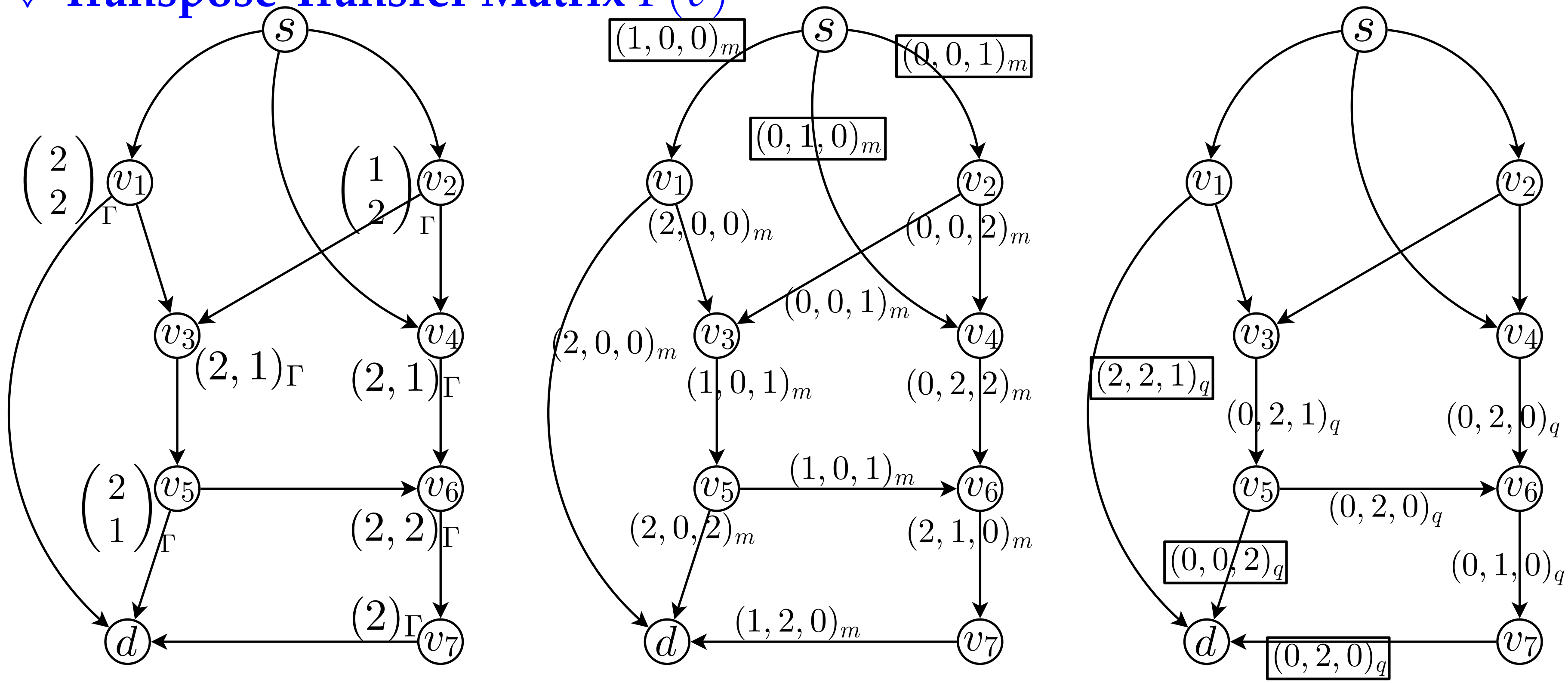
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on GF(3)

♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix $\Gamma(v)^T$**



New Approach of Coded Feedback

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

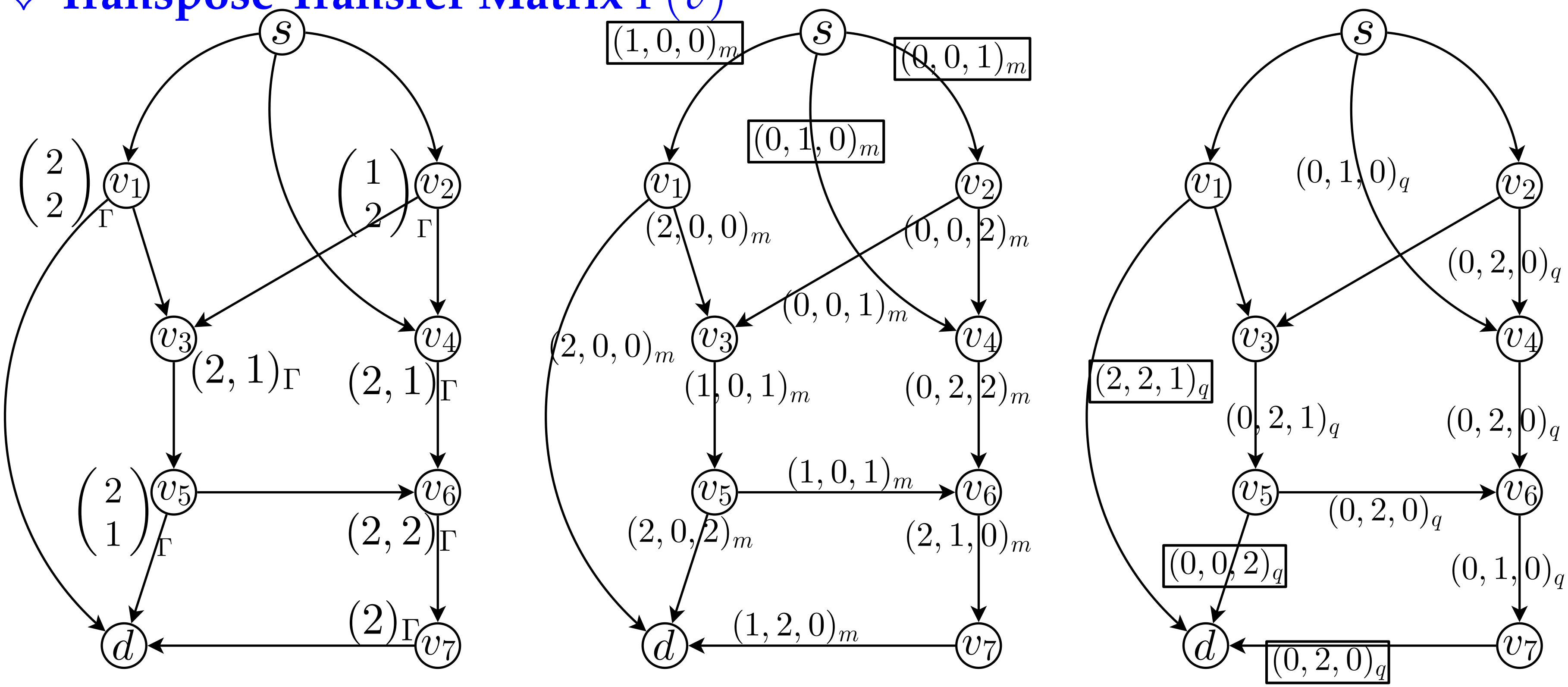
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on GF(3)

♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix** $\Gamma(v)^{\mathrm{T}}$



New Approach of Coded Feedback

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

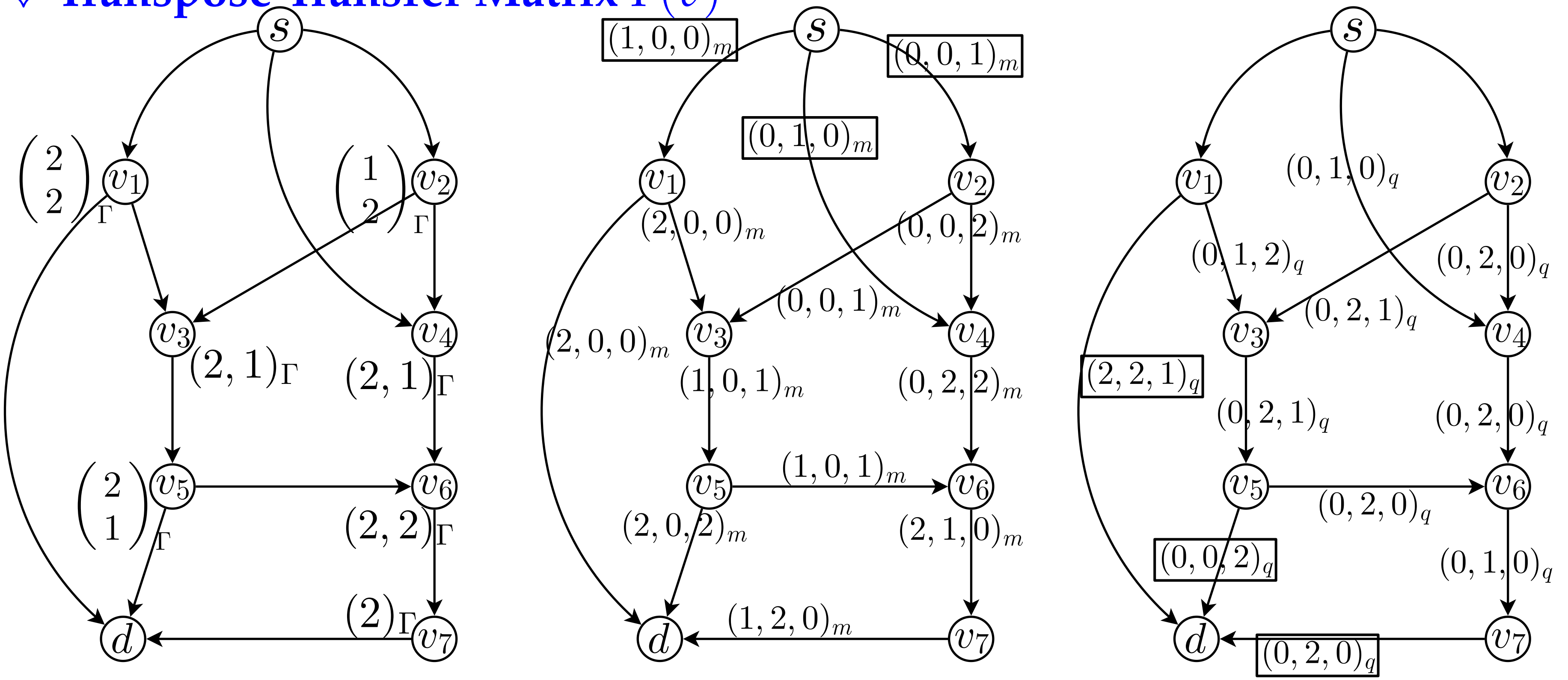
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on GF(3)

♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix $\Gamma(v)^T$**



New Approach of Coded Feed-back

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

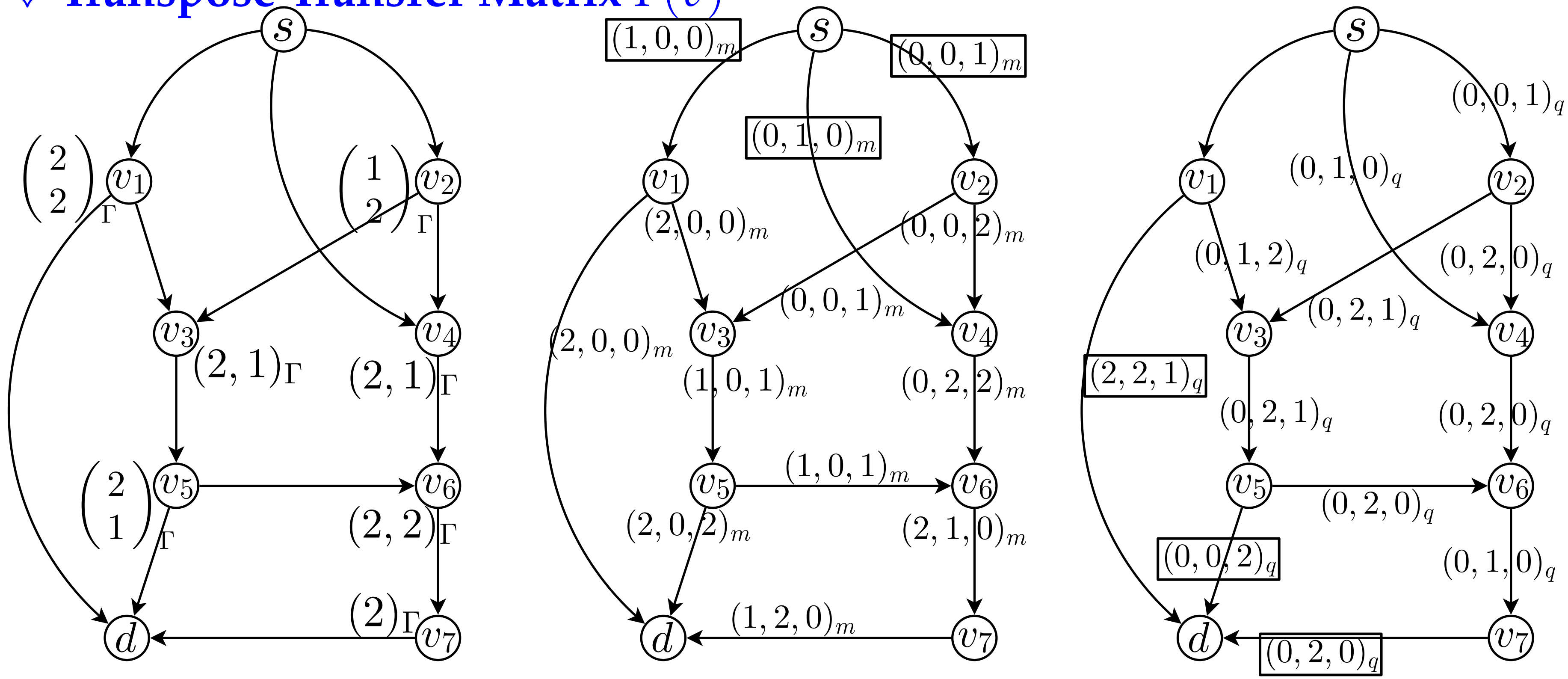
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on $\mathsf{GF}(3)$

♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix $\Gamma(v)^{\mathrm{T}}$**



New Approach of Coded Feedback

Step 1: Choose the $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ mixing matrix $\Gamma(v)$

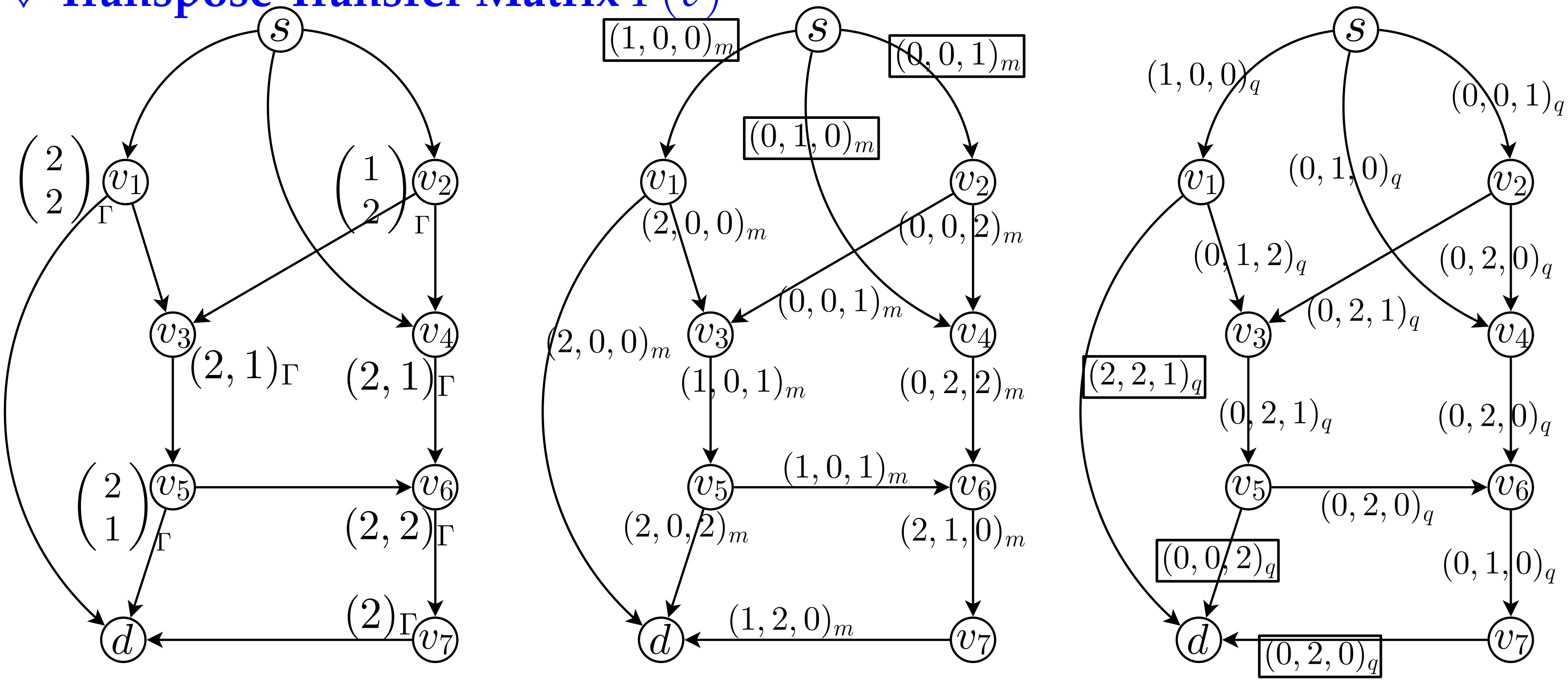
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on GF(3)

♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix $\Gamma(v)^{\mathrm{T}}$**



New Approach of Coded Feed-back

Step 1: Choose the $|\text{Out}(v)| \times |\text{In}(v)|$ mixing matrix $\Gamma(v)$

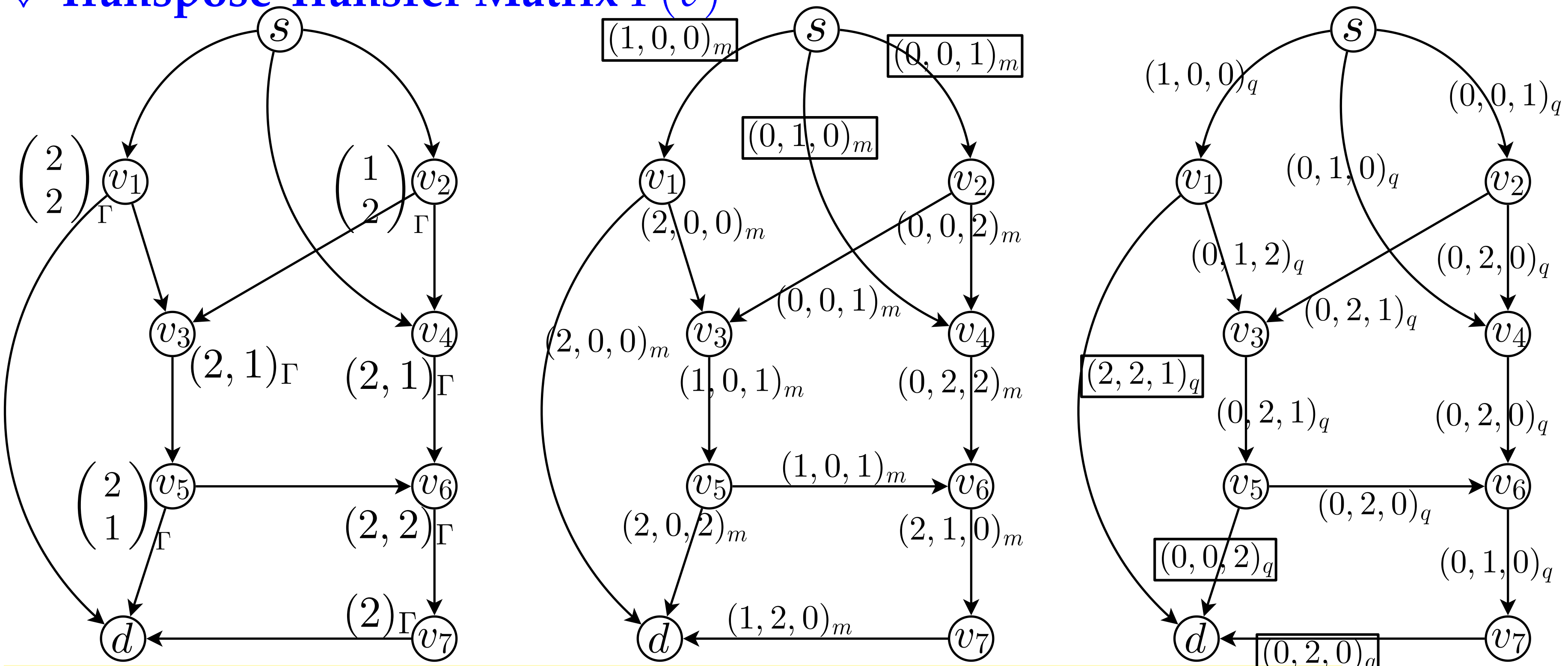
Step 2: Compute the coding vectors m_e

Step 3: Compute the coded feedback q_e

Network coding on GF(3)

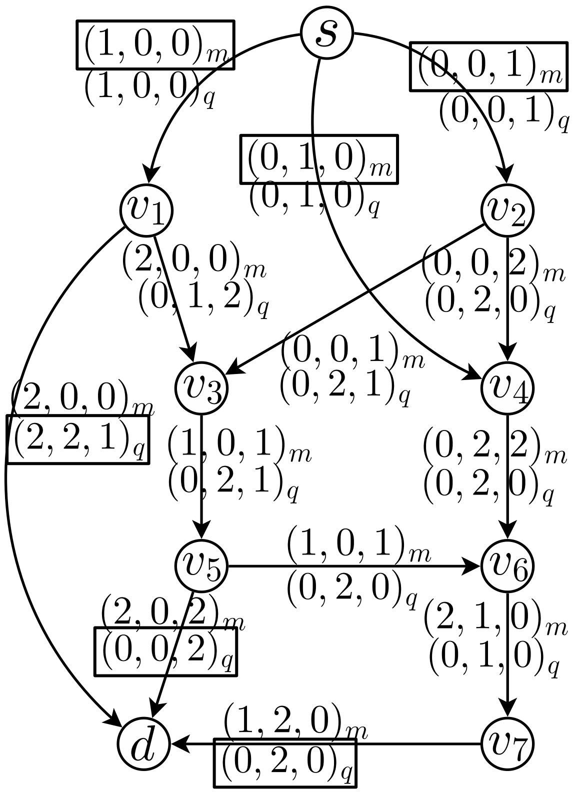
♡ **Orthogonal Coded Feedback**

♡ **Transpose Transfer Matrix** $\Gamma(v)^T$



Steps 1 and 2 are Normal Network Coding. Step 3 is new.

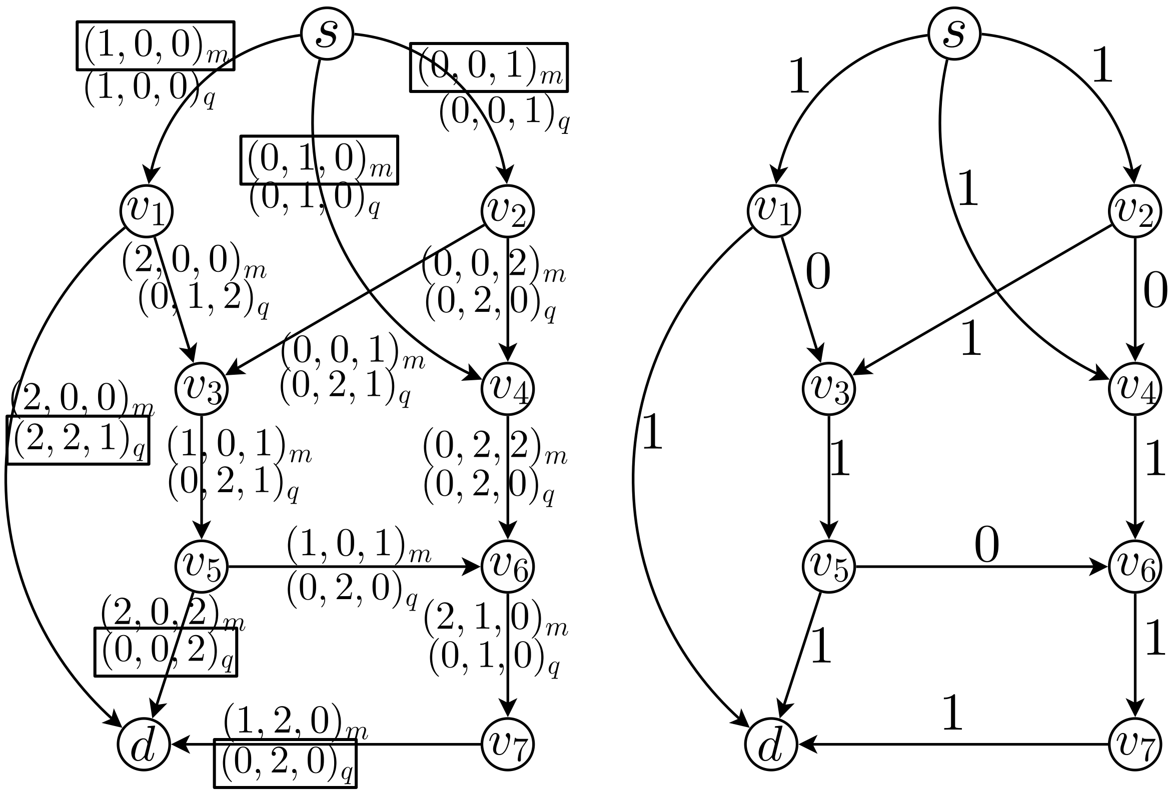
Cont'd



s
v1
v2
v3
v4
v5
v6
v7
d
(1, 0, 0)m
(1, 0, 0)q
(0, 1, 0)m
(0, 1, 0)q
(0, 0, 1)m
(0, 0, 1)q
(2, 0, 0)m
(0, 1, 2)q
(0, 0, 2)m
(0, 2, 0)q
(0, 0, 1)m
(0, 2, 1)q
(2, 0, 0)m
(2, 2, 1)q
(1, 0, 1)m
(0, 2, 1)q
(0, 2, 2)m
(0, 2, 0)q
(1, 0, 1)m
(0, 2, 0)q
(2, 0, 2)m
(0, 0, 2)q
(2, 1, 0)m
(0, 1, 0)q
(1, 2, 0)m
(0, 2, 0)q

Cont'd

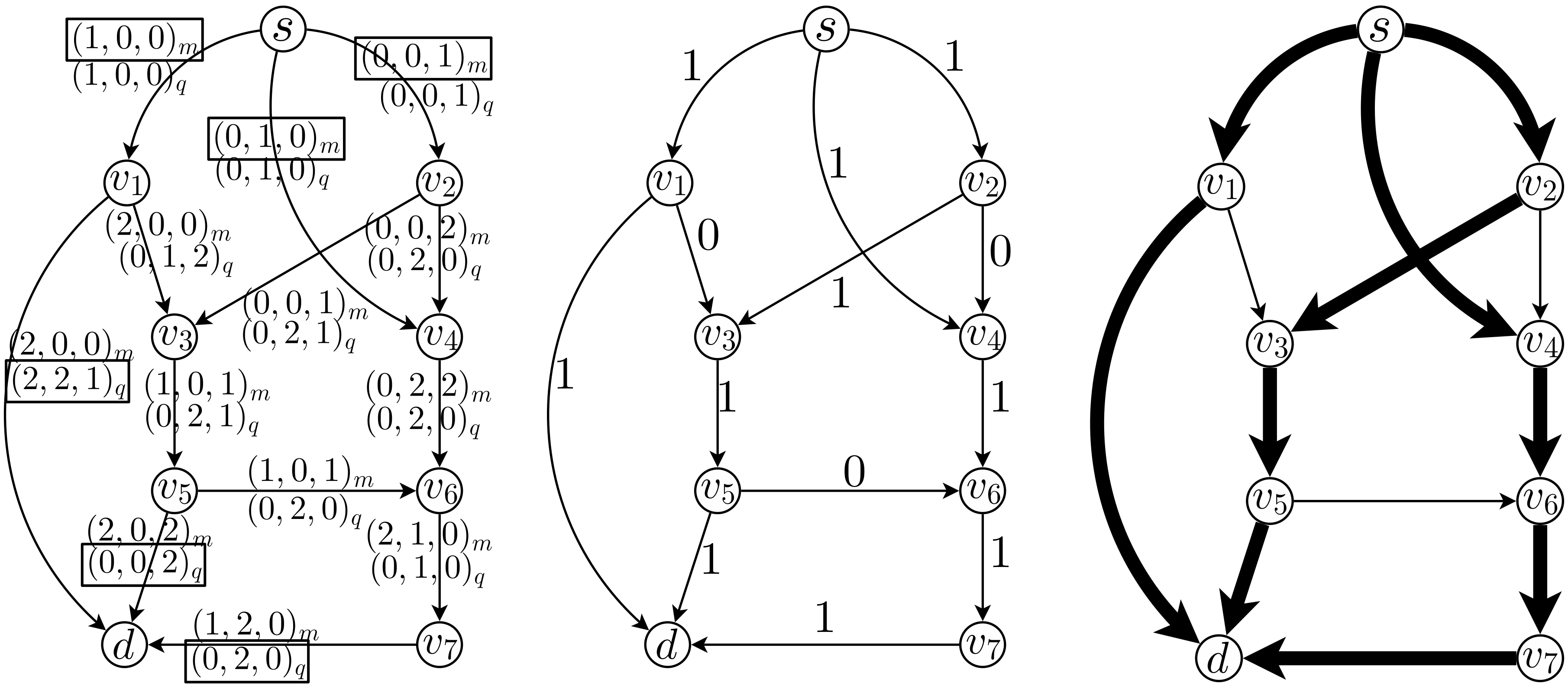
Step 4: Compute the inner products



Cont'd

Step 4: Compute the inner products

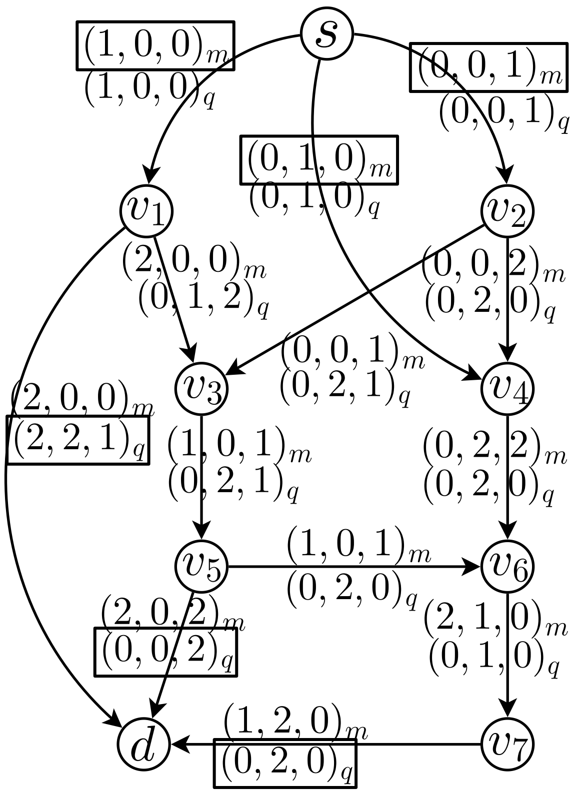
Comparison to the true max flow found offline

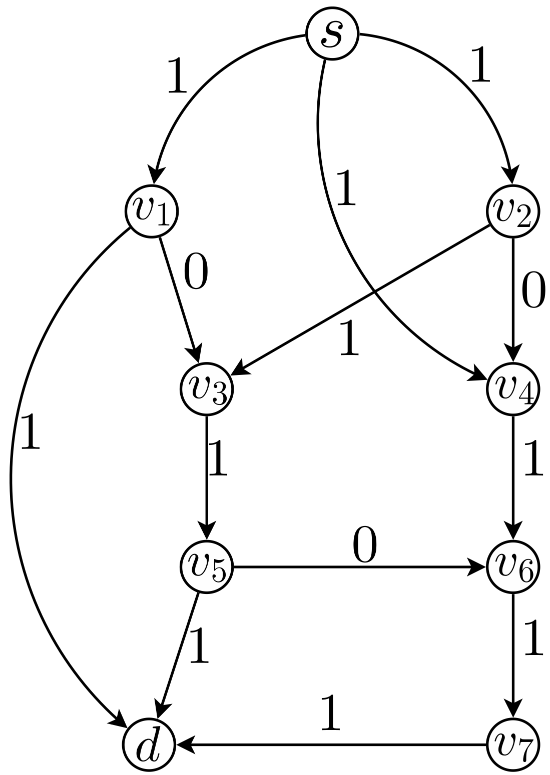


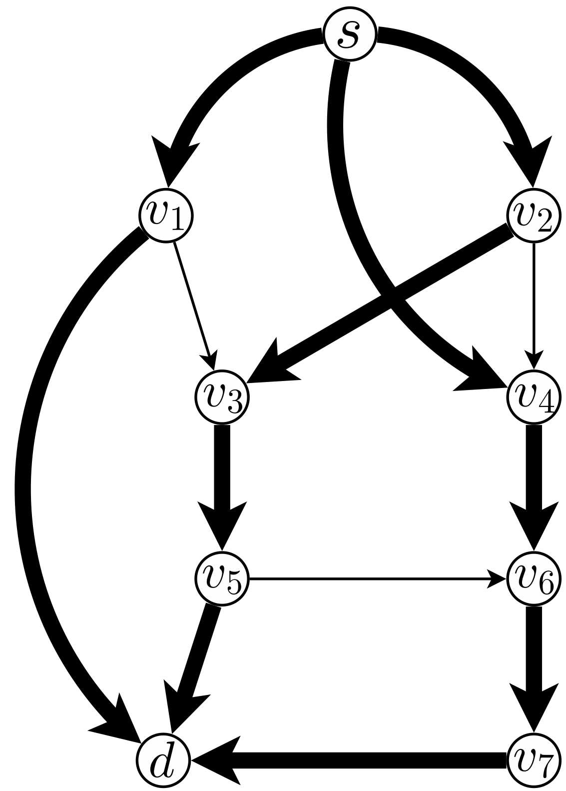
Cont'd

Step 4: Compute the inner products

Comparison to the true max flow found offline





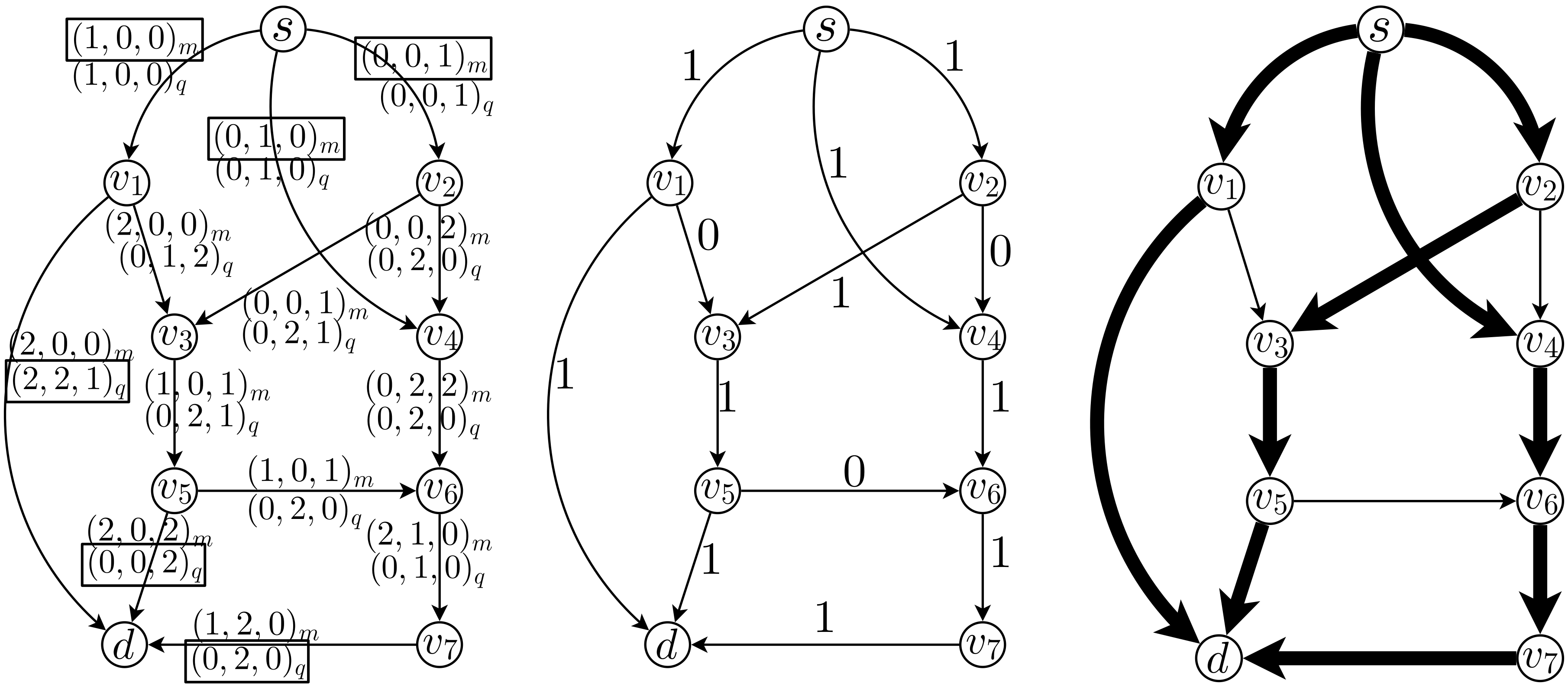


Voila!

Cont'd

Step 4: Compute the inner products

Comparison to the true max flow found offline



Voila!

Coded feedback helps identify redundant edges!!

A Provable Max-Flow Algorithm

High-level description:

Choose $\Gamma(v)$
loop
    Compute Forward Messages m_e
    Compute Coded Feedback q_e
    Find redundant edge set $E_R(v)$ by coded feedback
    if $E_R(v) \neq \emptyset$ **then**
        Remove $E_R(v)$.
    else
        return the remaining graph G
    end if
end loop

A Provable Max-Flow Algorithm

High-level description:

Choose $\Gamma(v)$
loop
    Compute Forward Messages m_e
    Compute Coded Feedback q_e
    Find redundant edge set $E_R(v)$ by coded feedback
    if $E_R(v) \neq \emptyset$ **then**
        Remove $E_R(v)$.
    else
        return the remaining graph G
    end if
end loop

- Zero overhead. Zero hardware requirement.

A Provable Max-Flow Algorithm

High-level description:

```
Choose Γ(v)
loop
    Compute Forward Messages m_e
    Compute Coded Feedback q_e
    Find redundant edge set E_R(v) by coded feedback
    if E_R(v) ≠ ∅ then
        Remove E_R(v).
    else
        return the remaining graph G
    end if
end loop
```

- Zero overhead. Zero hardware requirement.
- Distributiveness.

A Provable Max-Flow Algorithm

High-level description:

1: Choose $\Gamma(v)$
2: **loop**
3: Compute Forward Messages m_e
4: Compute Coded Feedback q_e
5: Find redundant edge set $E_R(v)$ by coded feedback
6: **if** $E_R(v) \neq \emptyset$ **then**
7: Remove $E_R(v)$.
8: **else**
9: **return** the remaining graph G
10: **end if**
11: **end loop**

- Zero overhead. Zero hardware requirement.
- Distributiveness.
- Minimal delay, no interruption to normal traffic.

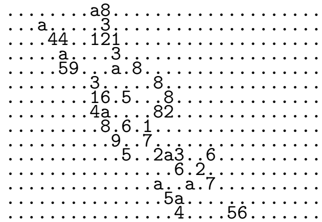
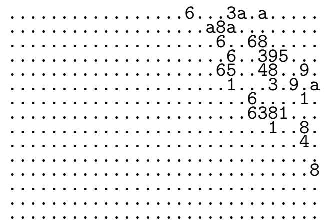
A Provable Max-Flow Algorithm

High-level description:

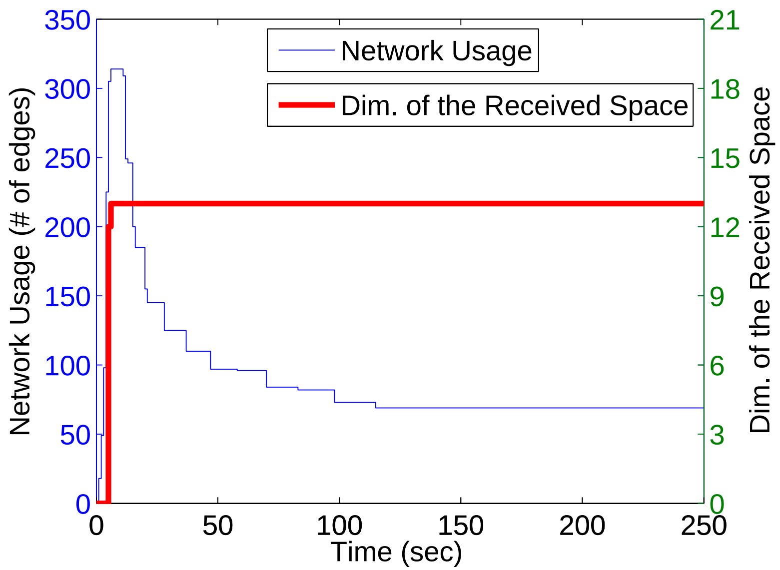
```
Choose Γ(v)
loop
    Compute Forward Messages m_e
    Compute Coded Feedback q_e
    Find redundant edge set E_R(v)
    by coded feedback
    if E_R(v) ≠ ∅ then
        Remove E_R(v).
    else
        return  the remaining graph G
    end if
end loop
```

- Zero overhead. Zero hardware requirement.
- Distributiveness.
- Minimal delay, no interruption to normal traffic.
- Fast convergence $\mathcal{O}(|V|^2)$ — no slower than push-&-relabel algorithm.

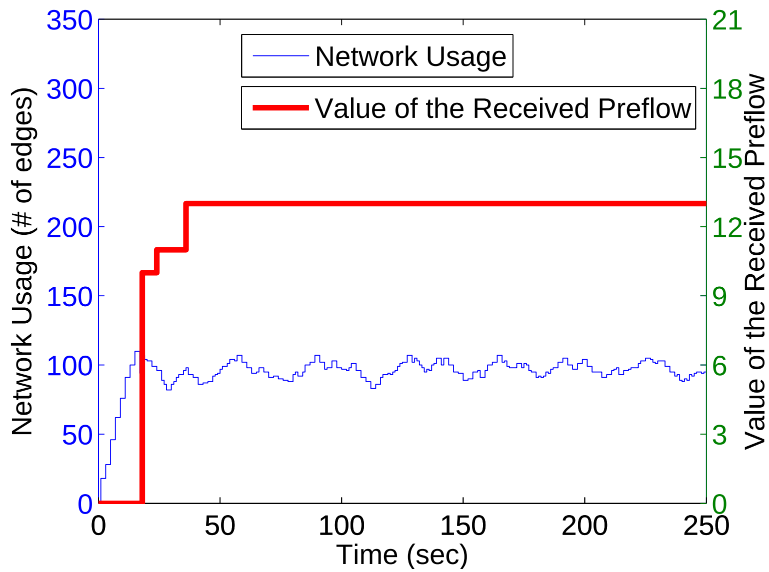
Simulation Results

A 30-node network with  

The coding-theoretic approach

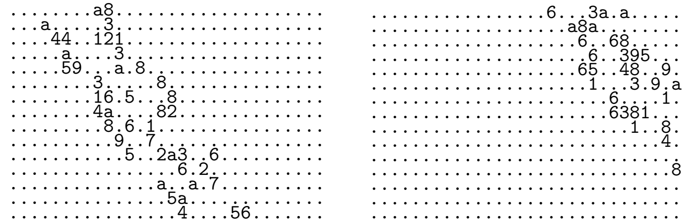


The push-&-relabel algorithm



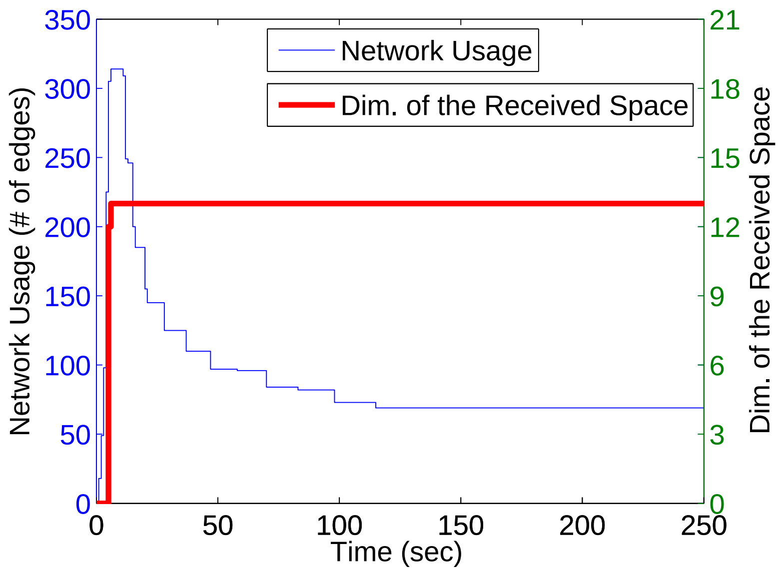
Simulation Results

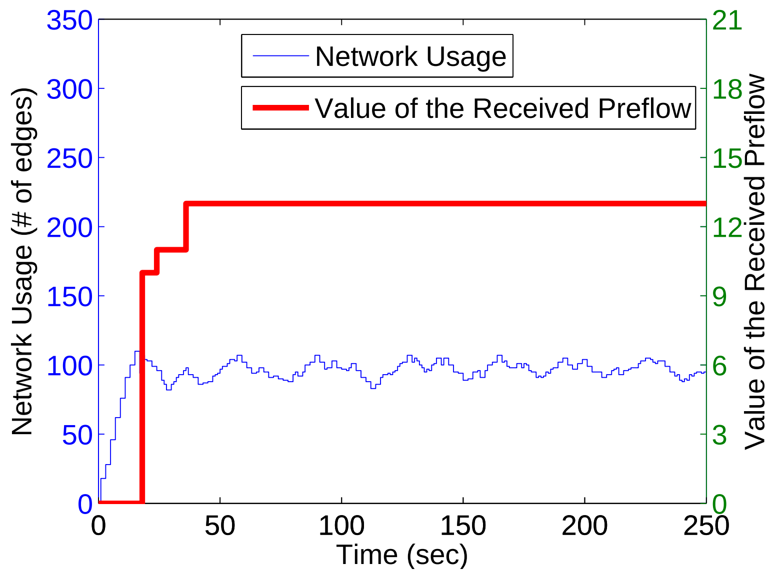
A 30-node network with



The coding-theoretic approach

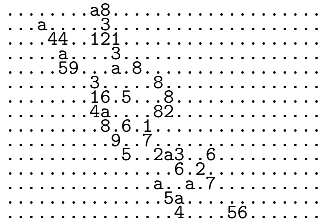
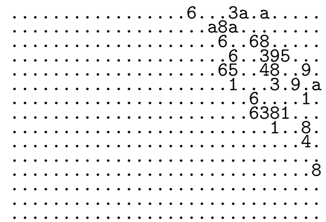
The push-&-relabel algorithm





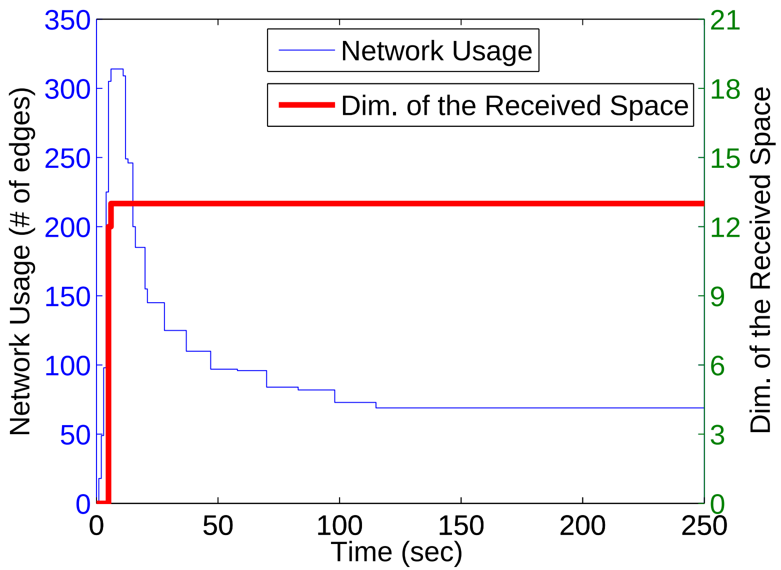
- Achieve the max-flow rate even before convergence.

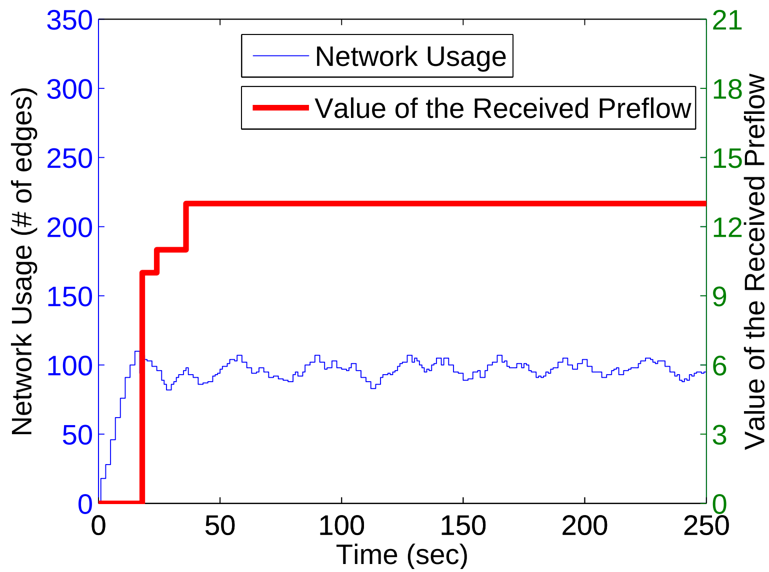
Simulation Results

A 30-node network with  

The coding-theoretic approach

The push-&-relabel algorithm





- Achieve the max-flow rate even before convergence.
- Monotonic traffic reduction vs. oscillating redirction of preflows.

Conclusion

- Graph-theoretic study of network coding.

Conclusion

- Graph-theoretic study of network coding.
- Characterization of pairwise intersession network coding
  - Paths with controlled edge overlap: A new basic unit for communications.

Conclusion

- Graph-theoretic study of network coding.
- Characterization of pairwise intersession network coding
  - Paths with controlled edge overlap: A new basic unit for communications.
- Algorithmic study of intrasession network coding.
  - The new max-flow algorithm: A practical application with solid foundation.