Network Coding (IN2315)

WiSe 2014/15

Technische Universität München

Department of Computer Science
Chair for Network Architectures and Services

Prof. Dr.-Ing. Georg Carle
Stephan M. Günther

Department of Electrical Engineering and Information Technology
Associate Institute for Signal Processing

Prof. Dr.-Ing. Wolfgang Utschick
Maximilian Riemensberger

Organizational issues

- Prof. Dr.-Ing. Georg Carle
- Email: carle@tum.de
- Office: MI 03.05.054
- Office hours: Mo 18:00 – 18:30 or after arrangement

- Prof. Dr.-Ing. Wolfgang Utschick
- Email: utschick@tum.de
- Office: N1122
- Office hours: after arrangement

- Dipl.-Ing. Stephan M. Günther, M. Sc.
- Email: guenther@tum.de
- Office: MI 03.05.061
- Office hours: anytime

- Maximilian Riemensberger, M. Sc. (hons)
- Email: riemensberger@tum.de
- Office: N1121
- Office hours: anytime

Lecture

- 6 ECTS (4 SWS)
- Targeted for Master students in Informatics
- Module number still tba (preliminary module is IN3300)
- Tuesday, 10:15 – 11:45, MI 03.07.023
- Thursday, 14:15 – 15:45, MI 03.07.023

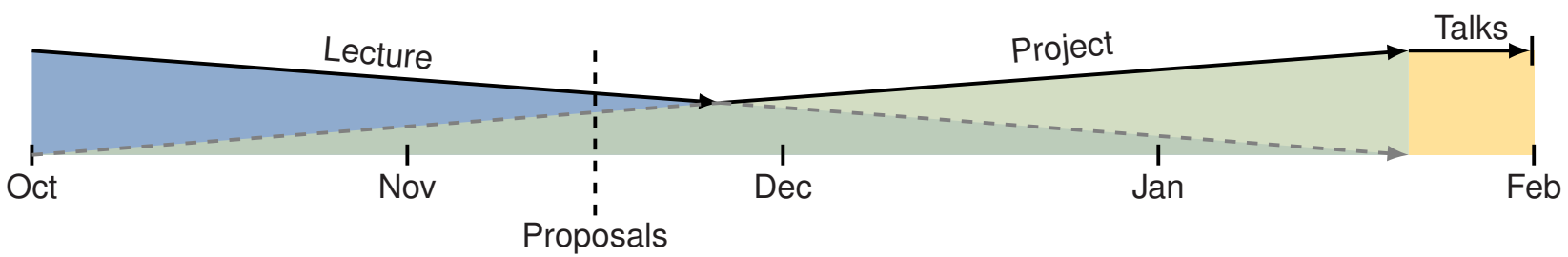
Exercises

- Lecture with integrated exercises
- Regular participation in lecture strongly recommended

Projects

- Individual projects (programming assignments in C)
- Should be worked on by teams of 2 students
- Topic proposals are welcome

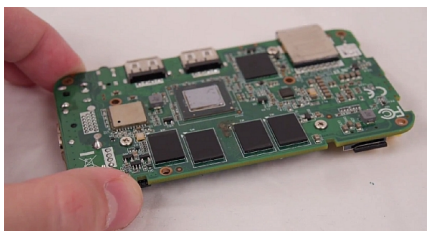
Time schedule



Supported by TUMLehrfonds

- The format of this course (lecture with integrated exercises, programming assignments, and final talks) are kindly supported by the TUMLehrfonds
- Benefit for you: we will purchase hardware to support you in creating innovative and interesting projects

(a) Odroid XU



(b) ZBox pico

(c) WLAN hardware

- We are still undecided which platform is the right one :)
- Hardware will be provided at the beginning of the project phase (December)

Exam

- Written exam at the end of the lecture period
  - 60 minutes / 60 credits
  - 1 sheet of paper (A4), hand-written (cheatsheet)
  - closed book otherwise
- No ”programming on paper”, promised
- Exam will be quite hard

Grading

- Exam gives 60 credits
- Project gives additional 40 bonus credits
- Credits earned in projects can only be added if you pass the exam with at least grade 4.0 (approximately 40% of 60 credits)

In a nutshell

- We strongly encourage you to participate in the projects
- Getting a top grade
  - should be easy if you participate in lecture and projects
  - but will be quite hard if you skip the projects

Grading example

Assuming that 40 % of 60 = 24 credits are needed to pass the written exam:

- Get 23 credits in the exam and 40 bonus credits from the project
  $\Rightarrow$ FAIL (thanks to APSO)
- Get 24 credits in the exam and 32 bonus credits
  $\Rightarrow$ Gives a total of 56 credits, which is most likely a 1.0
- Get 56 credits in the exam but 0 credits in the project
  $\Rightarrow$ Most likely a 1.0, same as above

Please note that this is only an example to illustrate the grading scheme.

Lecture material

The lecture material (slides, exercises) will be eventually made available on the course homepage. However:

- We will create an access-restricted git repository somewhere at `git@git.net.in.tum.de/nc/[somewildcard]`.
- It's so much more convenient, and we can provide you with material under copyright, e.g. book scans.
- Please send your RSA public key to `guenther@tum.de`.
- We will notify you as soon as the repository is online.

A note regarding attendance:

- We don't check your attendance in class.
- You should attend anyway:
  - Not everything may be on slides.
  - There will be discussions in class and presentations on the whiteboard.

What is Network Coding (NC)?

NC can be considered as a generalization of routing and forwarding:

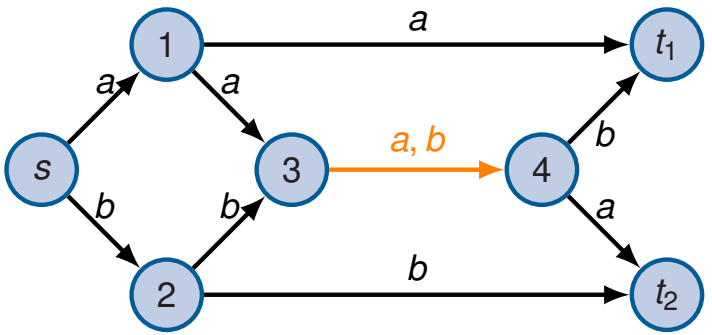
- Routing determines best-paths from source to destination.
- Forwarding switches packets along one of these paths.
- Forwarding merely creates replicas of incoming packets, i. e., a packet's payload remains unaltered.

NC drops this restriction:

- Outgoing packets are arbitrary combinations of previously received packets.
- The process of combining packets in such a way is referred to as coding.
- Since coding does not only happen at the source but on any node in the network, the network codes on packets.

Example 1: the famous butterfly network

Source s transmits 2 packets a, b to both t_1, t_2 (multicast):

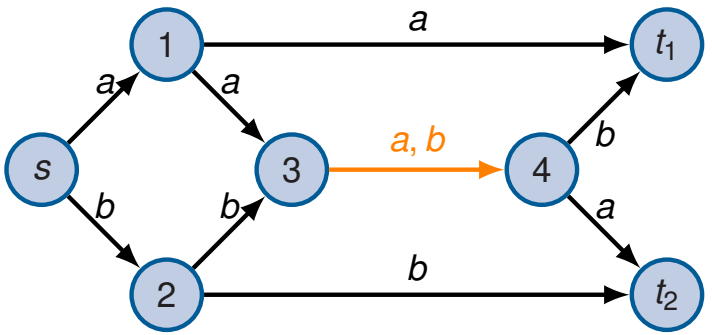


(a) Routing (with multicast)

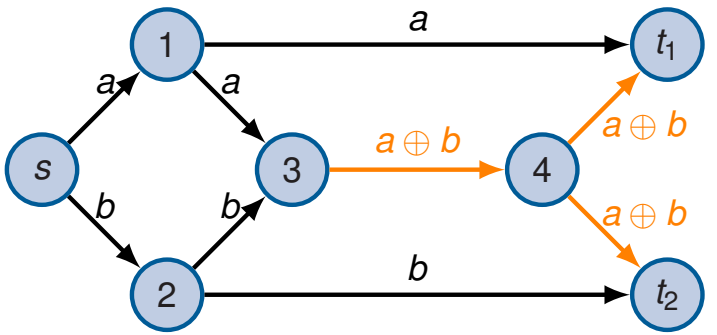
- The link $(3, 4)$ poses a bottleneck and must be used twice

Example 1: the famous butterfly network

Source s transmits 2 packets a, b to both t_1, t_2 (multicast):



(a) Routing (with multicast)



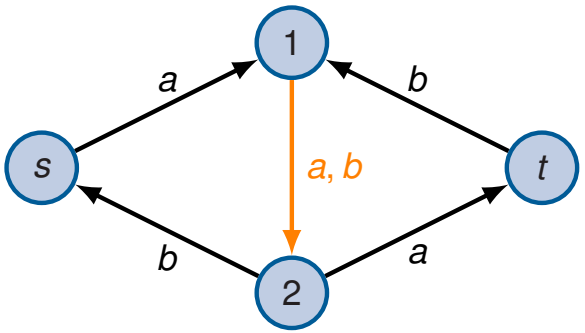
(b) Network Coding

- The link $(3, 4)$ poses a bottleneck and must be used twice

- NC saves one transmission on the critical link $(3, 4)$
- t_1, t_2 can decode the missing packet by XORing the coded packet with a and b, respectively

Example 2: diamond network

Nodes s, t want to communicate with each other (bidirectional unicasts):

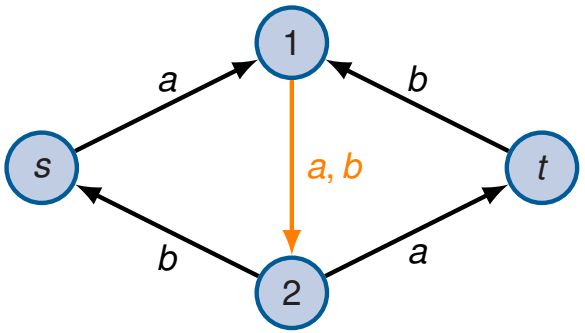


(a) Routing

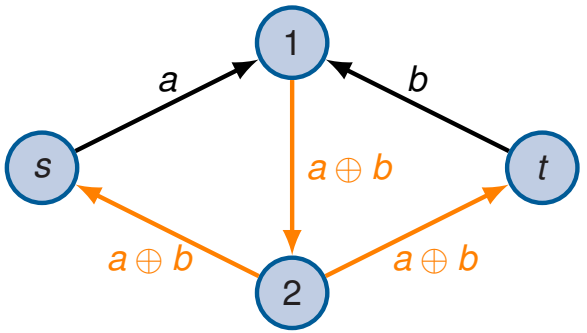
- The link $(1, 2)$ poses a bottleneck and must be used twice.

Example 2: diamond network

Nodes s, t want to communicate with each other (bidirectional unicasts):



(a) Routing



(b) Network Coding

- The link $(1, 2)$ poses a bottleneck and must be used twice.

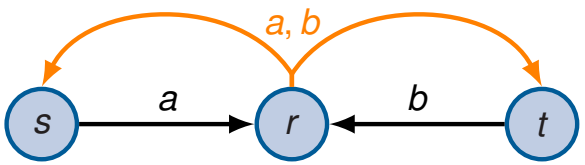
- NC saves again one transmission on the critical link $(1, 2)$.
- s, t know what they have sent and are thus able to decode.

Example 3: wireless relay network

Nodes s, t want to communicate with each other (bidirectional unicasts):

Note:

- Only 1 node can transmit at any time (otherwise transmissions would collide).
- A transmission by r is seen by both s, t (broadcast-nature of wireless networks).



(a) Routing

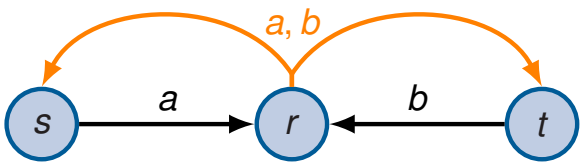
- The relay has to transmit a, b individually using 2 distinct broadcasts.
- Although s, t might overhear both transmissions, only one transmission is interesting for each node.

Example 3: wireless relay network

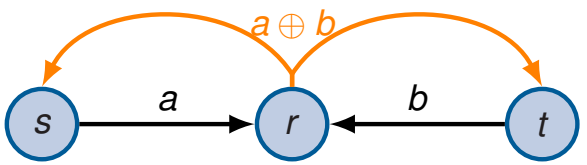
Nodes s, t want to communicate with each other (bidirectional unicasts):

Note:

- Only 1 node can transmit at any time (otherwise transmissions would collide).
- A transmission by r is seen by both s, t (broadcast-nature of wireless networks).



(a) Routing



(b) Network Coding

- The relay has to transmit a, b individually using 2 distinct broadcasts.
- Although s, t might overhear both transmissions, only one transmission is interesting for each node.

- With NC, the relay transmits $a \oplus b$.
- Both s, t know what they have sent and are thus able to decode the missing packet.

Applications of Network Coding

Throughput gain and reduced complexity

- Examples 1–3 already demonstrated the potential gain in throughput.
- May be even more interesting: in certain situations NC allows for a reduction in complexity:
  - The problem to find an optimal solution for Example 1 with routing results in the Steiner Tree problem, which is $\mathcal{NP}$.
  - With NC, a solution is found in polynomial time.

Applications of Network Coding

Throughput gain and reduced complexity

- Examples 1–3 already demonstrated the potential gain in throughput.
- May be even more interesting: in certain situations NC allows for a reduction in complexity:
  - The problem to find an optimal solution for Example 1 with routing results in the Steiner Tree problem, which is $\mathcal{NP}$.
  - With NC, a solution is found in polynomial time.

Robustness and adaptability

During the course of this class we will see that NC not only allows for

- more efficient channel usage but also
- reduces the cost of acknowledging and retransmitting packets.

Peer-to-peer content distribution (see Avalanche [4, 5])

Imagine a peer-to-peer network:

- A file is split into $n = 3$ blocks and spread over multiple nodes.
- Some node i has a set of $N(i) = \{1, 2, 3\}$ neighbors.
- For simplicity assume that each $j \in N(i)$ posseses the whole file.
- i asks each $j \in N(i)$ to send 1 of its blocks.
- Each $j \in N(i)$ chooses a packet independently and uniformely distributed.
- What is the probability that i gets the whole file?

Peer-to-peer content distribution (see Avalanche [4, 5])

Imagine a peer-to-peer network:

- A file is split into $n = 3$ blocks and spread over multiple nodes.
- Some node i has a set of $N(i) = \{1, 2, 3\}$ neighbors.
- For simplicity assume that each $j \in N(i)$ posseses the whole file.
- i asks each $j \in N(i)$ to send 1 of its blocks.
- Each $j \in N(i)$ chooses a packet independently and uniformely distributed.
- What is the probability that i gets the whole file?

$$p = 1 \cdot \frac{2}{3} \cdot \frac{1}{3} \approx 22\,\%$$

Peer-to-peer content distribution (see Avalanche [4, 5])

Imagine a peer-to-peer network:

- A file is split into $n = 3$ blocks and spread over multiple nodes.
- Some node i has a set of $N(i) = \{1, 2, 3\}$ neighbors.
- For simplicity assume that each $j \in N(i)$ posseses the whole file.
- i asks each $j \in N(i)$ to send 1 of its blocks.
- Each $j \in N(i)$ chooses a packet independently and uniformely distributed.
- What is the probability that i gets the whole file?

$$p = 1 \cdot \frac{2}{3} \cdot \frac{1}{3} \approx 22\,\%$$

Now assume the following:

- $j \in N(i)$ sends the XOR of a random number of blocks.
- To decide whether or not each of the blocks should be XORed, j flips a coin.
- The outcome of those trials is sent along with the XOR to i.
- i can obviously decode if those trials are linear independent.

Peer-to-peer content distribution (see Avalanche [4, 5])

Imagine a peer-to-peer network:

- A file is split into $n = 3$ blocks and spread over multiple nodes.
- Some node i has a set of $N(i) = \{1, 2, 3\}$ neighbors.
- For simplicity assume that each $j \in N(i)$ posseses the whole file.
- i asks each $j \in N(i)$ to send 1 of its blocks.
- Each $j \in N(i)$ chooses a packet independently and uniformely distributed.
- What is the probability that i gets the whole file?

$$p = 1 \cdot \frac{2}{3} \cdot \frac{1}{3} \approx 22\,\%$$

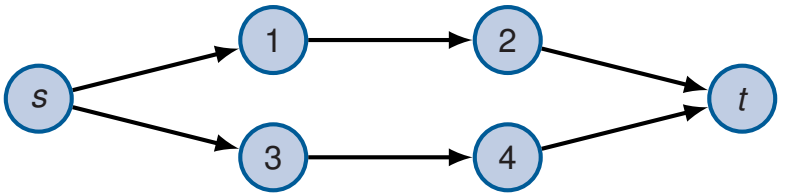
Now assume the following:

- $j \in N(i)$ sends the XOR of a random number of blocks.
- To decide whether or not each of the blocks should be XORed, j flips a coin.
- The outcome of those trials is sent along with the XOR to i.
- i can obviously decode if those trials are linear independent.

$$p' = \left(1 - \frac{1}{2^3}\right)\left(1 - \frac{1}{2^2}\right)\left(1 - \frac{1}{2}\right) \approx 32\,\%$$

Network security

- s wants to send messages to t.
- s knows that one of the four relay nodes is operated by an eavesdropper.



Routing:

- Since s does not know the eavesdropper, it has an odd by $1/2$ to choose the wrong path.
- Sending packets alternating over both paths might still yield information to the eavesdropper.

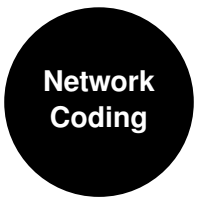
Network Coding:

- s splits every message to be sent into four packets p_i, $1 \leq i \leq 4$ of equal size.
- s then calculates

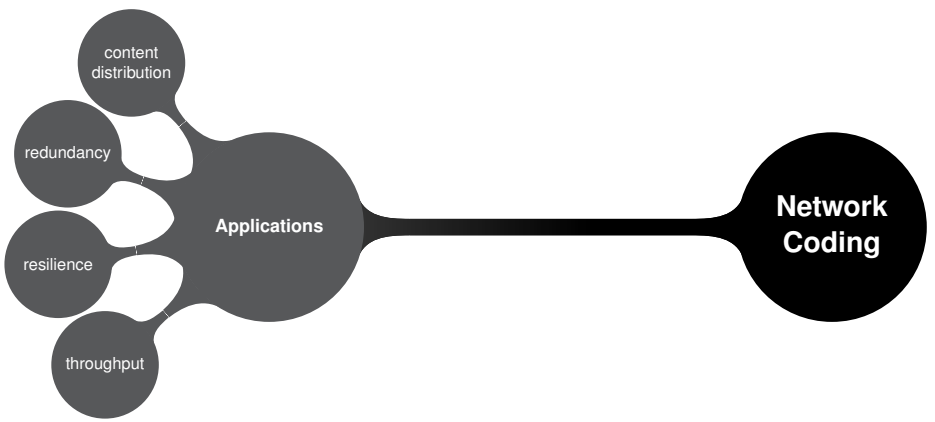
$$c_1 = p_1 \oplus p_2,\ c_2 = p_3 \oplus p_4,\ c_3 = p_1 \oplus p_4,\ c_4 = p_2 \oplus p_3$$

  and sends c_1, c_2 over one path and c_3, c_4 over the other one.
- As long as the eavesdropper is unable to guess the contents of at least one packet, decoding is impossible.

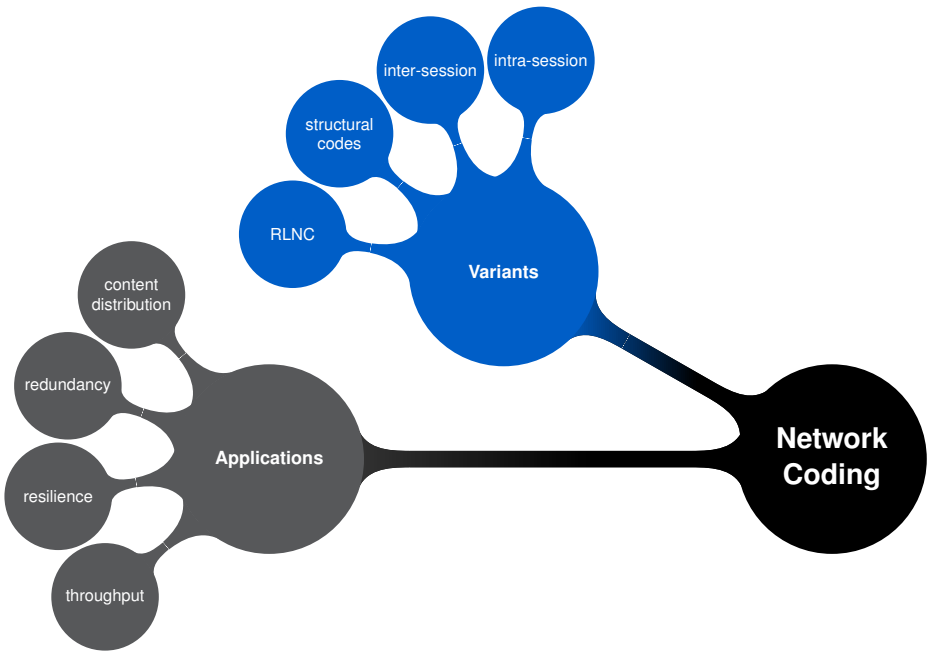
Mindmap: Network Coding and lecture outline



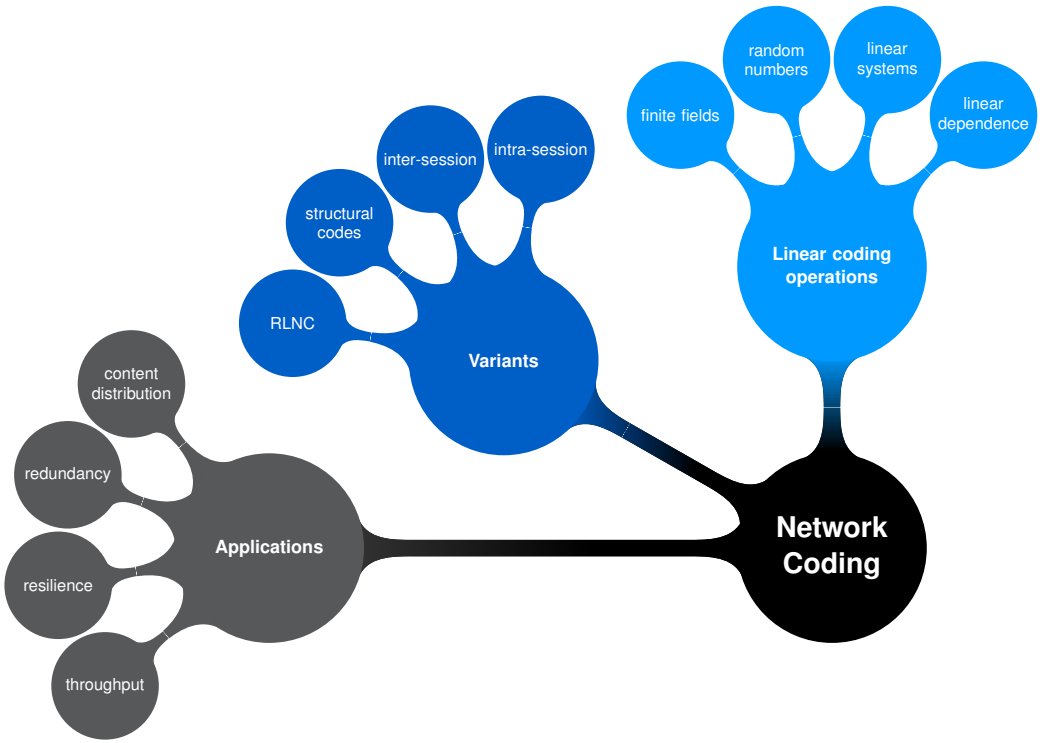
Mindmap: Network Coding and lecture outline



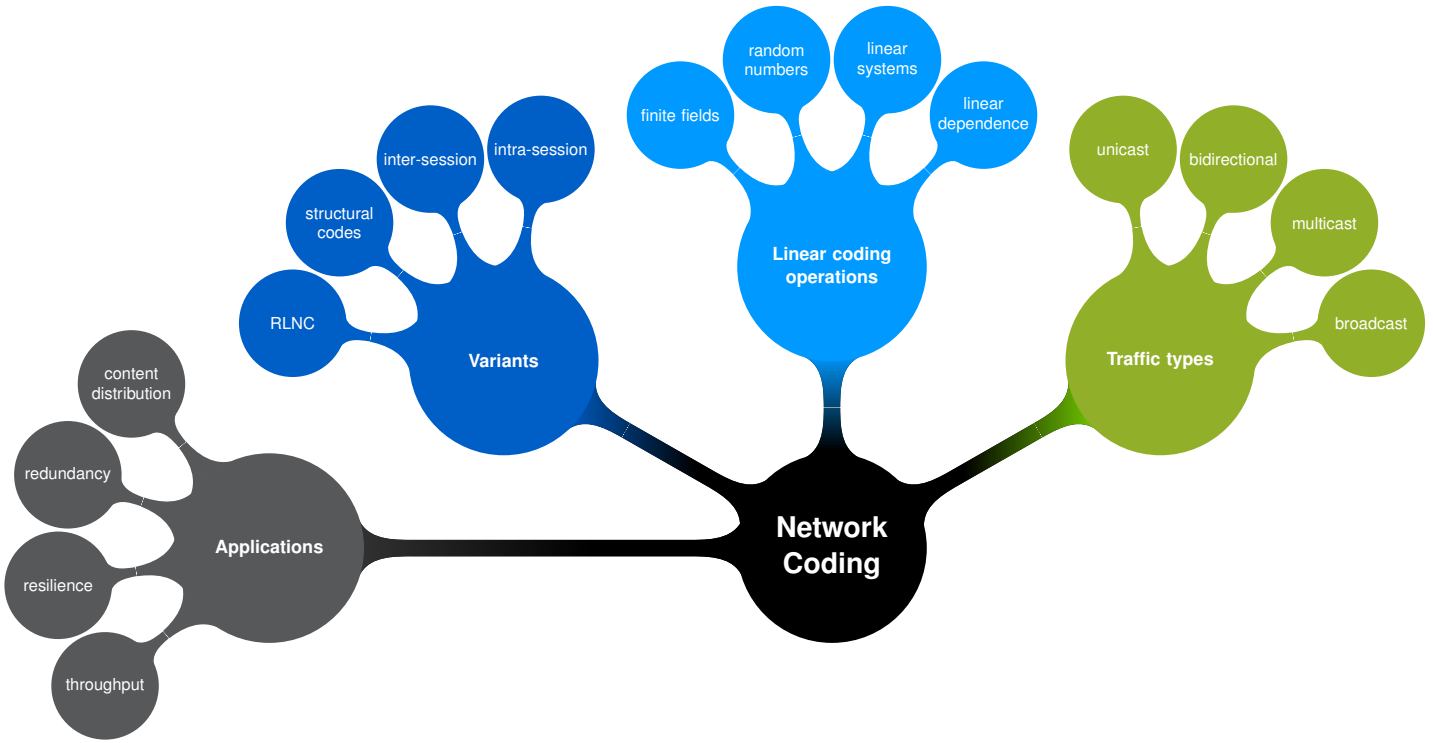
Mindmap: Network Coding and lecture outline



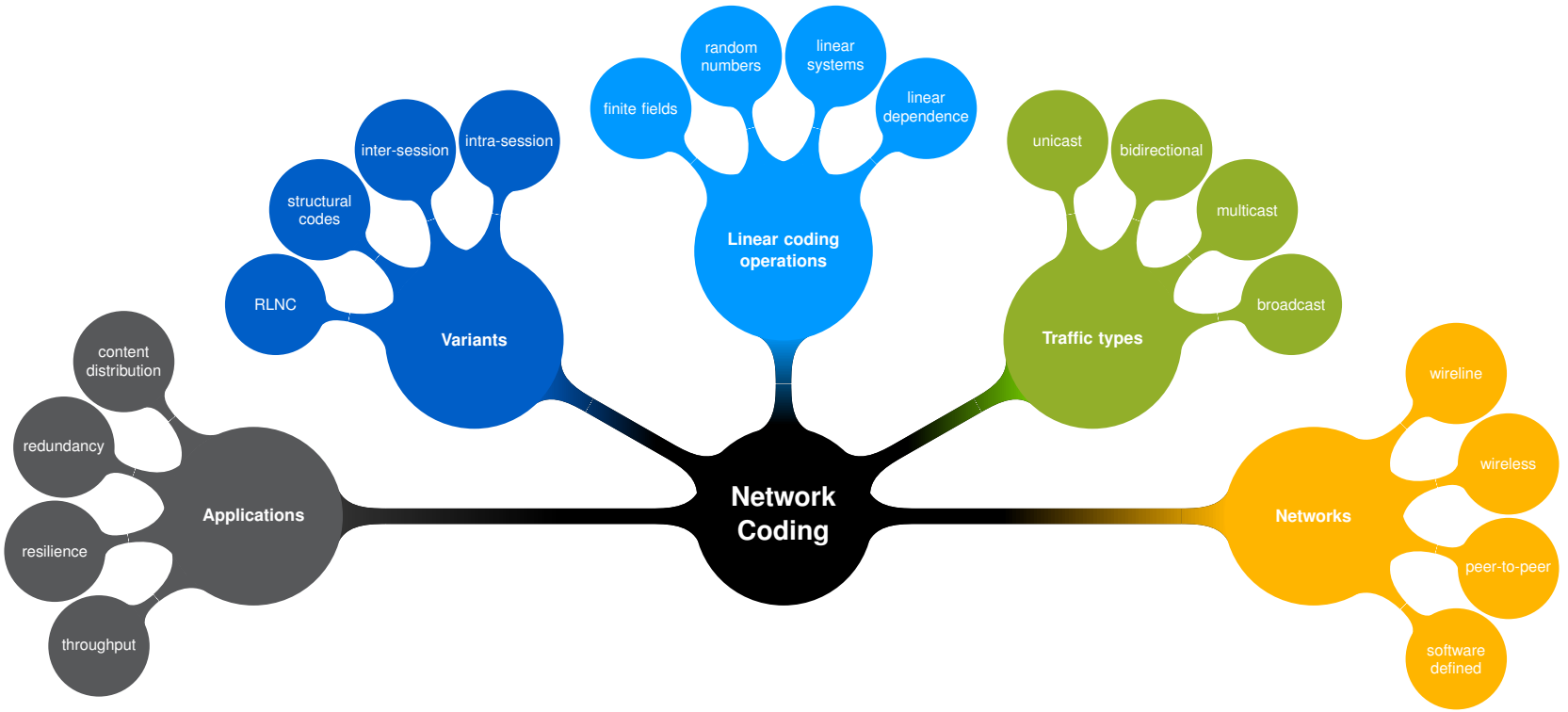
Mindmap: Network Coding and lecture outline



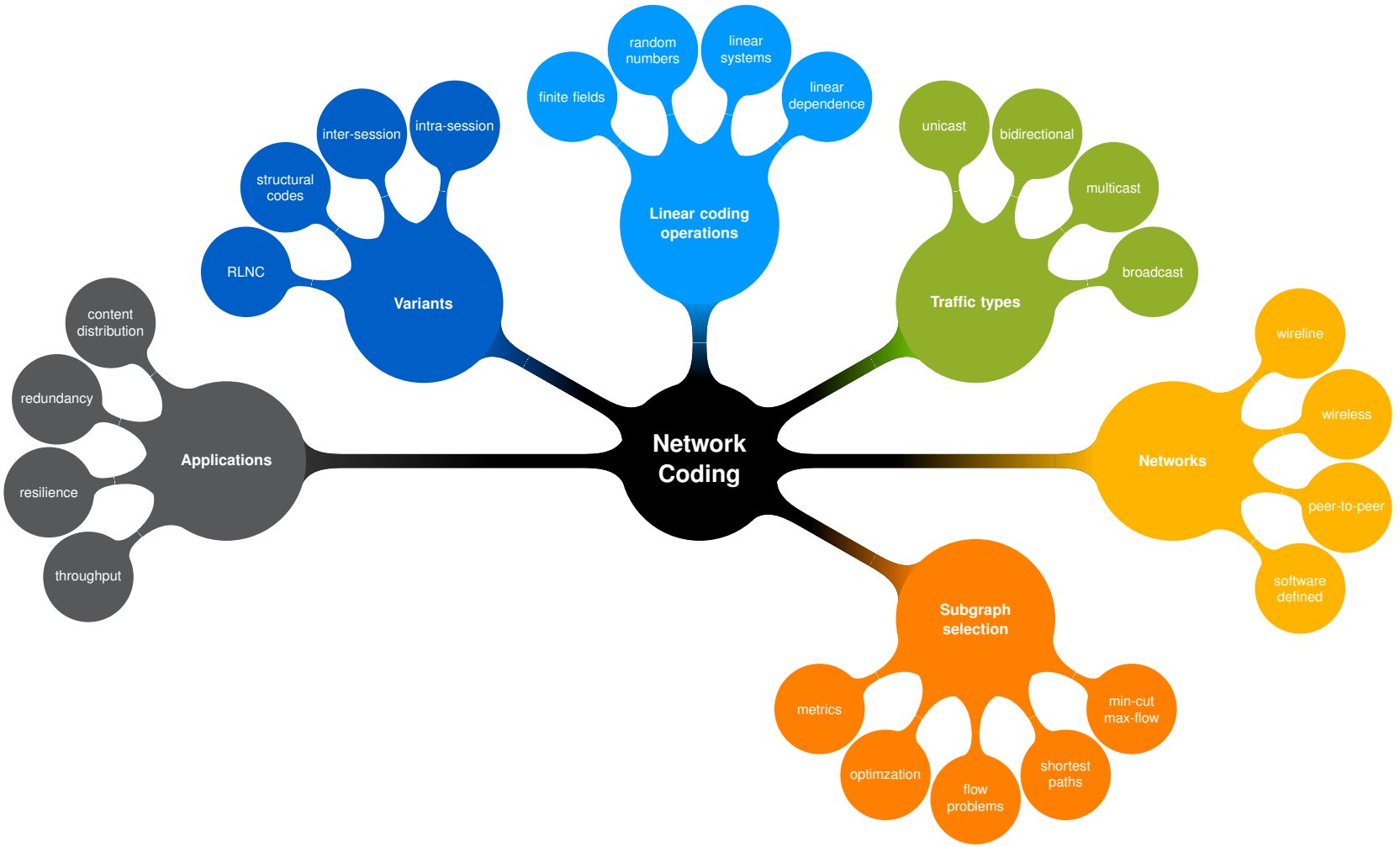
Mindmap: Network Coding and lecture outline



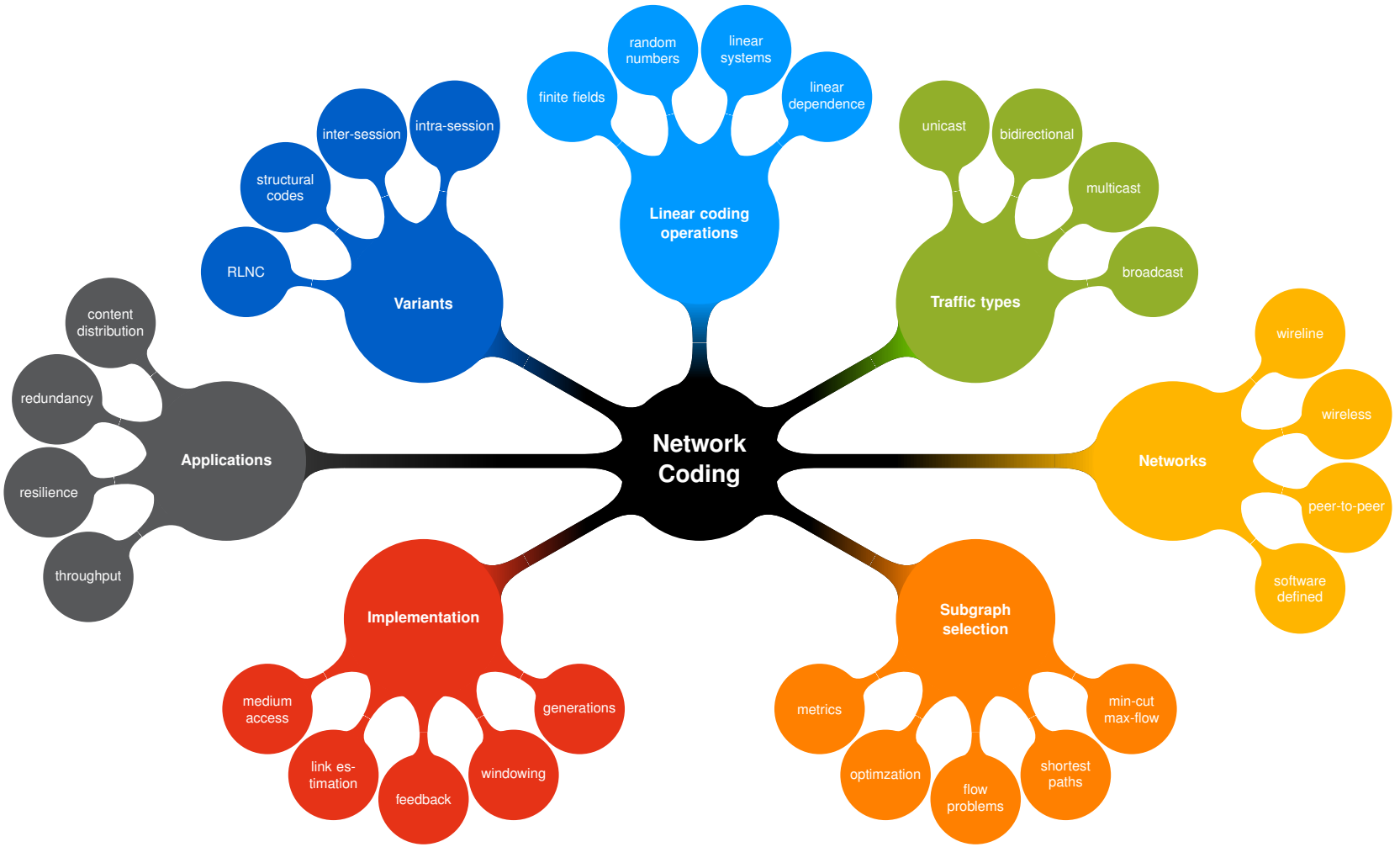
Mindmap: Network Coding and lecture outline



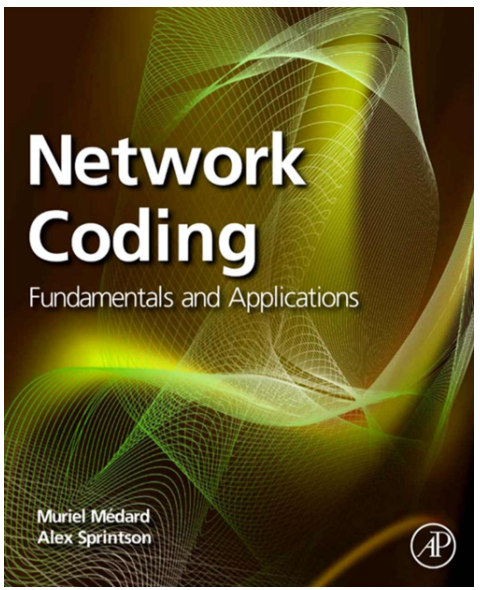
Mindmap: Network Coding and lecture outline



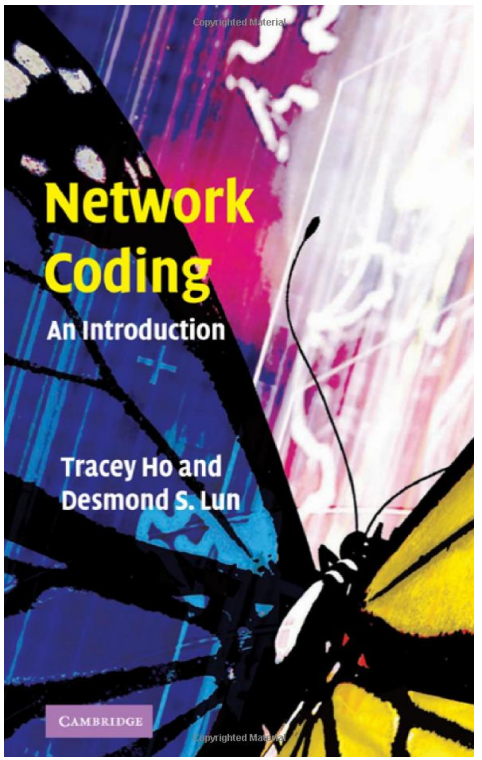
Mindmap: Network Coding and lecture outline



Literature



(a) Network Coding: Fundamentals and Applications [8]



(b) Network Coding: An Introduction [6]

And don't forget to study the Linux Kernel Coding Style [7]!

Finite fields

Remember finite fields $\langle \mathbb{F}_q, +, \cdot \rangle$ with $q = p^n$ elements where $p \in \mathbb{N}$ is prime?

Finite fields

Remember finite fields $\langle \mathbb{F}_q, +, \cdot \rangle$ with $q = p^n$ elements where $p \in \mathbb{N}$ is prime?

Definition: Finite fields (Galois fields)
A field $\mathbb{F}$ is a set of elements with two binary operators $(+, \cdot)$ such that $\langle \mathbb{F}, + \rangle$ and $\langle \mathbb{F} \backslash \{0\}, \cdot \rangle$ form Abelian groups with $\cdot$ being distributive over $+$. If the number of elements in $\mathbb{F}$ is finite, the field is said to be finite.

Finite fields

Remember finite fields $\langle \mathbb{F}_q, +, \cdot \rangle$ with $q = p^n$ elements where $p \in \mathbb{N}$ is prime?

Definition: Finite fields (Galois fields)
A field $\mathbb{F}$ is a set of elements with two binary operators $(+, \cdot)$ such that $\langle \mathbb{F}, + \rangle$ and $\langle \mathbb{F} \setminus \{0\}, \cdot \rangle$ form Abelian groups with $\cdot$ being distributive over $+$. If the number of elements in $\mathbb{F}$ is finite, the field is said to be finite.

If $q = p$, and we define for any $a, b \in \mathbb{F}_p$

$$a +_p b = (a + b) \bmod p, \text{ and}$$
$$a \cdot_p b = (a \cdot b) \bmod p,$$

then $\langle \mathbb{Z}_p, +_p, \cdot_p \rangle$ is a finite field.

Finite fields

Remember finite fields $\langle \mathbb{F}_q, +, \cdot \rangle$ with $q = p^n$ elements where $p \in \mathbb{N}$ is prime?

Definition: Finite fields (Galois fields)
A field $\mathbb{F}$ is a set of elements with two binary operators $(+, \cdot)$ such that $\langle \mathbb{F}, + \rangle$ and $\langle \mathbb{F} \backslash \{0\}, \cdot \rangle$ form Abelian groups with $\cdot$ being distributive over $+$. If the number of elements in $\mathbb{F}$ is finite, the field is said to be finite.

If $q = p$, and we define for any $a, b \in \mathbb{F}_p$

$$a +_p b = (a + b) \bmod p, \text{ and}$$
$$a \cdot_p b = (a \cdot b) \bmod p,$$

then $\langle \mathbb{Z}_p, +_p, \cdot_p \rangle$ is a finite field.

Beware:

- $\langle \mathbb{Z}_4, +_4, \cdot_4 \rangle$ is **not** a finite field as 4 is obviously not prime.
- However, there is a finite field with 4 elements, since 4 is a prime power. We just have to define addition and multiplication accordingly.

Example: the binary field $\langle \mathbb{F}_2, +, \cdot \rangle$

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

$\Rightarrow$ addition is a bitwise XOR, multiplication a bitwise AND.

- Cool for computers since those machines tend to work in bits.
- Even cooler if it was possible to work on their native block size, which are bytes[1] of 8 bit.
- However, $q = 256$ with + and · defined modulo-style is not a finite field. :(
- But wait: $q = 2^8$ is a prime power, and our definition claimed that there are finite fields for any p^n as long as p is prime. :)
- For the same reason there is also a finite field with 4 elements...

[1] Fun fact: the size of a byte is **not** defined. We merely assume it as 8 bit block since there are few computers around that work on 7 bit blocks.

Binary extension fields

Definition: binary extension fields

A binary extension field is a set of polynomials

$$F_q[x] = \left\{ \sum_{i=0}^{n-1} a_i x^i \;\middle|\; a_i \in \mathbb{F}_2 \right\}$$

of order $q = 2^n$ with appropriately defined binary operators $(+, \cdot)$.

Binary extension fields

Definition: binary extension fields

A binary extension field is a set of polynomials

$$F_q[x] = \left\{ \sum_{i=0}^{n-1} a_i x^i \;\middle|\; a_i \in \mathbb{F}_2 \right\}$$

of order $q = 2^n$ with appropriately defined binary operators $(+, \cdot)$.

Examples:

- $q = 2 \Rightarrow F_2[x] = \{0, 1\}$ (same as $\mathbb{F}_2$)
- $q = 4 \Rightarrow F_4[x] = \{0, 1, x, x + 1\}$

Binary extension fields

Definition: binary extension fields

A binary extension field is a set of polynomials

$$F_q[x] = \left\{ \sum_{i=0}^{n-1} a_i x^i \;\middle|\; a_i \in \mathbb{F}_2 \right\}$$

of order $q = 2^n$ with appropriately defined binary operators $(+, \cdot)$.

Examples:

- $q = 2 \Rightarrow F_2[x] = \{0, 1\}$ (same as $\mathbb{F}_2$)
- $q = 4 \Rightarrow F_4[x] = \{0, 1, x, x + 1\}$

Addition of any $a, b \in F_q[x]$ is defined as

$$a(x) + b(x) = \sum_{i=0}^{n-1} a_i x^i + \sum_{i=0}^{n-1} b_i x^i = \sum_{i=0}^{n-1} (a_i + b_i) x^i,$$

where + means XOR, which makes sense considering that the polynomials' coefficients are elements of $\mathbb{F}_2$.

What about multiplication?

- Multiplication of two polynomials of degree n and m yield another polynomial of degree at most $n + m$.
- Ordinary multiplication of $a, c \in F_q[x]$ would in general give a result $\notin F_q[x]$.

What about multiplication?

- Multiplication of two polynomials of degree n and m yield another polynomial of degree at most $n + m$.
- Ordinary multiplication of $a, c \in F_q[x]$ would in general give a result $\notin F_q[x]$.

The trick: reduce the multiplication result subject to some prime element (compare $\mathbb{F}_p$).

Definition: irreducible polynomial (reduction polynomial)
A polynomial of degree n over the binary field $\mathbb{F}_2$ is called irreducible if it cannot be represented as product of two polynomials $a, b \in F_q[x]$ of degree strictly less than n. Such a polynomial is guaranteed to exist and in general not unique.

Multiplication of any two $a, c \in F_q[x]$ is defined as

$$b(x) = (a(x) \cdot c(x)) \bmod r(x),$$

where $r(x)$ is irreducible.

Example: $F_4[x] = \{0, 1, x, x + 1\}$ and $r(x) = x^2 + x + 1$

Example: $F_4[x]$

- $F_4[x] = \{0, 1, x, x+1\}$
- $r(x) = x^2 + x + 1$

+	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

·	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

Example: $F_4[x]$

- $F_4[x] = \{0, 1, x, x+1\}$
- $r(x) = x^2 + x + 1$

+	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

·	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

- Except for the highlighted values, the tables should be clear.
- Here is an example for the reduction of $x \cdot x$:

$$x \cdot x \bmod x^2 + x + 1 = \begin{array}{r} x^2 \\ +x^2 + x + 1 \\ \hline x + 1 \end{array} \quad \bmod x^2 + x + 1$$

Homework: Do the same for $F_8[x]$, and does it work for $F_6[x]$?

Primitive element (generator)

Definition: primitive element

A primitive element $g \in \mathbb{F}_q$ is an polynomial such that

$$\bigcup_{i=1}^{q-1} \left\{ g^i \right\} = \mathbb{F}_q \backslash \{0\},$$

and is in general not unique.

Primitive elements are also referred to as generators. Each finite field has at least one. This also holds for extension fields $F_q[x]$.

Examples:

- For $\mathbb{F}_7$, $g = 3$ is a generator.
- For $F_4[x]$, $g(x) = x + 1$ is a generator.

Discrete logarithm

Primitive elements give raise for another multiplication algorithm: let

- $L[a]$ denote the discrete logarithm of $a \in F_q[x]$ and
- $A[a]$ the discrete power (antilog) of a.

Then, product and quotient of $a, b \in F_q[x]$ are given as

$$a \cdot b = A[L[a] + L[b]] \text{ and } a/b = A[L[a] - L[b]].$$

Discrete logarithm

Primitive elements give raise for another multiplication algorithm: let

- $L[a]$ denote the discrete logarithm of $a \in F_q[x]$ and
- $A[a]$ the discrete power (antilog) of a.

Then, product and quotient of $a, b \in F_q[x]$ are given as

$$a \cdot b = A[L[a] + L[b]] \text{ and } a/b = A[L[a] - L[b]].$$

First, we create two tables:

1. A containing all powers of g, i. e., $A[i] = g^i$
2. L containing the inverse elements, i. e., $L[g^i] = i$

For these steps we need to choose a reduction polyomial r. All calculations are done subject to this polynomial.

Multiplication $a \cdot b \bmod r$ is done as follows:

1. Determine $L[a]$ and $L[b]$, which yields the powers i, j such that $g^i = a$ and $g^j = b$.
2. Find $A[i + j]$, where addition is done modulo q (no XOR here).

Example: $F_{256}[x]$, $r(x) = x^8 + x^4 + x^3 + x + 1$, $g(x) = x + 1$

$$\underbrace{\left(x^2 + x\right)}_{\texttt{0x06}} \underbrace{\left(x^6 + x^4 + x + 1\right)}_{\texttt{0x53}} = ?$$

1. Lookup `0x06` and `0x53` in L, which yields `0x1a` and `0x30`, respectively.
2. Lookup `0x1a` + `0x30` mod `0xff` = `0x4a` in A.
3. This gives `0xf1` = $x^7 + x^6 + x^5 + x^4 + 1$, which is the result.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	01	03	05	0f	11	33	55	ff	1a	2e	72	96	a1	f8	13	35
1	5f	e1	38	48	d8	73	95	a4	f7	02	06	0a	1e	22	66	aa
2	e5	34	5c	e4	37	59	eb	26	6a	be	d9	70	90	ab	e6	31
3	53	f5	04	0c	14	3c	44	cc	4f	d1	68	b8	d3	6e	b2	cd
4	4c	d4	67	a9	e0	3b	4d	d7	62	a6	f1	08	18	28	78	88
5	83	9e	b9	d0	6b	bd	dc	7f	81	98	b3	ce	49	db	76	9a
6	b5	c4	57	f9	10	30	50	f0	0b	1d	27	69	bb	d6	61	a3
7	fe	19	2b	7d	87	92	ad	ec	2f	71	93	ae	e9	20	60	a0
8	fb	16	3a	4e	d2	6d	b7	c2	5d	e7	32	56	fa	15	3f	41
9	c3	5e	e2	3d	47	c9	40	c0	5b	ed	2c	74	9c	bf	da	75
10	9f	ba	d5	64	ac	ef	2a	7e	82	9d	bc	df	7a	8e	89	80
11	9b	b6	c1	58	e8	23	65	af	ea	25	6f	b1	c8	43	c5	54
12	fc	1f	21	63	a5	f4	07	09	1b	2d	77	99	b0	cb	46	ca
13	45	cf	4a	de	79	8b	86	91	a8	e3	3e	42	c6	51	f3	0e
14	12	36	5a	ee	29	7b	8d	8c	8f	8a	85	94	a7	f2	0d	17
15	39	4b	dd	7c	84	97	a2	fd	1c	24	6c	b4	c7	52	f6	01

(c) A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	00	ff	19	01	32	02	1a	c6	4b	c7	1b	68	33	ee	df	03
1	64	04	e0	0e	34	8d	81	ef	4c	71	08	c8	f8	69	1c	c1
2	7d	c2	1d	b5	f9	b9	27	6a	4d	e4	a6	72	9a	c9	09	78
3	65	2f	8a	05	21	0f	e1	24	12	f0	82	45	35	93	da	8e
4	96	8f	db	bd	36	d0	ce	94	13	5c	d2	f1	40	46	83	38
5	66	dd	fd	30	bf	06	8b	62	b3	25	e2	98	22	88	91	10
6	7e	6e	48	c3	a3	b6	1e	42	3a	6b	28	54	fa	85	3d	ba
7	2b	79	0a	15	9b	9f	5e	ca	4e	d4	ac	e5	f3	73	a7	57
8	af	58	a8	50	f4	ea	d6	74	4f	ae	e9	d5	e7	e6	ad	e8
9	2c	d7	75	7a	eb	16	0b	f5	59	cb	5f	b0	9c	a9	51	a0
10	7f	0c	f6	6f	17	c4	49	ec	d8	43	1f	2d	a4	76	7b	b7
11	cc	bb	3e	5a	fb	60	b1	86	3b	52	a1	6c	aa	55	29	9d
12	97	b2	87	90	61	be	dc	fc	bc	95	cf	cd	37	3f	5b	d1
13	53	39	84	3c	41	a2	6d	47	14	2a	9e	5d	56	f2	d3	ab
14	44	11	92	d9	23	20	2e	89	b4	7c	b8	26	77	99	e3	a5
15	67	4a	ed	de	c5	31	fe	18	0d	63	8c	80	c0	f7	70	07

(d) L

Linear coding operations

Encoding

- Data words are represented by polynomials $a \in F_q[x]$ over some binary extension field.
- A data packet is sequence of M words and thus a vector $\boldsymbol{a} \in F_q^M[x]$.
- A generation of N packets is written as matrix $\boldsymbol{A}^T = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$.
- A coded packet is obtained by

$$\boldsymbol{b} = \boldsymbol{A}^T \boldsymbol{c} = \sum_{i=1}^{N} c_i \boldsymbol{a}_i,$$

where $\boldsymbol{c} = [c_1, \ldots, c_N]^T \in F_q^N[x]$ denotes a vector of coding coefficients.

Transmission and recoding

- Packets are sent in general along with their coding vectors, i. e., $\boldsymbol{x}^T = \left[\boldsymbol{c}^T \; \boldsymbol{b}^T\right]$.
- Intermediate nodes may recode packets that have previously been received. Assume that some node has $k \leq N$ packets buffered, then

$$\boldsymbol{x}' = \sum_{i=0}^{k-1} c_i' \boldsymbol{x}_i$$

  is the recoded packet, where $c_i' \in F_q[x]$ are coding coefficient chosen by the intermediate node.
- Note that the original coding vector $\boldsymbol{c}$ is also recoded, i. e., the same linear transformation is applied to both the payload and the coding vector.

Decoding

- Packets are buffered at the receiver:

$$\begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_k^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1^T & \boldsymbol{b}_1^T \\ \vdots & \vdots \\ \boldsymbol{c}_k^T & \boldsymbol{b}_k^T \end{bmatrix} = \begin{bmatrix} c_{11} & \dots & c_{1N} & b_{11} & \dots & b_{1M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{k1} & \dots & c_{kN} & b_{k1} & \dots & b_{kM} \end{bmatrix} = [\boldsymbol{C}\ \boldsymbol{B}] = \boldsymbol{X}$$

- The receiver can decode if it has received N linear independent packets, i. e., $\boldsymbol{C} \in F_q^{N \times N}[x]$ and rank $\boldsymbol{C} = N$:

$$\boldsymbol{X}\boldsymbol{C}^{-1} = [\boldsymbol{C}\ \boldsymbol{B}]\,\boldsymbol{C}^{-1} = [\mathbf{1}\ \boldsymbol{A}]$$

- $\boldsymbol{C}^{-1}$ can be determined using Gaussian elimination.[2]
- Decoded packets are indicated by the rows of the unit matrix $\mathbf{1}$.
- Decoded packets are naturally in-order.

[2] Who not remembers Gaussian elemination should look it up until next time.

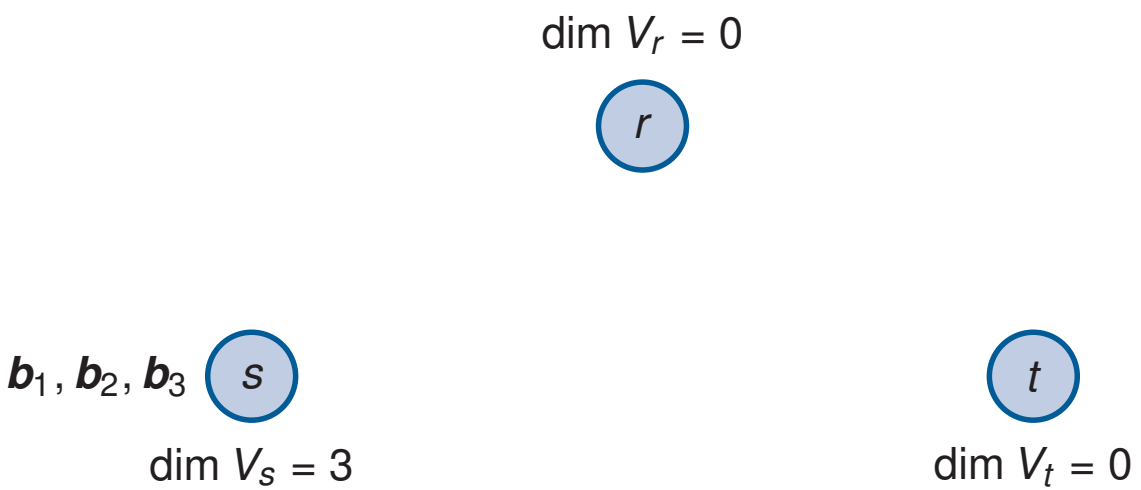
Observations:

- The coding matrix $\boldsymbol{C}$ fully describes the state of a generation:
  - rank $\boldsymbol{C}$ denotes the number of (eventually coded) packets in that generation.
  - $\boldsymbol{C} = \mathbf{1}$ means that all packets are decoded.
  - If $\boldsymbol{C}$ is in row-echelon[3] form, then a subset of packets may be decoded.
  - Recoding operations applied to $\boldsymbol{C}$ (or to a subset of its columns) are correspondingly applied to the packets (or a subset of those).
- The coding matrix at each node spans a vector space span $\boldsymbol{C}$, which is a subspace of the N-dimensional vector space $F_q^N[x]$.
- The dimension dim $\boldsymbol{C}$ and its changes over time are a concise description of the state of the network.

[3] more on that later

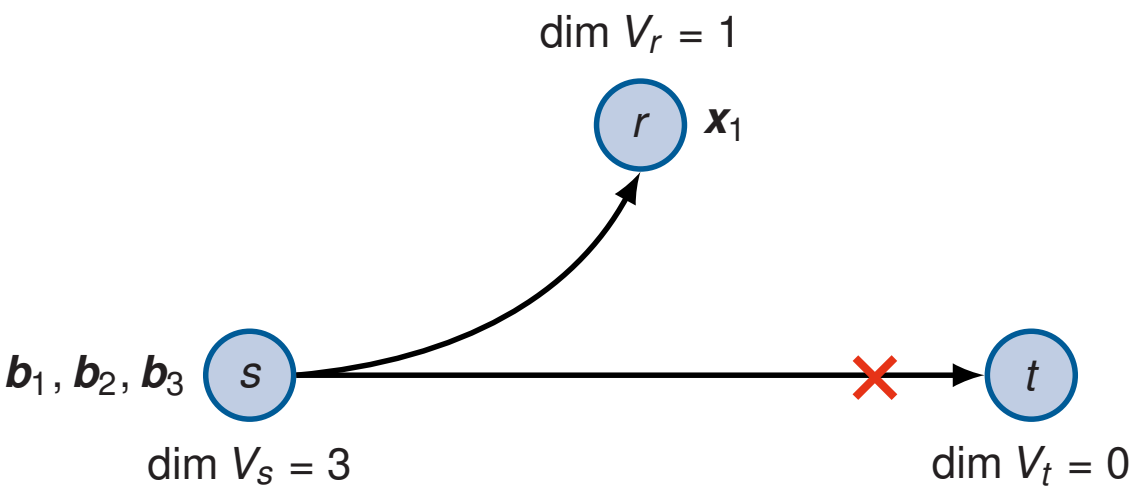
Example

- Erasure network with symmetric loss probabilities ϵ_{st}, ϵ_{sr}, and ϵ_{rt}.
- Unidirectional traffic from s to t.
- We assume that randomly choosen coding vectors are all linear independent.
- V_i is the subspace known to i, in particular we have dim $V_s = N$ and $V_r, V_t \subset V_s$.
- To keep it short we assume $N = 3$.



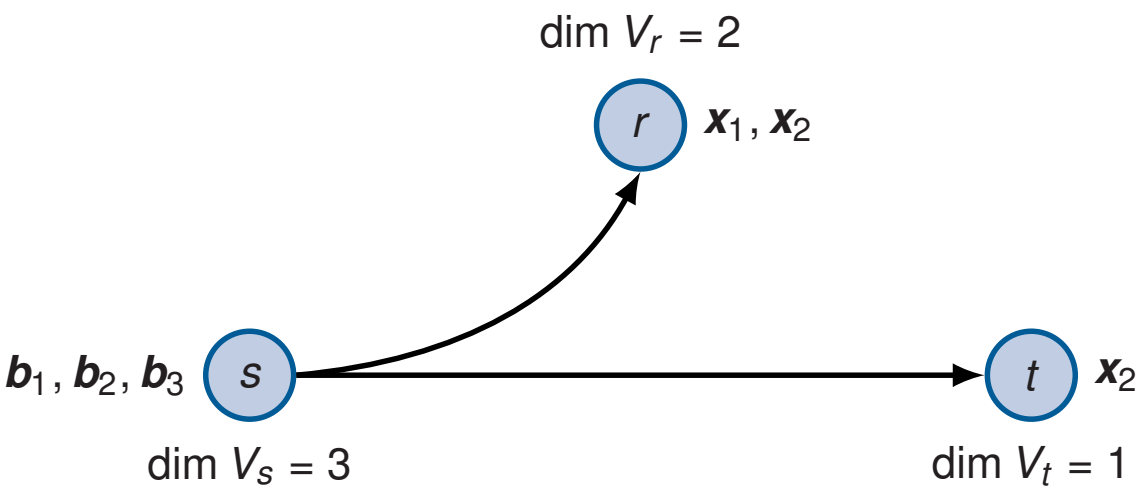
Example

- Erasure network with symmetric loss probabilities ϵ_{st}, ϵ_{sr}, and ϵ_{rt}.
- Unidirectional traffic from s to t.
- We assume that randomly choosen coding vectors are all linear independent.
- V_i is the subspace known to i, in particular we have dim $V_s = N$ and $V_r, V_t \subset V_s$.
- To keep it short we assume $N = 3$.



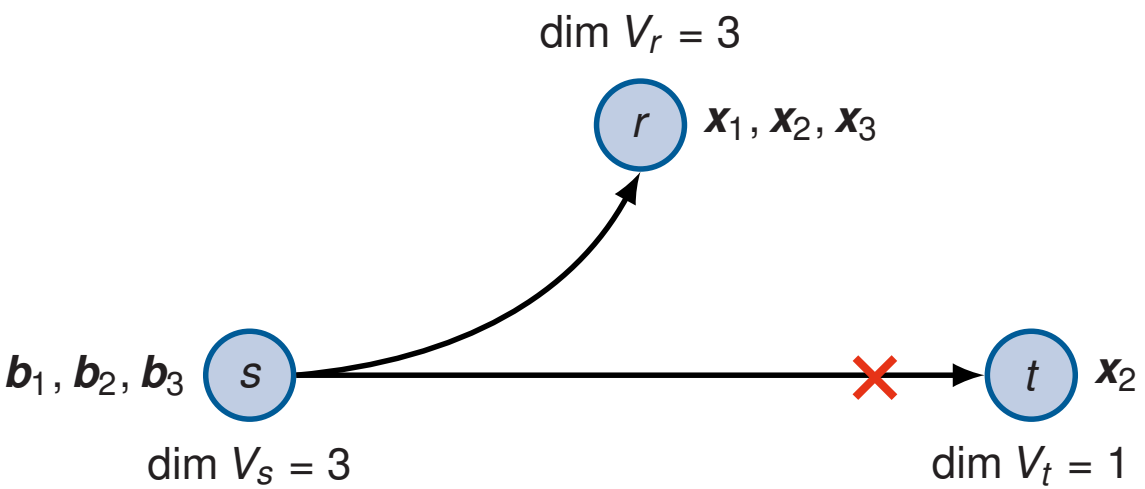
Example

- Erasure network with symmetric loss probabilities ϵ_{st}, ϵ_{sr}, and ϵ_{rt}.
- Unidirectional traffic from s to t.
- We assume that randomly choosen coding vectors are all linear independent.
- V_i is the subspace known to i, in particular we have dim $V_s = N$ and $V_r, V_t \subset V_s$.
- To keep it short we assume $N = 3$.



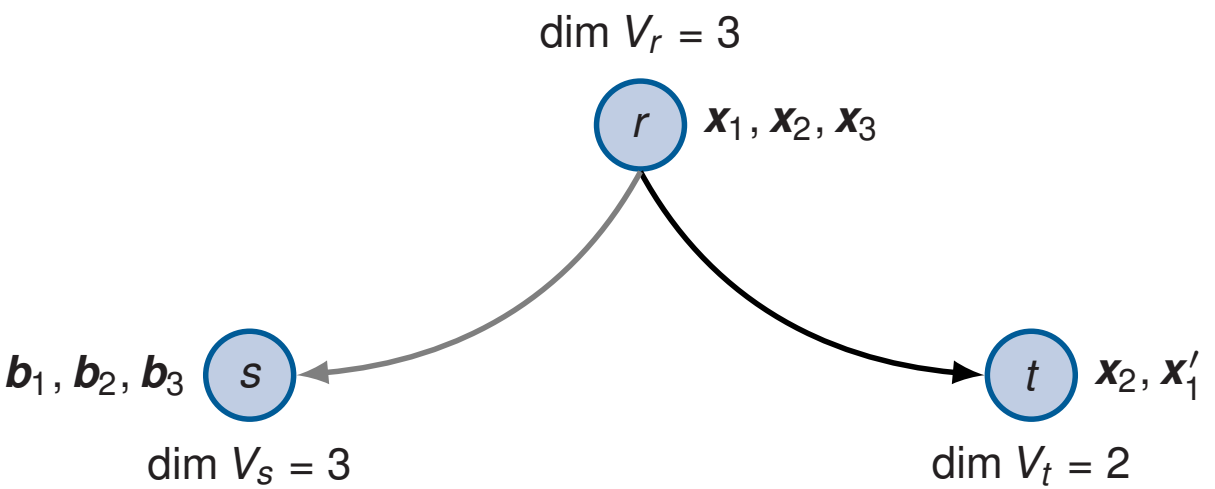
Example

- Erasure network with symmetric loss probabilities ϵ_{st}, ϵ_{sr}, and ϵ_{rt}.
- Unidirectional traffic from s to t.
- We assume that randomly choosen coding vectors are all linear independent.
- V_i is the subspace known to i, in particular we have dim $V_s = N$ and $V_r, V_t \subset V_s$.
- To keep it short we assume $N = 3$.



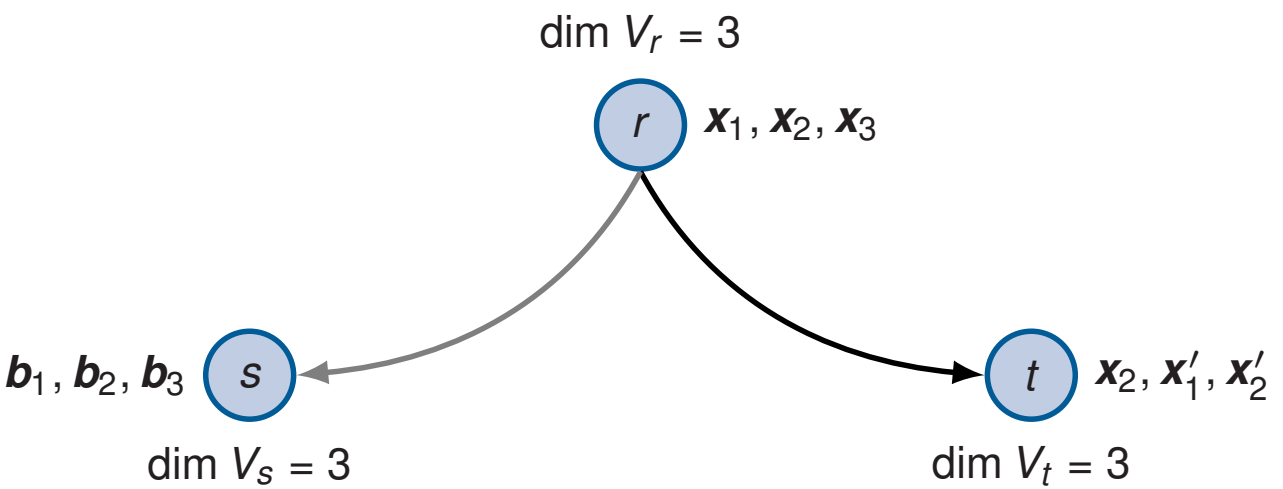
Example

- Erasure network with symmetric loss probabilities ϵ_{st}, ϵ_{sr}, and ϵ_{rt}.
- Unidirectional traffic from s to t.
- We assume that randomly choosen coding vectors are all linear independent.
- V_i is the subspace known to i, in particular we have dim $V_s = N$ and $V_r, V_t \subset V_s$.
- To keep it short we assume $N = 3$.



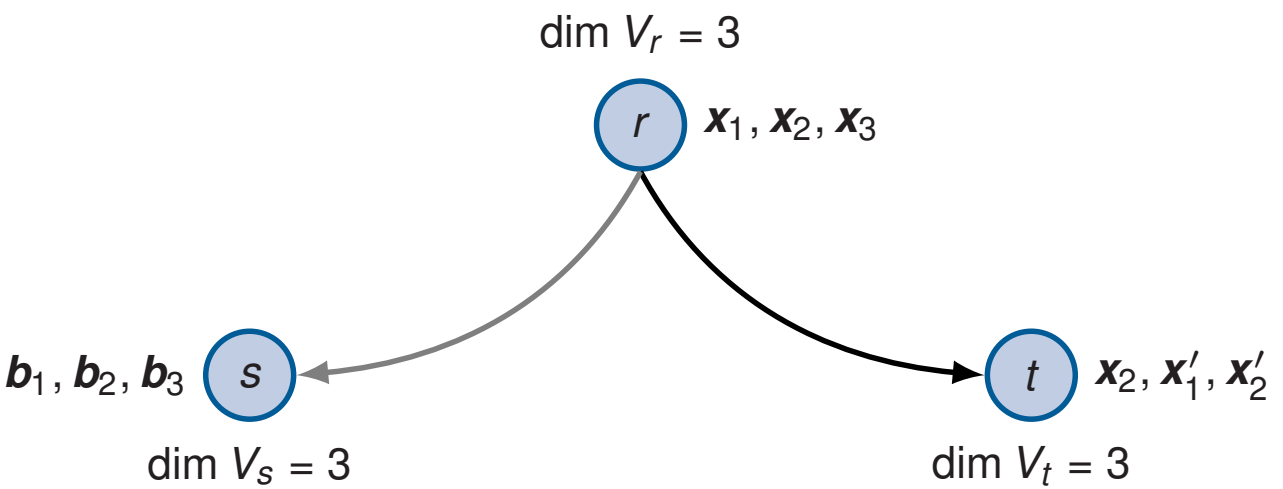
Example

- Erasure network with symmetric loss probabilities ϵ_{st}, ϵ_{sr}, and ϵ_{rt}.
- Unidirectional traffic from s to t.
- We assume that randomly choosen coding vectors are all linear independent.
- V_i is the subspace known to i, in particular we have dim $V_s = N$ and $V_r, V_t \subset V_s$.
- To keep it short we assume $N = 3$.



Example

- Erasure network with symmetric loss probabilities ϵ_{st}, ϵ_{sr}, and ϵ_{rt}.
- Unidirectional traffic from s to t.
- We assume that randomly choosen coding vectors are all linear independent.
- V_i is the subspace known to i, in particular we have dim $V_s = N$ and $V_r, V_t \subset V_s$.
- To keep it short we assume $N = 3$.



The recoded packets $\boldsymbol{x}'_j$ for $j \in \{1, 2\}$ can be written as $\boldsymbol{x}'_1 = \boldsymbol{X}^T \boldsymbol{c}'_1 = \sum_{i=1}^{3} c'_{1i} \boldsymbol{x}_i$.

Flows and sessions

Definition: (unicast) flow
In the context of network coding, we define an unicast flow (s, t) as the sequence of packets originating at some source node s and destined for precisely one destination node t.

Definition: (unicast) session
In the context of network coding, we define an unicast session as the tuple $\langle (s, t), (t, s) \rangle$ of two (unicast) flows in opposite directions.

Above definitions naturally extend to (single source) multicasts.

Intra-session vs. inter-session coding

Intra-session coding:

- Only packets belonging to the same session may be coded together.
- Flows belonging to the same session may be coded together provided that bidirectional coding is allowed.
- Easier to implement, but fewer coding opportunities and thus potentially lower coding gain.

Inter-session coding:

- Packets of arbitrary flows / sessions may be combined.
- More coding opportunities but more also more complex.

We will mainly focus on intra-session coding (particular in projects).

Connection to forward error correction (FEC)

What is FEC?

- FEC systematically adds redundancy to a transmission (at the symbol/packet/file/block-level depending on layer and application).
- It allows to detect and correct certain classes of transmission errors.
- We assume for now erasures, i. e., we want to correct lost (missing) symbols or blocks.

The code rate R is a measure of how much redundancy is added. Assuming an FEC code that maps k source symbols to $n \geq k$ output symbols, the code rate is defined as $R = k/n$.

We can differentiate between fixed-rate and rateless codes.

- Fixed-rate codes have a predefined R that remains constant. (Hamming codes, Reed-Solomon codes, Bose-Chaudhuri-Hocquenghem (BCH) codes)
- Rateless codes (fountain codes) are, in principle, able to output an endless sequence of coded symbols. R is not fixed but may vary over time for the very same code. (Luby transform (LT) codes, Raptor codes)

Example: Luby transform codes

The transmitter divides a packet into N blocks of equal size and then encodes blocks as follows:

1. Randomly choose $1 \leq d \leq N$ blocks, where d is the degree of the encoded block.
2. XOR exactly these d blocks.
3. Transmit the encoded block along with its degree, the list of indices (positions of blocks used for encoding), and a checksum.

It is obvious that the receiver can decode after receiving a sufficient number of encoded blocks.

Example: Luby transform codes

The transmitter divides a packet into N blocks of equal size and then encodes blocks as follows:

1. Randomly choose $1 \leq d \leq N$ blocks, where d is the degree of the encoded block.
2. XOR exactly these d blocks.
3. Transmit the encoded block along with its degree, the list of indices (positions of blocks used for encoding), and a checksum.

It is obvious that the receiver can decode after receiving a sufficient number of encoded blocks.

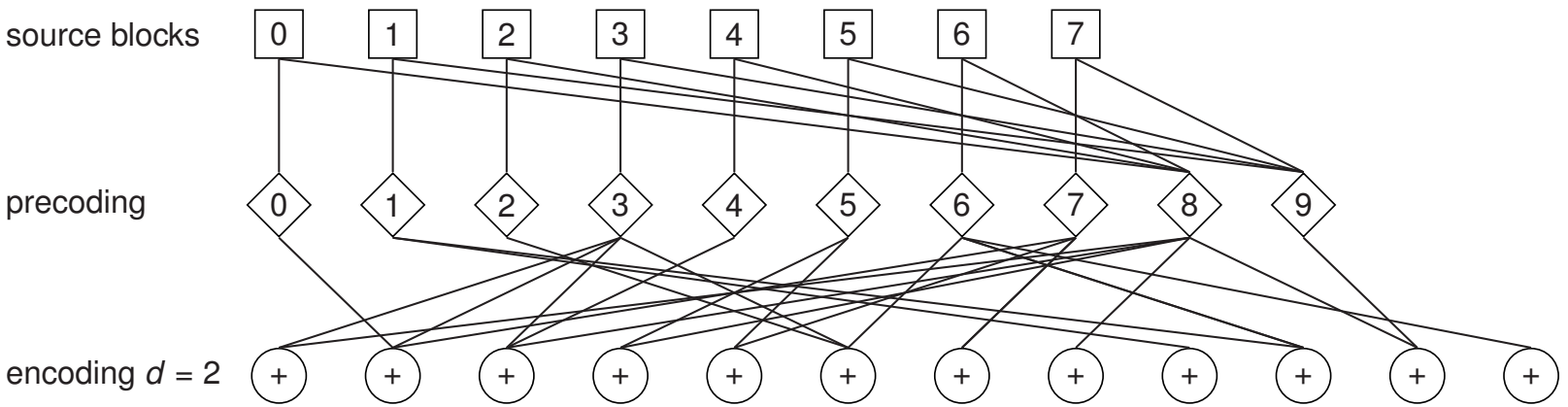
What's the difference to network coding?

- Coding operations are performed at the source node only. Alternatively, intermediate nodes may decode blocks and encode from scratch. With NC, intermediate nodes may re-encode packets without decoding.

Example: Raptor codes

Work similar to the LT codes but

1. use degree $d \ll N$ for combining blocks (inner code) and
2. employ a precoding stage (outer code) to recover blocks that are not transmitted due to sparsity.



Raptor codes are thus

- a concatenation of two codes (inner and outer), where
- the inner code is some kind of LT code.

Raptor codes vs. random linear network codes

- Encoding/Decoding
  - Raptor codes have a particular sparsity structure that enables fast encoding and decoding operations.
  - Network codes have no particular structure: decoding via Gauss elimination.
- Recoding at intermediate nodes
  - Raptor structure is lost when partially recoded at intermediate nodes. The structure could only be conserved if they knew all source data.
  - Network codes can be arbitrarily recoded from any set of coded blocks.

MORE

MORE – a MAC-independent opportunistic routing protocol

- Originally proposed by Chachulski [1, 2] in 2007
- Unidirectional intra-session network coding as discussed in Section 4
- Coding done between layer 2 and 3
- Opportunistic routing

MORE

MORE – a MAC-independent opportunistic routing protocol

- Originally proposed by Chachulski [1, 2] in 2007
- Unidirectional intra-session network coding as discussed in Section 4
- Coding done between layer 2 and 3
- Opportunistic routing

MORE framing:

MORE

MORE – a MAC-independent opportunistic routing protocol

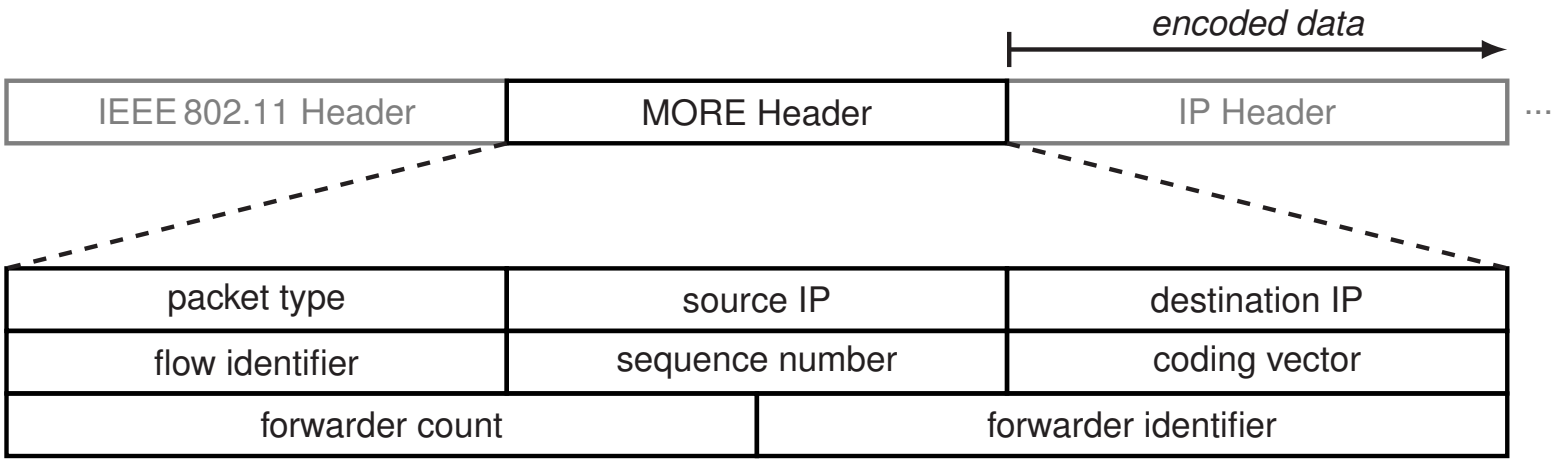
- Originally proposed by Chachulski [1, 2] in 2007
- Unidirectional intra-session network coding as discussed in Section 4
- Coding done between layer 2 and 3
- Opportunistic routing

MORE framing:

To achieve opportunistic routing, MORE has

- to send packets as MAC-layer broadcasts or
- use IEEE 802.11 MAC in promiscuous or monitor mode.

MORE header

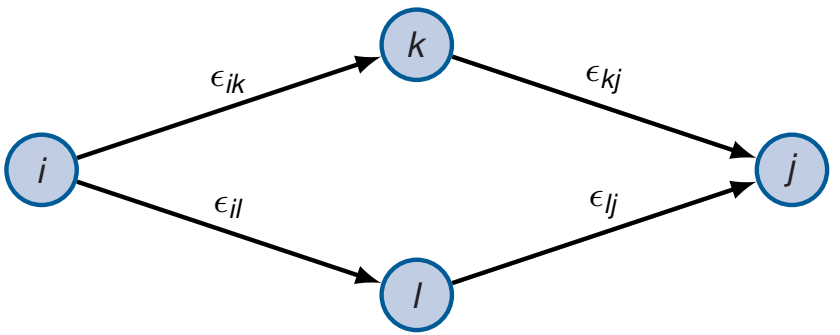


- packet type differentiates between data packets and acknowledgements
- flow identifier is used to identify the flow coded packets belong to
- sequence number is identifies a specific generation of packets within a flow
- coding vector is the vector $\boldsymbol{c} \in F_q^N[x]$ of random coefficients
- forwarder fields are used for routing $\leftarrow$ more on that later

Routing metric used by MORE

MORE uses the ETX (estimated transmission count) [3] metric:

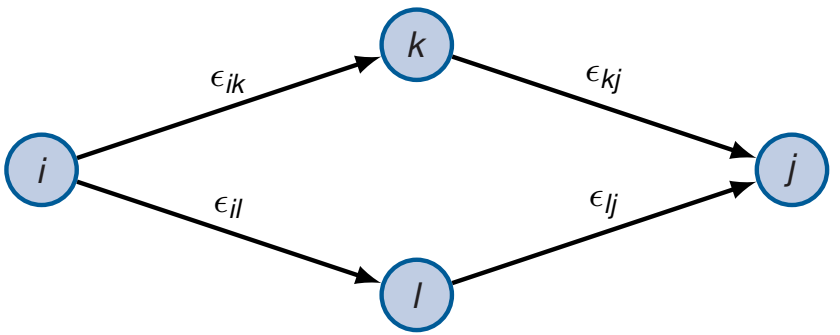
- The distance d_{ij} between two nodes i, j is the estimated number of transmissions along the best path between i, j.
- Consider the network below, assuming that there is no link between k and l:



Routing metric used by MORE

MORE uses the ETX (estimated transmission count) [3] metric:

- The distance d_{ij} between two nodes i, j is the estimated number of transmissions along the best path between i, j.
- Consider the network below, assuming that there is no link between k and l:



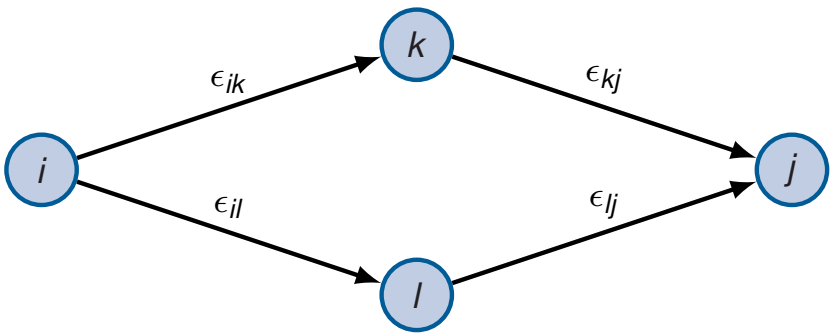
- Then the ETX distance d_{ij} in this example is

$$d_{ij} = \min \left\{ \frac{1}{(1-\epsilon_{ik})(1-\epsilon_{kj})}, \frac{1}{(1-\epsilon_{il})(1-\epsilon_{lj})}, \right\}$$

Routing metric used by MORE

MORE uses the ETX (estimated transmission count) [3] metric:

- The distance d_{ij} between two nodes i, j is the estimated number of transmissions along the best path between i, j.
- Consider the network below, assuming that there is no link between k and l:



- Then the ETX distance d_{ij} in this example is

$$d_{ij} = \min \left\{ \frac{1}{(1-\epsilon_{ik})(1-\epsilon_{kj})}, \frac{1}{(1-\epsilon_{il})(1-\epsilon_{lj})}, \right\}$$

Note: $d_{ik} = \frac{1}{1-\epsilon_{ik}}$ is the expectation of a random variable $X \sim \text{Geo}(1-\epsilon_{ik})$.

MORE uses opportunistic routing, i. e.,

- nodes overhear all transmissions and
- decide *somehow* whether or not to recode.

But *how*?

- Let z_i the number of transmissions by some $i \in \mathcal{N}$.
- Nodes estimate erasure probabilities by sending beacons.
- From this information the ETX distance between any $i \in \mathcal{N}$ and destination t can be computed.
- The expected number of packets that i receives from nodes with higher ETX distance is given as

$$R_j = \sum_{i>j} z_i \left(1 - \epsilon_{ij}\right).$$

Expected number of packets j has to forward for each packet sent by s:

- To avoid redundant transmissions node j should forward only if no other node k with lower distance to t has received a specific packet.
- The expected number of packets that node j has to send is therefore

$$L_j = \sum_{i>j} \left(z_i \left(1 - \epsilon_{ij}\right) \prod_{k<j} \epsilon_{ik} \right) .$$

- Note that $L_s = 1$.

Expected number of packets j has to forward for each packet sent by s:

- To avoid redundant transmissions node j should forward only if no other node k with lower distance to t has received a specific packet.
- The expected number of packets that node j has to send is therefore

$$L_j = \sum_{i>j} \left(z_i \left(1 - \epsilon_{ij}\right) \prod_{k<j} \epsilon_{ik} \right) .$$

- Note that $L_s = 1$.

Expected number of transmissions at node j:

- j has to transmit encoded packets until L_j packets have been received by nodes with lower ETX distance to t.
- That is given by

$$z_j = \frac{L_j}{1 - \prod_{k<j} \epsilon_{jk}} .$$

TX credit counter

- In order to keep track of how many packets nodes may send, each $i \in \mathcal{N}$ maintains a TX credit counter Z_i^{TX}.
- Z_i^{TX} is decremented by 1 for every packet that is transmitted.
- Node i stops transmitting if $Z_i^{\mathrm{TX}} < 0$ (note that $Z_i^{\mathrm{TX}} \in \mathbb{R}$).

TX credit counter

- In order to keep track of how many packets nodes may send, each $i \in \mathcal{N}$ maintains a TX credit counter Z_i^{TX}.
- Z_i^{TX} is decremented by 1 for every packet that is transmitted.
- Node i stops transmitting if $Z_i^{\mathrm{TX}} < 0$ (note that $Z_i^{\mathrm{TX}} \in \mathbb{R}$).

Problem:

- Each node has calculated its value z_i denoting the number of packets i has to send for each packet generated by s.
- However, i cannot know how many packets s generates.

TX credit counter

- In order to keep track of how many packets nodes may send, each $i \in \mathcal{N}$ maintains a TX credit counter Z_i^{TX}.
- Z_i^{TX} is decremented by 1 for every packet that is transmitted.
- Node i stops transmitting if $Z_i^{\mathrm{TX}} < 0$ (note that $Z_i^{\mathrm{TX}} \in \mathbb{R}$).

Problem:

- Each node has calculated its value z_i denoting the number of packets i has to send for each packet generated by s.
- However, i cannot know how many packets s generates.

Solution:

- In practice, Z_i^{TX} should be incremented whenever i receives a packet from some node with higher ETX distance.
- This incremental update is given by

$$\Delta Z_i^{\mathrm{TX}} = \frac{z_i}{R_i} = \frac{z_i}{\sum_{j>i} z_j \left(1 - \epsilon_{ji}\right)}.$$

Which nodes should act as forwarders?

With MORE, the subgraph that participates in forwarding packets from *s* to *t* is determine by *s*:

- Each packet sent by *s* contains a list of nodes that are closer to *t* than *s* ordered by their distance.
- Whenever an intermediate node overhears a coded packet, it first checks this forwarder list and discards the packet if it is not included in the list of possible forwarders.

The use of such a forwarder list

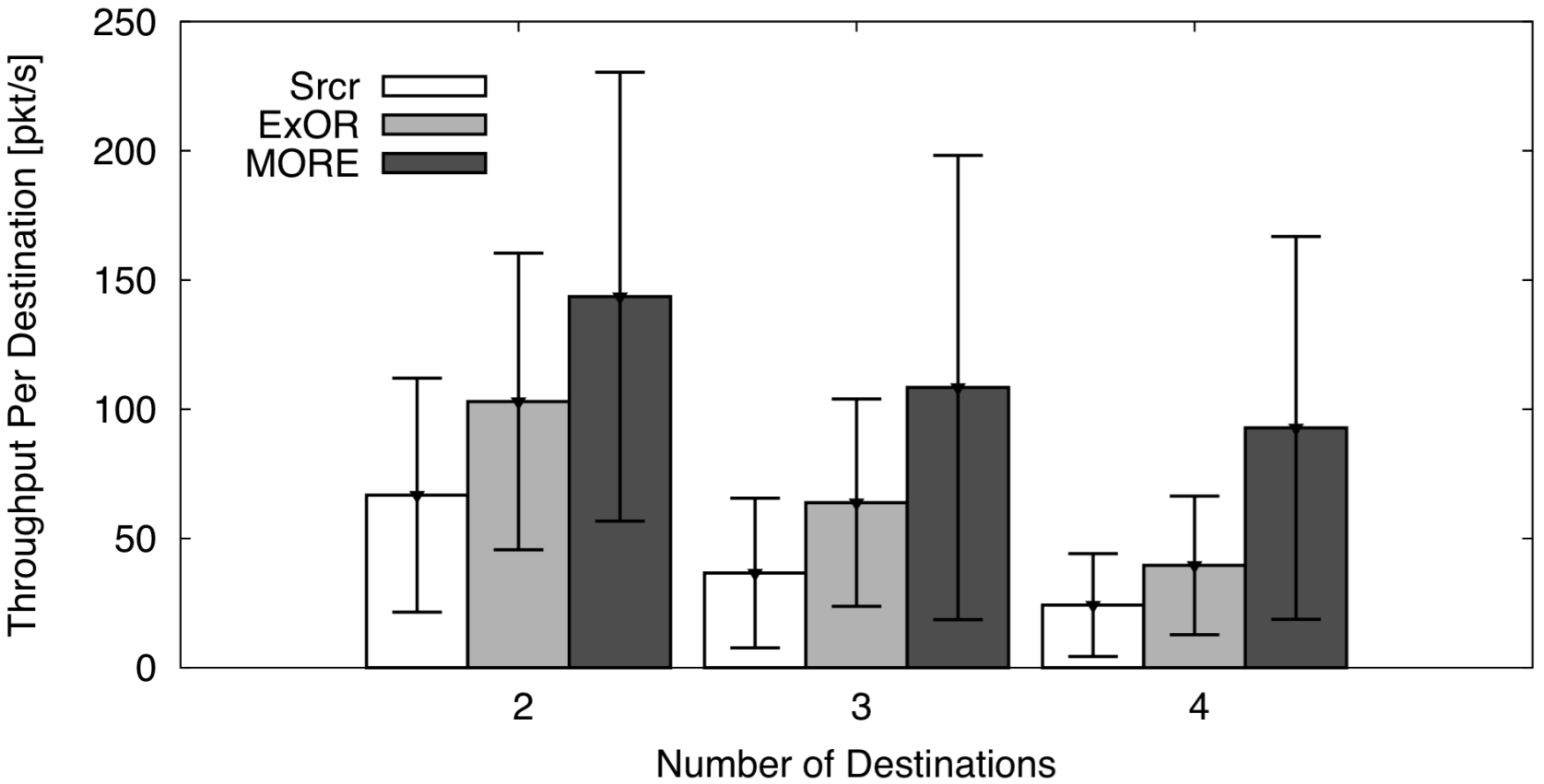
- resembles a variant of source routing and
- allows to dynamically setup routes instead of calculating all possible routes in advance,
- which limits the overhead induced by routing updates.

Acknowledgements

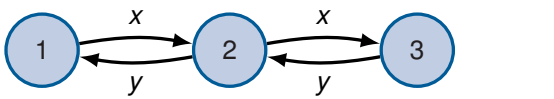
The source *s* keeps transmitting redundant frames until successful decoding is acknowledged by the destination *t*:

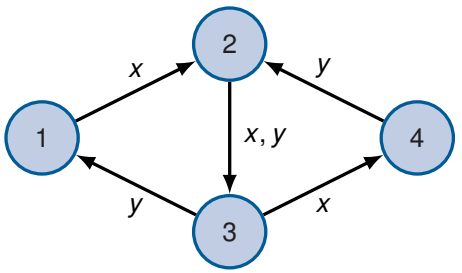
- *t* sends an uncoded acknowledgement back to *s* containing the generation sequence number.
- The acknowledgement is routed along the shortest path from *t* to *s* according to the ETX metric.
- For the acknowledgement, link layer retransmits provided by IEEE 802.11 are used.
- Forwarders overhearing the acknowledgement prune the corresponding generation.

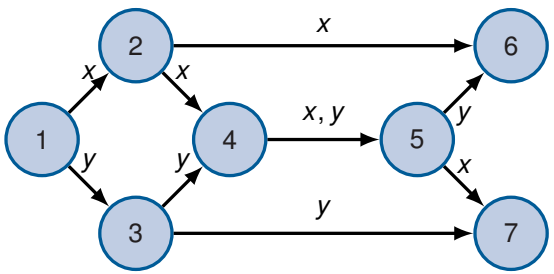
Throughput gain of MORE [2]



Networks as Graphs





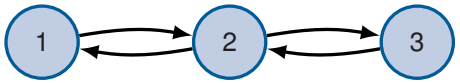


- Wired networks can be modeled as abstract graphs.
- Information flow in networks with routing & forwarding can be modeled as (multi-)commodity flow problem
- Gives nice problems (flow optimization problems) and algorithms (Dijkstra, Bellman-Ford, etc.)
- Special properties of “Information” (arbitrarily reproducible, coded representation, etc.) are not taken into account in the standard commodity model.

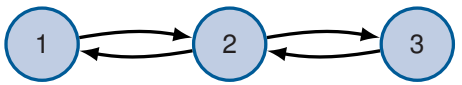
Graphs (directed) $G = (N, A)$

- Nodes $N = \{1, \ldots, n\}$
- Arcs $A = \{1, \ldots, m\}$
- Each arc $j \in A$ represents an ordered pair of nodes (a, b) with $a, b \in N, a \neq b$
- head$(j) = b$ and tail$(j) = a$
- Important structures:
  - Path (directed, undirected)
  - Tree (directed, undirected)
  - Cycle (directed, undirected)
- We assume G is connected, i.e., there exists an undirected path between any pair of nodes.

Examples



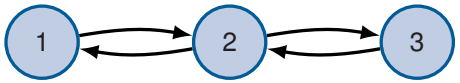
Examples

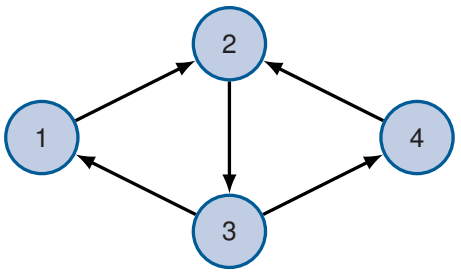


nodes $1, 2, 3$

arcs $(1, 2), (2, 1), (2, 3), (3, 2)$

Examples

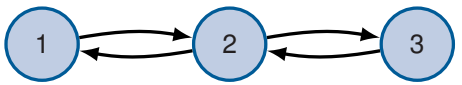




nodes $1, 2, 3$

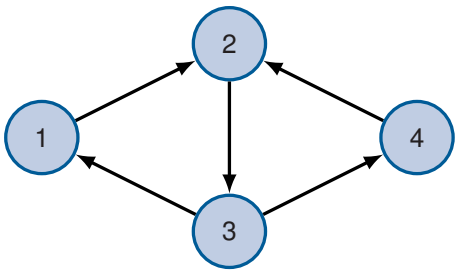
arcs $(1, 2), (2, 1), (2, 3), (3, 2)$

Examples



nodes $1, 2, 3$

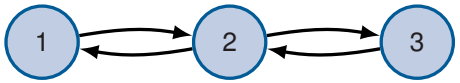
arcs $(1, 2), (2, 1), (2, 3), (3, 2)$



nodes $1, 2, 3, 4$

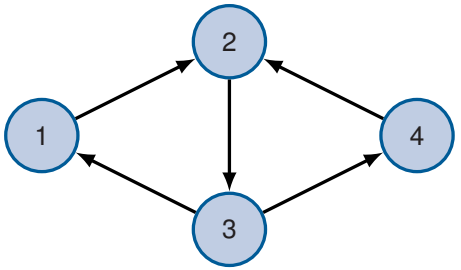
arcs $(1, 2), (2, 3), (3, 1), (3, 4), (4, 2)$

Examples



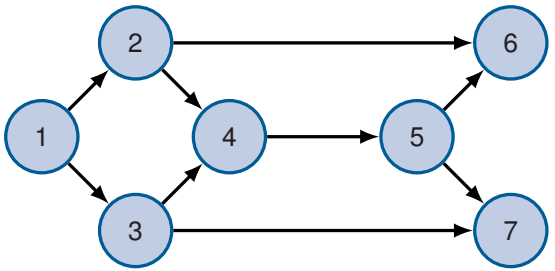
nodes $1, 2, 3$

arcs $(1, 2), (2, 1), (2, 3), (3, 2)$

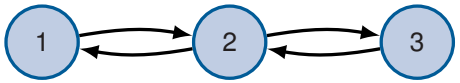


nodes $1, 2, 3, 4$

arcs $(1, 2), (2, 3), (3, 1), (3, 4), (4, 2)$

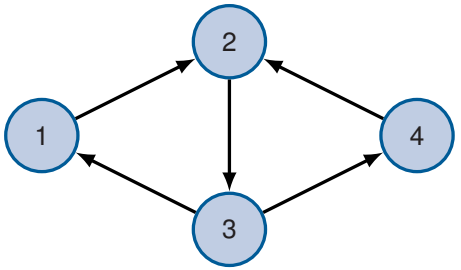


Examples



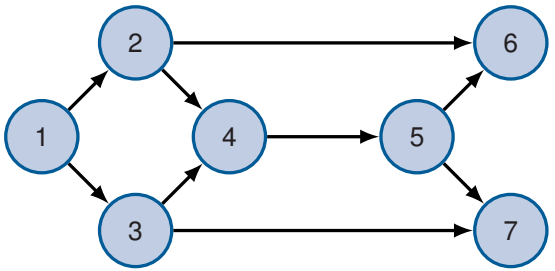
nodes 1, 2, 3

arcs (1, 2), (2, 1), (2, 3), (3, 2)



nodes 1, 2, 3, 4

arcs (1, 2), (2, 3), (3, 1), (3, 4), (4, 2)



nodes 1, 2, 3, 4, 5, 6, 7

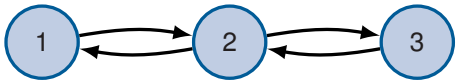
arcs (1, 2), (1, 3), (2, 4), (2, 6),
(3, 4), (3, 7), (4, 5), (5, 6), (5, 7)

Incidence matrix $\boldsymbol{M}$

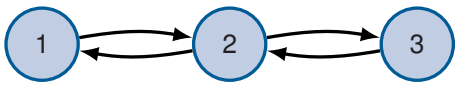
- $\boldsymbol{M}_{ij} = 1$ if tail(j) = i (arc j leaves node i)
- $\boldsymbol{M}_{ij} = -1$ if head(j) = i (arc j enters node i)

[4] Proof via undirected tree in G, adding any further arc creates a cycle
[5] Number of linearly independent undirected cycles

Examples

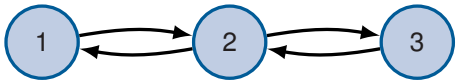


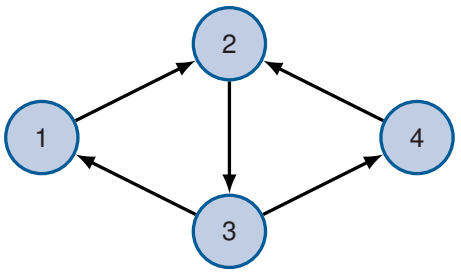
Examples



$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

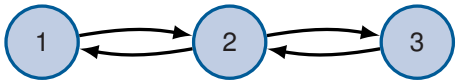
Examples



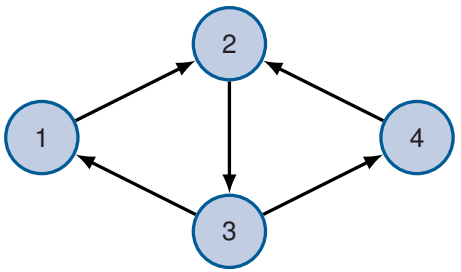


$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Examples

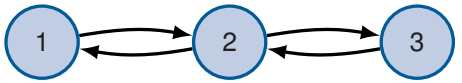


$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

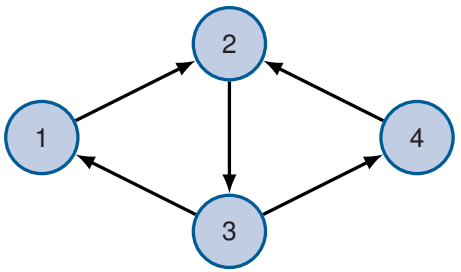


$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

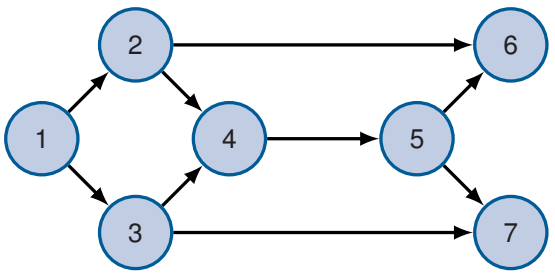
Examples



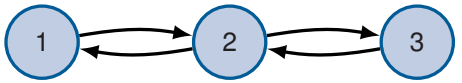
$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$



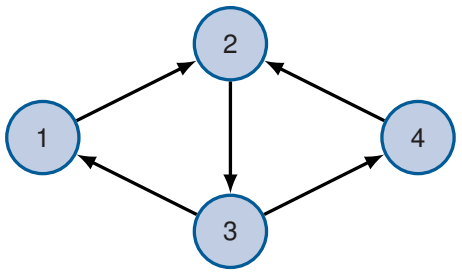
$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$



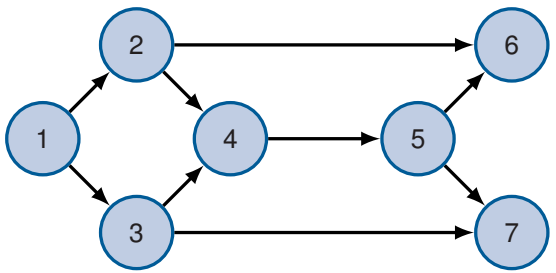
Examples



$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$



$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$



$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

Incidence matrix $\boldsymbol{M}$

- $\boldsymbol{M}_{ij} = 1$ if $\text{tail}(j) = i$ (arc j leaves node i)
- $\boldsymbol{M}_{ij} = -1$ if $\text{head}(j) = i$ (arc j enters node i)

Fundamental subspaces of $\boldsymbol{M}$ (G is connected)

$$\text{null}\, \boldsymbol{M}^{\mathsf{T}} = \text{span}\{\mathbf{1}\}$$

$$\text{null}\, \boldsymbol{M} = \text{span}\{\boldsymbol{x} : x_i = 1\ i \in C \text{ forward},\ x_i = -1\ i \in C \text{ backward},\ x_i = 0 \text{ else},\ C \text{ undirected cycle of } G\}$$

[4] Proof via undirected tree in G, adding any further arc creates a cycle
[5] Number of linearly independent undirected cycles

Incidence matrix $\boldsymbol{M}$

- $\boldsymbol{M}_{ij} = 1$ if tail$(j) = i$ (arc j leaves node i)
- $\boldsymbol{M}_{ij} = -1$ if head$(j) = i$ (arc j enters node i)

Fundamental subspaces of $\boldsymbol{M}$ (G is connected)

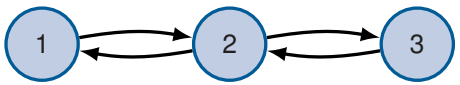
$$\operatorname{null} \boldsymbol{M}^{\mathsf{T}} = \operatorname{span}\{\mathbf{1}\}$$

$$\operatorname{null} \boldsymbol{M} = \operatorname{span}\{\boldsymbol{x} : x_i = 1 \; i \in C \text{ forward}, \; x_i = -1 \; i \in C \text{ backward}, \; x_i = 0 \text{ else}, \; C \text{ undirected cycle of } G\}$$

- $\operatorname{rank} \boldsymbol{M} = n - 1$ [4]
- $\dim \operatorname{null} \boldsymbol{M}^{\mathsf{T}} = 1$
- $\dim \operatorname{null} \boldsymbol{M} = m - n + 1$ [5]

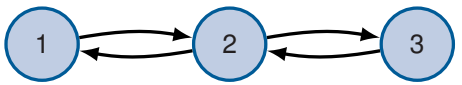
[4] Proof via undirected tree in G, adding any further arc creates a cycle
[5] Number of linearly independent undirected cycles

Examples



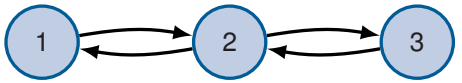
$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Examples

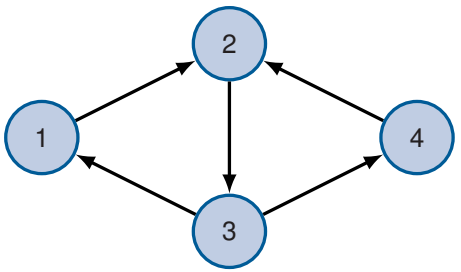


$$\text{null } \boldsymbol{M} = \text{span} \left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right\}$$

Examples

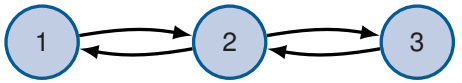


$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

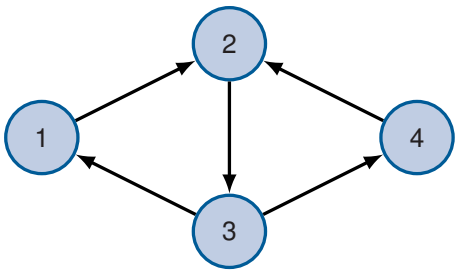


$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Examples

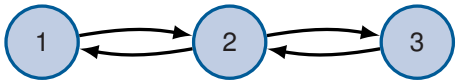


$$\text{null}\,\boldsymbol{M} = \text{span}\left\{\begin{bmatrix}1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\1\end{bmatrix}\right\}$$

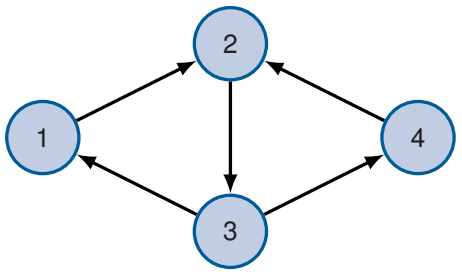


$$\text{null}\,\boldsymbol{M} = \text{span}\left\{\begin{bmatrix}1\\1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\1\\1\end{bmatrix}\right\}$$

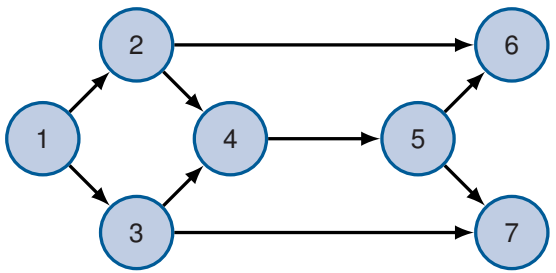
Examples



$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

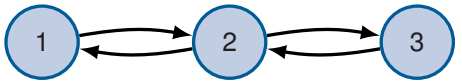


$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

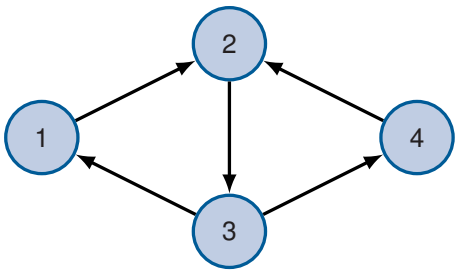


$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

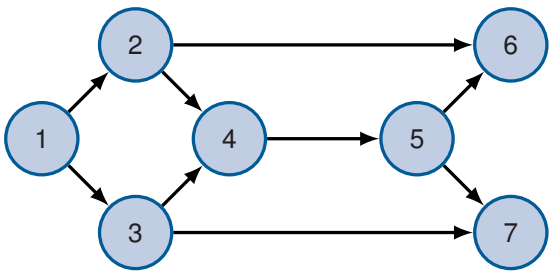
Examples



$$\text{null } \boldsymbol{M} = \text{span} \left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right\}$$



$$\text{null } \boldsymbol{M} = \text{span} \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right\}$$



$$\text{null } \boldsymbol{M} = \text{span} \left\{ \begin{bmatrix} 1 \\ -1 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ -1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Flows[6]

- The flow vector $\boldsymbol{x} = [x_1, \ldots, x_m]^T$ represents the amount of commodity (information) flow on each arc.
- The source vector $\boldsymbol{d} = [d_1, \ldots, d_n]^T$ represents the amount of commodity (information) that any node injects or consumes.
- Multiple information flows can be handled as a single commodity for routing/forwarding if they are
  - destined for a single common destination or
  - originate from a single common source.

[6] Single commodity flows

Flows

- Nonnegativity of flows

$$\boldsymbol{x} \geq \mathbf{0} \qquad \Leftrightarrow \qquad x_j \geq 0 \quad \forall j \in A$$

Flows

- Nonnegativity of flows

$$\boldsymbol{x} \geq \mathbf{0} \qquad \Leftrightarrow \qquad x_j \geq 0 \quad \forall j \in A$$

- Flow conservation law (Kirchhoff current law):

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \qquad \Leftrightarrow \qquad \sum_{j \in A:\text{tail}(j)=i} x_j - \sum_{j \in A:\text{head}(j)=i} x_j = d_i \quad \forall i \in N$$

Flows

- Nonnegativity of flows

$$\boldsymbol{x} \geq \mathbf{0} \qquad \Leftrightarrow \qquad x_j \geq 0 \quad \forall j \in A$$

- Flow conservation law (Kirchhoff current law):

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \qquad \Leftrightarrow \qquad \sum_{j \in A:\text{tail}(j)=i} x_j - \sum_{j \in A:\text{head}(j)=i} x_j = d_i \quad \forall i \in N$$

  - FCL can not be satisfied if $\mathbf{1}^{\mathsf{T}}\boldsymbol{d} \neq 0$ since $\mathbf{1}^{\mathsf{T}}\boldsymbol{M} = \mathbf{0}$.

Flows

- Nonnegativity of flows

$$\boldsymbol{x} \geq \mathbf{0} \qquad \Leftrightarrow \qquad x_j \geq 0 \quad \forall j \in A$$

- Flow conservation law (Kirchhoff current law):

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \qquad \Leftrightarrow \qquad \sum_{j \in A:\text{tail}(j)=i} x_j - \sum_{j \in A:\text{head}(j)=i} x_j = d_i \quad \forall i \in N$$

  - FCL can not be satisfied if $\mathbf{1}^\mathsf{T}\boldsymbol{d} \neq 0$ since $\mathbf{1}^\mathsf{T}\boldsymbol{M} = \mathbf{0}$.
  - FCL contains exactly one redundant constraint since rank $\boldsymbol{M} = n - 1$ (if graph is connected).

Flows

- Nonnegativity of flows

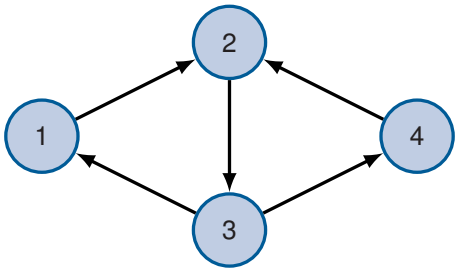
$$\boldsymbol{x} \geq \mathbf{0} \qquad \Leftrightarrow \qquad x_j \geq 0 \quad \forall j \in A$$

- Flow conservation law (Kirchhoff current law):

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \qquad \Leftrightarrow \qquad \sum_{j \in A:\text{tail}(j)=i} x_j - \sum_{j \in A:\text{head}(j)=i} x_j = d_i \quad \forall i \in N$$

  - FCL can not be satisfied if $\mathbf{1}^\mathsf{T}\boldsymbol{d} \neq 0$ since $\mathbf{1}^\mathsf{T}\boldsymbol{M} = \mathbf{0}$.
  - FCL contains exactly one redundant constraint since rank $\boldsymbol{M} = n - 1$ (if graph is connected).
  - Flows along directed cycles are independent of $\boldsymbol{d}$, i.e., flows that satisfy $\boldsymbol{M}\boldsymbol{x} = \mathbf{0}$, $\boldsymbol{x} \geq \mathbf{0}$.

Example 1: Diamond network from $s = 1$ to $t = 4$



- Incidence matrix and source vector

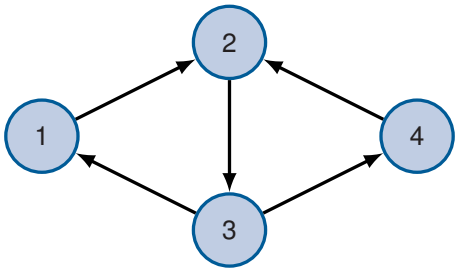
$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

- Feasible flows for $\boldsymbol{M}, \boldsymbol{d}$

$$\mathcal{F}(\boldsymbol{M}, \boldsymbol{d}) = \{\boldsymbol{x} : \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d}, \boldsymbol{x} \geq 0\}$$

- Flow solution(s) (Unique? How many solutions?)

Example 1: Diamond network from $s = 1$ to $t = 4$



- Incidence matrix and source vector

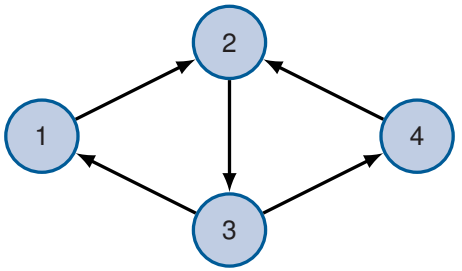
$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

- Feasible flows for $\boldsymbol{M}, \boldsymbol{d}$

$$\mathcal{F}(\boldsymbol{M}, \boldsymbol{d}) = \{\boldsymbol{x} : \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d}, \boldsymbol{x} \geq 0\}$$

- Flow solution(s) (Unique? How many solutions?)
  - $\boldsymbol{x}^{\mathsf{T}} = [1\ 1\ 0\ 1\ 0]$

Example 1: Diamond network from $s = 1$ to $t = 4$



- Incidence matrix and source vector

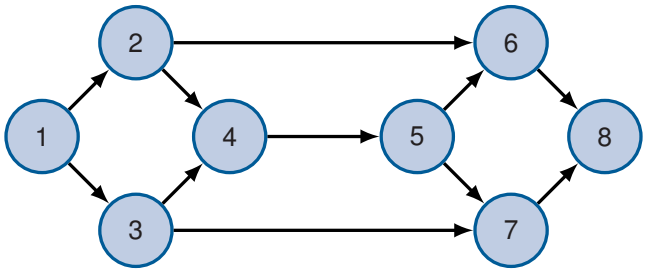
$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

- Feasible flows for $\boldsymbol{M}, \boldsymbol{d}$

$$\mathcal{F}(\boldsymbol{M}, \boldsymbol{d}) = \{\boldsymbol{x} : \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d}, \boldsymbol{x} \geq 0\}$$

- Flow solution(s) (Unique? How many solutions?)
  - $\boldsymbol{x}^{\mathsf{T}} = [1\ 1\ 0\ 1\ 0]$
  - $\boldsymbol{x}^{\mathsf{T}} = [1\ 1\ 0\ 1\ 0] + \alpha\,[1\ 1\ 1\ 0\ 0] + \beta\,[0\ 1\ 0\ 1\ 1],\ \alpha, \beta \geq 0$

Example 2: Extended butterfly from $s = 1$ to $t = 8$

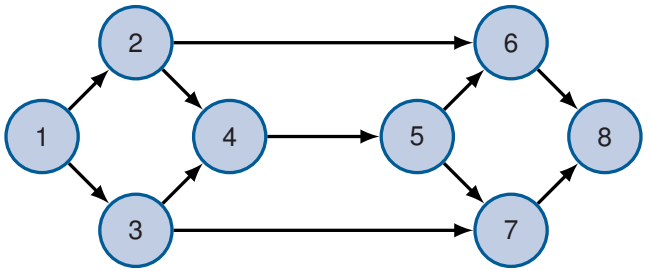


- Incidence matrix and source vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

- Flow solution(s) (Unique? How many?)

Example 2: Extended butterfly from $s = 1$ to $t = 8$

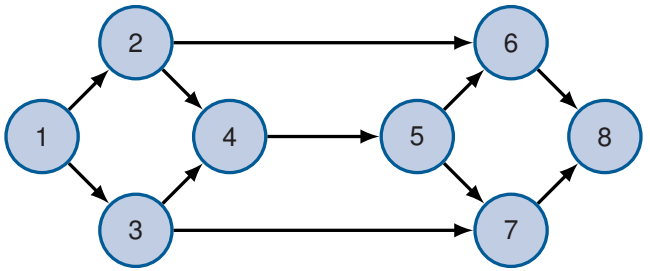


- Incidence matrix and source vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

- Flow solution(s) (Unique? How many?)
  - $\boldsymbol{x} = [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$

Example 2: Extended butterfly from $s = 1$ to $t = 8$

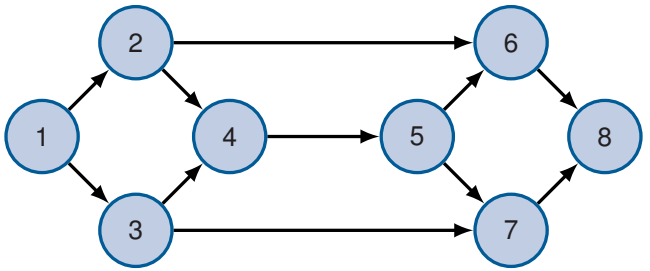


- Incidence matrix and source vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

- Flow solution(s) (Unique? How many?)
  - $\boldsymbol{x} = [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$
  - $\boldsymbol{x} = [0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1]$

Example 2: Extended butterfly from $s = 1$ to $t = 8$

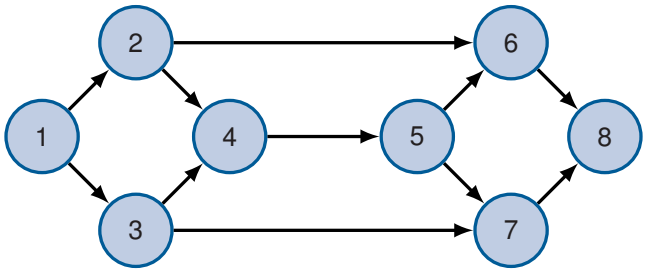


- Incidence matrix and source vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

- Flow solution(s) (Unique? How many?)
  - $\boldsymbol{x} = [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$
  - $\boldsymbol{x} = [0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1]$
  - $\boldsymbol{x} = \lambda [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0] + (1-\lambda)[0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1],\ \lambda \in [0,1]$

Example 2: Extended butterfly from $s = 1$ to $t = 8$

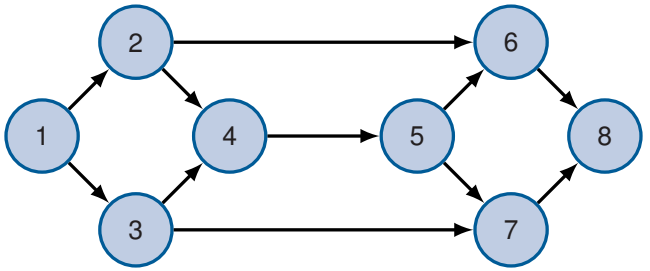


- Incidence matrix and source vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

- Flow solution(s) (Unique? How many?)
  - $\boldsymbol{x} = [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$
  - $\boldsymbol{x} = [0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1]$
  - $\boldsymbol{x} = \lambda [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0] + (1-\lambda)[0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1],\ \lambda \in [0,1]$
  - ...

Example 3: Flows from multiple sources to a single destination

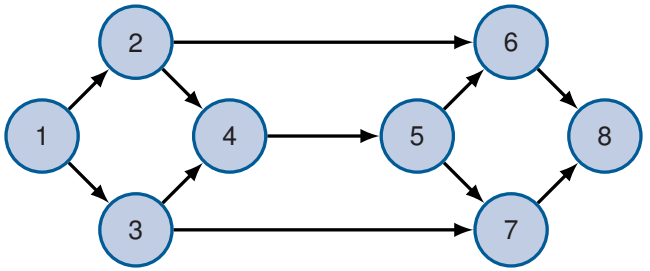


- Incidence matrix and source vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -4 \end{bmatrix}$$

- Flow solution(s) (Unique? How many?)

Example 3: Flows from multiple sources to a single destination

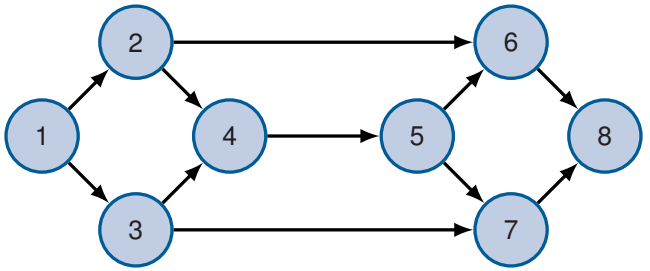


- Incidence matrix and source vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -4 \end{bmatrix}$$

- Flow solution(s) (Unique? How many?)
  - $\boldsymbol{x} = [0\ 1\ 0\ 1\ 1\ 1\ 2\ 1\ 1\ 2\ 2]$

Example 3: Flows from multiple sources to a single destination



- Incidence matrix and source vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -4 \end{bmatrix}$$

- Flow solution(s) (Unique? How many?)
  - $\boldsymbol{x} = [0\ 1\ 0\ 1\ 1\ 1\ 2\ 1\ 1\ 2\ 2]$
  - ...

Feasible flow region

$$\mathcal{F}(\boldsymbol{M}, \boldsymbol{d}) = \{\boldsymbol{x} : \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d}, \boldsymbol{x} \geq 0\}$$

- $\mathcal{F}(\boldsymbol{M}, \boldsymbol{d})$ is a closed[7] polyhedral[8] convex[9] set.
- $\mathcal{F}(\boldsymbol{M}, \boldsymbol{d})$ is nonempty $\mathbf{1}^{\mathsf{T}}\boldsymbol{d} = 0$ (if G is connected).
- $\mathcal{F}(\boldsymbol{M}, \boldsymbol{d})$ is bounded[10] if G is acyclic (contains no directed cycles), i.e., $\mathcal{F}(\boldsymbol{M}, \mathbf{0}) = \{\mathbf{0}\}$.
- In general, $\mathcal{F}(\boldsymbol{M}, \boldsymbol{d})$ contains infinitely many solutions. Which one is the best?

[7] A set $\mathcal{X}$ is closed if it contains all its limit points.
[8] A set $\mathcal{X}$ is a polyhedron if it is defined by a finite number of affine (in)equalities, i.e., $\mathcal{X} = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$.
[9] A set $\mathcal{X}$ is convex if for any two points $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ and any real scalar $\lambda \in [0, 1]$, $\lambda\boldsymbol{x} + (1 - \lambda)\boldsymbol{y} \in \mathcal{X}$.
[10] A set $\mathcal{X}$ is bounded if it is contained in some ball around the origin, i.e., $\mathcal{X} \subset B_r(\mathbf{0})$ for some $r > 0$.

Minimum cost flow problem

- Cost per unit flow on arcs: $\boldsymbol{c} = [c_1, \ldots, c_m]^T$

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

[11] Not all flow solutions to these two problems describe shortest paths, but at least one does.

Minimum cost flow problem

- Cost per unit flow on arcs: $\boldsymbol{c} = [c_1, \ldots, c_m]^\mathsf{T}$

$$\begin{aligned} \min \quad & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Example: Shortest path[11]

- $\boldsymbol{c}$ "length" of each arc, e.g., $\boldsymbol{c} = \boldsymbol{1}$ (number of hops metric)
- Shortest path from s to t: $d_s = 1$, $d_t = -1$, $d_i = 0 \; \forall \; i \neq s, t$
- Simultaneous shortest paths to t: $d_t = -n + 1$, $d_i = 1 \; \forall \; i \neq t$

[11] Not all flow solutions to these two problems describe shortest paths, but at least one does.

Minimum cost flow problem

- Cost per unit flow on arcs: $\boldsymbol{c} = [c_1, \ldots, c_m]^T$

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Minimum cost flow problem

- Cost per unit flow on arcs: $\boldsymbol{c} = [c_1, \ldots, c_m]^T$

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Solution approaches

- General purpose linear programming solver (Simplex, Interior point, etc.)
- Specialized algorithms (Dijkstra, Bellman-Ford, network simplex, etc.) exploiting graph structure and recursive structure of the optimal solution (if available)

Maximum s-t flow problem

- Source vector: $d_s = 1$, $d_t = -1$, $d_i = 0 \; \forall \; i \neq s, t$
- Capacity vector (maximum flow on arcs): $\boldsymbol{z} = [z_1, \ldots, z_m]^{\mathsf{T}}$

$$\begin{aligned} \max \quad & r \\ \text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d} \\ & \boldsymbol{x} \leq \boldsymbol{z} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Maximum s-t flow problem

- Source vector: $d_s = 1$, $d_t = -1$, $d_i = 0 \; \forall \; i \neq s, t$
- Capacity vector (maximum flow on arcs): $\boldsymbol{z} = [z_1, \ldots, z_m]^{\mathsf{T}}$

$$\begin{aligned} \max \quad & r \\ \text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d} \\ & \boldsymbol{x} \leq \boldsymbol{z} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Solution approaches

- General purpose linear programming solver (Simplex, Interior point, etc.)
- Lagrangian duality approaches (selectively relax one constraint)
- Specialized algorithms (Ford-Fulkerson) exploiting graph structure and relation to min-cut

Max-flow min-cut theorem

- An s-t cut is a subset of nodes $S \subset N$ such that $s \in S$ and $t \notin S$.
- An arc $j \in A$ crosses S if tail(j) $\in S$ and head(j) $\notin S$. $A(S)$ denotes all crossing arcs.
- The value of an s-t cut given the capacity vector $\boldsymbol{z}$ is defined as

$$v(S) = \sum_{j \in A(S)} z_j$$

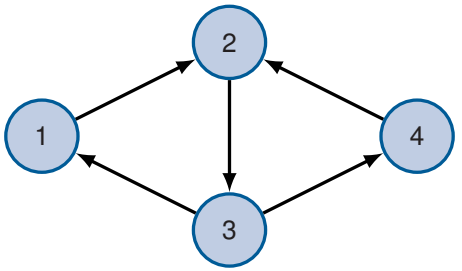
- The value of any s-t cut upper bounds the maximum s-t flow.

Max-flow min-cut theorem

The value of the minimum s-t cut equals the value of the maximum s-t flow, i.e.,

$$\max\{r : \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d},\ \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{z}\} = \min\{v(S) : S \text{ is } s\text{-}t \text{ cut}\}$$

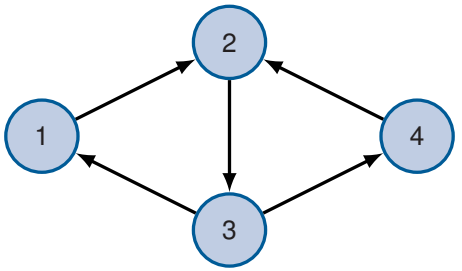
Example 1: Diamond network from $s = 1$ to $t = 4$



- Incidence matrix, source vector, capacity vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} \qquad \boldsymbol{z} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Example 1: Diamond network from $s = 1$ to $t = 4$



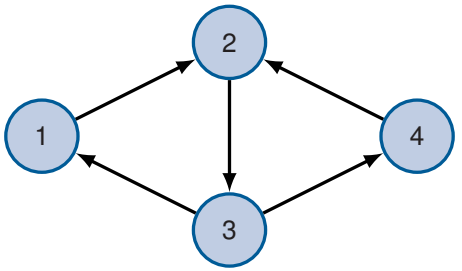
- Incidence matrix, source vector, capacity vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} \qquad \boldsymbol{z} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Max-flow

$$\max\{r : \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d},\ \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{z}\} = 1$$

Example 1: Diamond network from $s = 1$ to $t = 4$



- Incidence matrix, source vector, capacity vector

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} \qquad \boldsymbol{z} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Max-flow

$$\max\{r : \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d},\ \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{z}\} = 1$$

- Min-cut

$$\min\{v(S) : S \text{ is } s\text{-}t \text{ cut}\} = 1$$

Multicast in Networks as Flow Problems

- Multicast communication is identified by its terminal set $T \subset N$.
- We can consider one or multiple sources (there is no big difference form a theoretical perspective).
- Special cases:

Multicast in Networks as Flow Problems

- Multicast communication is identified by its terminal set $T \subset N$.
- We can consider one or multiple sources (there is no big difference form a theoretical perspective).
- Special cases:
  - Unicast (one source, one terminal)
  - Bidirectional communication (two nodes that are sources and terminals)
  - Broadcast (all nodes are terminals)

Multicast in Networks as Flow Problems

- Multicast communication is identified by its terminal set $T \subset N$.
- We can consider one or multiple sources (there is no big difference form a theoretical perspective).
- Special cases:
  - Unicast (one source, one terminal)
  - Bidirectional communication (two nodes that are sources and terminals)
  - Broadcast (all nodes are terminals)

How is multicast treated in networks?

- Convert to unicast (replicate packets at source and store-forward at all other nodes)
- Allow replication at all nodes (multicast tree/Steiner tree based forwarding)
- Allow coding at all nodes (network coding)

Preliminaries: Single commodity flow problems

- Source and flow vector: $\boldsymbol{d}$, $\boldsymbol{x}$
- Capacity and cost vector: $\boldsymbol{z}$, $\boldsymbol{c}$

$$\begin{aligned} \min \quad & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} \\ \text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{x} \leq \boldsymbol{z} \end{aligned}$$

Preliminaries: Single commodity flow problems

- Source and flow vector: $\boldsymbol{d}$, $\boldsymbol{x}$
- Capacity and cost vector: $\boldsymbol{z}$, $\boldsymbol{c}$

$$\begin{aligned} \min \quad & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = \boldsymbol{d} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{x} \leq \boldsymbol{z} \end{aligned}$$

Special cases

- Maximum *s*-*t* flow (see tutorial)
- Minimum cost flow (capacitated $\boldsymbol{z} < \infty$, uncapacitated $\boldsymbol{z} = \infty$)

Preliminaries: Multicommodity flow problems

- Commodities $C = \{1, \ldots, c\}$
- Source, flow, and cost vector of commodity k: $\boldsymbol{d}_k$, $\boldsymbol{x}_k$, $\boldsymbol{c}_k$
- Capacity shared across all commodities: $\boldsymbol{z}$

$$\min \sum_{k \in C} \boldsymbol{c}_k^T \boldsymbol{x}_k$$

$$\text{s. t.} \quad \boldsymbol{M}\boldsymbol{x}_k = \boldsymbol{d}_k \quad \forall k \in C$$

$$\boldsymbol{x}_t \geq \boldsymbol{0} \quad \forall k \in C$$

$$\sum_{k \in C} \boldsymbol{x}_k \leq \boldsymbol{z}$$

Preliminaries: Multicommodity flow problems

- Commodities $C = \{1, \ldots, c\}$
- Source, flow, and cost vector of commodity k: $\boldsymbol{d}_k$, $\boldsymbol{x}_k$, $\boldsymbol{c}_k$
- Capacity shared across all commodities: $\boldsymbol{z}$

$$\min \sum_{k \in C} \boldsymbol{c}_k^T \boldsymbol{x}_k$$

$$\text{s. t.} \quad \boldsymbol{M}\boldsymbol{x}_k = \boldsymbol{d}_k \quad \forall k \in C$$

$$\boldsymbol{x}_t \geq \boldsymbol{0} \quad \forall k \in C$$

$$\sum_{k \in C} \boldsymbol{x}_k \leq \boldsymbol{z}$$

Properties

- Flow conservation applies to all commodities individually
- Capacity is shared among all commodities

Preliminaries: Multicommodity flow problems

- Commodities $C = \{1, \ldots, c\}$
- Source, flow, and cost vector of commodity k: $\boldsymbol{d}_k$, $\boldsymbol{x}_k$, $\boldsymbol{c}_k$
- Capacity shared across all commodities: $\boldsymbol{z}$

$$\min \sum_{k \in C} \boldsymbol{c}_k^T \boldsymbol{x}_k$$

$$\text{s. t.} \quad \boldsymbol{M}\boldsymbol{x}_k = \boldsymbol{d}_k \quad \forall k \in C$$

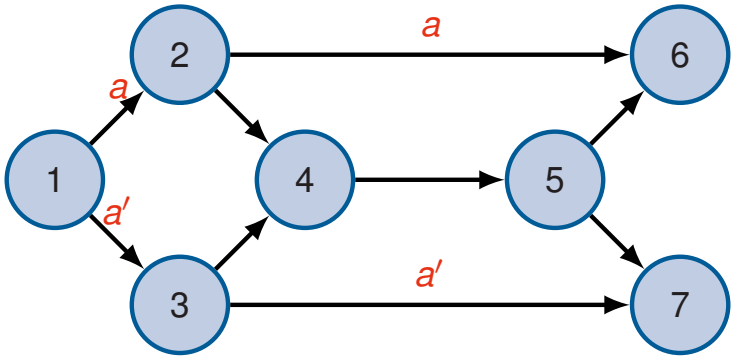
$$\boldsymbol{x}_t \geq \boldsymbol{0} \quad \forall k \in C$$

$$\sum_{k \in C} \boldsymbol{x}_k \leq \boldsymbol{z}$$

Solution approaches

- General purpose linear programming solver
- Lagrangian duality approaches (selectively relax one constraint, mostly the capacity constraint which couples all flows)

Store-forward multicast



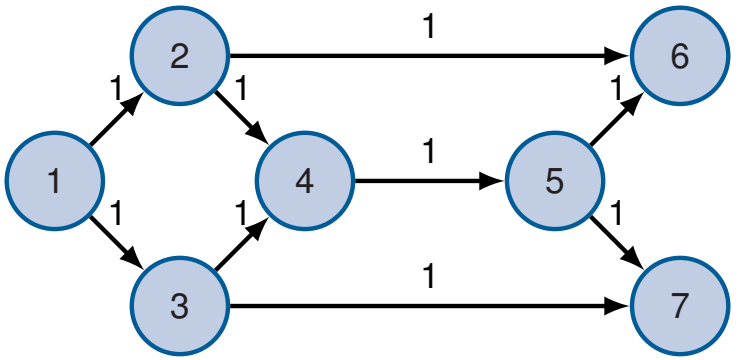
- The flows to the terminals are independent of each other.
- Capacity needs to be split among all flows.

Store-forward multicast: Max s–T flow problem

- One commodity for each terminal $t \in T$
- Source vector $\boldsymbol{d}_{st}$ such that $d_{st,s} = 1$, $d_{st,t} = -1$, and $d_{st,i} = 0$ otherwise
- Capacity vector $\boldsymbol{z}$ split among commodities

$$\begin{aligned} \max\ r \quad \text{s.t.} \quad \boldsymbol{M}\boldsymbol{x}_t &= r\boldsymbol{d}_{st} && \forall t \in T \\ \boldsymbol{x}_t &\geq \boldsymbol{0} && \forall t \in T \\ \sum_{t \in T} \boldsymbol{x}_t &\leq \boldsymbol{z} \end{aligned}$$

Store-forward multicast



- Optimal flow solutions
  - $\boldsymbol{x}_6 = [1\,0\,0\,1\,0\,0\,0\,0\,0]^T$
  - $\boldsymbol{x}_7 = [0\,1\,0\,0\,0\,1\,0\,0\,0]^T$
- Total flow which is capacity relevant
  - $\boldsymbol{x}_6 + \boldsymbol{x}_7 = [1\,1\,0\,1\,0\,1\,0\,0\,0]^T$

Maximum Multicast s–T Flow = 1

Preliminaries: Maximum *s*-*t* flow (flow formulation)

- Source vector $\boldsymbol{d}_{st}$ such that $d_s = 1$, $d_t = -1$, $d_i = 0 \; \forall \; i \neq s, t$
- Capacity vector (maximum flow on arcs): $\boldsymbol{z}$
- Flow vector $\boldsymbol{x}$
- Flow conservation $\boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d}_{st}$

$$
\begin{aligned}
\max \quad & r \\
\text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d}_{st} \\
& \boldsymbol{x} \leq \boldsymbol{z} \\
& \boldsymbol{x} \geq \boldsymbol{0}
\end{aligned}
$$

Preliminaries: Maximum *s*-*t* flow (path formulation)

- Paths $\mathrm{P}_{st} = \{1, \ldots, K\}$
- Path incidence vector $\boldsymbol{x}_k$ such that $x_{k,j} = 1$ if arc j is in the k-th path, otherwise $x_{k,j} = 0$
- All incidence vectors satisfy flow conservation with *s*-*t* source vector $\boldsymbol{d}$, i.e., $\boldsymbol{M}\boldsymbol{x}_k = \boldsymbol{d}_{st}$

$$\begin{aligned} \max \quad & \sum_{k \in \mathrm{P}_{st}} r_k \\ \text{s.t.} \quad & \sum_{k \in \mathrm{P}_{st}} r_k \boldsymbol{x}_k \leq \boldsymbol{z} \\ & r_k \geq 0 \quad \forall k \in \mathrm{P}_{st} \end{aligned}$$

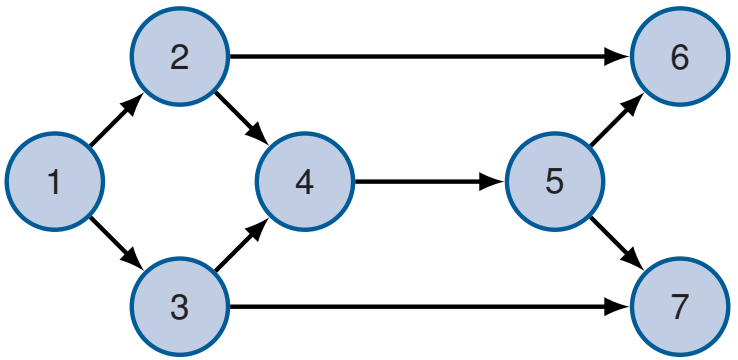
Preliminaries: Maximum *s*-*t* flow (path formulation)

- Paths $\mathrm{P}_{st} = \{1, \ldots, K\}$
- Path incidence vector $\boldsymbol{x}_k$ such that $x_{k,j} = 1$ if arc j is in the k-th path, otherwise $x_{k,j} = 0$
- All incidence vectors satisfy flow conservation with *s*-*t* source vector $\boldsymbol{d}$, i.e., $\boldsymbol{M}\boldsymbol{x}_k = \boldsymbol{d}_{st}$

$$\begin{aligned} \max \quad & \sum_{k \in \mathrm{P}_{st}} r_k \\ \text{s. t.} \quad & \sum_{k \in \mathrm{P}_{st}} r_k \boldsymbol{x}_k \leq \boldsymbol{z} \\ & r_k \geq 0 \quad \forall k \in \mathrm{P}_{st} \end{aligned}$$

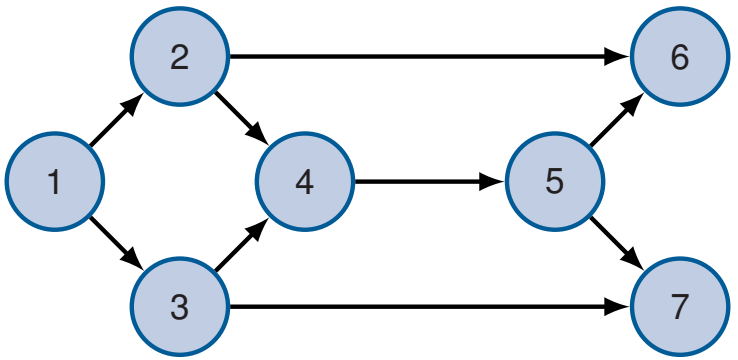
- Set of paths may be large.
- Path formulation based solution approaches generate only a few good paths during the problem solution.

Multicast tree based forwarding



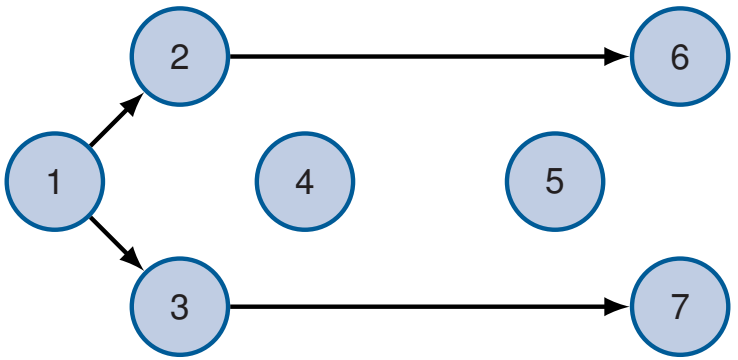
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).

Multicast tree based forwarding



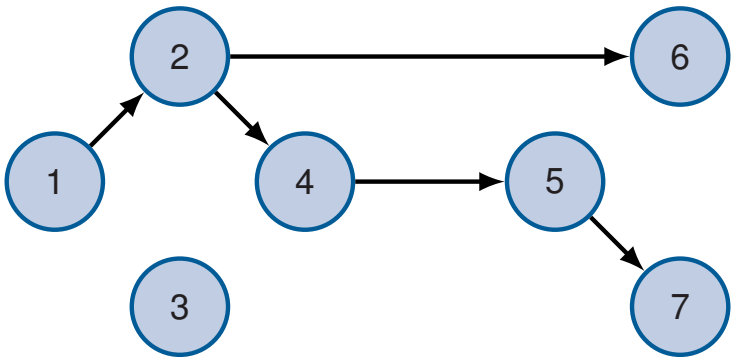
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Example 1–$\{6, 7\}$ multicast trees:

Multicast tree based forwarding



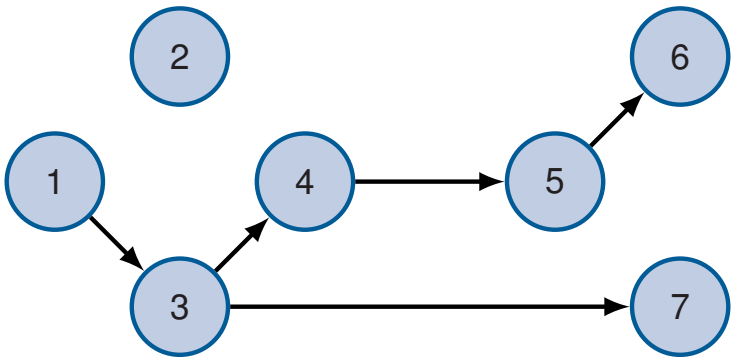
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Example 1–$\{6, 7\}$ multicast trees:
  - $(1, 2), (1, 3), (2, 6), (3, 7)$

Multicast tree based forwarding



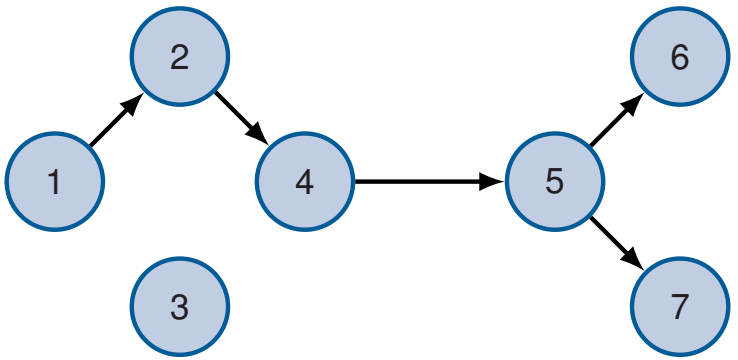
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Example 1–$\{6, 7\}$ multicast trees:
  - $(1, 2), (1, 3), (2, 6), (3, 7)$
  - $(1, 2), (2, 4), (2, 6), (4, 5), (5, 7)$

Multicast tree based forwarding



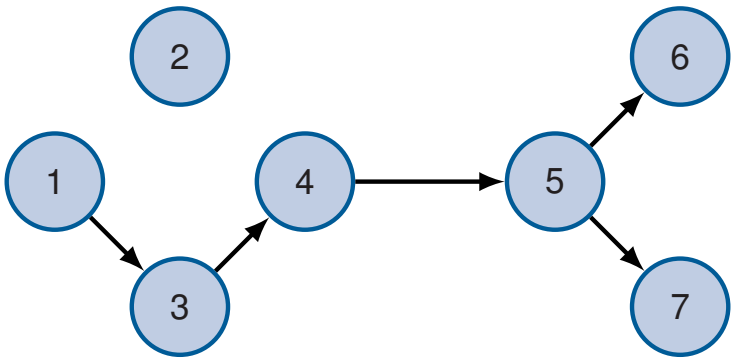
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Example 1–$\{6, 7\}$ multicast trees:
  - $(1, 2), (1, 3), (2, 6), (3, 7)$
  - $(1, 2), (2, 4), (2, 6), (4, 5), (5, 7)$
  - $(1, 3), (3, 4), (3, 7), (4, 5), (5, 6)$

Multicast tree based forwarding



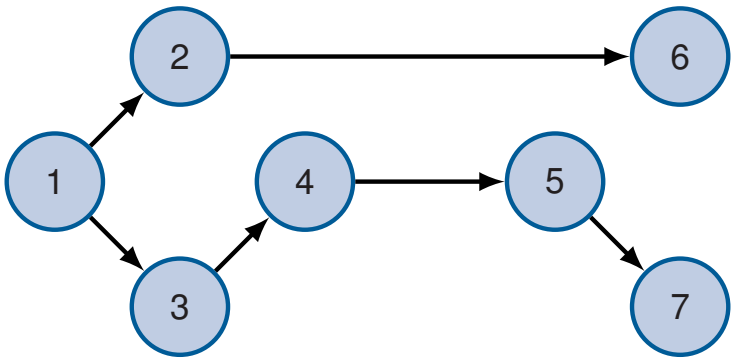
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Example 1–$\{6, 7\}$ multicast trees:
  - $(1, 2), (1, 3), (2, 6), (3, 7)$
  - $(1, 2), (2, 4), (2, 6), (4, 5), (5, 7)$
  - $(1, 3), (3, 4), (3, 7), (4, 5), (5, 6)$
  - $(1, 2), (2, 4), (4, 5), (5, 6), (5, 7)$

Multicast tree based forwarding



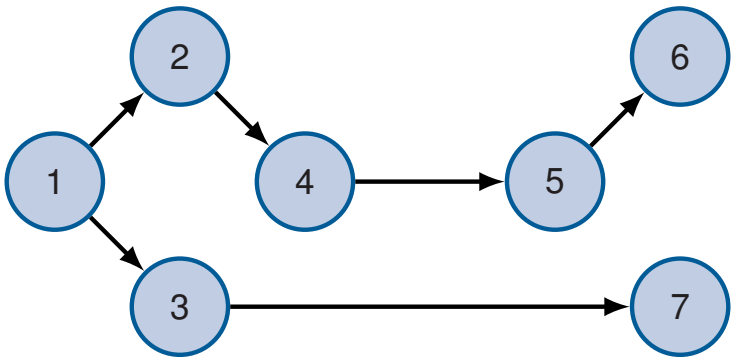
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Example 1–$\{6, 7\}$ multicast trees:
  - $(1, 2), (1, 3), (2, 6), (3, 7)$
  - $(1, 2), (2, 4), (2, 6), (4, 5), (5, 7)$
  - $(1, 3), (3, 4), (3, 7), (4, 5), (5, 6)$
  - $(1, 2), (2, 4), (4, 5), (5, 6), (5, 7)$
  - $(1, 3), (3, 4), (4, 5), (5, 6), (5, 7)$

Multicast tree based forwarding



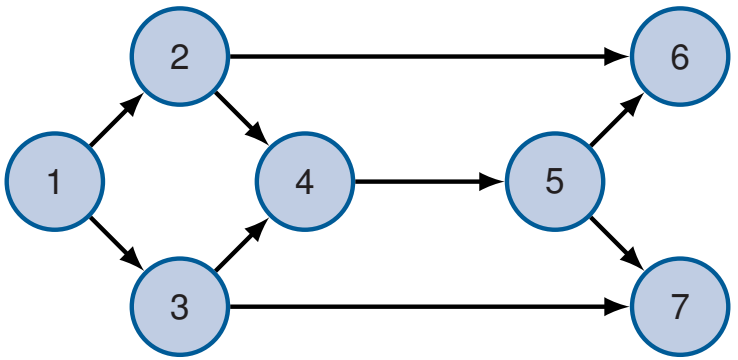
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Example 1–$\{6, 7\}$ multicast trees:
  - $(1, 2), (1, 3), (2, 6), (3, 7)$
  - $(1, 2), (2, 4), (2, 6), (4, 5), (5, 7)$
  - $(1, 3), (3, 4), (3, 7), (4, 5), (5, 6)$
  - $(1, 2), (2, 4), (4, 5), (5, 6), (5, 7)$
  - $(1, 3), (3, 4), (4, 5), (5, 6), (5, 7)$
  - $(1, 2), (1, 3), (2, 6), (3, 4), (4, 5), (5, 7)$

Multicast tree based forwarding



- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Example 1–$\{6, 7\}$ multicast trees:
  - $(1, 2), (1, 3), (2, 6), (3, 7)$
  - $(1, 2), (2, 4), (2, 6), (4, 5), (5, 7)$
  - $(1, 3), (3, 4), (3, 7), (4, 5), (5, 6)$
  - $(1, 2), (2, 4), (4, 5), (5, 6), (5, 7)$
  - $(1, 3), (3, 4), (4, 5), (5, 6), (5, 7)$
  - $(1, 2), (1, 3), (2, 6), (3, 4), (4, 5), (5, 7)$
  - $(1, 2), (1, 3), (2, 4), (3, 7), (4, 5), (5, 6)$

Multicast tree based forwarding



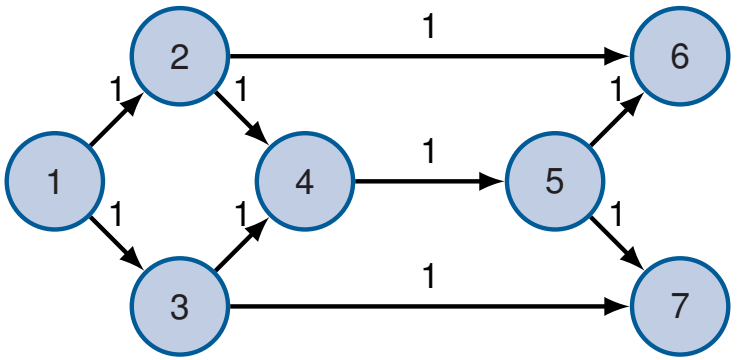
- s–T multicast tree: A tree rooted at s such that there exists a directed path to each $t \in T$ (arcs belong to at least one path).
- Unit flow on multicast tree delivers one unit (the same unit) of information to each terminal.
- Finding all/best multicast trees (directed Steiner trees) is a hard problem.
- Capacity needs to be split among all trees.

Multicast tree based forwarding: Max s–T flow problem

- Multicast trees $\mathrm{MT}_{sT} = \{1, \ldots, K\}$
- Multicast tree incidence vector $\boldsymbol{x}_k$ such that $x_{k,j} = 1$ if arc j is in the k-th multicast tree, otherwise $x_{k,j} = 0$
- One commodity for the multicast, no flow conservation constraint.
- Capacity vector $\boldsymbol{z}$ split among all trees.

$$\begin{aligned} \max \quad & \sum_{k \in \mathrm{MT}_{sT}} r_k \\ \text{s.t.} \quad & \sum_{k \in \mathrm{MT}_{sT}} r_k \boldsymbol{x}_k \leq \boldsymbol{z} \\ & r_k \geq 0 \quad \forall k \in \mathrm{MT}_{sT} \end{aligned}$$

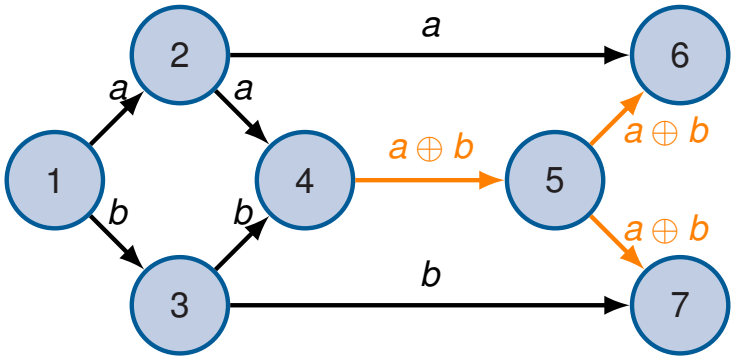
Multicast tree based forwarding



- Trees in optimal solution
  - $(1,2),(1,3),(2,6),(3,7)$ $\quad \boldsymbol{x}_1 = [1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0]^T$
  - $(1,2),(2,4),(2,6),(4,5),(5,7)$ $\quad \boldsymbol{x}_2 = [1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1]^T$
  - $(1,3),(3,4),(3,7),(4,5),(5,6)$ $\quad \boldsymbol{x}_2 = [0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0]^T$
- Each tree carries rate .5.
- Total flow which is capacity relevant
  - $0.5(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3) = [1\ 1\ .5\ 1\ .5\ 1\ 1\ .5\ .5]^T$

Maximum Multicast s–T Flow = 1.5

Network coding



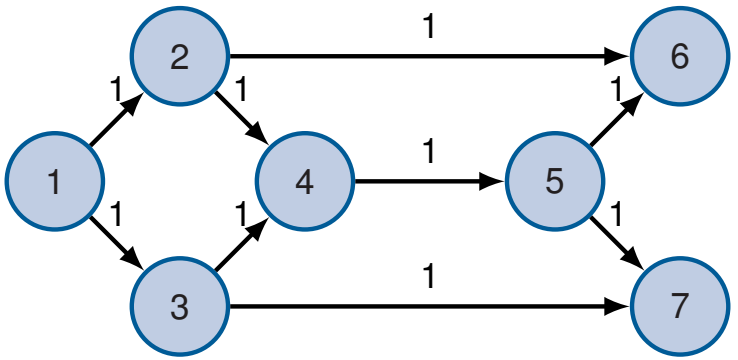
- A single packet (coded unit of information) may serve multiple terminals simultaneously.
- Consider flow to each terminal separately.
- But capacity is shared among all flows, i.e., each flow can use the full capacity on each arc.
- Example $(4, 5)$: Flow 1–6 and 1–7 transmit unit of information over this arc, but only one coded packet is transmitted.

Network coding: Max s–T flow problem

- One commodity flow $\boldsymbol{x}_t$ for each terminal $t \in T$
- Source vector $\boldsymbol{d}_{st}$ for each terminal $t \in T$
- Capacity vector $\boldsymbol{z}$ is shared for all flows, i.e., capacity on each arc can be fully exploited by each commodity flow.

$$\begin{aligned} \max\ r \quad \text{s.t.}\ \ \boldsymbol{M}\boldsymbol{x}_t &= r\boldsymbol{d}_{st} && \forall t \in T \\ \boldsymbol{x}_t &\geq \boldsymbol{0} && \forall t \in T \\ \boldsymbol{x}_t &\leq \boldsymbol{z} && \forall t \in T \end{aligned}$$

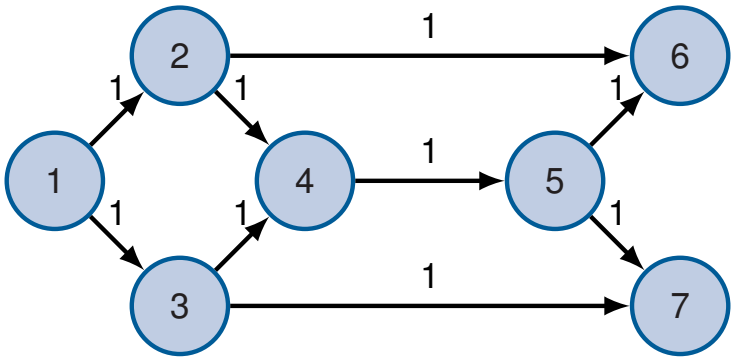
Multicast tree based forwarding



- Optimal flow solutions
  - $\boldsymbol{x}_6 = [\,1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\,]^T$
  - $\boldsymbol{x}_7 = [\,1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\,]^T$
- Total flow which is capacity relevant
  - $\max(\boldsymbol{x}_6, \boldsymbol{x}_7) = [\,1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\,]^T$

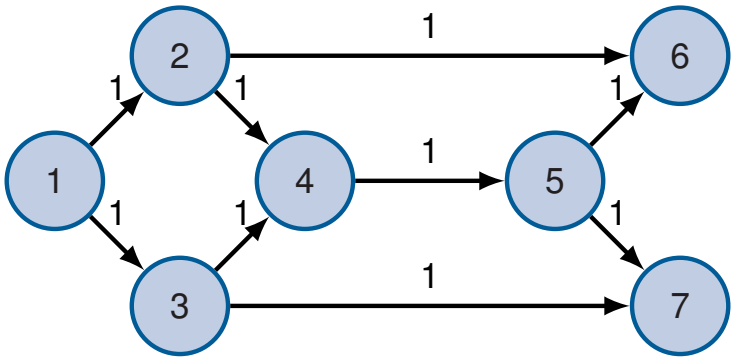
Maximum Multicast s–T Flow = 2

Comparison for Butterfly



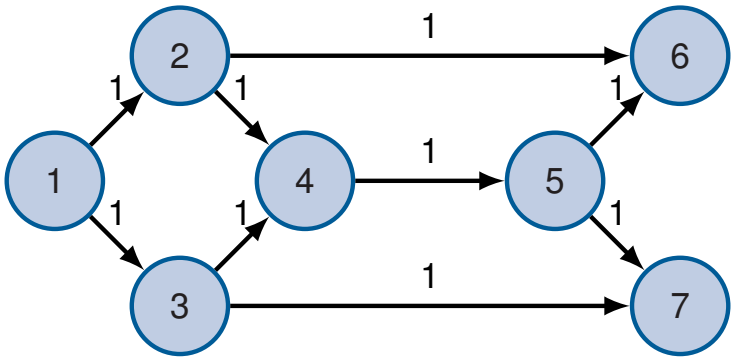
- Store-forward 1
- Multicast tree 1.5
- Network coding 2
- Can we do even better?

Comparison for Butterfly



- Store-forward 1
- Multicast tree 1.5
- Network coding 2
- Can we do even better? No!

Comparison for Butterfly



- Store-forward 1
- Multicast tree 1.5
- Network coding 2
- Can we do even better? No! Why?

Min cut upper bound on multicast rate TODO

Max-flow min-cut theorem

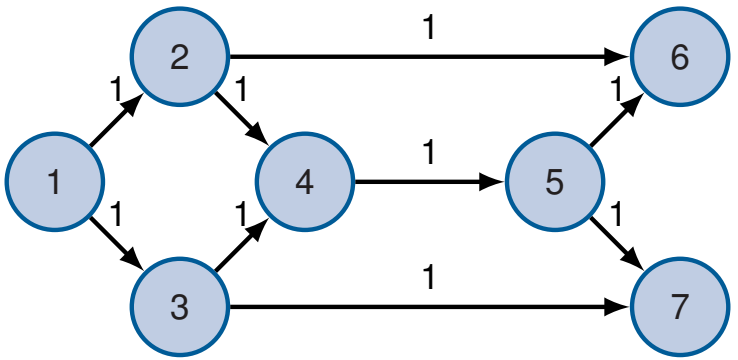
The value of the minimum s-t cut for all terminals $t \in T$ equals the value of the maximum s-T flow with network coding, i.e.,

$$\max\{r : \boldsymbol{M}\boldsymbol{x}_t = r\boldsymbol{d}_{st},\ \mathbf{0} \leq \boldsymbol{x}_t \leq \boldsymbol{z},\ \forall t \in T\}$$

$$=$$

$$\min_{t \in T} \min\{v(S) : S \text{ is } s\text{-}t \text{ cut}\}$$

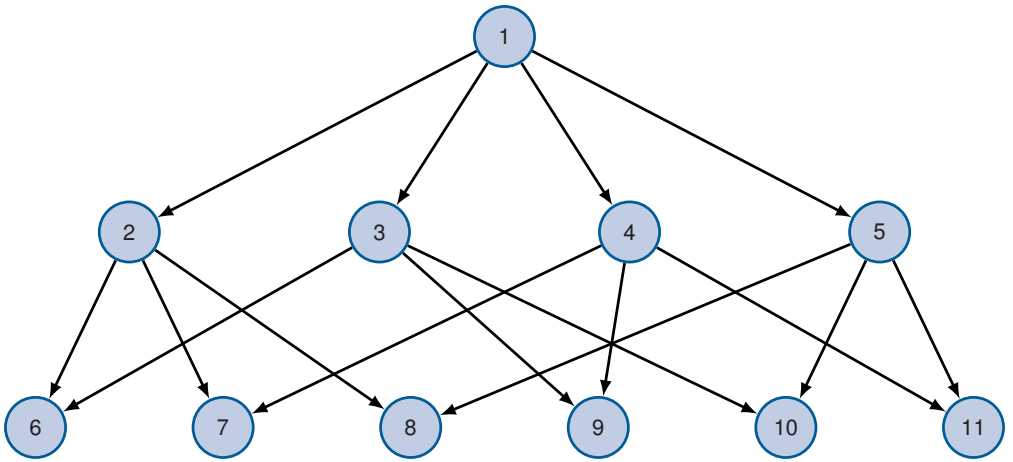
Min-cut for Butterfly



- $\min\{v(S) : S \text{ is } 1\text{–}6 \text{ cut}\} = 2$
- $\min\{v(S) : S \text{ is } 1\text{–}7 \text{ cut}\} = 2$

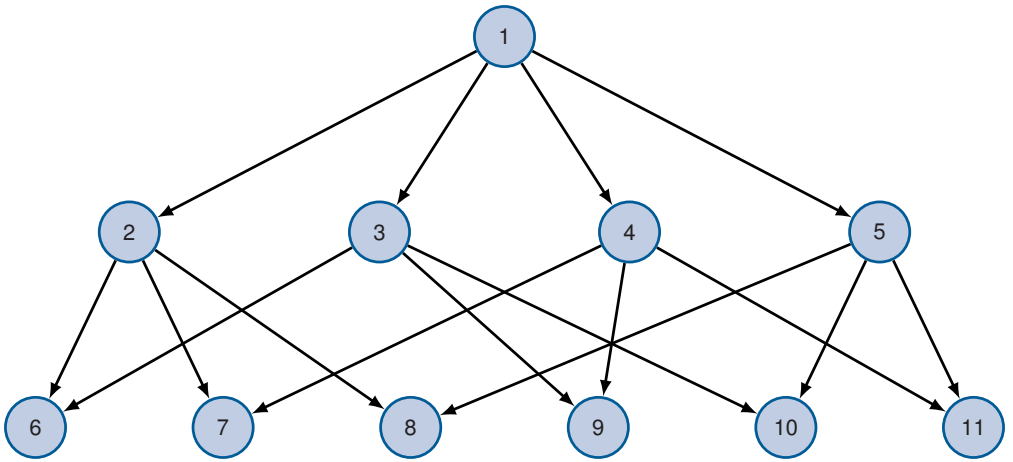
Multicast s–T Capacity = 2

Example: Combination network



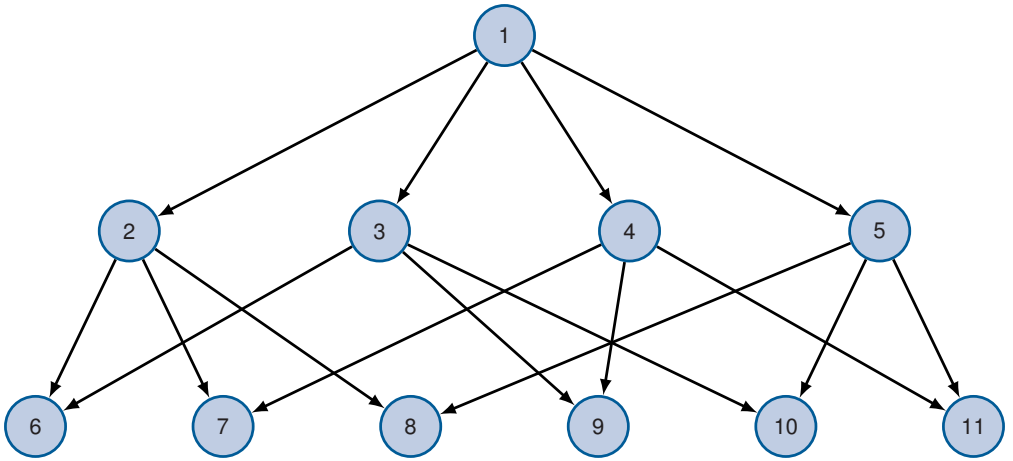
- Multicast $s = 1$ and $T = \{6, \ldots, 11\}$
- Unit arc capacities

Example: Combination network



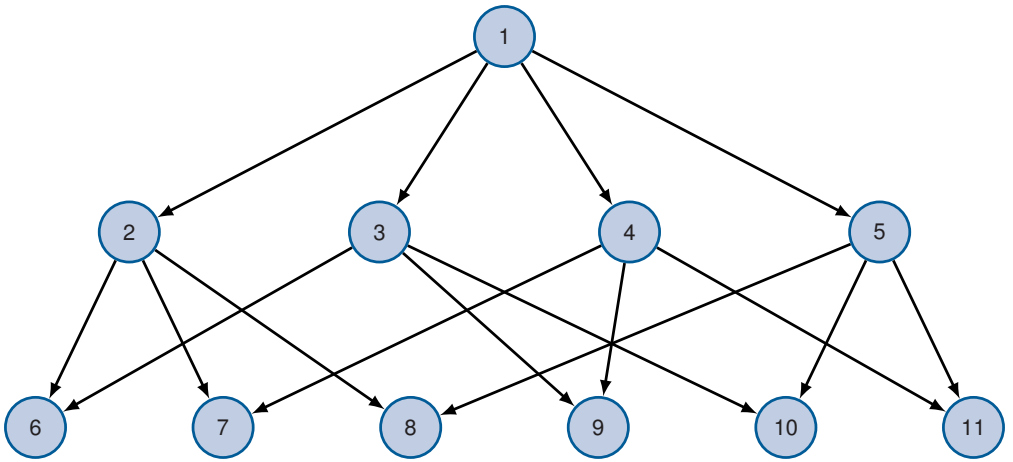
- Min-cut
- Store-forward
- Multicast tree
- Network coding (any code)
- Network coding (XOR only)

Example: Combination network



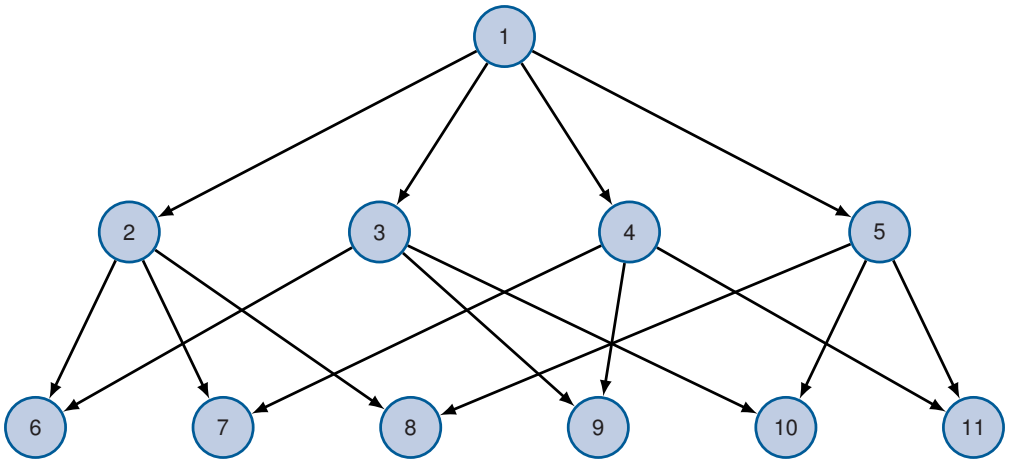
- Min-cut 2
- Store-forward
- Multicast tree
- Network coding (any code)
- Network coding (XOR only)

Example: Combination network



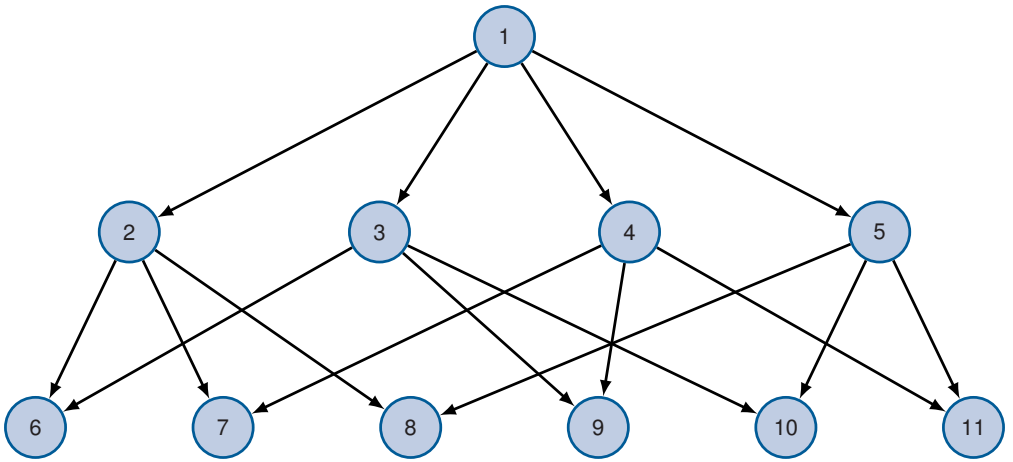
- Min-cut 2
- Store-forward $\frac{2}{3}$
- Multicast tree
- Network coding (any code)
- Network coding (XOR only)

Example: Combination network



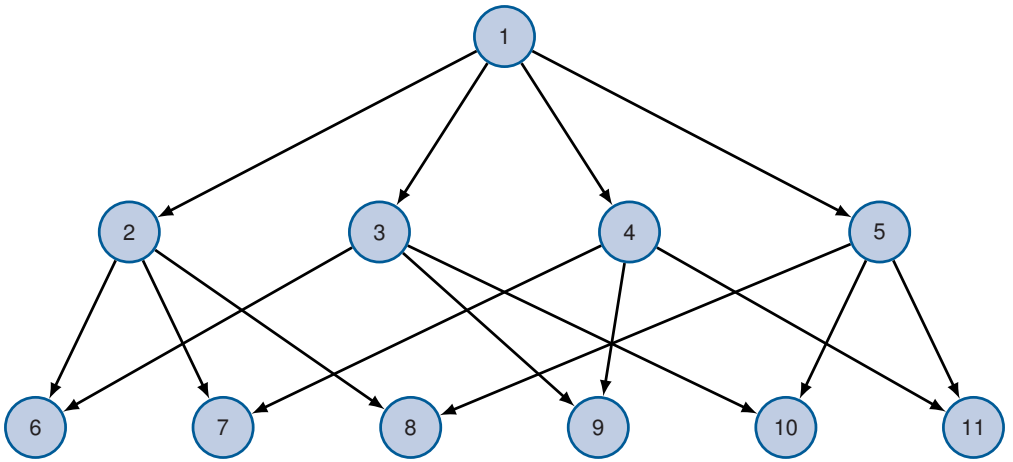
- Min-cut 2
- Store-forward $\frac{2}{3}$
- Multicast tree $\frac{4}{3}$
- Network coding (any code)
- Network coding (XOR only)

Example: Combination network



- Min-cut 2
- Store-forward $\frac{2}{3}$
- Multicast tree $\frac{4}{3}$
- Network coding (any code) 2 (Use flow model/min-cut)
- Network coding (XOR only)

Example: Combination network



- Min-cut 2
- Store-forward $\frac{2}{3}$
- Multicast tree $\frac{4}{3}$
- Network coding (any code) 2 (Use flow model/min-cut)
- Network coding (XOR only) $\frac{4}{3} \leq ?? < 2$ (Can not use flow model!)

Wired vs. Wireless — Typical Properties

Wired Networks

- Wired networks are composed from individual point-to-point links, which do not interact and share no resources.
- Links are next to lossless and error-free.
- Wired networks can be modeled as abstract graphs with perfect capacitated links for throughput calculation.

Wireless Networks

- Wireless networks share a common transmission medium.
- The medium is shared and omnidirectional, which turns it into a broadcast medium and causes interference.
- Wireless transmission are prone to errors leading to packet errors or packet loss.
- How can we model wireless networks? Graph?

Packet Networks

- Information is encoded into packets, which are protected by an
  - error correcting code on the physical layer (channel code) for removing inevitable transmission errors and an
  - error detecting code (e.g. CRC) to detect any residual errors or decoding failures of the channel code.

Wireless Packet Networks

- Nodes agree who is transmitting at which end and how long its transmission endures (medium access).
  - Simultaneous transmissions may cause interference (BAD).
  - No simultaneous transmissions may waste resources (BAD).
  - Medium access needs to be organized (central or distributed).
- Transmitted packets are randomly lost (not decodable).
  - Loss may be due to imperfections of wireless communication (fading, mobility, etc.).
  - Loss may also be due to interference (packet collisions).
- Transmitted packets are not only received by one (intended) node but by multiple nodes (wireless broadcast advantage).
  - Need to model selective overhearing of individual packets. Who gets which packet?
  - Need to integrate packet loss and packet overhearing with each other.

Capacitated Graph Model for Wireless Packet Networks

- Ignore wireless broadcast advantage: Each packet is intended for one particular receiver and all other nodes ignore it.
- Include medium access and interference in arc capacities
- Include packet losses into arc capacities

Wireless network model

- Graph (N, A)
- Arc capacity vector $\boldsymbol{z}$
- Region of admissible capacity vectors $\mathcal{Z}$ from which capacity vector $\boldsymbol{z}$ can be selected
  - Each capacity vector $\boldsymbol{z} \in \mathcal{Z}$ corresponds to a different trade-off between all arcs.
  - Trade-off is necessary due to shared resources and interference.
  - Compare to wired networks, where each arc capacity depends only on the properties of the underlying link ($\mathcal{Z} = \{\boldsymbol{z}\}$)

Model 1: Simple graph model with orthogonal medium access

- All packets use the same code rate (simplification).
- All resources are equal and can be split arbitrarily fine.
- No simultaneous transmissions
- No interference
- Shared transmission time/frequency resources: Resource share τ_j of arc j s.t. total resource shares add up to 1.
- Packet loss (due to fading/noise/mobility/...): Packet loss probability $\varepsilon_j \in [0, 1]$ on arc j ($\varepsilon_j = 0$ for all j means no packet loss)

Arc Capacity Region (NC or ACK/NACK)

$$\mathcal{Z} = \bigcup_{\substack{\boldsymbol{\tau} \geq \mathbf{0}: \\ \mathbf{1}^{\mathsf{T}} \boldsymbol{\tau} \leq 1}} \{\boldsymbol{z} : z_j = \tau_j(1 - \varepsilon_j)\}$$

Example: Skewed Diamond Network

Maximum s-t Flow

- Source vector $\boldsymbol{d}_{st}$
- Incidence matrix $\boldsymbol{M}$
- Arc capacity region $\mathcal{Z}$

$$
\begin{aligned}
\max \quad & r \\
\text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d}_{st} \\
& \boldsymbol{x} \geq \boldsymbol{0} \\
& \boldsymbol{x} \leq \boldsymbol{z} \\
& \boldsymbol{z} \in \mathcal{Z}
\end{aligned}
$$

Maximum s-T Multicast Flow (Network Coding)

- Source vector $\boldsymbol{d}_{st}$ for all $t \in T$
- Incidence matrix $\boldsymbol{M}$
- Arc capacity region $\mathcal{Z}$

$$\begin{aligned} \max\ r \quad \text{s.t.}\quad \boldsymbol{M}\boldsymbol{x}_t &= r\boldsymbol{d}_{st} && \forall t \in T \\ \boldsymbol{x}_t &\geq \boldsymbol{0} && \forall t \in T \\ \boldsymbol{x}_t &\leq \boldsymbol{z} && \forall t \in T \\ \boldsymbol{z} &\in \mathcal{Z} \end{aligned}$$

Multicast Max-Flow Min-Cut Theorem (Model 1)

The value of the minimum s-t cut for all terminals $t \in T$ equals the value of the maximum s-T flow with network coding, i.e.,

$$\max\left\{ r : \boldsymbol{M}\boldsymbol{x}_t = r\boldsymbol{d}_{st},\ \mathbf{0} \leq \boldsymbol{x}_t \leq \boldsymbol{z},\ \forall t \in T, \boldsymbol{z} \in \mathcal{Z} \right\}$$

$$=$$

$$\max_{\boldsymbol{z}\in\mathcal{Z}} \min_{t\in T} \min\left\{ v(S) = \sum\nolimits_{j\in A(S)} z_j : S \text{ is } s\text{-}t \text{ cut} \right\}$$

Example: Skewed Diamond Network

Hypergraphs (directed) $G = (N, H)$

- Nodes $N = \{1, \ldots, n\}$
- Hyperarcs $H = \{1, \ldots, m\}$
- Each hyperarc $j \in H$ represents an ordered pair (a, B) of a node $a = \text{Tail}(j) \in N$ and a subset of nodes $B = \text{Head}(j) \subset N$ with $a \notin B$. (Notation $j \equiv (a, B)$)
- Graph (N, A) (directed) induced by hypergraph (N, H) consists of all arcs k with $a = \text{tail}(k)$ and $b = \text{head}(k)$ such that there exists $j \in H$ with $a = \text{Tail}(j)$ and $b \in \text{Head}(j)$. (Notation $k \equiv (a, b)$)
- Arcs A_j which are generated by hyperarc j
- Hyperarc-arc incidence matrix $\boldsymbol{N}$: $N_{jk} = 1$ if $k \in A_j$, $N_{jk} = 0$ else
- Important structures in hypergraphs (via induced graph (N, A)):[12]
  - Path (directed, undirected)
  - Tree (directed, undirected)
  - Cycle (directed, undirected)
- We assume G is connected, i.e., there exists an undirected path between any pair of nodes.

[12] ... but may not make so much sense any more

Example: Skewed Diamond Network

Model 2: Lossless hypergraph model with orthogonal MAC

- Hypergraph (N, H)
- One hyperarc per node (simplification), enumerated according to tail nodes $(N = H)$
- Inherits MAC properties from model 1
- Each node gets a resource share $\tau_i \geq 0$ such that $\sum_{i \in N} \tau_i \leq 1$
- Packets transmitted on a hyperarc $j \equiv (a, B)$ (i.e., by its tail node a) are received by all head nodes $b \in B$.
- No packets are lost.

Information Flow in Lossless Hypergraphs (N, H) (Model 2)

- Information flow vector $\boldsymbol{x}$ on induced graph (N, A)
- Flow must be conserved on induced graph

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{d}$$

- Receivers of a hyperarc get identical packets over this hyperarc
- Each piece of information can only be used once (by one node)

$$\boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{z} \quad \Leftrightarrow \quad \sum_{k \in A_j} x_k \leq z_j \quad \forall j \in H$$

- Lossless Hyperarc Capacity Region (NC)

$$\mathcal{Z} = \bigcup_{\substack{\boldsymbol{\tau} \geq \mathbf{0}: \\ \mathbf{1}^{\mathrm{T}} \boldsymbol{\tau} \leq 1}} \{\boldsymbol{z} : z_j = \tau_j\}$$

Example: Skewed Diamond Network

Hyperarc Maximum *s*-*t* Flow (Routing/Network Coding)

- Source vector $\boldsymbol{d}_{st}$
- Incidence matrix $\boldsymbol{M}$
- Hyperarc-arc incidence matrix $\boldsymbol{N}$
- Hyperarc capacity region $\mathcal{Z}$

$$
\begin{aligned}
\max \quad & r \\
\text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d}_{st} \\
& \boldsymbol{x} \geq \boldsymbol{0} \\
& \boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{z} \\
& \boldsymbol{z} \in \mathcal{Z}
\end{aligned}
$$

Hyperarc Maximum s-T Multicast Flow (Network Coding)

- Source vector $\boldsymbol{d}_{st}$
- Incidence matrix $\boldsymbol{M}$
- Hyperarc-arc incidence matrix $\boldsymbol{N}$
- Hyperarc capacity region $\mathcal{Z}$

$$\begin{aligned} \max\ r \quad \text{s.t.}\ \ \boldsymbol{M}\boldsymbol{x}_t &= r\boldsymbol{d}_{st} && \forall t \in T \\ \boldsymbol{x}_t &\geq \boldsymbol{0} && \forall t \in T \\ \boldsymbol{N}\boldsymbol{x}_t &\leq \boldsymbol{z} && \forall t \in T \\ \boldsymbol{z} &\in \mathcal{Z} \end{aligned}$$

- We can use each hyperarc (packet) only once for each terminal.
- But we can use each hyperarc differently for each terminal.

Hyperarc Min-Cut Model

- An s-t cut is a subset of nodes $S \subset N$ such that $s \in S$ and $t \notin S$.
- A hyperarc $j \in H$ crosses S if Tail(j) $\in S$ and Head(j) $\notin S$. $H(S)$ denotes all crossing arcs.
- The value of any s-t cut upper bounds the maximum s-t flow.
- The value of an s-t cut given the capacity vector $\boldsymbol{z}$ is defined as

$$v(S) = \sum_{j \in H(S)} z_j$$

- Model 2 (only one HA per node, $z_j = \tau_{\text{tail}(j)}$)

$$v(S) = \sum_{j \in H(S)} \tau_{\text{tail}(j)}$$

Multicast Max-Flow Min-Cut Theorem (Model 2)

The value of the minimum s-t cut for all terminals $t \in T$ equals the value of the maximum s-T flow with network coding, i.e.,

$$\max\left\{ r : \boldsymbol{M}\boldsymbol{x}_t = r\boldsymbol{d}_{st},\ \boldsymbol{0} \leq \boldsymbol{x}_t,\ \boldsymbol{N}\boldsymbol{x}_t \leq \boldsymbol{z},\ \forall t \in T, \boldsymbol{z} \in \mathcal{Z} \right\}$$

$$=$$

$$\max_{\boldsymbol{z} \in \mathcal{Z}} \min_{t \in T} \min\left\{ v(S) = \sum\nolimits_{j \in H(S)} z_j : S \text{ is } s\text{-}t \text{ cut} \right\}$$

Example: Skewed Diamond Network

Model 3: Lossy hypergraph model with orthogonal MAC

- Hypergraph (N, H) with induced graph (N, A)
- All possible hyperarcs for each node a up to some maximal set N_a (neighbors): $j \in H$ with $j \equiv (a, B)$ if $B \subset N_a$ ($2^{|N_a|}$ hyperarcs)
- Inherits MAC properties from model 1
- Each node gets a resource share $\tau_i \geq 0$ such that $\sum_{i \in N} \tau_i \leq 1$
- Packet loss is independent across all receivers (simplification)
- Packets from a to b are lost with probability ε_k where $k \equiv (a, b)$
- Packets transmitted by a are transmitted on hyperarc $j \equiv (a, B)$ (i.e., they are received by $B \subset N_a$ and lost by all other nodes $N_a \setminus B$) with probability

$$\Pr[\text{hyperarc } j \mid \text{Tail}(j) \text{ transmits}] = \prod_{k \in A_j} (1 - \varepsilon_k) \prod_{\substack{k \notin A_j \\ \text{tail}(k)=\text{Tail}(j)}} \varepsilon_k$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Information flow vector $\boldsymbol{x}$ on induced graph (N, A)
- Flow must be conserved on induced graph

$$\boldsymbol{Mx} = \boldsymbol{d}$$

- Receivers of a hyperarc get identical packets over this hyperarc provided they have not lost the packets
- Each piece of information can only be used by one node of those which received it
- The total flow to a set of nodes must be smaller than the total fraction of different received packets of this set of nodes

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Single receiver set $b \in N_a$ and $k \equiv (a, b)$
  - Hyperarcs that bring packets from a to b: j with $a = \text{Tail}(j)$ and $b \in \text{Head}(j)$, or equivalently, $j : k \in A_j$
  - Flow bound

$$x_k \leq \sum_{j:k \in A_j} \tau_{\text{tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \text{tail}(\ell)=\text{Tail}(j)}} \varepsilon_\ell$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Single receiver set $b \in N_a$ and $k \equiv (a, b)$
  - Hyperarcs that bring packets from a to b: j with $a = \text{Tail}(j)$ and $b \in \text{Head}(j)$, or equivalently, $j : k \in A_j$
  - Flow bound

$$x_k \leq \sum_{j:k \in A_j} \tau_{\text{tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \text{tail}(\ell)=\text{Tail}(j)}} \varepsilon_\ell$$

- Two receiver set $B = \{b, c\} \subset N_a$ and $k_1 \equiv (a, b)$, $k_2 \equiv (a, c)$
  - Hyperarcs that bring packets from a to either b or c: j with $a = \text{Tail}(j)$ and $B \cap \text{Head}(j) \neq \emptyset$, or equivalently, $j : \{k_1, k_2\} \cap A_j \neq \emptyset$
  - Joint flow bound for both receivers

$$x_{k_1} + x_{k_2} \leq \sum_{j \in H : \{k_1, k_2\} \cap A_j \neq \emptyset} \tau_{\text{tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \text{tail}(\ell)=\text{Tail}(j)}} \varepsilon_\ell$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Single receiver set $b \in N_a$ and $k \equiv (a, b)$
  - Hyperarcs that bring packets from a to b: j with $a = \text{Tail}(j)$ and $b \in \text{Head}(j)$, or equivalently, $j : k \in A_j$
  - Flow bound

$$x_k \leq \sum_{j:k \in A_j} \tau_{\text{tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \text{tail}(\ell)=\text{Tail}(j)}} \varepsilon_\ell$$

- Two receiver set $B = \{b, c\} \subset N_a$ and $k_1 \equiv (a, b)$, $k_2 \equiv (a, c)$
  - Hyperarcs that bring packets from a to either b or c: j with $a = \text{Tail}(j)$ and $B \cap \text{Head}(j) \neq \emptyset$, or equivalently, $j : \{k_1, k_2\} \cap A_j \neq \emptyset$
  - Joint flow bound for both receivers

$$x_{k_1} + x_{k_2} \leq \sum_{j \in H : \{k_1, k_2\} \cap A_j \neq \emptyset} \tau_{\text{tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \text{tail}(\ell)=\text{Tail}(j)}} \varepsilon_\ell$$

- ...

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Multiple receiver set: Each pair (a, B) corresponds to some hyperarc j', i.e., $j' \equiv (a, B)$

$$\sum_{k \in A_{j'}} x_k \leq \sum_{j \in H : A_{j'} \cap A_j \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Multiple receiver set: Each pair (a, B) corresponds to some hyperarc j', i.e., $j' \equiv (a, B)$

$$\sum_{k \in A_{j'}} x_k \leq \sum_{j \in H: A_{j'} \cap A_j \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell$$

- Hyperarc capacity

$$z_j = \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Multiple receiver set: Each pair (a, B) corresponds to some hyperarc j', i.e., $j' \equiv (a, B)$

$$\sum_{k \in A_{j'}} x_k \leq \sum_{j \in H: A_{j'} \cap A_j \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell$$

- Hyperarc capacity

$$z_j = \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell$$

- Hyperarc capacity region

$$\mathcal{Z} = \bigcup_{\substack{\boldsymbol{\tau} \geq \mathbf{0} \\ \mathbf{1}^{\mathsf{T}} \boldsymbol{\tau} \leq 1}} \left\{ \boldsymbol{z} : z_j = \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell \quad \forall j \in H \right\}$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Reformulation of the flow bound for receiver set $\mathrm{Head}(j')$

$$\sum_{k \in A_{j'}} x_k \leq \tau_{\mathrm{Tail}(j')} \left(1 - \prod_{\ell \in A_{j'}} \varepsilon_\ell \right)$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Reformulation of the flow bound for receiver set Head(j')

$$\sum_{k \in A_{j'}} x_k \leq \tau_{\text{Tail}(j')} \left(1 - \prod_{\ell \in A_{j'}} \varepsilon_\ell \right)$$

- Lossy flow bound (for all receiver set)

$$\sum_{k \in A_j} x_k \leq \tau_{\text{Tail}(j)} \left(1 - \prod_{\ell \in A_j} \varepsilon_\ell \right) \quad \forall j \in H$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Reformulation of the flow bound for receiver set Head(j')

$$\sum_{k \in A_{j'}} x_k \leq \tau_{\text{Tail}(j')} \left(1 - \prod_{\ell \in A_{j'}} \varepsilon_\ell \right)$$

- Lossy flow bound (for all receiver set)

$$\sum_{k \in A_j} x_k \leq \tau_{\text{Tail}(j)} \left(1 - \prod_{\ell \in A_j} \varepsilon_\ell \right) \quad \forall j \in H$$

- Broadcast capacity region

$$\mathcal{Y} = \bigcup_{\substack{\boldsymbol{\tau} \geq \mathbf{0} \\ \mathbf{1}^{\mathsf{T}} \boldsymbol{\tau} \leq 1}} \left\{ \boldsymbol{y} : y_j = \tau_{\text{Tail}(j)} \left(1 - \prod_{\ell \in A_j} \varepsilon_\ell \right) \quad \forall j \in H \right\}$$

Information Flow in Lossy Hypergraphs (N, H) (Model 3)

- Hyperarc-arc incidence matrix $\boldsymbol{N}$:
  $N_{jk} = 1$ if $k \in A_j$, $N_{jk} = 0$ else.
- Hyperarc-hyperarc incidence matrix $\boldsymbol{Q}$:
  $Q_{ij} = 1$ if $A_i \cap A_j \neq \emptyset$, $Q_{ij} = 0$ else.
- Hyperarc-to-broadcast transformation

$$\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{z}$$

- Lossy hyperarc flow bound with hyperarc capacity region

$$\boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{Q}\boldsymbol{z}$$

- Lossy hyperarc flow bound with broadcast capacity region

$$\boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{y}$$

Lossy Hyperarc Maximum *s*-*t* Flow (Opportunistic RT/NC)

- Source vector $\boldsymbol{d}_{st}$
- Incidence matrix $\boldsymbol{M}$
- Hyperarc-arc incidence matrix $\boldsymbol{N}$
- Broadcast capacity region $\mathcal{Y}$

$$
\begin{aligned}
\max \quad & r \\
\text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x} = r\boldsymbol{d}_{st} \\
& \boldsymbol{x} \geq \boldsymbol{0} \\
& \boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{y} \\
& \boldsymbol{y} \in \mathcal{Y}
\end{aligned}
$$

Lossy Hyperarc Maximum s-T Multicast Flow (NC)

- Source vector $\boldsymbol{d}_{st}$
- Incidence matrix $\boldsymbol{M}$
- Hyperarc-arc incidence matrix $\boldsymbol{N}$
- Broadcast capacity region $\mathcal{Y}$

$$\begin{aligned} \max\ r \quad \text{s.t.}\quad \boldsymbol{M}\boldsymbol{x}_t &= r\boldsymbol{d}_{st} && \forall t \in T \\ \boldsymbol{x}_t &\geq \boldsymbol{0} && \forall t \in T \\ \boldsymbol{N}\boldsymbol{x}_t &\leq \boldsymbol{y} && \forall t \in T \\ \boldsymbol{y} &\in \mathcal{Y} \end{aligned}$$

Lossy Hyperarc Min-Cut Model

- An s-t cut is a subset of nodes $S \subset N$ such that $s \in S$ and $t \notin S$.
- A hyperarc $j \in H$ crosses S if Tail(j) $\in S$ and Head(j) $\notin S$. $H(S)$ denotes all crossing arcs.
- The value of any s-t cut upper bounds the maximum s-t flow.
- The value of an s-t cut given the capacity vector $\boldsymbol{z}$ is defined as

$$v(S) = \sum_{j \in H(S)} z_j$$

Lossy Hyperarc Min-Cut Model

- An s-t cut is a subset of nodes $S \subset N$ such that $s \in S$ and $t \notin S$.
- A hyperarc $j \in H$ crosses S if $\text{Tail}(j) \in S$ and $\text{Head}(j) \notin S$. $H(S)$ denotes all crossing arcs.
- The value of any s-t cut upper bounds the maximum s-t flow.
- The value of an s-t cut given the capacity vector $\boldsymbol{z}$ is defined as

$$v(S) = \sum_{j \in H(S)} z_j$$

- Cut value of Model 3:

$$v(S) = \sum_{j \in H(S)} \tau_{\text{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \text{tail}(\ell)=\text{Tail}(j)}} \varepsilon_\ell$$

Lossy Hyperarc Min-Cut Model

- $A_i(S)$: Set of arcs $k \in A$ s.t. tail$(k) = i$, head$(i) \in (N \setminus S)$
- $H_i(S)$: Set of hyperarcs $j \in H$ s.t. Tail$(j) = i$, Head$(j) \cap (N \setminus S) \neq \emptyset$
- Characterize $H(S)$:

$$
\begin{aligned}
H(S) &= \{j \in H : \text{Tail}(j) \in S, \text{Head}(j) \cap (N \setminus S) \neq \emptyset\} \\
&= \bigcup_{i \in S} H_i(S) \\
&= \bigcup_{i \in S} \{j \in H : A_j \cap A_i(S) \neq \emptyset\}
\end{aligned}
$$

Lossy Hyperarc Min-Cut Model

- $A_i(S)$: Set of arcs $k \in A$ s.t. tail$(k) = i$, head$(i) \in (N \setminus S)$
- $H_i(S)$: Set of hyperarcs $j \in H$ s.t. Tail$(j) = i$, Head$(j) \cap (N \setminus S) \neq \emptyset$
- Characterize $H(S)$:

$$\begin{aligned} H(S) &= \{j \in H : \text{Tail}(j) \in S, \text{Head}(j) \cap (N \setminus S) \neq \emptyset\} \\ &= \bigcup_{i \in S} H_i(S) \\ &= \bigcup_{i \in S} \{j \in H : A_j \cap A_i(S) \neq \emptyset\} \end{aligned}$$

- Cut value of Model 3 (looks very much like flow bound)

$$v(S) = \sum_{i \in S} \sum_{j \in H : A_j \cap A_i(S) \neq \emptyset} \tau_{\text{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \text{tail}(\ell) = \text{Tail}(j)}} \varepsilon_\ell$$

Lossy Hyperarc Min-Cut Model

- Flow bound

$$\sum_{k \in A_{j'}} x_k \leq \sum_{j \in H : A_{j'} \cap A_j \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell \quad \forall j' \in H$$

- Cut value of Model 3

$$v(S) = \sum_{i \in S} \sum_{j \in H : A_j \cap A_i(S) \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell$$

Lossy Hyperarc Min-Cut Model

- Flow bound

$$\sum_{k \in A_{j'}} x_k \leq \sum_{j \in H : A_{j'} \cap A_j \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell \quad \forall j' \in H$$

$$\sum_{k \in A_{j'}} x_k \leq \tau_{\mathrm{Tail}(j')} \left(1 - \prod_{\ell \in A_{j'}} \varepsilon_\ell \right) = y_{j'} \quad \forall j' \in H$$

- Cut value of Model 3

$$v(S) = \sum_{i \in S} \sum_{j \in H : A_j \cap A_i(S) \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell$$

Lossy Hyperarc Min-Cut Model

- Flow bound

$$\sum_{k \in A_{j'}} x_k \leq \sum_{j \in H : A_{j'} \cap A_j \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell \quad \forall j' \in H$$

$$\sum_{k \in A_{j'}} x_k \leq \tau_{\mathrm{Tail}(j')} \left(1 - \prod_{\ell \in A_{j'}} \varepsilon_\ell \right) = y_{j'} \quad \forall j' \in H$$

- Cut value of Model 3

$$v(S) = \sum_{i \in S} \sum_{j \in H : A_j \cap A_i(S) \neq \emptyset} \tau_{\mathrm{Tail}(j)} \prod_{\ell \in A_j} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_j \\ \mathrm{tail}(\ell) = \mathrm{Tail}(j)}} \varepsilon_\ell$$

$$v(S) = \sum_{i \in S} \tau_i \left(1 - \prod_{\ell \in A_i(S)} \varepsilon_\ell \right) = \sum_{\substack{j' \in H : \mathrm{Tail}(j') \in S, \\ \mathrm{Head}(j) = N_{\mathrm{Tail}(j')} \setminus S}} y_{j'}$$

Multicast Max-Flow Min-Cut Theorem (Model 3)

The value of the minimum s-t cut for all terminals $t \in T$ equals the value of the maximum s-T flow with network coding, i.e.,

$$\max\left\{ r : \boldsymbol{M}\boldsymbol{x}_t = r\boldsymbol{d}_{st},\ \boldsymbol{0} \leq \boldsymbol{x}_t,\ \boldsymbol{N}\boldsymbol{x}_t \leq \boldsymbol{y},\ \forall t \in T, \boldsymbol{y} \in \mathcal{Y} \right\}$$

$$=$$

$$\max_{\boldsymbol{y} \in \mathcal{Y}} \min_{t \in T} \min\left\{ v(S) = \sum_{\substack{j \in H : \mathrm{Tail}(j) \in S \\ \mathrm{Head}(j) = N_{\mathrm{Tail}(j)} \setminus S}} y_j : S \text{ is } s\text{-}t \text{ cut} \right\}$$

Example: Skewed Diamond Network

Compare Flow Bounds and Capacity Regions

Model 1 (Graph)

$$\boldsymbol{x} \leq \boldsymbol{z} \qquad \mathcal{Z} = \bigcup_{\boldsymbol{\tau} \geq \mathbf{0} : \mathbf{1}^{\mathsf{T}} \boldsymbol{\tau} \leq 1} \{\boldsymbol{z} : z_j = \tau_j (1 - \varepsilon_j)\}$$

Model 2 (Lossless Hypergraph)

$$\boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{z} \qquad \mathcal{Z} = \bigcup_{\boldsymbol{\tau} \geq \mathbf{0} : \mathbf{1}^{\mathsf{T}} \boldsymbol{\tau} \leq 1} \{\boldsymbol{z} : z_j = \tau_j\}$$

Model 3 (Lossy Hypergraph)

$$\boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{Q}\boldsymbol{z} = \boldsymbol{y} \qquad \mathcal{Y} = \bigcup_{\substack{\boldsymbol{\tau} \geq \mathbf{0} \\ \mathbf{1}^{\mathsf{T}} \boldsymbol{\tau} \leq 1}} \left\{ \boldsymbol{y} : y_j = \tau_{\mathrm{Tail}(j)} \left(1 - \prod_{\ell \in A_j} \varepsilon_\ell \right) \quad \forall j \in H \right\}$$

Model 2 $\in$ Model 3:

- Model 2 hyperarcs H_2: $j \equiv (a, N_a)\ \forall a \in N$ with capacity $z_j = \tau_{\text{Tail}(j)}$
- Add hyperarcs $j \equiv (a, B)\ \forall B \subsetneq N_a$ with capacity $z_j = 0 \Rightarrow H_3$
- Flow bound (Model 2)

$$\sum_{k \in A_j} x_k \leq z_j = \tau_{\text{Tail}(j)} \quad \forall j \in H_2 \ \ (\Leftrightarrow \forall j \equiv (a, N_a), a \in N)$$

- Equivalent to the following bound since $z_j = 0$ for all $j \notin H_2$:

$$\sum_{k \in A_j} x_k \leq \sum_{\substack{j' \in H_3: \\ A_j \cap A_{j'} \neq \emptyset}} z_{j'} = \tau_{\text{Tail}(j)} \quad \forall j \in H_3$$

- Corresponds to Model 3 with $\varepsilon_k = 0$ for all $k \in A$

$$\sum_{k \in A_j} x_k \leq \sum_{\substack{j' \in H_3: \\ A_j \cap A_{j'} \neq \emptyset}} \tau_{\text{Tail}(j')} \prod_{\ell \in A_{j'}} (1 - \varepsilon_\ell) \prod_{\substack{\ell \notin A_{j'} \\ \text{tail}(\ell) = \text{Tail}(j')}} \varepsilon_\ell = \tau_{\text{Tail}(j)} \quad \forall j \in H_3$$

Medium Access Schemes

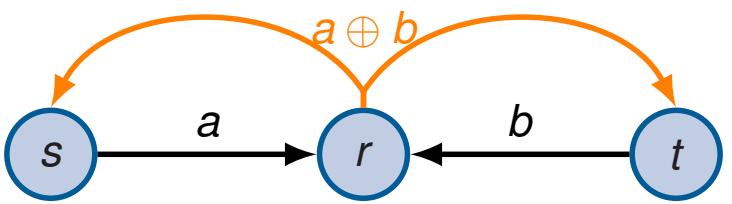
- Orthogonal MAC
- Centralized Scheduling
- Aloha Random Access (slotted/unslotted)
- Carrier Sense Multiple Access (slotted/unslotted)

Include MAC into Model 3:

- Different/More complex constraints on τ_i
- Packet loss probabilities ε_k depend on active transmitters

Bidirectional Communication

- Two nodes s and t exchange information, all other nodes help by relaying.
- Equivalent to a multicast with terminal set $T = \{s, t\}$ and two sources s and t.
- Maximize rate r_s (source s) and rate r_t (source t) jointly.



- Packet a from s to t and packet b from t to a.
- Equivalent to s and t want to have both packets a and b.

Flow model for the bidirectional rate region (Model 1)

$$\mathcal{R} = \bigcup_{\boldsymbol{z}\in\mathcal{Z}} \Big\{ [r_s, r_t]^{\mathsf{T}} \geq \mathbf{0} : \boldsymbol{M}\boldsymbol{x}_t = r_s \boldsymbol{d}_{st}, \mathbf{0} \leq \boldsymbol{x}_t \leq \boldsymbol{z}, \\ \boldsymbol{M}\boldsymbol{x}_s = r_t \boldsymbol{d}_{ts}, \mathbf{0} \leq \boldsymbol{x}_s \leq \boldsymbol{z}, \Big\}$$

Cut model for the bidirectional rate region (Model 1)

$$\mathcal{R} = \bigcup_{\boldsymbol{z}\in\mathcal{Z}} \Big\{ [r_s, r_t]^{\mathsf{T}} \geq \mathbf{0} : r_s \leq v(S) = \sum_{j\in A(S)} z_j \; \forall S \text{ is } s\text{-}t \text{ cut} \\ r_t \leq v(S) = \sum_{j\in A(S)} z_j \; \forall S \text{ is } t\text{-}s \text{ cut} \Big\}$$

Max-Flow Min-Cut: Both rate regions are equal!

Weighted Maximum Bidirectional Flow (Model 1)

- Source vectors $\boldsymbol{d}_{st}, \boldsymbol{d}_{ts}$
- Incidence matrix $\boldsymbol{M}$
- Weights $\alpha_s, \alpha_t \geq 0$
- Arc capacity region $\mathcal{Z}$

$$
\begin{aligned}
\max \quad & \alpha_s r_s + \alpha_t r_t \\
\text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x}_t = r_s \boldsymbol{d}_{st} \\
& \boldsymbol{M}\boldsymbol{x}_s = r_t \boldsymbol{d}_{ts} \\
& \boldsymbol{x}_t \leq \boldsymbol{z} \\
& \boldsymbol{x}_s \leq \boldsymbol{z} \\
& \boldsymbol{x}_s, \boldsymbol{x}_t \geq \boldsymbol{0}, \;\; r_s, r_t \geq 0 \\
& \boldsymbol{z} \in \mathcal{Z}
\end{aligned}
$$

Flow model for the bidirectional rate region (Model 2 & 3)

$$\mathcal{R} = \bigcup_{\boldsymbol{y} \in \mathcal{Y}} \Big\{ [r_s, r_t]^{\mathsf{T}} \geq \mathbf{0} : \boldsymbol{M}\boldsymbol{x}_t = r_s \boldsymbol{d}_{st}, \mathbf{0} \leq \boldsymbol{x}_t, \boldsymbol{N}\boldsymbol{x}_t \leq \boldsymbol{y}, \\ \boldsymbol{M}\boldsymbol{x}_s = r_t \boldsymbol{d}_{ts}, \mathbf{0} \leq \boldsymbol{x}_s, \boldsymbol{N}\boldsymbol{x}_s \leq \boldsymbol{y}, \Big\}$$

Cut model for the bidirectional rate region (Model 1)

$$\mathcal{R} = \bigcup_{\boldsymbol{y} \in \mathcal{Y}} \Bigg\{ [r_s, r_t]^{\mathsf{T}} \geq \mathbf{0} : r_s \leq v(S) = \sum_{\substack{j \in H : \mathrm{Tail}(j) \in S \\ \mathrm{Head}(j) = N_{\mathrm{Tail}(j)} \setminus S}} y_j \; \forall S \text{ is } s\text{-}t \text{ cut} \\ r_t \leq v(S) = \sum_{\substack{j \in H : \mathrm{Tail}(j) \in S \\ \mathrm{Head}(j) = N_{\mathrm{Tail}(j)} \setminus S}} y_j \; \forall S \text{ is } t\text{-}s \text{ cut} \Bigg\}$$

Max-Flow Min-Cut: Both rate regions are equal!

Weighted Maximum Bidirectional Flow (Model 2 & 3)

- Source vectors $\boldsymbol{d}_{st}, \boldsymbol{d}_{ts}$
- Incidence matrix $\boldsymbol{M}$, hyperarc-arc incidence matrix $\boldsymbol{N}$
- Weights $\alpha_s, \alpha_t \geq 0$
- Broadcast capacity region $\mathcal{Y}$

$$
\begin{aligned}
\max \quad & \alpha_s r_s + \alpha_t r_t \\
\text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x}_t = r_s \boldsymbol{d}_{st} \\
& \boldsymbol{M}\boldsymbol{x}_s = r_t \boldsymbol{d}_{ts} \\
& \boldsymbol{N}\boldsymbol{x}_t \leq \boldsymbol{y} \\
& \boldsymbol{N}\boldsymbol{x}_s \leq \boldsymbol{y} \\
& \boldsymbol{x}_s, \boldsymbol{x}_t \geq \boldsymbol{0}, \;\; r_s, r_t \geq 0 \\
& \boldsymbol{y} \in \mathcal{Y}
\end{aligned}
$$

Max-Min Bidirectional Flow (Model 2 & 3)

- Source vectors $\boldsymbol{d}_{st}, \boldsymbol{d}_{ts}$
- Incidence matrix $\boldsymbol{M}$, hyperarc-arc incidence matrix $\boldsymbol{N}$
- Broadcast capacity region $\mathcal{Y}$

$$
\begin{aligned}
\max \quad & \min\{r_s, r_t\} \\
\text{s. t.} \quad & \boldsymbol{M}\boldsymbol{x}_t = r_s \boldsymbol{d}_{st} \\
& \boldsymbol{M}\boldsymbol{x}_s = r_t \boldsymbol{d}_{ts} \\
& \boldsymbol{N}\boldsymbol{x}_t \leq \boldsymbol{y} \\
& \boldsymbol{N}\boldsymbol{x}_s \leq \boldsymbol{y} \\
& \boldsymbol{x}_s, \boldsymbol{x}_t \geq \boldsymbol{0}, \;\; r_s, r_t \geq 0 \\
& \boldsymbol{y} \in \mathcal{Y}
\end{aligned}
$$

Bidirectional Max-Flow Min-Cut Theorem (Model 2 & 3)

The value of the minimum s-t and t-s cut equals the value of the max-min bidirectional flow with network coding, i.e.,

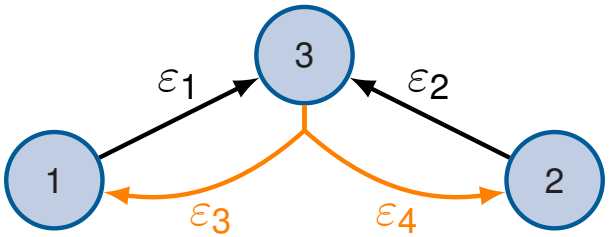
$$\max\Big\{\min\{r_s, r_t\} : \boldsymbol{M}\boldsymbol{x}_t = r_s\boldsymbol{d}_{st},\ \boldsymbol{0} \le \boldsymbol{x}_t,\ \boldsymbol{N}\boldsymbol{x}_t \le \boldsymbol{y},$$

$$\boldsymbol{M}\boldsymbol{x}_s = r_t\boldsymbol{d}_{ts},\ \boldsymbol{0} \le \boldsymbol{x}_s,\ \boldsymbol{N}\boldsymbol{x}_s \le \boldsymbol{y},\ \boldsymbol{y} \in \mathcal{Y}\Big\}$$

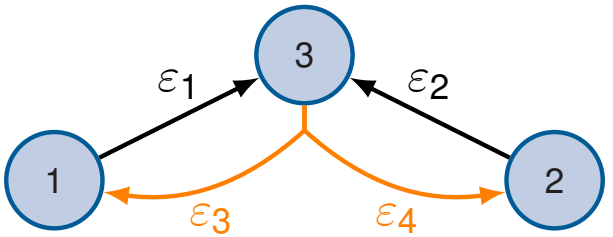
$$=$$

$$\max_{\boldsymbol{y}\in\mathcal{Y}} \min\left\{ v(S) = \sum_{\substack{j\in H:\mathrm{Tail}(j)\in S \\ \mathrm{Head}(j)=N_{\mathrm{Tail}(j)}\setminus S}} y_j : S \text{ is } s\text{-}t \text{ or } t\text{-}s \text{ cut} \right\}$$

Example: Two-way relay network

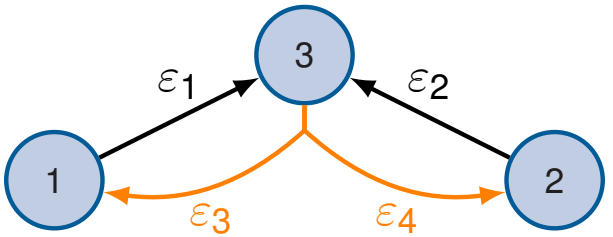


Example: Two-way relay network



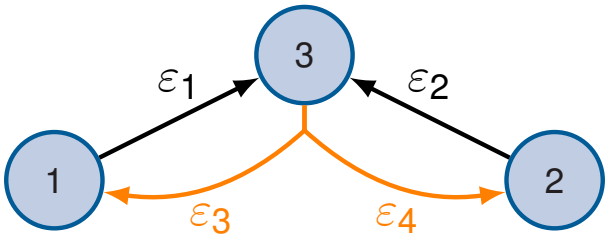
- Cuts
  - 1–2:
  - 2–1:

Example: Two-way relay network



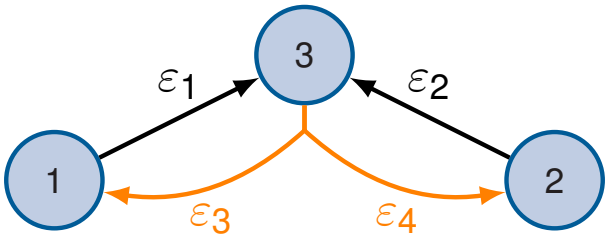
- Cuts
  - 1–2: $\{1\},\{1,3\}$
  - 2–1: $\{2\},\{2,3\}$
- Cut values

Example: Two-way relay network



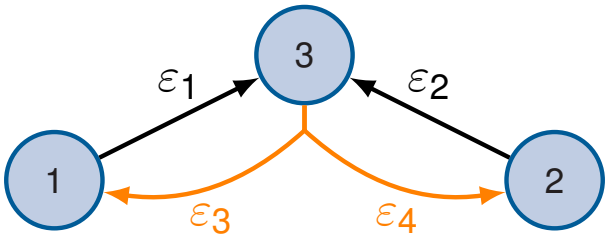
- Cuts
  - 1–2: $\{1\},\{1,3\}$
  - 2–1: $\{2\},\{2,3\}$
- Cut values
  - $v(\{1\}) = \tau_1(1-\varepsilon_1)$

Example: Two-way relay network



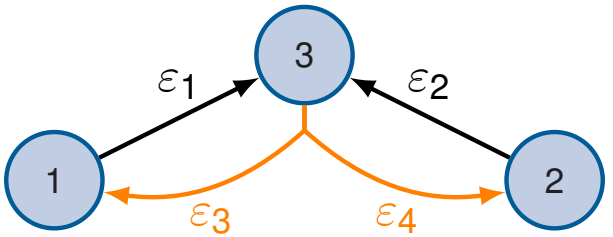
- Cuts
  - 1–2: $\{1\},\{1,3\}$
  - 2–1: $\{2\},\{2,3\}$
- Cut values
  - $v(\{1\}) = \tau_1(1-\varepsilon_1)$
  - $v(\{1,3\}) = \tau_3(1-\varepsilon_4)$

Example: Two-way relay network



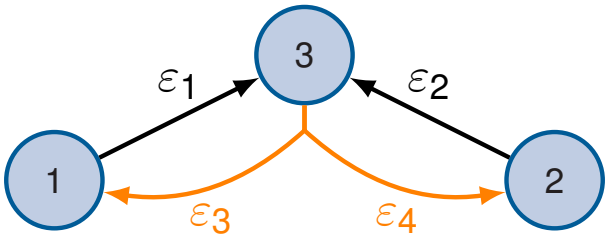
- Cuts
  - 1–2: $\{1\},\{1,3\}$
  - 2–1: $\{2\},\{2,3\}$
- Cut values
  - $v(\{1\}) = \tau_1(1-\varepsilon_1)$
  - $v(\{1,3\}) = \tau_3(1-\varepsilon_4)$
  - $v(\{2\}) = \tau_2(1-\varepsilon_2)$

Example: Two-way relay network



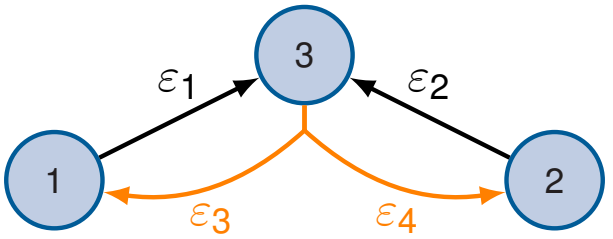
- Cuts
  - 1–2: $\{1\},\{1,3\}$
  - 2–1: $\{2\},\{2,3\}$
- Cut values
  - $v(\{1\}) = \tau_1(1-\varepsilon_1)$
  - $v(\{1,3\}) = \tau_3(1-\varepsilon_4)$
  - $v(\{2\}) = \tau_2(1-\varepsilon_2)$
  - $v(\{2,3\}) = \tau_3(1-\varepsilon_3)$
- Rate bounds

Example: Two-way relay network



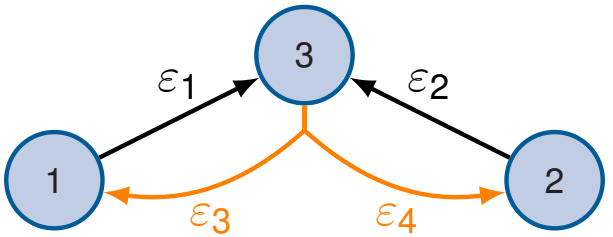
- Cuts
  - 1–2: $\{1\},\{1,3\}$
  - 2–1: $\{2\},\{2,3\}$
- Cut values
  - $v(\{1\}) = \tau_1(1-\varepsilon_1)$
  - $v(\{1,3\}) = \tau_3(1-\varepsilon_4)$
  - $v(\{2\}) = \tau_2(1-\varepsilon_2)$
  - $v(\{2,3\}) = \tau_3(1-\varepsilon_3)$
- Rate bounds
  - $r_1 \leq \min\{v(\{1\}), v(\{1,3\})\} = \min\{\tau_1(1-\varepsilon_1), \tau_3(1-\varepsilon_4)\}$

Example: Two-way relay network



- Cuts
  - 1–2: $\{1\},\{1,3\}$
  - 2–1: $\{2\},\{2,3\}$
- Cut values
  - $v(\{1\}) = \tau_1(1-\varepsilon_1)$
  - $v(\{1,3\}) = \tau_3(1-\varepsilon_4)$
  - $v(\{2\}) = \tau_2(1-\varepsilon_2)$
  - $v(\{2,3\}) = \tau_3(1-\varepsilon_3)$
- Rate bounds
  - $r_1 \leq \min\{v(\{1\}), v(\{1,3\})\} = \min\{\tau_1(1-\varepsilon_1), \tau_3(1-\varepsilon_4)\}$
  - $r_2 \leq \min\{v(\{2\}), v(\{2,3\})\} = \min\{\tau_2(1-\varepsilon_2), \tau_3(1-\varepsilon_3)\}$

Example: Two-way relay network



- Rate region

$$\mathcal{R} = \{[r_1, r_2]^T \in \mathbb{R}^2 : r_1 \leq \tau_1(1-\varepsilon_1),\ r_1 \leq \tau_3(1-\varepsilon_4),$$
$$r_2 \leq \tau_2(1-\varepsilon_2),\ r_2 \leq \tau_3(1-\varepsilon_3),$$
$$\tau_1 + \tau_2 + \tau_3 \leq 1,\ \tau_1, \tau_2, \tau_3 \in [0,1]\}$$

Multicast Rate Region for Terminal Set $T \subset N$

- Max-Flow Region (Network Coding)

$$\mathcal{R} = \bigcup_{\boldsymbol{y} \in \mathcal{Y}} \left\{ \boldsymbol{r} \geq \boldsymbol{0} : \boldsymbol{M}\boldsymbol{x}_t = \sum_{s \in N \setminus \{t\}} r_s \boldsymbol{d}_{st}, \boldsymbol{0} \leq \boldsymbol{x}_t, \boldsymbol{N}\boldsymbol{x}_t \leq \boldsymbol{y} \;\; \forall t \in T \right\}$$

- Min-Cut Region

$$\mathcal{R} = \bigcup_{\boldsymbol{y} \in \mathcal{Y}} \left\{ \boldsymbol{r} \geq \boldsymbol{0} : \sum_{s \in S} r_s \leq v(S) = \sum_{\substack{j \in H : \mathrm{Tail}(j) \in S \\ \mathrm{Head}(j) = N_{\mathrm{Tail}(j)} \setminus S}} y_j \;\; \forall t \in T, S \subset N \setminus \{t\} \right\}$$

- Max-Flow Min-Cut Theorem: Both are equal!

IEEE 802.11 frame format

IEEE 802.11 uses three different frametypes:

- Data frames
  - Contain data of any kind (both user data and ”management traffic” such as ARP, neighbor discovery, DNS, etc.)
  - Payload may be encrypted
  - Various subtypes (e.g. QoS and many special formats for networks with AP)
- Management frames
  - Management traffic between stations, in particular to associate to an AP
  - No encryption
  - Various subtypes (e.g. beacons, association requests, etc.)
- Control frames
  - Frames assisting in media access
  - No encryption
  - Various subtypes (e.g. RTS / CTS, ACK, etc.)

Each frame type (even subtypes) has custom headers
$\Rightarrow$ variable length header (without explicit length specification)

The generic frame format looks as follows:

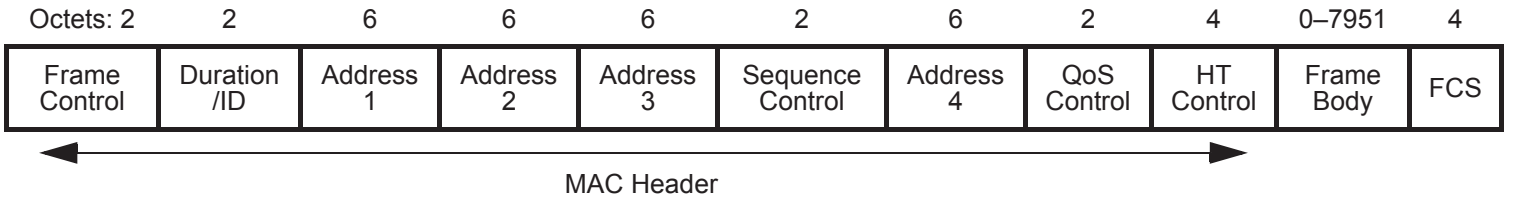


Figure: IEEE 802.11 generic header [**?**]

- Frame control
  - Defines frame type and subtype
  - Controls how MAC addresses shall be interpreted
  - Fragmentation control
  - Indicates whether or not a the payload is encrypted (but not how it is encrypted)
  - etc.

The generic frame format looks as follows:

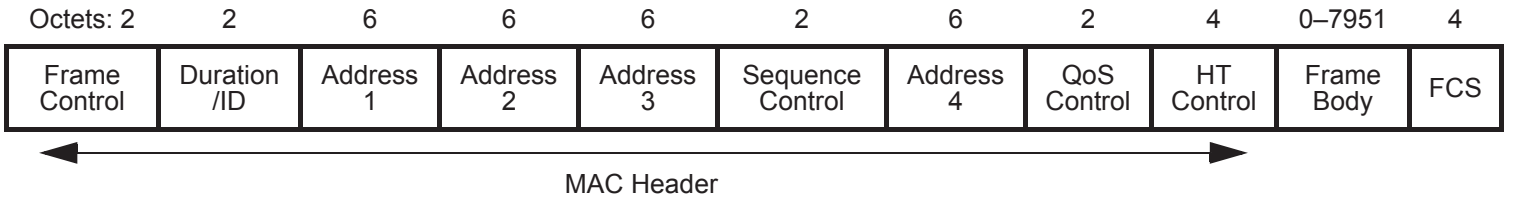


Figure: IEEE 802.11 generic header [**?**]

- Duration / ID
  - Meaning and content differs between frame types
  - One application is to assist in virtual carrier sensing, i. e., the expected duration of a transmission is specified

The generic frame format looks as follows:

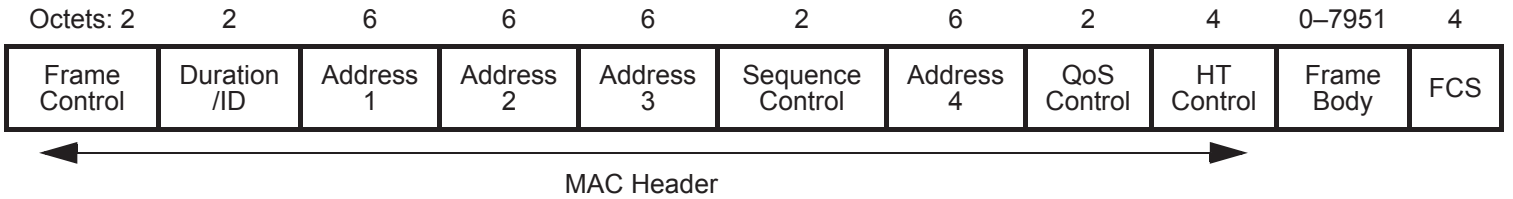


Figure: IEEE 802.11 generic header [**?**]

- 4 MAC addresses
  - Interpretation depends on the ToDS / FromDS bits in the frame control field
  - Not all addresses may be present (infrastructure mode commonly uses 3 addresses)
  - MAC addresses are compatible with IEEE 802.3

The generic frame format looks as follows:

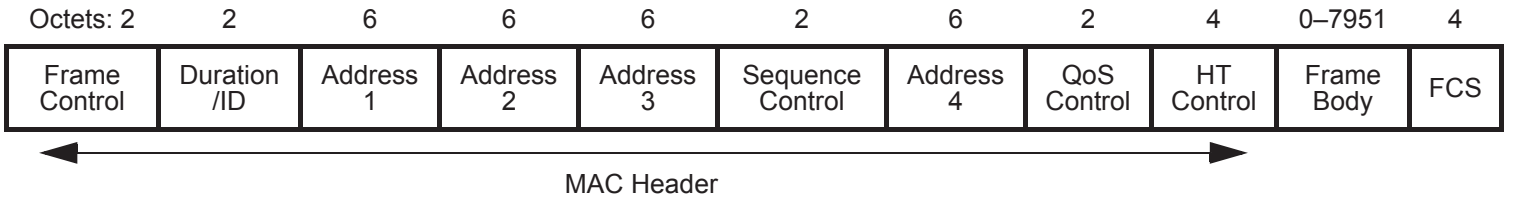


Figure: IEEE 802.11 generic header [**?**]

- Sequence Control
  - Consists of a fragment number (4 bit) amd a sequence number (12 bit)
  - Fragment number is used fragmentation and reassemblation of frames
  - Sequence number is needed for link-layer acknowledgements

The generic frame format looks as follows:

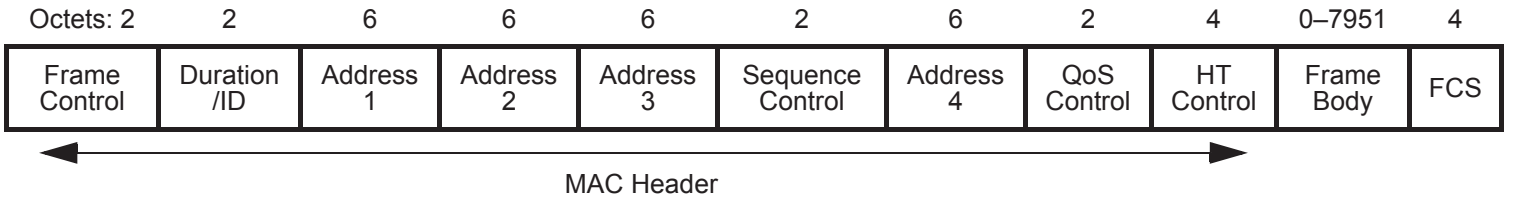


Figure: IEEE 802.11 generic header [**?**]

- QoS control
  - Used for quality of service (traffic classes, priorities, etc.)

The generic frame format looks as follows:

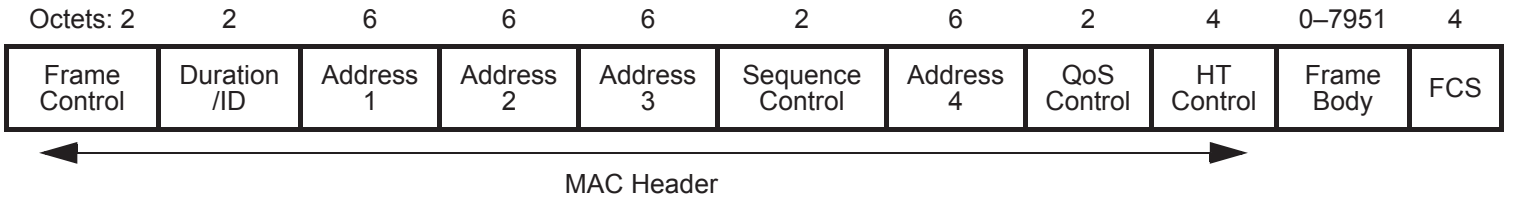


Figure: IEEE 802.11 generic header [**?**]

- Frame body
  - Everything that is considered as payload
  - May be encrypted
  - Contains other headers (even before the layer 3 header)

The generic frame format looks as follows:

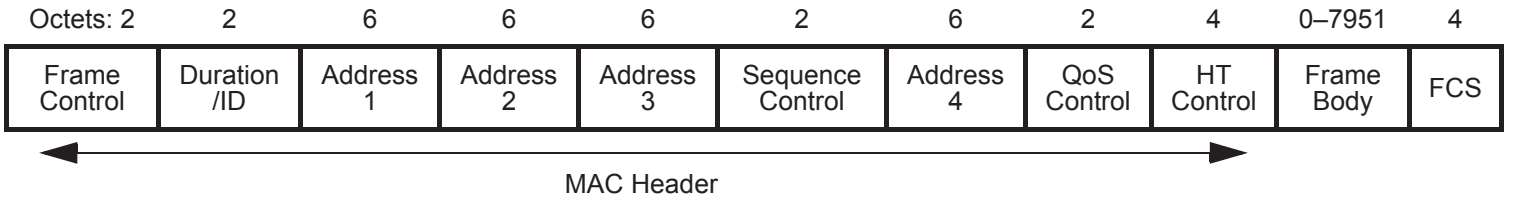


Figure: IEEE 802.11 generic header [**?**]

- FCS
  - Frame check sequence to detect transmission errors
  - 32 bit CRC

There (at least) 2 weird things on this format:

1. There is nothing that specifies the next layer protocol
2. The maximum frame body size of 7951 B exceeds the common MTU of 1500 B quite a bit

The first one is quickly explained:

- The frame body contains a SNAP header (subnetwork access protocol)
- It specifies the next layer protocol, whatever it might be
- Unfortunately the SNAP header is again of variable length
- There might be encryption headers before the SNAP header

The second one takes a bit longer, more on that later.

IEEE 80211 MAC

CSMA / CA is used:

- Sense the medium for ongoing transmission before transmitting (”listen before talk”)
- Since collisions cannot be reliably detected (hidden stations, sensing while transmitting), collisions have to be avoided

How is collision avoidance implemented in IEEE 802.11:

- So called coordination functions define the collision avoidance scheme
- The most basic method is the distributed coordination function (DCF)
- All other methods are based or derived from the DCF
- Optionally, nodes may use RTS /CTS protection

Distributed coordination function (DCF)

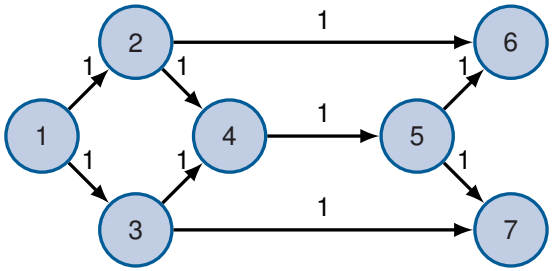


Figure: Media access via distributed coordination function (DCF)

Assuming that a node is backlogged:

1. Medium is sensed until it becomes idle
2. The medium has to be idle for a specific minimum idle time (called inter frame spacing)
3. The node draws a independently and unformly distributed number from a contention window
4. The node further defers transmission for this number of time slots
   - 4.1 After this backoff and if the medium is still idle, the node starts transmitting
   - 4.2 Otherwise transmission is deferred and the process starts from scratch when the medium becomes idle again

- In contrast to IEEE 802.3, the contention window $W = \{0, 1, \ldots, m\}$ has a minimum size of $m > 0$.
- If a transmission error eoccurs, i. e., a data frame is not acknowledged by the receiver, the contention window increased:

$$m \equiv C(n) = \min\left\{2^{n+k} - 1, 255\right\},$$

  where k defines the minimum size (depends on the coordination function) and n is the number of failed transmission attempts.
- A common value for $C(0)$ is 15.

How severe is it?

- Let the random variable C_n denote the number of backoff slots drawn for a given transmission attempt.
- Assuming that only one node is backlogged and no transmission errors, there is an additional idle time of $\mathrm{E}[C_0]$.

Example: HT mixed mode, 5 GHz (802.11n)

- Slot time is 9 µs, $C(0) = 15 \Rightarrow 67{,}5$ µs
- Inter frame spacing with DCF adds another 34 µs
- The average total delay for media access (without PHY headers) is therefore $\Delta t = 110{,}5$ µs

[13] MAC PDU

Example: HT mixed mode, 5 GHz (802.11n)

- Slot time is 9 µs, $C(0) = 15 \Rightarrow 67{,}5$ µs
- Inter frame spacing with DCF adds another 34 µs
- The average total delay for media access (without PHY headers) is therefore $\Delta t = 110{,}5$ µs

How much time is needed for the actual tansmission?

- Assume an MPDU[13] of $l = 1500$ B, and forget about any other overhead that might exsist
- Assume a bit rate of $r = 150 \frac{\text{Mbit}}{\text{s}}$ (maximum rate of 802.11n with one antenna)
- The actual transmission lasts only $t = l/r = 80$ µs

[13] MAC PDU

Example: HT mixed mode, 5 GHz (802.11n)

- Slot time is 9 µs, $C(0) = 15 \Rightarrow 67{,}5$ µs
- Inter frame spacing with DCF adds another 34 µs
- The average total delay for media access (without PHY headers) is therefore $\Delta t = 110{,}5$ µs

How much time is needed for the actual tansmission?

- Assume an MPDU[13] of $l = 1500$ B, and forget about any other overhead that might exsist
- Assume a bit rate of $r = 150 \frac{\text{Mbit}}{\text{s}}$ (maximum rate of 802.11n with one antenna)
- The actual transmission lasts only $t = l/r = 80$ µs

such efficient, very speed, wow!

[13] MAC PDU

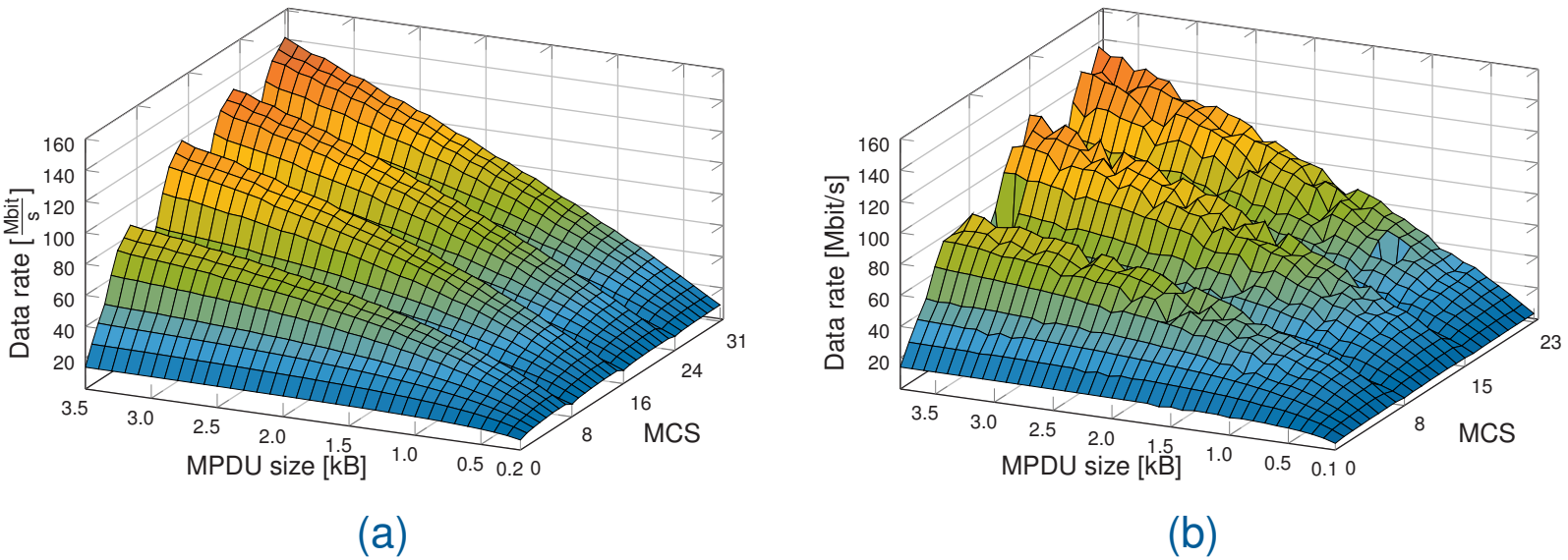


Figure: TX simulation (a) and RX measurement (b) using AR9390 chipsets

Besides large MPDUs, IEEE 802.11 has several mechanisms to reduce MAC delays:

- APs help to coordinate medium access
- Stations may aggregate multiple frames into an AMPDU
- etc.

Most mechanisms require an AP, i. e., infrastructure mode, and more complex coordination functions.

IEEE 802.11 service sets

- Basic service set (BSS) or infrastructure mode consists of an AP with all associated STAs
  - Identified by its BSSID (usually the MAC address of the AP)
  - STAs do not communicate directly with each other, the AP relays messages

IEEE 802.11 service sets

- Basic service set (BSS) or infrastructure mode consists of an AP with all associated STAs
  - Identified by its BSSID (usually the MAC address of the AP)
  - STAs do not communicate directly with each other, the AP relays messages
- Extended service set (ESS) or distribution system (DS) is a set of connected APs (e.g. over Ethernet) and their associated STAs
  - Identified by its ESSID (that is what you see when searching for networks)
  - APs relay messages to other APs

IEEE 802.11 service sets

- Basic service set (BSS) or infrastructure mode consists of an AP with all associated STAs
  - Identified by its BSSID (usually the MAC address of the AP)
  - STAs do not communicate directly with each other, the AP relays messages
- Extended service set (ESS) or distribution system (DS) is a set of connected APs (e.g. over Ethernet) and their associated STAs
  - Identified by its ESSID (that is what you see when searching for networks)
  - APs relay messages to other APs
- Independent basic service set (IBSS) or ad-hoc mode is a set of STAs communicating directly with each other without AP
  - STAs can communicate only with other STAs in range
  - STAs do not automatically relay messages on behalf of others
  - An IBSS may form a mesh network when suitable routing protocols are installed

How is a BSS formed?

- The AP broadcasts beacons in regular intervals, which contain
  - the BSSID (and ESSID),
  - channel, frequency, supported hardware modes, data rates,
  - and many more information.
- When an STA *joins* a BSS, a four-way-handshake is performed.
- Afterwards, the STA is associated, i. e., link-layer connectivity is established.

After association many more things might happen, e.g.

- negotiation of encryption, authentication etc.,
- obtaining a network layer address,
- ...

libmoep80211

What is libmoep80211

libmoep80211 is a shared library written in C that allows to

- inject cooked IEEE 802.11 frames (native mode) or
- frames based on a proprietary, extensible frame format to develop and evaluate custom link-layer protocols,
- and it is primarily developed and maintained by Maurice Leclaire.

What is libmoep80211

libmoep80211 is a shared library written in C that allows to

- inject cooked IEEE 802.11 frames (native mode) or
- frames based on a proprietary, extensible frame format to develop and evaluate custom link-layer protocols,
- and it is primarily developed and maintained by Maurice Leclaire.

Why not opening raw sockets?

What is libmoep80211

libmoep80211 is a shared library written in C that allows to

- inject cooked IEEE 802.11 frames (native mode) or
- frames based on a proprietary, extensible frame format to develop and evaluate custom link-layer protocols,
- and it is primarily developed and maintained by Maurice Leclaire.

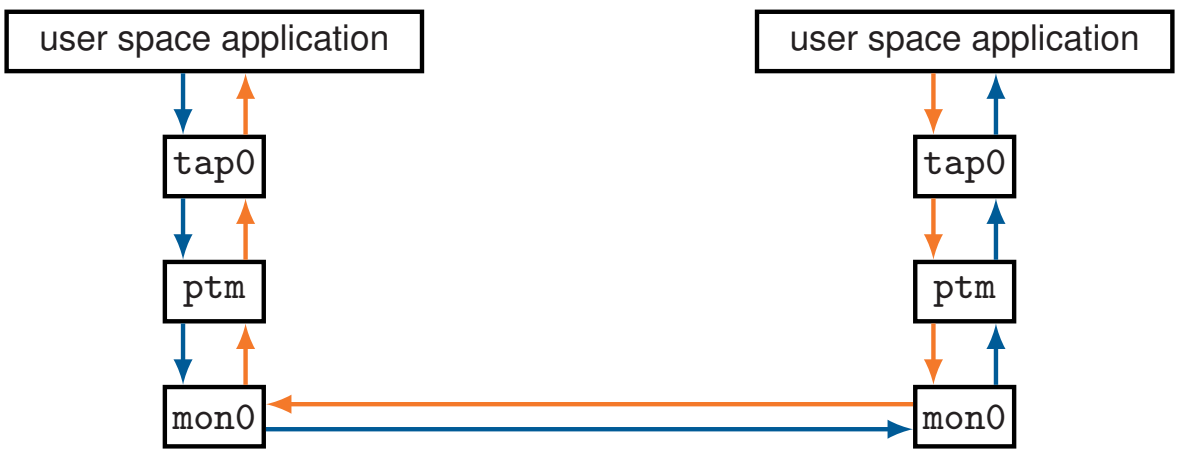
Why not opening raw sockets? ...libmoep80211 uses raw sockets but

- it hides most of the complexity of
  - creating monitor mode interfaces,
  - setting interface parameters,
  - parsing radiotap headers, etc.,
- and allows a convenient way to *pair* a monitor interface with a TAP interface.

Example: `ptmsimple`

The `ptmsimple` (PTM stands for packet transfer module) is

- the most simple kind of *module* using libmoep80211 to
- relay packets by
  - accepting IEEE 802.3 frames over a virtual Ethernet interface (tap0),
  - converting those frames to a custom format suitable for wireless transmission,
  - sending those frames over a monitor interface, and
  - translating incoming frames from the monitor interface back to valid IEEE 802.3 frames.



Example: `ptmsimple`

The tap interface presents itself like a physical Ethernet device, i.e.,

- it has a MAC address and
- can be assigned IP(v6) addresses:

```
tap0: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast ...
    link/ether 06:36:10:3e:a8:b0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::436:10ff:fe3e:a8b0/64 scope link
```

Example: `ptmsimple`

The tap interface presents itself like a physical Ethernet device, i.e.,

- it has a MAC address and
- can be assigned IP(v6) addresses:

```
tap0: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast ...
    link/ether 06:36:10:3e:a8:b0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::436:10ff:fe3e:a8b0/64 scope link
```

Advantages:

- Applications are completely unaware of the translation.
- It works with any kind of traffic (even ARP).
- We have any control about the radio interface we can ever have without writing custom device drivers.

How is it implemented?

1. Create a tap and a monitor device:

```
if (!(tap = moep_dev_ieee8023_tap_open(args.addr, &args.ip, 24,
            args.mtu + sizeof(struct ether_header)))) {
    fprintf(stderr, "ptmsimple: error: %s\n", strerror(errno));
    return -1;
}

if (!(rad = moep_dev_moep80211_open(args.rad, args.freq,
            moep80211_chan_width_20_noht,
            0, 0, args.mtu + radiotap_len(-1) +
            sizeof(struct moep80211_hdr) +
            sizeof(struct moep80211_hdr_pctrl)))) {
    fprintf(stderr, "ptmsimple: error: %s\n", strerror(errno));
    moep_dev_close(tap);
    return -1;
}
```

2. Set `rx_handler` for both devices that will be used as callbacks upon frame arrival:

```
moep_dev_set_rx_handler(tap, taphandler);
moep_dev_set_rx_handler(rad, radhandler);
```

3. *Pair* both devices and turn control to libmoepgf:

```
moep_dev_pair(tap, rad);
moep_run(sigh);
```

- The call to `moep_run()` turns control to libmoep80211gf.
- The internal event loop is essentially a wrapper for `pselect()`.
- Depending on which interface a frame is received, the appropriate hand ler is called:
  - If a frame arrives at the tap interface, the `taphandler()` is called and the received frame is passed to this hand ler.
  - The handler can translate the frame to a suitable format and sche dule it for transmission on the radio interface

- The call to `moep_run()` turns control to libmoep80211gf.
- The internal event loop is essentially a wrapper for `pselect()`.
- Depending on which interface a frame is received, the appropriate hand ler is called:
  - If a frame arrives at the tap interface, the `taphandler()` is called and the received frame is passed to this hand ler.
  - The handler can translate the frame to a suitable format and sche dule it for transmission on the radio interface

Do we have to turn control over to libmoepgf?

- Of course not.
- There is `moep80211_select()`, which works just like `pselect()` but still supports `rx_handlers`.
- More on that later ...

moep80211 frame format

There are two different ways to create radio interfaces:

- `moep_dev_ieee80211_open()`
  - Frames passed to the `rx_handler` will be ordinary IEEE 802.11 frames, including their link-layer headers.
  - The radiotap header will be in `moep80211_radiotap` since `ieee80211_radiotap` sucks.
- `moep_dev_moep80211_open()`
  - Frames passed to the `rx_handler` will be custom format that is based on the generic IEEE 802.11 header for data frames.
  - The radiotap header is again `moep80211_radiotap`.

In both cases, libmoepgf uses common format:

```
struct moep_frame {
    struct moep_frame_ops l1_ops;
    struct moep_frame_ops l2_ops;
    void *l1_hdr;
    void *l2_hdr;
    u8 *payload;
    size_t payload_len;
};
```

- `struct moep_frame` is private, typedeffed to `moep_frame_t`, and thus **not** made accessible.
- Use the interfaces instead:

```
// Returns the radiotap header
struct moep80211_radiotap *moep_frame_radiotap(moep_frame_t frame);

// Returns the IEEE80211 header (generic format, your have to parse it)
struct ieee80211_hdr_gen *moep_frame_ieee80211_hdr(moep_frame_t frame);

// Returns the moep80211_hdr common to all our custom frames
struct moep80211_hdr *moep_frame_moep80211_hdr(moep_frame_t frame);
```

- `struct moep_frame` is private, typedeffed to `moep_frame_t`, and thus **not** made accessible.
- Use the interfaces instead:

```
// Returns the radiotap header
struct moep80211_radiotap *moep_frame_radiotap(moep_frame_t frame);

// Returns the IEEE80211 header (generic format, your have to parse it)
struct ieee80211_hdr_gen *moep_frame_ieee80211_hdr(moep_frame_t frame);

// Returns the moep80211_hdr common to all our custom frames
struct moep80211_hdr *moep_frame_moep80211_hdr(moep_frame_t frame);
```

Warning

Never try to assemble a whole frame in memory and pass it to libmoepgf. It will not work, and it is meant that way. The library will serialize the frame for you before passing it to the driver.

The generic moep80211 header

When not operating in native mode, **all** radio frames will have this common header:

```
struct moep80211_hdr {
    u16 frame_control;
    u16 duration_id;
    u8 ra[IEEE80211_ALEN];
    u8 ta[IEEE80211_ALEN];
    u32 disc;
    u16 txseq;
    u16 seq_ctrl;
} __attribute__((packed));
```

The generic moep80211 header

When not operating in native mode, **all** radio frames will have this common header:

```
struct moep80211_hdr {
    u16 frame_control;
    u16 duration_id;
    u8 ra[IEEE80211_ALEN];
    u8 ta[IEEE80211_ALEN];
    u32 disc;
    u16 txseq;
    u16 seq_ctrl;
} __attribute__((packed));
```

- `frame_control` has the same meaning as for ordinary IEEE 802.11 frames.
- We set it to `FTYPE_DATA | STYPE_DATA` for **any** of our frames to avoid unexpected behavior of hardware.

The generic moep80211 header

When not operating in native mode, **all** radio frames will have this common header:

```
struct moep80211_hdr {
    u16 frame_control;
    u16 duration_id;
    u8 ra[IEEE80211_ALEN];
    u8 ta[IEEE80211_ALEN];
    u32 disc;
    u16 txseq;
    u16 seq_ctrl;
} __attribute__((packed));
```

- `duration_id` may be interpreted by other STAs.
- We set it to zero for now.

The generic moep80211 header

When not operating in native mode, **all** radio frames will have this common header:

```
struct moep80211_hdr {
    u16 frame_control;
    u16 duration_id;
    u8 ra[IEEE80211_ALEN];
    u8 ta[IEEE80211_ALEN];
    u32 disc;
    u16 txseq;
    u16 seq_ctrl;
} __attribute__((packed));
```

- `ra` is the 6 B receiver address of this frame.
- If we exploit the wireless broadcast advantage, we set it to the MAC broadcast address.

The generic moep80211 header

When not operating in native mode, **all** radio frames will have this common header:

```
struct moep80211_hdr {
    u16 frame_control;
    u16 duration_id;
    u8 ra[IEEE80211_ALEN];
    u8 ta[IEEE80211_ALEN];
    u32 disc;
    u16 txseq;
    u16 seq_ctrl;
} __attribute__((packed));
```

- `ta` is the 6 B transmitter address of this frame.
- This is **not** the MAC of our wireless interface but of the tap interface. Think about it!

The generic moep80211 header

When not operating in native mode, **all** radio frames will have this common header:

```
struct moep80211_hdr {
    u16 frame_control;
    u16 duration_id;
    u8 ra[IEEE80211_ALEN];
    u8 ta[IEEE80211_ALEN];
    u32 disc;
    u16 txseq;
    u16 seq_ctrl;
} __attribute__((packed));
```

- `disc` is a 4 B field that we call frame discriminator.
- In IEEE 802.11 data frames this would be the third MAC address.
- We choose a value that should be invalid as MAC adress.
- This way we can differentiate our own frames from normal IEEE 802.11 traffic.

The generic moep80211 header

When not operating in native mode, **all** radio frames will have this common header:

```
struct moep80211_hdr {
    u16 frame_control;
    u16 duration_id;
    u8 ra[IEEE80211_ALEN];
    u8 ta[IEEE80211_ALEN];
    u32 disc;
    u16 txseq;
    u16 seq_ctrl;
} __attribute__((packed));
```

- `txseq` are the latter 2 B of the third MAC address in IEEE 802.11 data frames.
- We use it as per-node TX sequence number, e.g. to estimate erasure probabilities.

The generic moep80211 header

When not operating in native mode, **all** radio frames will have this common header:

```
struct moep80211_hdr {
    u16 frame_control;
    u16 duration_id;
    u8 ra[IEEE80211_ALEN];
    u8 ta[IEEE80211_ALEN];
    u32 disc;
    u16 txseq;
    u16 seq_ctrl;
} __attribute__((packed));
```

- `seq_ctrl` is fragment number / sequence number of the normal IEEE 802.11 data frame header.
- Problem with this field is that the NIC's driver may play with it.
- It is safer to set it to zero and to ignore it on reception.

Only data frames?

Of course not. We use extension headers, e.g. the packet control header:

```
struct moep80211_hdr_pctrl {
    struct moep80211_hdr_ext hdr;
    u16 type;  // corresponding to the Ethertype
    u16 len;   // explicit length of the frame's payload
} __attribute__((packed));

struct moep80211_hdr_ext {
    u8 type;   // type of the extension header, e.g. MOEP80211_HDR_PCTRL
    u8 len;    // total length of the extension header
} __attribute__((packed));
```

- After the `moep80211_hdr` at **least one** extension header must follow.
- Bit 7 in the extension header's type field indicates whether or not another extension header follows.
- Type and length field precisely specify the extension header, and allow anyone to skip unknown extension headers.

How to add extension headers?

- Extension headers are part of the `l2_header` in the private `struct moep_frame`.
- How exactly extension headers are stored within a typedeffed `moep_frame_t` is not your business.

How to add extension headers?

- Extension headers are part of the `l2_header` in the private `struct moep_frame`.
- How exactly extension headers are stored within a typedeffed `moep_frame_t` is not your business.

Just let libmoep80211 do it for you:

- `moep_frame_add_moep80211_hdr_ext()`
  Add a new extension header to an existing frame.
- `moep_frame_set_moep80211_hdr_ext()`
  Replace an existing extension header by a new one
- `moep_frame_del_moep80211_hdr_ext()`
  Delete an extension header.
- `moep_frame_moep80211_hdr_ext()`
  Get a pointer to a specific extension header (or `NULL` if it does not exist).

Bibliography I

[1] S. Chachulski. Trading Structure for Randomness in Wireless Opportunistic Routing. M.sc. thesis, Massachusetts Institute of Technology, 2007.

[2] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *ACM SIGCOMM*, pages 169–180, 2007.

[3] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-throughput Path Metric for Multi-hop Wireless Routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom 2003, pages 134–146, New York, NY, USA, 2003. ACM.

[4] C. Gkantsidis and M. Goldberg. Avalanche: File Swarming with Network Coding. `http://research.microsoft.com/en-us/projects/avalanche/`.

[5] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOMM).*, volume 4, pages 2235–2245, March 2005.

[6] T. Ho and D. Lun. *Network Coding: An Introduction*. Cambridge University Press, 2008.

[7] G. Kroah. Linux Kernel Coding Syle. `https://www.kernel.org/doc/Documentation/CodingStyle`.

Bibliography II

[8] M. Médard and A. Sprintson. *Network Coding: Fundamentals and Applications*. Academic Press, 2011.